

ct. op 86. 8 Januari
school.

test d.m.g.
MOTOR Computer.
meenemen.



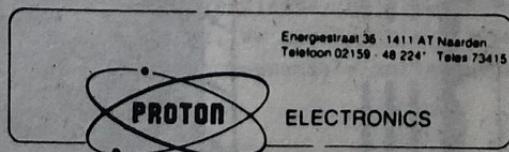
USER MANUAL

PC-1 MICROCOMPUTER

KURSUS INDUSTRIËLE MICROPROCESSORS

INHOUD

- Installatie
- Overzicht PC-1 commando's
- Memory-map
- PC-1 monitor



Energiestraat 36 1411 AT Naarden
Telefoon 02159 - 48 224 Telex 73415

1

PROTON PC-I UNIVERSELE COMPACT COMPUTER

In totaal 16 functie toetsen, waarvan er 12 geschikt zijn voor uitgebreide simulatie doelen.

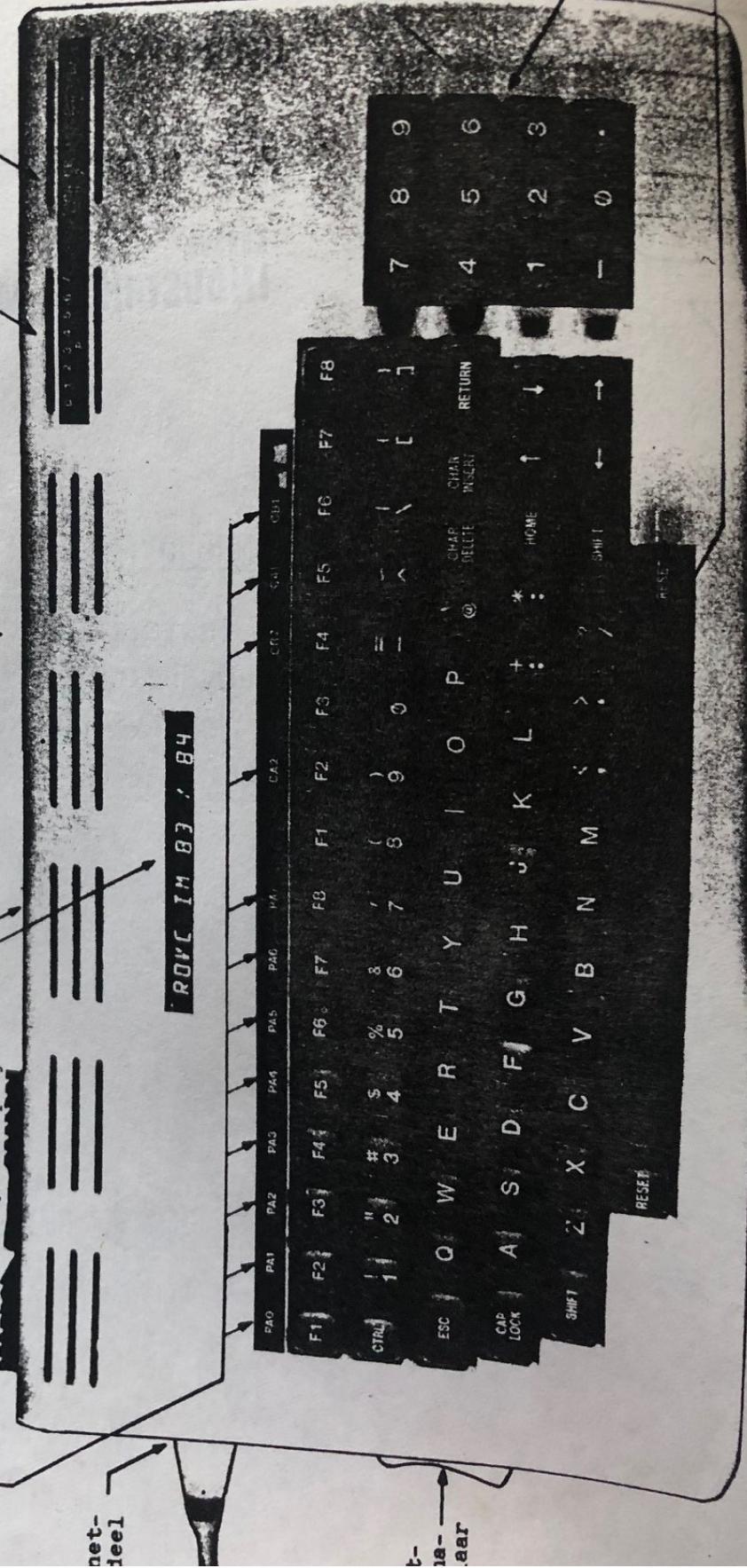
De eerste 8 toetsen blijven na bediening in de gewenste stand staan (bistabiel), terwijl de 8 meest rechtse toetsen pulsbediend zijn (monostabiel). LET OP een bediende toets betekent een nul.

flatcable I/O connector

koelplaat

DIN-aansluiting cassette recorder
16-karacter alfanumeriek display

10 met LED's uitgeruste simulatie uitgangen.



INSTALLATIE.

In figuur 1 is het complete bovenaanzicht van de mikrokomputer afgebeeld, zoals deze op de cursus gebruikt gaat worden.

funktie. Zie figuur 1. Zo treft u op de linker zijkant een opening voor het netsnoer aan, om de mikrokomputer van stroom te voorzien en tevens een robuust uitgevoerde netschakelaar waarmee de mikrokomputer aan en uit gezet kan worden.

Links achter bevindt zich de zogenaamde koelplaat. Deze koelplaat heeft de functie een groot deel van de in het apparaat optredende warmte af te staan aan de buitenlucht, waarvoor eveneens de ventilatie-gaatjes in de bovenkant van het apparaat bestemd zijn. Vooral de koelplaat kan in de praktijk lekker warm worden. Ca ± 60°C. Reden, dat u er altijd voor moet zorgen, dat de lucht goed bij de koelplaat kan komen. Naast de koelplaat vindt u op de achterkant een opening, waarop de kassettorecorder door middel van een zogenaamde DIN-plug kan worden aangesloten. Daarover straks meer. Naast de kassettorecorderaansluiting bevindt zich de flatcable aansluiting voor het aansluiten van extra I/O zoals relais, etc.

Op de bovenzijde van de komputer bevindt zich in het midden onderaan het grote toetsenbord met letters en cijfers zoals ook op een typemachine aanwezig is. Dat toetsenbord noemen we vanaf nu het ASCII-keyboard (zeg aski twee ki boort). Rechts daarvan is nog een klein toetsenbord met o.a. cijfertoetsen aangebracht. Dit toetsenbord zullen we in eerste instantie nog niet gebruiken. Overigens noemen we dat het decimal keypad (spreek uit: dissie mel ki pat). Met beide keyboards kunnen we de mikrokomputer vertellen wat we willen doen. De bovenste rij toetsen van het grote keyboard hoort niet bij het ASCII-schrijfmachinegedeelte, maar dient om eindschakelaars en signaalgevers te simuleren. De bovenste rij toetsen worden funktietoetsen genoemd.

Boven de funktietoetsen bevindt zich het zogenaamde display (spreek uit: dispree). Op het display kan de mikrokomputer ons vertellen wat wij aan het doen zijn. Hiermee kunnen we onder andere ook zien wat er in het geheugen staat en wat er in de mikroprocessor gebeurt. Daarover later meer. Boven het decimal keypad bevinden zich 10 rode lampjes of ook wel LED's. Deze LED's dienen ervoor aan te geven of een uitgang van de komputer een één of een nul is. Daarmee kunnen uitvoerorganen worden gesimuleerd.

Tot zover de benamingen en globaal de functie van de diverse onderdelen. Verbindt het relais I/O board op de juiste wijze met de mikrokomputer. Steek de stekker in het stopkontakt en de andere kant in het netdeel en schakel de mikrokomputer in met de hoofdschakelaar.

Hierdoor licht één van de display's een moment fel op waarna de naam van de mikrokomputer op het display verschijnt. Gebeurt dit niet - kans van 1 op 100 - druk dan even beide resettoetsen - die zich links en rechts van de lange toets bevinden (spatiebalk) - tegelijk in en laat deze onmiddellijk weer los.

PC-1 kommando's.

Om de mikrokomputer wat zinnigs te laten doen, moeten we op één of andere manier met hem kunnen praten. Dat is nodig om de komputer te kunnen vertellen wat hij moet doen. Hiervoor is de mikrokomputer voorzien van een vast speciaal programma (monitor), welke ervoor zorgt, dat er bij het indrukken van toetsen iets gebeurt. Dat kan bijvoorbeeld zijn het laden van een bepaalde geheugenplaats, het starten van een zelfgeschreven programma, het schrijven op cassette, enz. We spreken hierbij van kommando's. In figuur 2 zijn deze kommando's weergegeven. Deze kommando's zullen we aan de hand van een aantal voorbeelden uitdiepen.

Mocht tijdens het uitproberen van de voorbeelden er één op de laatste display verschijnen, dan heeft u een verkeerde - niet kommando - toets bediend. En wel op een moment, dat u wel een kommandotoets had moeten bedienen. Om dan verder te kunnen werken, helpen alleen de toetsen M, ←, → en de beide resettoetsen nog, waarbij we u aanbevelen de M te gebruiken of het pijltje →.

2

KOMMANDO	FUNKTIE
M van memory location	hiermee kan een adres worden opgegeven. toon de inhoud van het huidige adres na het adres met M te hebben opgegeven en/of schrijf een nieuwe inhoud weg naar het geheugen en toon afhankelijk van het laatst bediende pijltje het volgende of het vorige adres.
→	toon na het indrukken van de RETURN-toets de volgende geheugenplaats totdat het pijltje ← wordt bediend. Tevens het herhaald uitlezen van een adres.
←	toon na het indrukken van de RETURN-toets de vorige geheugenplaats totdat het pijltje → wordt bediend. Tevens het herhaald uitlezen van een adres.
G (van go):	start een programma op het door u opgegeven adres gevolgd door een RETURN.
S (van single step):	voer één instruktie uit en geef de inhoud van
O (van offset):	bereken de offset van een branch-instruktie door het gewenste adres op te geven. Na het bedienen van de RETURN-toets wordt de berekende offset op de gewenste geheugenplaats gezet.
<u>CHAR INSERT:</u>	voeg een geheugenpositie tussen.
<u>CHAR DELETE:</u>	verwijder een geheugenpositie.
R (van read):	lees een programma van de kassette in het geheugen. Wel eerst het nummer van het programma opgeven op adres \$ 0100.
W (van write):	schrijf een programma vanuit het geheugen naar de kassette. Vergeet niet op adres \$ 0100 het nummer van het programma op te geven, alsmede het eind- en startadres. schakel de line-assembler in. Hierdoor kunnen instructies in MNEMONIC vorm worden ingetoetst.
I (Instruct)	andere monitor functies weer actief. disassembler een instructie op een van te voren opgegeven adres en geef deze in MNEMONIC weer.
ESC (Escape)	geef de inhoud van de microprocessor registers weer. geef een adres op in de line-assembler mode.
K	verwijder de vorige ingetoetste key in MNEMONIC-code. Hexadecimale cijfertoetsen.
? (SHIFT + /)	
* (SHIFT + :)	
← bij gebruik van de line-assembler.	
O - 9 & A - F	

Het bekijken en wijzigen van het geheugen

Wanneer we de monitor vragen de inhoud van het geheugen te tonen, moeten we eerst opgeven welke geheugenplaats we willen zien. We moeten het ADRES opgeven met de toets M van memory opgeven.

Wanneer het adres opgegeven is vragen we de computer de inhoud van dit adres te laten zien met de RETURN toets. Stel: we vragen de inhoud van adres \$ 0200.

toets in	display toont
M	0000
2	0002
0	0020
0	0200
RETURN	0200 XX

geef nu het adres op.

XX = de inhoud van \$0200

Nu zien we wat de inhoud van geheugenplaats \$0200 is. Willen we dit veranderen, dan toetsen we op dit moment de nieuwe waarde in. Deze waarde wordt dan na het indrukken van de RETURN toets op de geheugenplaats \$0200 geschreven.

Tikt u bijvoorbeeld "88" (is \$88) in en daarna RETURN, dan staat er op adres \$ 0200 88. Het display toont hierbij geen dollar teken maar geeft wel alvast het volgende adres weer. In dit geval dus \$ 0201.

Om nu te kontrolieren of er inderdaad \$88 op adres \$ 0200 staat, kunt u op twee manieren te werk gaan.

1. toets in display toont

M	0000
2	0002
0	0020
0	0200
RETURN	0200 88

2. Alvorens met deze methode het eerst genoemde probleem op te lossen moeten we wel even terug naar adres \$ 0201 door eventueel na het proberen van oplossing 1 nog éénmaal op de RETURN toets te drukken.

Handel dan als volgt:

toets in display toont

← (terug)	0201 ?? willekeurige inhoud.
RETURN	0200 88

Met andere woorden in plaats van vooruit in het geheugen na het bedienen van de RETURN toets te gaan, gaan we door het bedienen van ← juist terug in het geheugen na het bedienen van de RETURN toets. Wanneer we daarin tegen →, M of de beide resettoetsen (tegelijk) bedienen wordt automatisch weer verder gelezen -na het bedienen van de RETURN toets-.

Om wat met de pijltjes en de RETURN toets te oefenen gaan we eens kijken wat er op adres \$ 0300 staat. Vervolgens doorlopen we het geheugen zowel op en neer en doen dit met behulp van de pijltjes toetsen en de RETURN toets.

Merk hierbij op, dat wanneer de RETURN toets ingedrukt houdt, automatisch die opdracht door de microcomputer wordt herhaald.

Het intoetsen van een programma gescheidt op dezelfde manier als in het voor-gaande voorbeeld. Dit doen we door op de gewenste adressen met de machine instrukties en bijbehorende data in te vullen.

Wanneer nu na het intikken van een programma blijkt dat we een locatie hebben overgeslagen, kunnen we de overgeslagen locatie vrij maken door alles vanaf dat punt in het geheugen tot aan het einde van het programma één positie op te schuiven. Hiervoor moeten we echter eerst het eindadres opgeven aan de monitor. Tot waar geschoven moet worden anders zou het gehele geheugen opgeschoven worden. Dit eindadres moeten we invullen op locatie \$102/\$103 (EOL/EOH). Stel ons programma beslaat de adressen \$0200 t/m \$0260 en we zijn op locatie \$0221 iets vergeten.

We vullen eerst het eindadres in:

toets in	display toont	
M	0000	we nemen \$0260 als eindadres
1	0001	en geven dat aan de monitor.
0	0010	
2	0102	
RETURN	0102 XX	geef het laatste deel (low byte)
6	0102 X6	
0	0102 60	
RETURN	0103 XX	nu het eerste deel (high byte)
0	0103 X0	
2	0103 02	
RETURN	0104 XX	en klaar.

Nu weet de monitor tot waar het geheugen verschoven dient te worden en moet hij alleen nog weten welke locatie moet worden vrijgemaakt.

Stel we zijn op locatie \$221 het getal \$12 vergeten:

toets in	display toont	
M	0000	
2	0002	geef de 'insert'-locatie
2	0022	
1	0221	
RETURN	0221 XX	
CHAR INSERT	0221 YY	
1	0221 Y1	nu schuift alles van \$221 t/m \$260
2	0221 12	één plaats op.
RETURN	0222 XX	voer nu het vergeten getal in

We hebben nu op locatie \$0221 het getal \$12 tussengevoegd en alle posities van \$0221 t/m \$0260 één plaats opgeschoven.

De omgekeerde bewerking van INSERT is DELETE. Hiermee wordt een locatie die 'te veel' is weggegooid en alle locaties daarachter tot EOL/EOH teruggeschoven. Willen we nu bij het bovenstaande voorbeeld van locatie \$0221 het ingevoegde getal (\$12) weer weghalen, dan toetsen we:

toets in	display toont	
M	0000	
2	0002	geeft het 'verwijder'-adres
2	0022	
1	0221	
RETURN	0221 12	
CHAR DELETE	0221 XX	en delete

Het in uitvoer brengen van een programma

Hiervoor zijn 2 methoden:

G = 'GO' Starten op een adres.

S = 'SST' Single step van één instructie.

Vóór het geven van een GO of een SST moet eerst een startadres worden opgegeven aan de monitor. Dit doen we door middel van:

M xxxx RETURN . Daarna drukken we op de GO of SST toets.

Bij GO start de monitor het programma, te beginnen bij het opgegeven startadres. Het programma stopt bij het uitvoeren van een BRK instructie (\$00), waarna de monitor weer actief wordt.

Met SST wordt er één instructie uitgevoerd en hierna op het display de inhoud van alle microprocessor-registers na die instructie weergegeven. De volgorde van de registers is van links naar rechts:

- Program counter { 4 cijfers};
- Accumulator { 2 cijfers};
- X-register { 2 cijfers};
- Y-register { 2 cijfers};
- Program status { 2 cijfers}.

Bij langdurig indrukken van de SST-toets wordt er om de $\frac{1}{2}$ seconde één instructie uitgevoerd en het resultaat weergegeven op het display.

De inhoud van de registers is op elk moment ook zonder het SST-kommando opvraagbaar door het bedienen van het kommando "?" wat door tegelijk indrukken van SHIFT en / gegeven kan worden.

Offset berekening

De monitor heeft de functie-toets 0 (niet nul) voor het berekenen van sprongafstanden (branch-offset). Deze branch-offset is het verschil tussen twee adressen. Wanneer een programma een Branchinstructie heeft, wordt het sprongadres opgegeven en moet achter de instructie code de offset worden ingevuld.

Bijvoorbeeld: Op adres \$ 0200 staat BMI naar adres \$ 0210.

De monitor kan de 'offset' op locatie \$0201 zelf berekenen uit het sprongadres en het huidige adres. De berekening is:

$$\text{OFFSET} = \$0210 - \$0201 - 1 = \$0E \quad (\text{OFFSET} = \text{TARGET-ADDR} - 1).$$

Het toetsvoorbeeld is:

toets in	display toont	
M	0000	we schrijven vanaf \$0200
2	0002	
0	0020	
0	0200	
RETURN	0200 XX	
3	0200 X3	
0	0200 30	code voor 'BMI'
RETURN	0201 XX	Hier moet de offset komen
0 en geen 2 nul	0000	geef nu het 'target'adres (sprongadres)
1	0002	
0	0021	
RETURN	0201 OE	de offset is nu berekend en kan worden afgelezen.
RETURN	0202 XX	de offset is ingevuld op adres \$ 0201.

Wanneer het bestemmingsadres te ver weg is voor de (8-bits) offset, worden in het data-veld twee puntjes getoond.

Het gebruik van een cassette recorder voor het opslaan van programma's.

Met een cassette of tape recorder kunnen op een relatief goedkope manier programma's worden opgenomen en weergegeven. Dit gebeurt als volgt:

Na de instructiedump komen er toontjes uit de mikroprocessor, welke kunnen worden opgenomen op band. Deze toontjes stellen énen en nullen voor

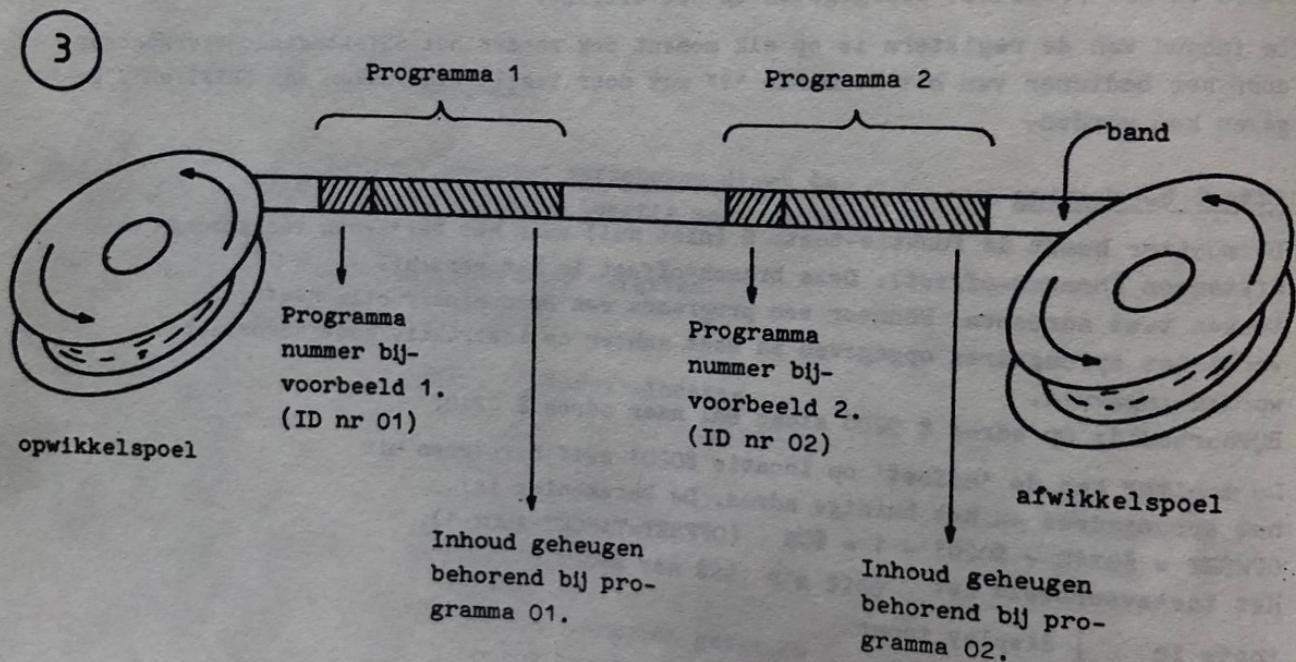
Bij het opnemen (laden) van een programma gebeurt het volgende:

de toontjes op band worden nu aan de computer toegevoerd, verwerkt en in het geheugen gezet. Omdat de mikroproces-

sor zelf niets weet, moet hem door middel van instruk-

ties verteld worden wat er gedumpt of geladen moet worden. Hiervoor zijn verschillende gegevens nodig.

Op de eerste plaats welke geheugenplaatsen uit de mikroprocessor op de band moeten wor- den gezet en omdat er op een cassette meerdere programma's gezet kunnen worden, het pro-gramma wat hij moet laden. In figuur 3 is een schematische voorstelling gegeven van twee programma's, die op een band staan.



Een onhebbelijkheid van cassette-bandjes is, dat er stukjes in zitten, die geen magne-tisch signaal opnemen. Zulke stukjes worden "dropouts" genoemd. Dropouts bij het dumpen van een programma kunnen grote problemen veroorzaken. Kies dus voor het opslaan van pro-gramma's een goede kwaliteitsband, bijvoorbeeld:

TDK, AGFA, BASF C 60, dus geen C 90 of erger C 120.

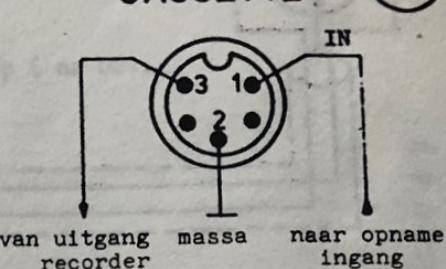
Aan de cassette recorder worden geen hoge eisen gesteld. Een eenvoudig huis/tuin/keuken apparaat voldoet goed. Let er echter op, dat de bandsnelheid wel konstant is, daar an-ders bij het afdraaien, de opgenomen toontjes lager of hoger worden. De mikroproces-sor kan dat verschil in toonhoogte niet goed verwerken.

Het aansluiten van de cassette-recorder op de mikroprocessor.

Vanwege de grote hoeveelheid cassette-en tape-recorders, die er in de handel zijn, is het niet mogelijk voor alle types de juiste aansluiting op de mikroprocessor te tekenen. Eén ding hebben de meeste recorders echter gemeen en dat is een ingang voor opname en een uitgang voor weergave. In figuur 4 is de aansluiting voor een recorder op de mikroprocessor gegeven. Hierbij dient punt 1, die een spanning aangeeft van ± 10 mV uit de mikroprocessor verbonden te worden met de microfooningang van de recorder of met de line low-ingang van een recorder met een gemeenschappelijke in/uitgangsplug.

Punt 3 van de ingangsplug dient verbonden te worden met de uitgang line, earphone of speaker (spr) van de recorder. Punt 2 is een gemeenschappelijke massa-aansluiting. Voor verschillende typen recorders zijn speciale verloepsnoeren verkrijgbaar. In figuur 5 tot en met 7 zijn de meest voorkomende type recorderaansluitingen getekend met vermelding van het type verloepsnoer. De snoeren zijn van het merk AMROH en verkrijgbaar bij de meeste HIFI-zaken.

CASSETTE 4

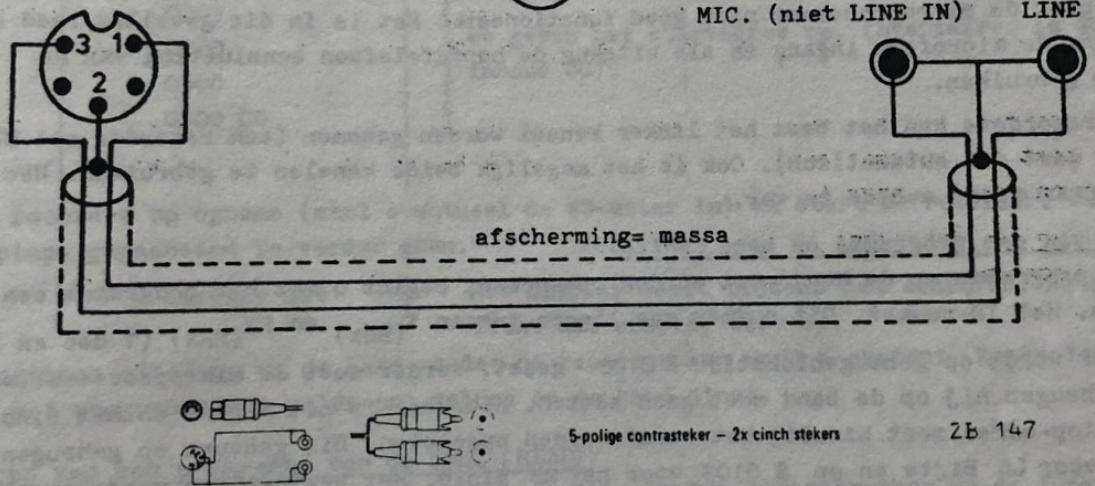


MICROPROCESSOR

5

RECODER

MIC. (niet LINE IN) LINE OUT



5-polige contraststeker - 2x cinch stekers

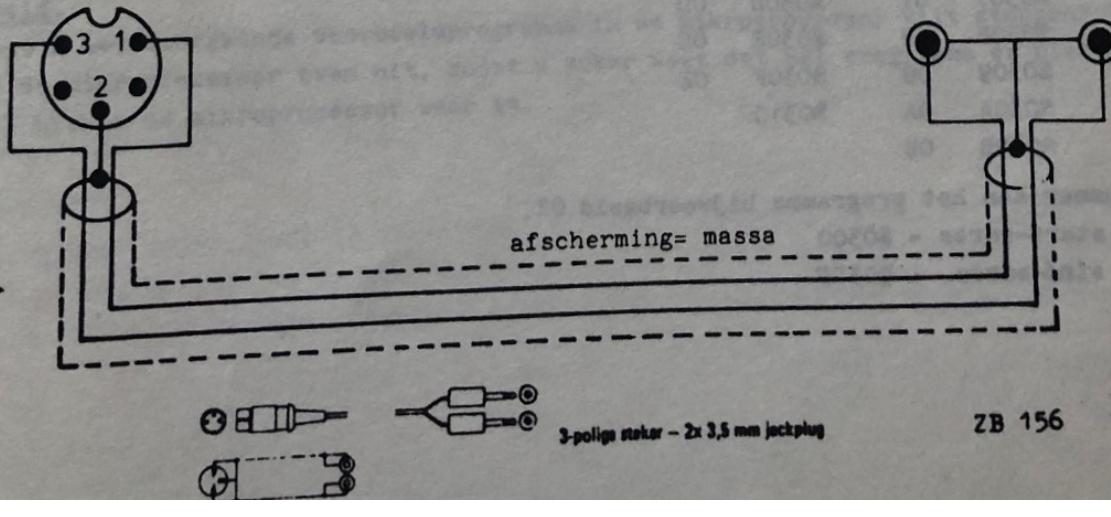
ZB 147

MICROPROCESSOR

6

RECODER

MIC. SPR. of EAR.



3-polige steker - 2x 3,5 mm jackplug

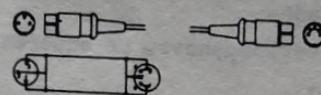
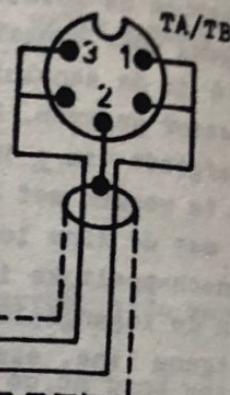
ZB 156

⑦

MICROPROCESSOR



RECODER (DIN)



3-polige steker - 5-polige steker

ZB 716

Het gebruik van een recorder met DIN aansluiting kan soms grote problemen opleveren bij het afspeLEN van de band. Dit komt omdat sommige DIN recorders een te lage uitgangsspanning afgeven, waardoor de microprocessor niet goed functioneert. Het is in die gevallen raadzaam om als ingang de microfoon ingang en als uitgang de hoofdtelefoon aansluiting van de recorder te gebruiken.

Bij stereo recorders kan het best het linker kanaal worden genomen (een recorder met DIN-aansluiting doet dit automatisch). Ook is het mogelijk beide kanalen te gebruiken. Het aansluiten daarvan voert echter te ver.

Het opnemen van een programma op band (WRITE).

Wanneer een programma op de band moet worden opgenomen, begint u met het programma een nummer te geven. Het ID-nummer. Dit nummer mag liggen tussen 1 (Hex) en FE (HEX) (1 dec en 254 dec). Dit ID-nummer wordt op geheugenlokatie \$0100 gezet. Verder moet de mikroprocessor weten welk stuk geheugen hij op de band moet gaan zetten. Hiervoor is het begin- en het eind-adres nodig. Het stop-adres moet hierbij het eerst worden opgegeven. Dit gebeurt op geheugenlokatie \$0102 voor LS Bijte en op \$0103 voor het MS Bijte. Het begin-adres wordt als laatste opgegeven, waarna gedumpt kan worden.

Voorbeeld.

In het geheugen staat het volgende programma dat gedumpt moet worden. (Toets dit eventueel om uit te proberen in).

\$0300	00	\$0306	06	\$030C	0C
\$0301	01	\$0307	07	\$030D	0D
\$0302	02	\$0308	08	\$030E	0E
\$0303	03	\$0309	09	\$030F	0F
\$0304	04	\$030A	0A	\$0310	
\$0305	05	\$030B	0B		

1. Geef een nummer aan het programma bijvoorbeeld 02.
2. Bepaal het start-adres = \$0300
3. Bepaal het eind-adres = \$030F

Handel nu als volgt:

toets in	display toont	kommentaar
M	0000	
1	0001	
0	0010	
0	0100	
RETURN	0100 XX	
0	0100 X0	
2	0100 02	
RETURN	0101 XX	
M	0000	
1	0001	
0	0010	
2	0102	
RETURN	0102 XX	
0	0102 X0	
F	0102 OF	
RETURN	0103 XX	
0	0103 X0	
3	0103 03	
RETURN	0104 XX	
M	0000	
3	0003	
0	0030	
0	0300	
RETURN	0300 CO	

Zet de recorder op opname (stel eventueel de VU-meter in) en start de recorder. Pas als de aanlooptape gepasseerd is verder gaan.

W | display uit | Het display dooft tijdens het schrijven. Na het schrijven licht het display weer op.

Het programma staat nu op de band, zodat de recorder kan worden gestopt. Kontroleer echter wel of het programma met voldoende volume op de band staat.

Het lezen van een programma van de band (READ).

Om een programma van de band in de mikroprocessor te krijgen gebruiken we de functie READ. De mikroprocessor moet ook hier weer verteld worden welk programma hij moet verwerken. Op adres \$0100 moet weer het ID-nummer ingevuld worden. Bij de functie READ hebben de ID-nummers 00 en FF een speciale betekenis evenals de inhoud van adres \$0101. Normaal zijn ze echter niet nodig. Voor de liefhebbers vindt u er aan het eind van de bijlage meer over.

Voorbeeld.

Stel dat u het voorgaande voorbeeldprogramma in de mikroprocessor wilt stoppen? Zet dan als eerste de mikroprocessor even uit, zodat u zeker weet dat het programma er niet meer in zit. Schakel hierna de mikroprocessor weer in.

Handel dan als volgt:

toets in	display toont	kommentaar
M	0000	
1	0001	
0	0010	
0	0100	
RETURN	0100 XX	
0	0100 XO	{ We geven het ID-nummer op van het programma dat
2	0100 02	geladen moet worden.
RETURN	0101 XX	

Zet de recorder klaar maar start de recorder nog niet. (Denk er om dat u niet halverwege het programma begint maar liefst een stuk er voor).

R

Het display gaat nu uit.

Start de recorder. Zodra het display oplicht, is het programma geladen en kan de recorder gestopt worden. Kontroleer of het programma in het geheugen zit.

Speciale mogelijkheden met het binnenvullen van programma's in de mikroprocessor.

Wanneer voor het laden het ID-nummer op \$00 gezet wordt, laadt de mikroprocessor het eerstvolgende programma wat hij op de band tegenkomt. Dus onafhankelijk wat voor een ID-nummer dat programma heeft.

Wordt voor het laden het ID-nummer op \$FF gezet, dan laadt de mikroprocessor evenals bij \$00 het eerste programma dat hij tegenkomt. Echter u kunt zelf het startadres opgeven. Dus is een programma geschreven op de adressen \$0300 tot \$0350 dan kunt u met het ID-nummer \$FF het programma laden op bijvoorbeeld de adressen \$0200 tot \$0250. Wanneer men een programma laadt en op andere adressen plaatst, dan moeten wel alle jumpadressen worden aangepast.

Om te controleren of het juiste programma geladen is, kan adres \$ 0101 bekijken worden. Hier staat namelijk het ID-nummer van het programma dat als laatste is geladen.
Voorbeeld.

Stel dat u onderstaand programma met het ID-nummer 02 van de tape in de mikroprocessor wilt laden. Echter in plaats van op adres \$0300 tot en met \$030F op adres \$0250 tot en met \$025F.

\$0300 00	\$0306 06	\$030C 0C
0301 01	0307 07	030D 0D
0302 02	0308 08	030F 0F
0303 03	0309 09	
0304 04	030A 0A	
0305 05	030B 0B	

Handel nu als volgt.

1. Geef aan de processor op dat een bestaand programma op tape op een ander adres geschreven moet worden,
2. Geef op welk adres hij dan wel moet beginnen.

toets in	display toont	kommentaar
M	0000	
1	0001	
0	0010	
0	0100	
RETURN	0100	
F	XX	
F	FF	
RETURN	0100	
R	FF	
RETURN	0101	
M	0000	
2	0002	
5	0025	
0	0250	
RETURN	0250	
R	XX	
	display uit	Start de band en wacht tot het display oplicht.

Het programma is dan geladen op de adressen \$0250 tot en met \$025F. Kontroleer dit. Om te kijken of het juiste programma is geladen kan op adres \$0101 gekeken worden. Hier moet \$02 staan. Namelijk het ID-nummer van het geladen programma.

ASSEMBLER.

Behalve in machine-code kan een programma ook in MNEMONIC-vorm worden ingevoerd. Dit gebeurt met een zogenaamde "line-assembler", die instructie naam vertaald in de juiste machine-code, rekening houdend met de toegepaste addressing mode. Hierbij dienende operanden als hexadecimale getallen opgegeven te worden, waarbij het dollar teken moet worden weggelaten in verband met de lengte van het display.

De assembler wordt gestart met de toets I te bedienen. Hierna kan het werkadres worden opgegeven met de toets # gevolgt door het gewenste adres en een RETURN.

De gewenste instructie kan nu in MNEMONIC- vorm worden ingevoerd en worden voorzien van de gewenste operand, waarbij de schrijfwijze van de operand bepaald in wat voor addressing mode er gewerkt moet worden. Zie de onderstaande voorbeelden.

- immediate	LDA # 10	- Z-page,X STA 00,X
- absolute	LDA 0300	- Z-page,Y STX 10,Y
- zero page	LDA 01	- Absolute,X STA 0300,X
- accum	ASL A (van ACCU)	- Absolute,Y STA 0400,Y
- implied	BRK	- Indexed indirect (inhoud Z-page adres + X)
- relative	BNE 10 (offsetwaarde)	(IND, X) STA (10,X)
	BNE 0300 (absolute adres)	- Indirect indexed (inhoud van twee op één Z-page adressen + Y)
- indirect	JMP (0400)	(IND),Y STA (20),Y

Tussen de code en operanden moeten geen spaties worden gebruikt daar de assembler daar ook voor zorgt. Na het intoetsen kan op de spatiebalk worden gedrukt waarop voor een $\frac{1}{2}$ seconde de door assembler gemaakte code verschijnt. Wordt er na het intoetsen van de operand op de RETURN gedrukt, dan komt onmiddellijk de volgende vrije regel voor de volgend instructie op het display te staan. Komt er na het indrukken van een toets een ERROR melding op het display te staan dan bestaat of de instructiecode niet of is de operand niet goed ingevoerd. De assembler kan verlaten worden door de ESC (escape) toets te bedienen of de beide RESET-toetsen gelijktijdig in te drukken.

DISASSEMBLER.

Een programma welke reeds is ingevoerd kan weer in MNEMONIC worden uitgelezen met de toets K. Dit geschied door het beginadres op te geven, gevolgt door een RETURN waarna de toets K wordt bediend. De gezochtte instructie verschijnt dan automatisch op het display.

PC-1 MEMORY-MAP	
Adres in hexadeci- malecode	Indeling
\$ 0000	ZERO PAGE (PAGINA NUL)
\$ 00EF	
\$ 00FO	ZERO PAGE VOOR DE MONITOR
\$ 0OFF	
\$ 0100	GEBRUIKT DOOR DE MONITOR
\$ 0138	
\$ 0139	STACK
\$ 01FF	
\$ 0200	RAM VOOR DE GEBRUIKER
\$ 07FF	
\$ 0800	LEEG GEBIED
\$ DFFF	
\$ E000	I/O - BLOK VOOR INTERN GEBRUIK
\$ EOF	
\$ EO10	I/O - BLOK VOOR DE GEBRUIKER
\$ EO1F	
\$ EO20	LEEG GEBIED
\$ EFFF	
\$ FO00	MONITOR IN EPROM
\$ FFFF	

8

In figuur 9 is de memory-map (adres indeling) van de PC - 1 uC weergegeven. De Zeropage en de Ram zijn vrij door de programmeur te gebruiken. Programma's kunnen het beste vanaf \$0200 worden geschreven, terwijl de data van deze programma's in de zeropage kan staan.

De Monitor Ram is het werkgeheugen van de PC - 1 monitor. Er bevinden zich in die geheugenplaatsen een aantal belangrijke gegevens:

\$F0	NMIVEC	non-maskable interrupt
\$F2	INTVEC	vector voor IRQ
\$F6	IRQVEC	vector na een IRQ

PROCESSOR REGISTERS

\$F8, F9	PC	low byte, high byte
\$FA	ACCU	
\$FB	INDEX REGISTER X	
\$FC	INDEX REGISTER Y	
\$FD	STATUS REGISTER	
\$FE	STACK POINTER	

CASSETTE

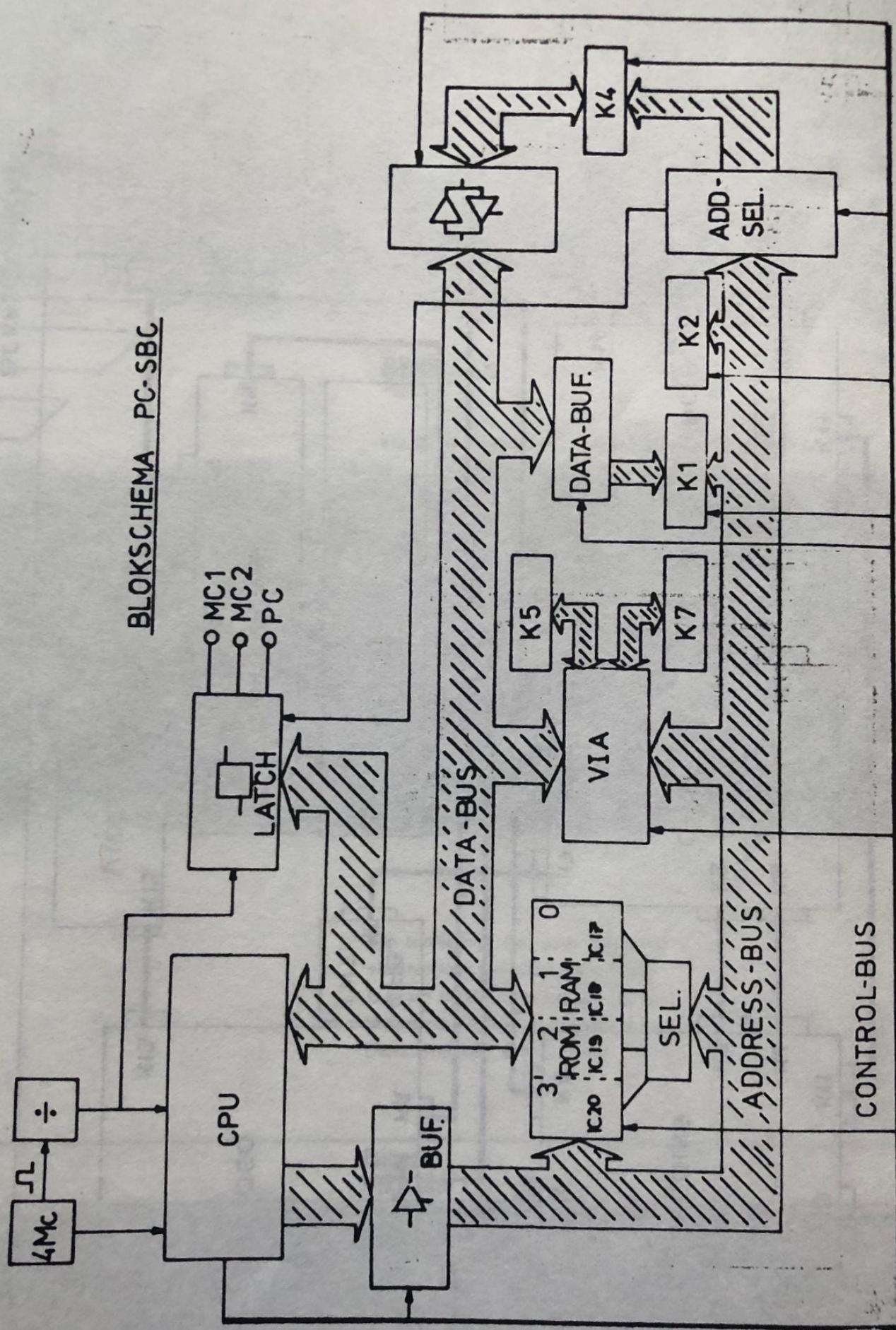
\$0100	IDnr van de cassettefile
\$0101	TAPID IDnr van de laatste file
\$0102,0103	EOL,EOH einde van het programma

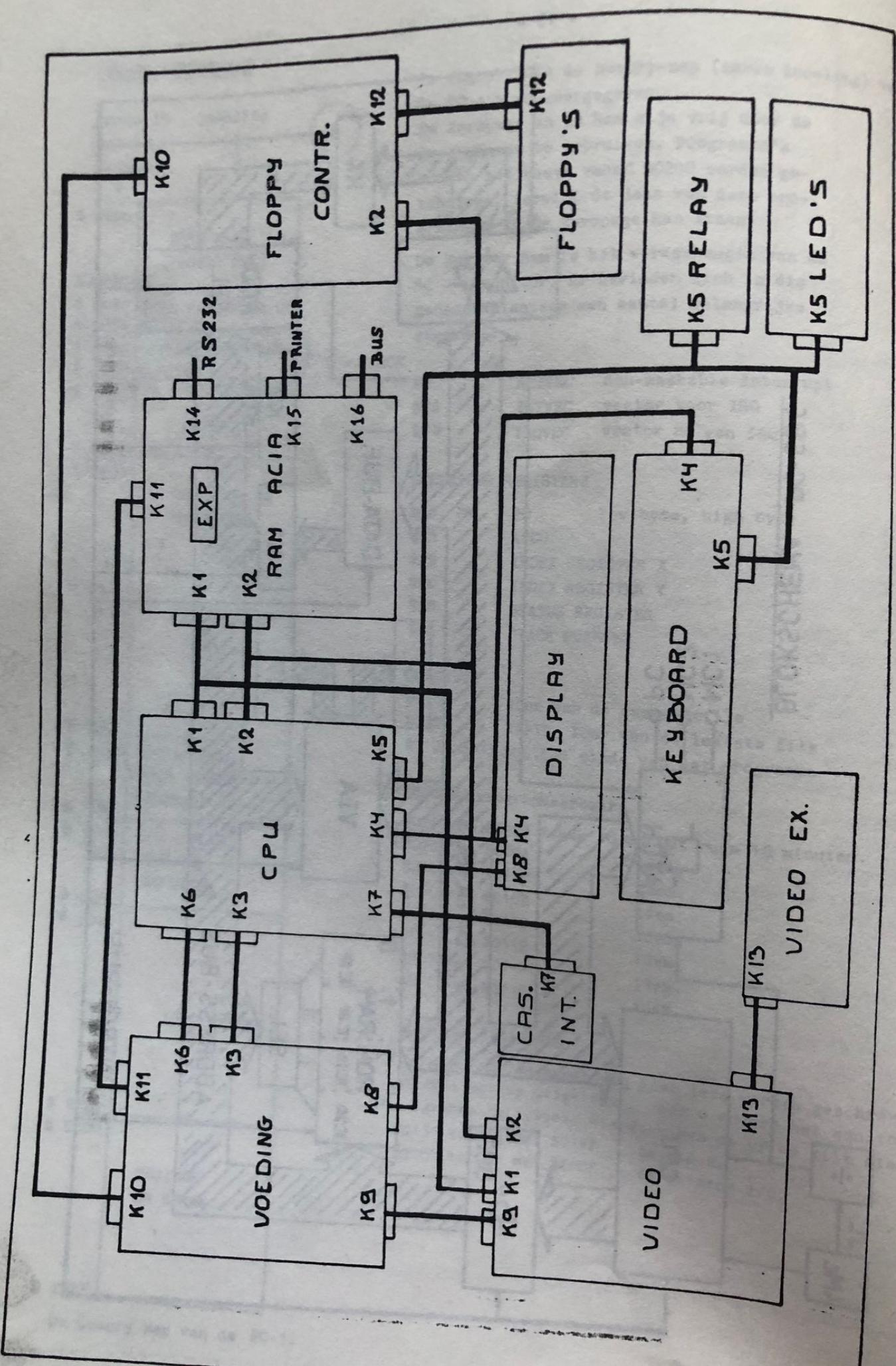
TIMERS (programmeerbaar)

\$0107 en \$0108	timer 10 mS tot ruim 10 minuten.
\$0109 en \$010A	idem.
\$010B en \$010C	idem.
\$010D en \$010E	idem.
\$010F en \$0110	idem.
\$0111 en \$0112	idem.
\$0113 en \$0114	idem.
\$0115 en \$0116	idem.

VERBODEN REGISTERS

In verboden registers mag niet iets worden geschreven of iets worden uitgelezen, daar u anders het monitor programma beïnvloed. Hierdoor kan de uP op tilt slaan. \$0137 tot en met \$01FF de stack. \$E000 tot en met \$EOF de interne I/O.





BEKABELING

28-7-1982

KOMPONENTENLIJST PIM82 SBC

PAGE: 0001

FILE = SBC82# SYSTEEM-DOCUMENTATIE

Halfgeleiders

IC1 = CD4040
IC2 = 74LS367
IC3 = 74LS260
IC4 = 74LS02
IC5 = TBP28L22
IC6 = 74LS245
IC7 = 74LS245
IC8 = 74LS245
IC9 = 74LS374
IC10 = 74LS86
IC11 = 74LS393
IC12 = 74LS04
IC13 = R6502
IC14 = R6522
IC15 = 74LS138
IC16 = 74LS245
IC17 = (P)ROM/RAM 1
IC18 = (P)ROM/RAM 2
IC19 = (P)ROM/RAM 3
IC20 = (P)ROM/RAM 4
IC21 = 7438

Dip-switches

S1 = 2 x maak
S2 = 8 x maak
S3 = 10 x maak
S4 = 10 x maak

Transistor

T 1 = 21133 (klein)

Weerstanden

R1 = 10 K
R2-6 = 3,3 K
R7,8 = 2,2 K
R9 = 1,5 K
R10-19 = 3,3 K

Condensatoren

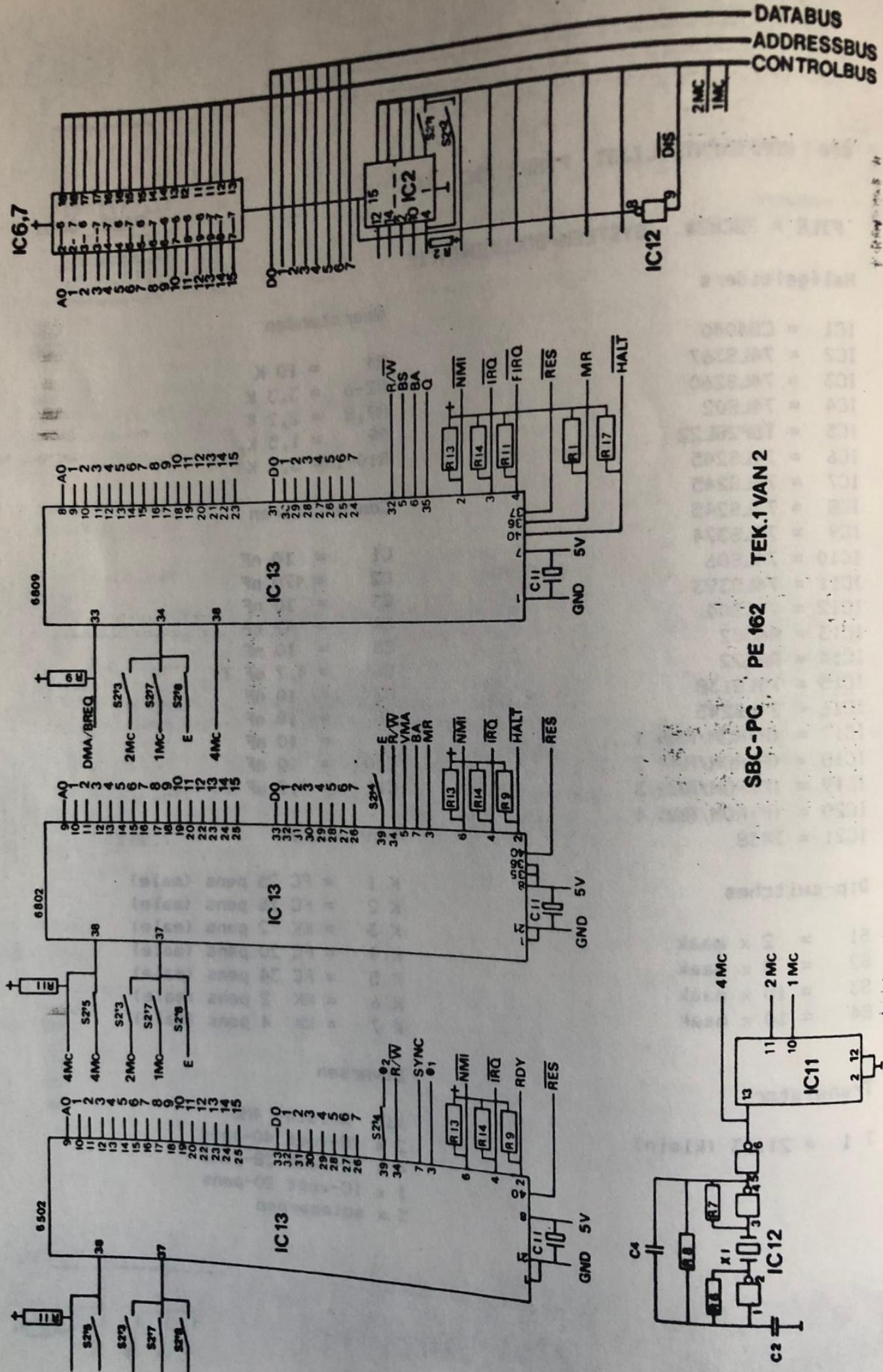
C1 = 10 nF
C2 = 470 pF
C3 = 10 nF
C4 = 47 nF
C5 = 10 nF
C6 = 4,7 uF T
C7 = 10 nF
C8 = 10 nF
C9 = 10 nF
C10 = 10 nF
C11 = 4,7 uF T

Konnektoren

K 1 = FC 26 pens (male)
K 2 = FC 16 pens (male)
K 3 = KK 2 pens (male)
K 4 = FC 20 pens (male)
K 5 = FC 34 pens (male)
K 6 = KK 2 pens (male)
K 7 = KK 4 pens (male)

Diversen

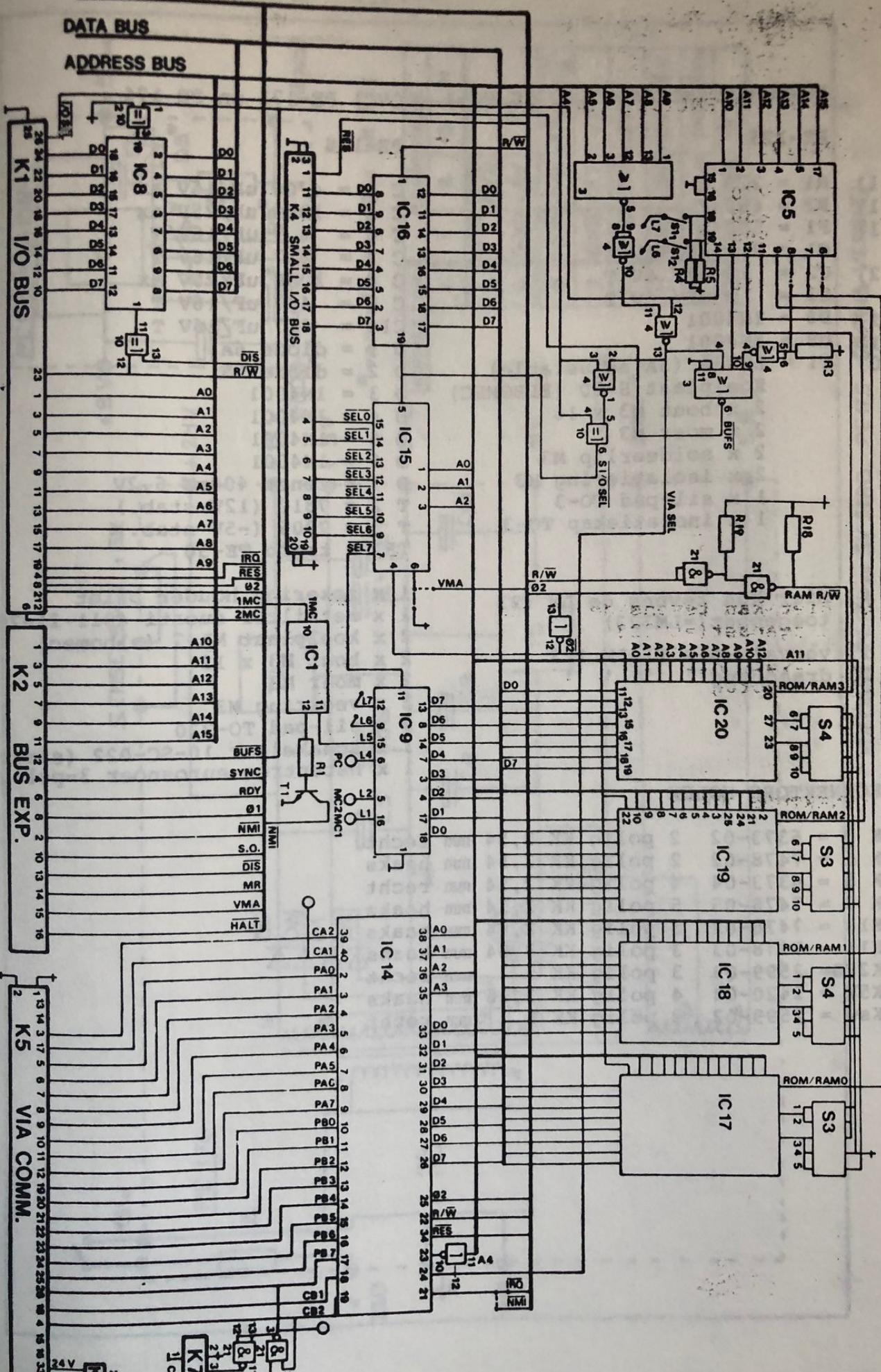
X1 Kristal 4Mc
2 x IC-voet 40-pens
4 x IC-voet 28-pens
1 x IC-voet 20-pens
5 x solddeerpen



CONTROL BUS

DATA BUS

ADDRESS BUS



KOMPONENTENLIJST VOEDING PC-1 (ROVC) PE-175 en PE-176

PE-176

PE-175

- 1) R1 = 270 E
1) R2 = 680 E
1) P1 = 500 E
C1 = 0,1 uF/16V T
2) C2 = 10 uF/ 6V T
C3 = 1 uF/ 6V T
1) D8 = 1N4001
1) D9 = 1N4001
0) T1 = LM350 (3A adjustable)
Koelplaat S-30 (ELBOMECH)
2 x bout M3 x 16
2 x moer M3
2 x soldeerlip M3
2 x isolatiering M3
1 x sil-pad TO-3
1 x isolatiekap TO-3

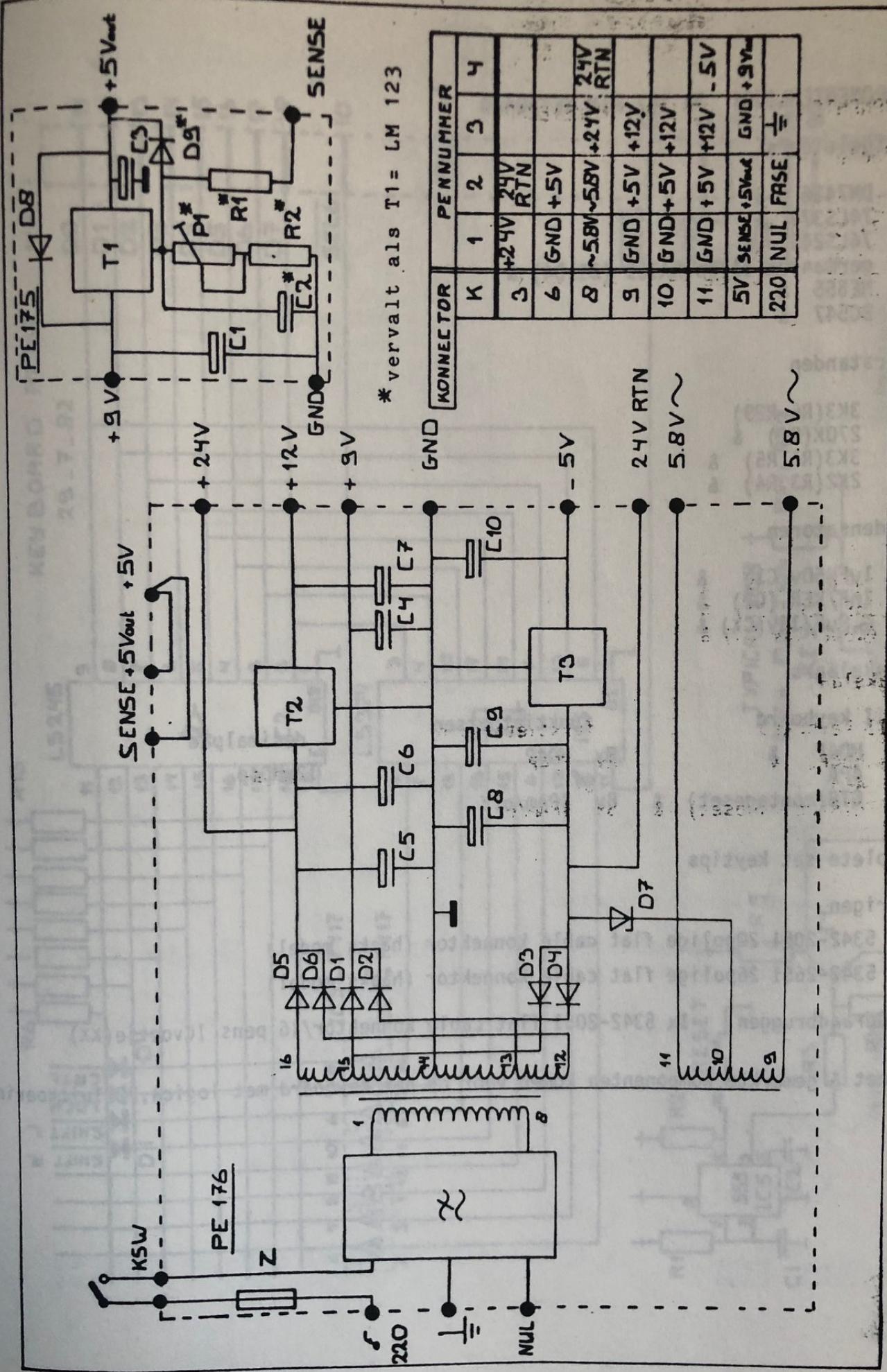
- C 4 = 4700 uF/16V ax
C 5 = 1000 uF/25V ax
C 6 = 4,7 uF/16V T
C 7 = 4,7 uF/16V T
C 8 = 1000 uF/25V ax
C 9 = 4,7 uF/16V T
C10 = 4,7 uF/16V T
D 1 = diode 6A
D 2 = diode 6A
D 3 = 1N4001
D 4 = 1N4001
D 5 = 1N4001
D 6 = 1N4001
D 7 = zener 400mW 6,2V
T 2 = 7812 (12V stab.)
T 3 = 7905 (-5V stab.)
TR1 = trafo TE-50

- 0) Hier kan tevens de LM 123
toegepast (=LM323)
1) vervalt als T1=LM 123
2) draadbrug ' ' ' '

- 1 x zekeringhouder print
1 x netfilter ducati 4011-21-7002
2 x koelplaat ML-7 (elbomec)
2 x bout M3 x 12
2 x moer M3
2 x veerring M3
2 x sil-pad TO-220
1 x schakelaar 10-SC-022 (samar)
1 x netentree eurosnoer 3-polig

KONNEKTORS MOLEX

- K 3 = 6373-02 2 polig KK 2,54 mm recht
K 6 = 7478-02 2 polig KK 2,54 mm haaks
K 8 = 6373-04 4 polig KK 2,54 mm haaks
K 9 = 7478-05 5 polig KK 2,54 mm recht
K10 = 7478-03 3 polig KK 2,54 mm haaks
K11 = 7478-03 3 polig KK 2,54 mm haaks
K220 = 2599-03 3 polig KK 2,54 mm haaks
K5V = 2420-04 4 polig KK 5 mm recht
Ksw = 2599-02 2 polig KK 3,96 mm haaks
 5 mm recht



* vervalt als T1 = LM 123

KONNEKTOR	PENNNUMMER				
	K	1	2	3	4
3	+24V	24V	RTN		
6	GND	+5V			
8	~5.8V	~5.8V	+24V	24V	RTN
9	GND	+5V	+12V		
10	GND	+5V	+12V		
11	GND	+5V	+12V	-5V	
5V	SENSE	+5Vout	GND	+9V	
220	NUL	FASE			

VOEDING PE176 en SV STAB. PE175 (BASIS VERSIE)

2-8-'82

KOMPONENTENLIJST PE 157 PIM KEYBOARD

halfgeleiders

2x DM7416 &
1x 74LS374 &
1x 74LS245 &
4x germanium diode AA119 (D1-D4) &
1x NE555 &
1x BC547 &

weerstanden

24x 3K3(R6-R29)
 1x 270K(R1) &
 2x 3K3(R1,R5) &
 2x 2K2(R3,R4) &

condensatoren

1x 1uF/50v(C1) &
 1x 1nF/KER.(C2) &
 1x 6,8uF/16v(CX) &

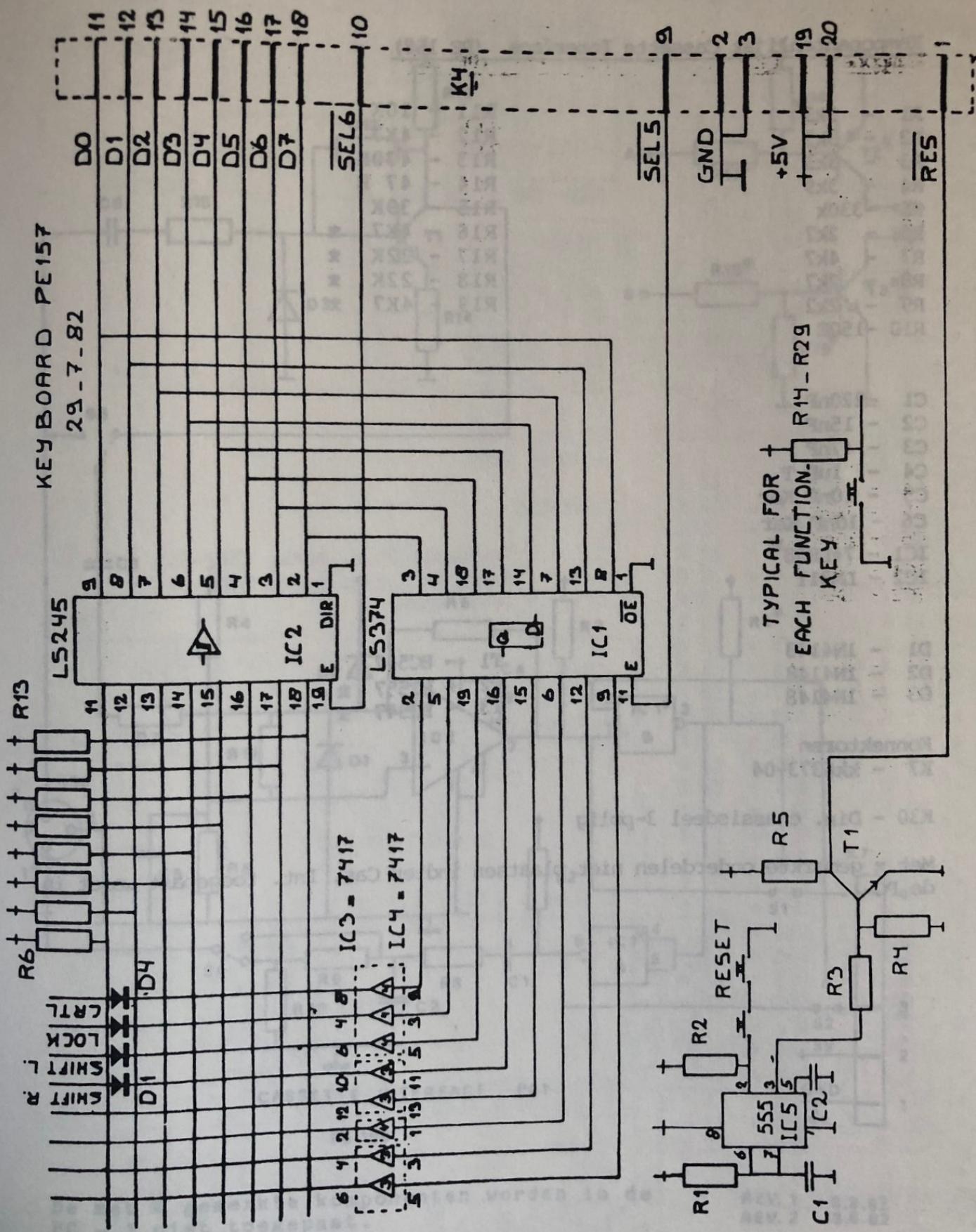
schakelaars

<u>ASCII keyboard</u>	<u>funktietoetsen</u>	<u>decimalpad</u>
62x MD4P &	8x MD4P	
1x 4PA	of	
1x GT8(montageset) &	8x 4PAm/off	12xMD4p

komplete set keytips

overigen

1x 5342-2051 20polige flat cable konnektor (haaks model)
1x 5342-2651 26polige flat cable konnektor (haaks model)
+ 50draadbruggen 1x 5342-2051 flat cable konnektor/16 pens ICvoetje(KX)
de met & gemerkte komponenten komen voor op het keyboard met logica, PC-luitvoering



Komponentenlijst Cassette Interface (PE 168)

R1 - 3k3^x
R2 - 3k3
R3 - 3k3
R4 - 3k9
R5 - 330k
R6 - 2k7
R7 - 4k7
R8 - 2k7
R9 - 2k2
R10 - 150E

R11 - 10K
R12 - 4K7
R13 - 470K
R14 - 47 E
R15 - 39K
R16 - 4K7 ^{*}
R17 - 22K ^{*}
R18 - 22K ^{*}
R19 - 4K7 ^{*}

C1 - 220nF
C2 - 15nF
C3 - 47nF
C4 - 1uF T
C5 - 10nF Ker
C6 - 10nF Ker
IC1 - 74LS38^x
IC2 - LM311

D1 - 1N4148
D2 - 1N4148
D3 - 1N4148

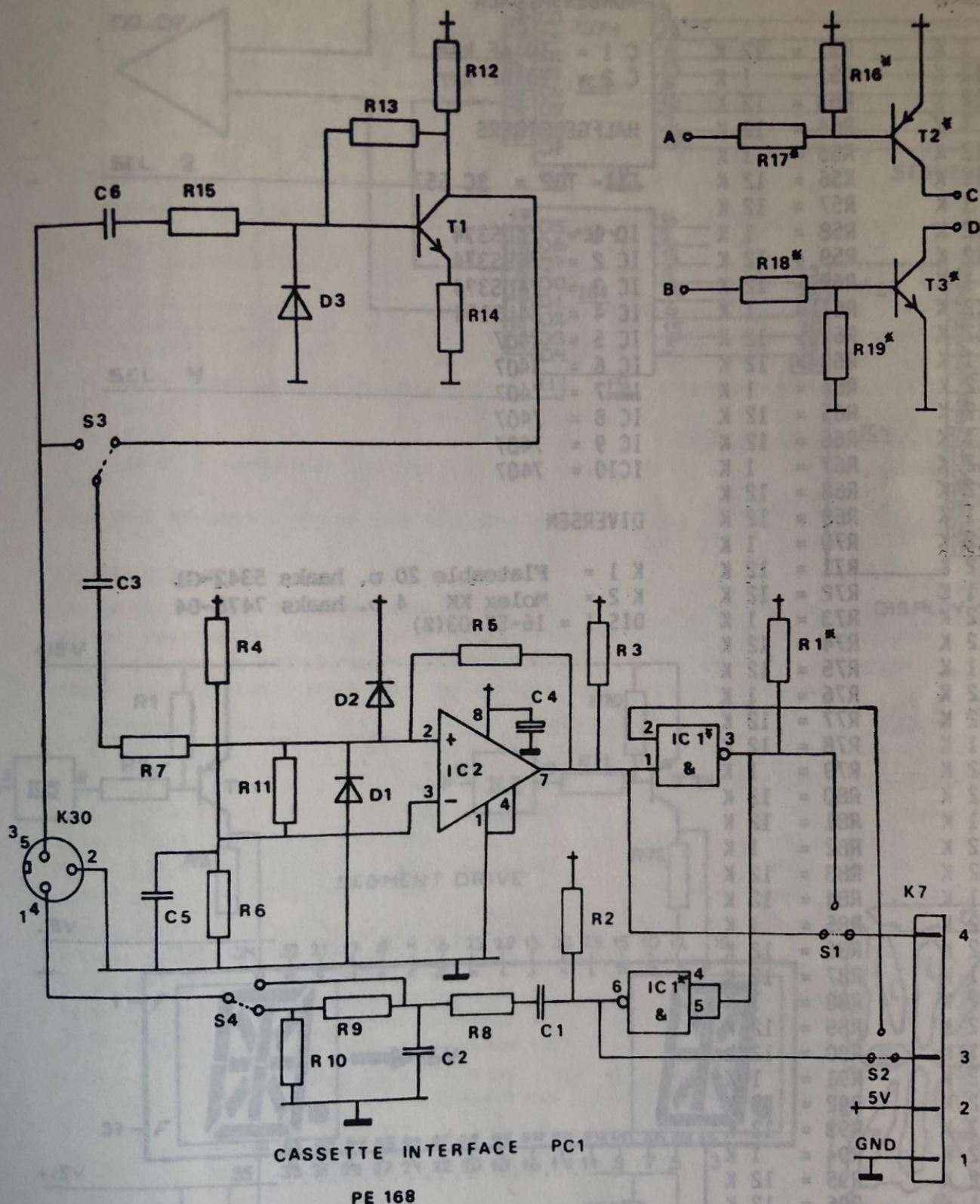
T1 - BC547
T2 - BC557 ^{*}
T3 - BC547 ^{*}

Konnektoren

K7 - kk6373-04

K30 - Din. chassisdeel 3-polig

Met ^{*} gemerkte onderdelen niet plaatsen indien Cas. Int. toegepast wordt in de PC-1.



De met * gemerkte komponenten worden in de
PC - 1 niet toegepast.

REV. 1 6.8.82
REV. 2 13.8.82

KOMPONENTENLIJST DISPLAYPRINT PE 161 16 DIGITS 14 SEGMENTS met punt en komma 21/06/82/1s

WEERSTANDEN

R 1 = 1 K R51 = 12 K
R 2 = 12 K R52 = 1 K
R 3 = 12 K R53 = 12 K
R 4 = 1 K R54 = 12 K
R 5 = 12 K R55 = 1 K
R 6 = 12 K R56 = 12 K
R 7 = 1 K R57 = 12 K
R 8 = 12 K R58 = 1 K
R 9 = 12 K R59 = 12 K
R10 = 1 K R60 = 12 K
R11 = 12 K R61 = 1 K
R12 = 12 K R62 = 12 K
R13 = 1 K R63 = 12 K
R14 = 12 K R64 = 1 K
R15 = 12 K R65 = 12 K
R16 = 1 K R66 = 12 K
R17 = 12 K R67 = 1 K
R18 = 12 K R68 = 12 K
R19 = 1 K R69 = 12 K
R20 = 12 K R70 = 1 K
R21 = 12 K R71 = 12 K
R22 = 1 K R72 = 12 K
R23 = 12 K R73 = 1 K
R24 = 12 K R74 = 12 K
R25 = 1 K R75 = 12 K
R26 = 12 K R76 = 1 K
R27 = 12 K R77 = 12 K
R28 = 1 K R78 = 12 K
R29 = 12 K R79 = 1 K
R30 = 12 K R80 = 12 K
R31 = 1 K R81 = 12 K
R32 = 12 K R82 = 1 K
R33 = 12 K R83 = 12 K
R34 = 1 K R84 = 12 K
R35 = 12 K R85 = 1 K
R36 = 12 K R86 = 12 K
R37 = 1 K R87 = 12 K
R38 = 12 K R88 = 1 K
R39 = 12 K R89 = 12 K
R40 = 1 K R90 = 12 K
R41 = 12 K R91 = 1 K
R42 = 12 K R92 = 12 K
R43 = 1 K R93 = 12 K
R44 = 12 K R94 = 1 K
R45 = 12 K R95 = 12 K
R46 = 1 K R96 = 12 K
R47 = 12 K
R48 = 12 K
R49 = 1 K
R50 = 12 K

KONDENSATOREN

C 1 = 10 nF ker
C 2 = 10 nF ker

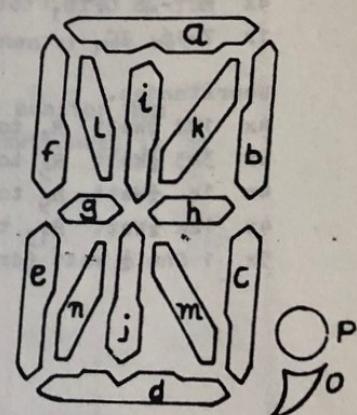
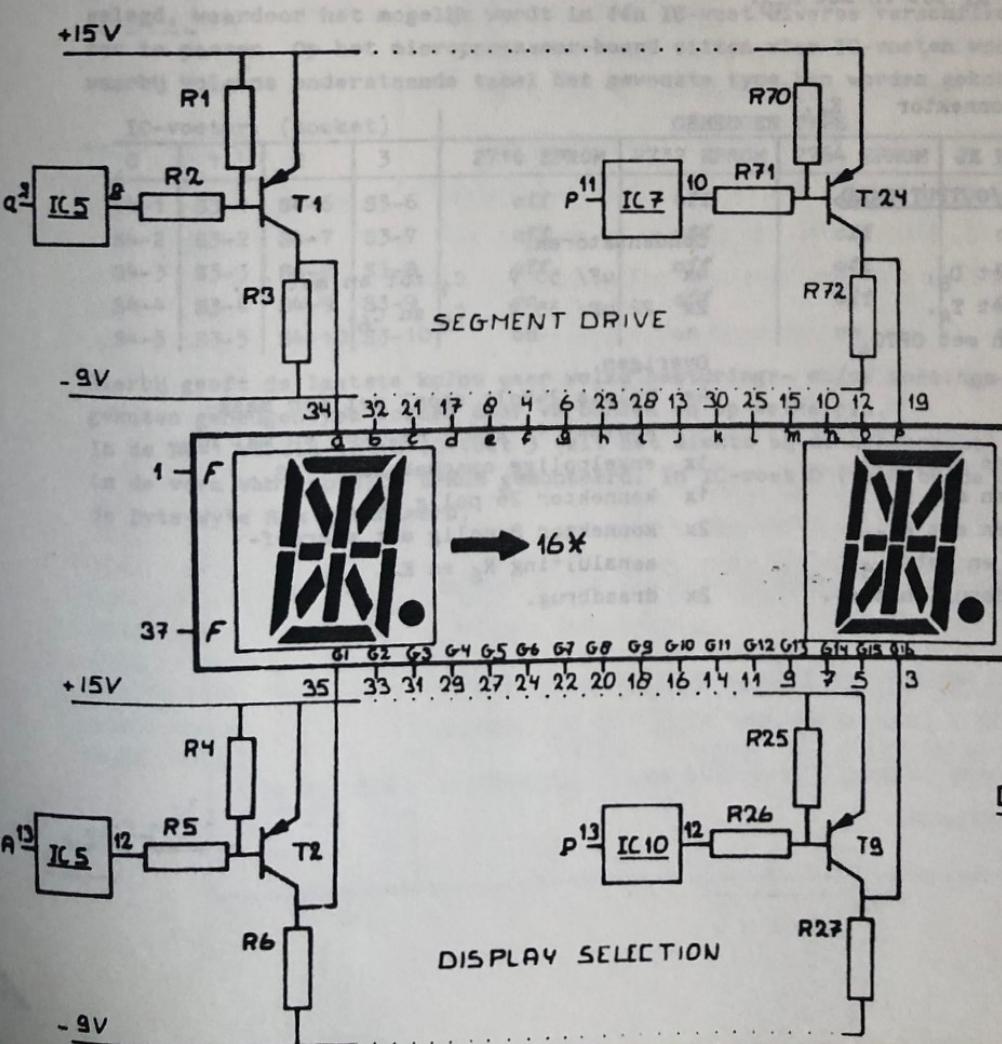
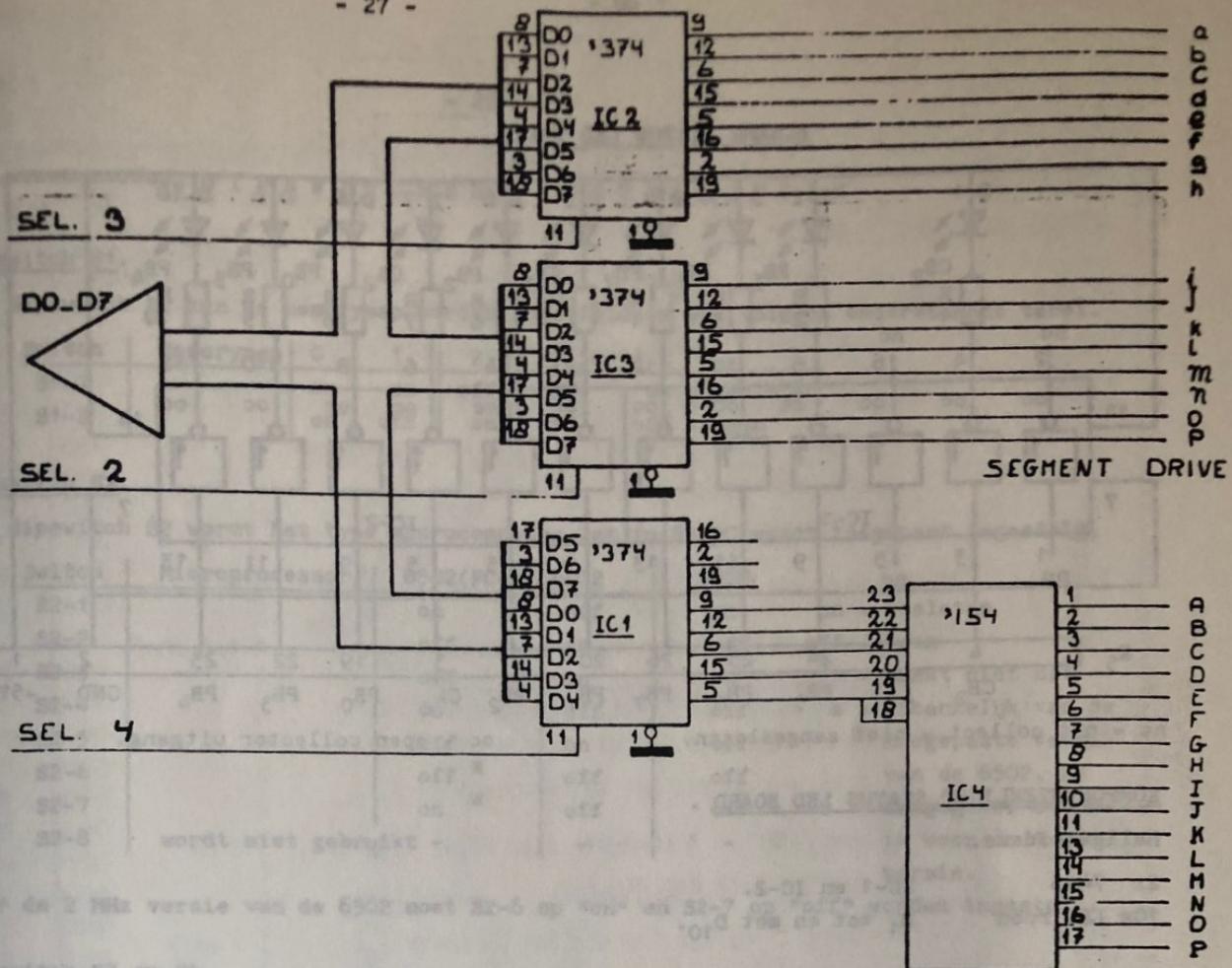
HALFGELEIDERS

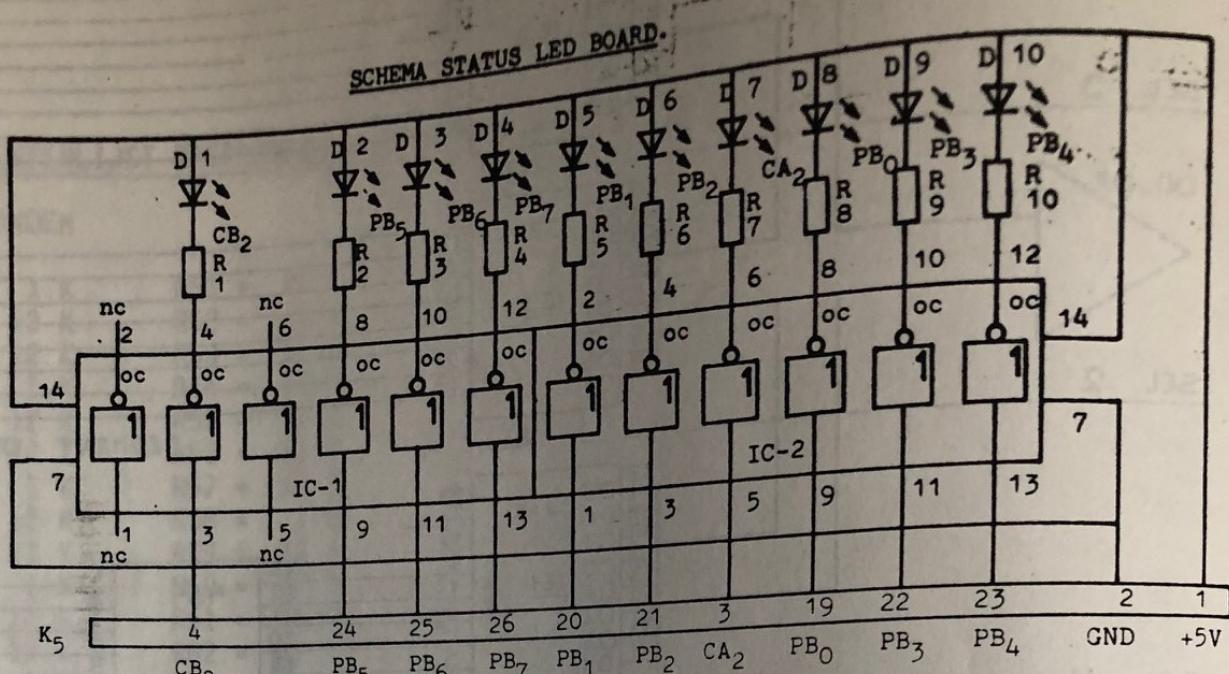
T1 - T32 = BC 557

IC 1 = 74LS374
IC 2 = 74LS374
IC 3 = 74LS374
IC 4 = 74LS154
IC 5 = 7407
IC 6 = 7407
IC 7 = 7407
IC 8 = 7407
IC 9 = 7407
IC10 = 7407

DIVERSEN

K 1 = Flatcable 20 p. haaks 5342-G1
K 2 = Molex KK 4 p. haaks 7478-04
DIS 1 = 16-SY-03(Z)





nc = not collect = niet aangesloten.

oc = open collector uitgang.

KOMPONENTENLIJST STATUS LED BOARD .

Halfgeleiders.

- 2x 7416 IC-1 en IC-2.
 10x LED rood D₁ tot en met D₁₀.

Weerstanden.

- 10x 220 Ohm 1/8 Watt R₁ tot en met R₁₀.

Overigen.

- 26 polige flat cable konnektor K₅.

KOMPONENTENLIJST INPUT/OUTPUTBOARD.

Halfgeleiders.

- 8x 1N4001 D₁ tot en met D₈.
 4x BC557 T₁ tot en met T₄.
 4x MCT-2E OPTO₁ tot en met OPTO₄.
 1x 7416 IC₁.

Weerstanden.

- 4x 1k2 ½Watt R₁ tot en met R₄.
 4x 3k3 ½Watt R₅ tot en met R₈.
 4x 1k ½Watt R₉ tot en met R₁₂.
 4x 12k ½Watt R₁₃ tot en met R₁₆.
 5x 1 Ohm ½ Watt (draadbrugfunktie).

Condensatoren.

- 4x 1 uF/ 50 V C₁ tot en met C₄.
 2x 22 uF/ 35 V C₅ en C₆.

Overigen.

- 4x Relais 24Volt spoel met één maak-
 kontakt 1A 220 V RE₁ tot en met RE₄.
 1x enkelpolige omschakelaar S₁.
 1x konnektor 26 polig K₅.
 2x konnektor 8 polig met schroef-
 aansluiting K₆ en K₇.
 2x draadbrug.

----- D I P S W I T C H E S I N S T E L L I N G E N -----

Dipswitch S1.

Met dipswitch S1 kan de memorymap worden ingesteld en wel volgens onderstaande tabel.

Switch	Memorymap	0	1	2	3(PC-1)
S1-1		on	on	off	off on = gesloten
S1-2		on	off	on	off off = open

Dipswitch S2.

Met dipswitch S2 wordt het type microcomputer wat in de PC wordt toegepast ingesteld.

Switch	Microprocessor	6502(PC-1)	6802	6809	
S2-1		on	off	on	on = gesloten
S2-2		off	on	off	off = open
S2-3		off	on	on	- = maakt niet uit
S2-4		on	off	off	* = afhankelijk van de
S2-5		off	on	off	toegepaste versie
S2-6		off *	off	off	van de 6502. De
S2-7		on *	off	off	aangegeven stand
S2-8	wordt niet gebruikt -	-	-	-	is voor de 1 MHz versie.

Voor de 2 MHz versie van de 6502 moet S2-6 op "on" en S2-7 op "off" worden ingesteld.

Dipswitch S3 en S4.

Met de dipswitches S3 en S4 kunnen bepaalde besturings- en voedingslijnen worden omgelegd, waardoor het mogelijk wordt in één IC-voet diverse verschillende geheugens toe te passen. Op het microprocessor-board zitten vier IC-voeten voor geheugens waarbij volgens onderstaande tabel het gewenste type kan worden gekozen.

IC-voetnr. (socket)				GEHEUGEN TYPE				SOCKET	
0	1	2	3	2716 EPROM	2732 EPROM	2764 EPROM	2K BW-RAM	pin 23	pin 27
S4-1	S3-1	S4-6	S3-6	off	off	on	off		+ 5V
S4-2	S3-2	S4-7	S3-7	off	off	off	off		R/W
S4-3	S3-3	S4-8	S3-8	off	off	off	on	R/W	
S4-4	S3-4	S4-9	S3-9	on	off	off	off	+ 5V	
S4-5	S3-5	S4-10	S3-10	on	on	on	off	A-11	

Hierbij geeft de laatste kolom weer welke besturings- en/of voedings-lijnen er met het gekozen geheugentype worden door verbonden en op welke pin.

In de PC-1 versie is in IC-voet 3 (zit het dichts bij de achterkant) de monitor ROM in de vorm van een 2732 EPROM gemonteerd. In IC-voet 0 (vlak bij de dipswitches) is de Byte-Wyte RAM gemonteerd.

SUBROUTINE	ADRES	FUNKTIE	VERANDERT REGISTERS
GETKEY	\$F003	Wacht tot er een toets is ingedrukt en komt met de ASCII-toetswaarde in A en \$0134 terug.	A en P
SCNKEY	\$F006	Is er geen toets ingedrukt, dan is het carry-bit geset. Is er een toets ingedrukt, dan is het carrybit geset en is de ASCII-toetswaarde in A en \$0134.	A en P
CLRDSP	\$F009	Wist het display. X wordt \$00, A wordt \$20	A,X en P
WBYTE	\$F00C	Byte in A komt op positie X en X+1 op het display. X wordt X+2.	A,X en P
RBYTE	\$F00F	Leest het byte op het display met positie X en X+1 en zet het hexresultaat in A.	A,X en P
WCHAR	\$F012	Hexcijfer in A (laagste vier bits) komt op positie X op het display.	X en P
RCHAR	\$F015	Leest cijfer van display op positie X. De hexwaarde komt in A.	A en P
TPRINT	\$F018	Text in tabel komt op het display na uitvoering van CLRDSP. De text moet eindigen met \$00. Het beginadres van de texttabel staat in Y (high-byte) en A (low-byte).	A,Y en P
BLKDIS	\$F01B	De NMI die elke 512 mS komt wordt uitgeschakeld. A wordt \$10. De display is dan uit en de timers worden niet meer bijgehouden.	A en P
RELDIS	\$F01E	De NMI wordt ingeschakeld. Display en timers funktionieren weer. A wordt \$00.	A en P
DUMP	\$F021	Zet het geheugen op cassette van het adres ADL (\$F8),ADH (\$F9) tot EAL (\$0102),EAH (\$0103) met ID-nummer van \$0100.	A,X,Y en P
LOAD	\$F024	Laad geheugen vanaf de cassette met ID-nummer op adres \$0100.	A,X,Y en P
OUTALL	\$F027	Zet karakter in A op eerstvolgende vrije plaats (aangewezen door \$0137) op het display.	A,X,Y en P
MONITR	\$F000	Ingang in monitorprogramma vanuit gebruikersprogramma.	P A,X,Y,P en S

PASS 1

PASS 2

***** PC-1 HEX-MONITOR ***** PROTON 650X ASSEMBLER V4.4 PAGE: 0001

0001 0000
0002 0000
0003 0000
0004 0000
0005 0000
0006 0000
0007 0000
0008 0000
0009 0000
0010 0000
0011 0000
0012 0000
0013 0000
0014 0000
0015 0000
0016 0000
0017 0000
0018 0000
0019 0000
0020 0000



KURSUS
INDUSTRIËLE MICROPROCESSORS

0021 0000 : 00000000 000000 00
0022 0000 : 00 00 00 00000000 00000000
0023 0000 : 00 00 00 00000000 00000000
0024 0000 : 00000000 00 00000000 00000000 00
0025 0000 : 00 00 00 00000000 00000000 00
0026 0000 : 00 00 00 00000000 00000000 00
0027 0000 : 00 000000 00000000 00000000 00000000
0028 0000 :
0029 0000 :
0030 0000 :
0031 0000 :
0032 0000 :
0033 0000 :
0034 0000 :
0035 0000 :
0036 0000 :
0037 0000 :
0038 0000 : PROTON behoudt zich het recht voor, zonder
0039 0000 : voorafkondiging, wijzigingen die de functie
0040 0000 : dan wel het gebruik van de PC-1 veranderen
0041 0000 : aan te brengen. PROTON is niet aansprakelijk
0042 0000 : voor gevolgen, voortkomende uit de toepassing
0043 0000 : van de PC-1 en haar programmatuur, noch kunnen
0044 0000 : aan deze listing rechten worden ontleend voor
0045 0000 : eigen toepassing.
0046 0000 : Copyright PROTON ELECTRONICS Naarden 1983
0047 0000 :
0048 0000 : .TITLE * ***** PC-1 HEX-MONITOR *****
0049 0000 :

0050 0000	;			
0051 0000	;	REVISION DD: 24-09-82	SK	FILE=PC21\$
0052 0000	;	REVISION DD: 27-06-83	SK	FILE=PC23\$
0053 0000	;	REVISION DD: 14-07-83	SK	FILE=PC24\$; INSTANT ASM/X
0054 0000	;			
0055 0000	;	\$		
0056 0000	;	\$		
0057 0000	;	\$		
0058 0000	;	\$\$\$\$		\$\$\$\$\$
0059 0000	;	\$\$\$\$ PC-1 HEXADECIMAL MONITOR	\$\$\$\$	\$\$\$\$\$
0060 0000	;	\$\$\$\$		\$\$\$\$\$
0061 0000	;	\$\$\$\$ PROTON ELECTRONICS	\$\$\$\$	\$\$\$\$\$
0062 0000	;	\$\$\$\$ ENERGIESTRAAT 36	\$\$\$\$	\$\$\$\$\$
0063 0000	;	\$\$\$\$ 1411 AT NAARDEN	\$\$\$\$	\$\$\$\$\$
0064 0000	;	\$\$\$\$ TEL: 02159-48224	\$\$\$\$	\$\$\$\$\$
0065 0000	;	\$\$\$\$		\$\$\$\$\$
0066 0000	;	\$		
0067 0000	;	\$		
0068 0000	;			
0069 0000	;	KEYBOARD HAS INVERTED DRIVERS	25-08-1982	
0070 0000	;			

*** COMMAND DESCRIPTION ***

0072 0000	;			
0073 0000	;	DISPLAY NEXT/PREVIOUS LOCATION,ENTER ADDRESS		
0074 0000	;	GO TO USER'S PROGRAM		
0075 0000	;	INPUT ADDRESS		
0076 0000	;	INSERT 1 BYTE INTO TO PROGRAM		
0077 0000	;	DELETE 1 BYTE FROM THE PROGRAM		
0078 0000	;	CALCULATE BRANCH OFFSET		
0079 0000	;	SINGLE STEP INSTRUCTION(S)		
0080 0000	;	LOAD CODE FROM AUDIO-CASSETTE		
0081 0000	;	DUMP CODE TO CASSETTE		
0082 0000	;			
0083 0000	;	INSTANT ASSEMBLER & DISASSEMBLER (14-07-83 V2.X)		
0084 0000	;			
0085 0000	;	THE MONITOR DRIVES A 16 CHARACTER DISPLAY,		
0086 0000	;	HANDLES INTERRUPTS, BREAKPOINTS AND INCORPORATES		
0087 0000	;	EIGHT 10-MILLISECOND TIMERS.		
0088 0000	;			
0089 0000	;			

*** SYSTEM ADDRESSES ***

0091 0000	;			
0092 0000	;			
0093 0000	;	ROMBAS: =\$F000		
0094 0000	;	ROMSIZ: =\$1000		START OF PROGRAM
0095 0000	;	I0SEL: =\$E000		: 4 K (E)PROM
		MONRAM: =\$F0		: I/O SELECT

0096 0000

*** VIA ADDRESSING ***

0098 0000	;	
0099 0000		\$=IDSEL+\$10
0100 E010	;	
0101 E010	;VIA #1	
0102 E010	DRB1: \$=\$+1	
0103 E011	DRA1: \$=\$+1	
0104 E012	DDRBI1: \$=\$+1	
0105 E013	DDRA1: \$=\$+1	
0106 E014	T1CL1: \$=\$+1	
0107 E015	T1CH1: \$=\$+1	
0108 E016	T1LL1: \$=\$+1	
0109 E017	T1LH1: \$=\$+1	
0110 E018	T2LL1: \$=\$+1	
0111 E019	T2CH1: \$=\$+1	
0112 E01A	SR1: \$=\$+1	
0113 E01B	ACR1: \$=\$+1	
0114 E01C	PCR1: \$=\$+1	
0115 E01D	IFRI: \$=\$+1	
0116 E01E	IER1: \$=\$+1	
0117 E01F	APORT1: \$=\$+1	
0118 E020	;	
0119 E020	;	

*** ASSEMBLER CONSTANTS ***

0121 E020	ENTER	=\$0D	
0122 E020	ESC	=\$1B	
0123 E020	;		
0124 E020	; COMMAND-KEY VALUE'S		
0125 E020	MEMCMD:	=\$0C	: FORM,SPACE
0126 E020	MINCMD:	=\$08	: BACK,SPACE
0127 E020	ADR CMD:	=\$4D	: 'M' = MEMORY SHOW/ALTER
0128 E020	SSTCMD:	=\$53	: 'S' = SINGLE-STEP A PROGRAM
0129 E020	GOCMD:	=\$47	: 'G' = GO TO USER PROGRAM
0130 E020	LOCMD:	=\$52	: 'R' = READ FROM CASSETTE
0131 E020	DMPCMD:	=\$57	: 'W' = WRITE TO CASSETTE
0132 E020	OFFCMD:	=\$4F	: 'O' = CALCULATE BRANCH-OFFSET
0133 E020	DISCMD:	=\$4B	: 'K' = DISASSEMBLE
0134 E020	ASSCMD:	=\$49	: 'I' = ASSEMBLE
0135 E020	REGCMD:	=\$3F	: '?' = SHOW REGISTERS
0136 E020	INSCMD:	=\$11	: 'CHAR INSERT' = INSERT A BYTE IN MEMORY
0137 E020	DELCMD:	=\$12	: 'CHAR DELETE' = REMOVE A BYTE FROM MEMORY
0138 E020	TSTCMD:	=\$14	: CTRL-T = KEYBOARD --> DISPLAY ECHO TEST
0139 E020	;		
0140 E020	; HARDWARE CONSTANTS		
0141 E020	;		

			; CONTROL-LATCH
0142	E020	NMICTR	=IOSSEL+7
0143	E020	;	
0144	E020	CHSEL	=IOSSEL+4
0145	E020	SEGMO	=IOSSEL+3
0146	E020	SEGML	=IOSSEL+2
0147	E020	;	
0148	E020	INKB	=IOSSEL+6
0149	E020	OUTKB	=IOSSEL+5
0150	E020	;	
0151	E020	;	\$=MONRAM
0152	00F0	NMIVEC	\$=\$+2
0153	00F2	INTVEC	\$=\$+2
0154	00F4	TEMP	\$=\$+2
0155	00F6	IRQVEC	\$=\$+2
0156	00F8	ADL	\$=\$+1
0157	00F9	ADH	\$=\$+1
0158	00FA	AC	\$=\$+1
0159	00FB	XR	\$=\$+1
0160	00FC	YR	\$=\$+1
0161	00FD	PS	\$=\$+1
0162	00FE	SPTR	\$=\$+1
0163	00FF	PRVCMD	\$=\$+1
0164	0100	;	
0165	0100	;	\$=\$\$ FROM HERE IN PAGE 1 \$=\$\$
0166	0100	;	
0167	0100	;	\$=\$100
0168	0100	ID	\$=\$+1
0169	0101	TAPID	\$=\$+1
0170	0102	EAL	\$=\$+1
0171	0103	EAH	\$=\$+1
0172	0104	DIV1	\$=\$+1
0173	0105	CURCMD	\$=\$+1
0174	0106	GKX:	\$=\$+1
0175	0107	TIMER	\$=\$+16
0176	0117	DBCNTR	\$=\$+2
0177	0119	ASAW	\$=\$+1
0178	011A	XSAW	\$=\$+1
0179	011B	YSAW	\$=\$+1
0180	011C	INDADR	
0181	011C	INDADL	\$=\$+1
0182	011D	INDADH	\$=\$+1
0183	011E	CHKSML	\$=\$+1
0184	011F	CHKSMH	\$=\$+1
0185	0120	;	
0186	0120	DIBUFL	=16
0187	0120	DIBUFF	\$=\$+\$+DIBUFL
0188	0130	DCPTR	\$=\$+1
0189	0131	DMASK	\$=\$+1
0190	0132	IMASK	\$=\$+1
0191	0133	KEYNR	\$=\$+1
0192	0134	LSTKEY	\$=\$+1
0193	0135	CPATRN	\$=\$+1

; DISPLAYBUFFER SIZE
 ; DISPLAY REFRESH BUFFER
 ; DISPLAY POINTER
 ; MASK FOR KEYBOARD-OUTPUT
 ; MASK FOR INPUT-DECODE
 ; LOGICAL NUMBER OF THE KEY
 ; ASCII VALUE OF CURRENT & LAST KEY
 ; PATTERN FOR CONTROL-KEYS

0194	0136	REPTO	\$=#+1	
0195	0137	CURPOS	\$=#+1	; REPEAT-TIME COUNTER
0196	0138	DXSAV	\$=#+1	; DISPLAY CURSOR POSITION
0197	0139		;	; X-SAV FOR OUTALL
0198	0000		.EX1	
0199	0000		;	
0200	0000		;	
0201	0000		\$=ROMBAS	
0202	F000		;	
0203	F000		;	VECTOR-LIST FOR USER-ENTRIES
0204	F000		;	
0205	F000	4C78F0	JMP MONITR	; RE-ENTER THE MONITOR
0206	F003	4C95F2	JMP GETKEY	; WAIT FOR A KEY
0207	F006	4CC3F2	JMP SCNKEY	; TEST FOR A KEY & RETURN
0208	F009	4CFBF4	JMP CLRDSP	; CLEAR DISPLAY
0209	F00C	4CE5F4	JMP WBYTE	; WRITE A BYTE (INDEXED BY .X)
0210	F00F	4CD4F4	JMP RBYTE	; READ A BYTE
0211	F012	4C07F5	JMP WCHAR	; WRITE A HEX-CHR (INDEXED BY .X)
0212	F015	4C49F5	JMP RCHAR	; READ A HEX-CHR
0213	F018	4C73F5	JMP TPRINT	; PRINT A TEXT
0214	F01B	4C57F6	JMP BLKDIS	; BLOCK DISPLAY INTERRUPTS
0215	F01E	4C62F6	JMP RELDIS	; RELEASE DISPLAY INTERRUPTS
0216	F021	4CE8F6	JMP DUMP	; DUMP MEMORY TO TAPE
0217	F024	4CE6F7	JMP LOAD	; READ FROM TAPE TO MEMORY
0218	F027	4C1CF5	JMP OUTALL	; OUTPUT A ASCII CHAR TO THE DISPLAY
0219	F02A		;	
0220	F02A		;	
0221	F02A		;	***** RESET-ENTRY *****
0222	F02A		;	
0223	F02A		;	INITIALISE WORK-SPACE IF COLD RESET
0224	F02A		;	
0225	F02A	A2FF	RESET: LDX #\$FF	; INITIAL STACK VALUE
0226	F02C	9A	TXS	
0227	F02D	78	SEI	; AND BLOCK INTERRUPTS
0228	F02E	2057F6	JSR BLKDIS	; BLOCK DISPLAY-INTERRUPTS
0229	F031	B6FE	STX SPTR	; USER STACK
0230	F033	D8	CLD	; BINairy MODE
0231	F034		;	TEST FOR WARM RESTART
0232	F034	A5F0	LDA NMIVEC	
0233	F036	CDDFF0	CMP VECTAB	
0234	F039	D007	BNE RS3A	
0235	F03B	A5F1	LDA NMIVEC+1	
0236	F03D	CDE0F0	CMP VECTAB+1	
0237	F040	F02C	BED WARM	; WARM RESTART
0238	F042		;	COLD START
0239	F042	A000	RS3A LDY #VECLEN	
0240	F044	B9DFF0	RS1 LDA VECTAB,Y	
0241	F047	99F000	STA MONRAM,Y	
0242	F04A	BB	DEY	
0243	F04B	10F7	BPL RS1	
0244	F04D	A900	LDA #0	
0245	F04F	A205	LDX #TIMER+1-EAH	

```

0246 F051 9D0301 RS2 STA EAH,X
0247 F054 CA DEX
0248 F055 10FA BPL RS2
0249 F057 A9FF LDA #6FF
0250 F059 8D0201 STA EAL
0251 F05C A906 LDA #106
0252 F05E 8D0301 STA EAH
0253 F061 A901 LDA #1
0254 F063 8D0001 STA ID ; DEFAULT = 1
0255 F066 8D0401 STA DIVI
0256 F069 58 CLI ; ENABLE INTERRUPTS
0257 F06A 0B PHP
0258 F06B 68 PLA
0259 F06C 85FD STA PS ; SETUP A VALID PROGRAM-STATUS
0260 F06E ; MARM LDY #>HDRMSG ; PRINT HEADER-MESSAGE
0261 F06E A0F0 MARM LDA #KHDRMSG
0262 F070 A9ED JSR TPRINT
0263 F072 2073F5 JSR VIAINI ; RELEASE PB7 BY C82=LOW
0264 F075 2015F7
0265 F078 ;
```

*** MONITOR COMMAND LOOP ***

```

0267 F078 ;
0268 F078 ; GET A COMMAND FROM THE KEYBOARD AND
0269 F078 ; START THE APPROPRIATE COMMAND FUNCTION ROUTINE.
0270 F078 ;
0271 F078 A6FE MONITR LDY SPTR
0272 F07A 9A TXS
0273 F07B 2062F6 JSR RELDIS ; RELEASE DISPLAY
0274 F07E D8 CLD
0275 F07F 2067F2 JSR GETUKY ; GET UPPER-CASE KEY
0276 F082 ; JUMPER LDY SPTR ; SET SYSTEM STACK
0277 F082 A6FE JUMPER LDY SPTR ; SET SYSTEM STACK
0278 F084 9A TXS
0279 F085 8D0501 STA CURCMD
0280 F088 A20D LDY #MONMDS
0281 F08A JP1
0282 F08A DDB5F0 CMP CMDTAB,X
0283 F08D F00E BEQ JP2
0284 F08F CA DEX
0285 F090 10FB BPL JP1
0286 F092 A93F LDA #'?'
0287 F094 8D0501 STA CURCMD
0288 F097 20A8F1 JSR NCMD1 ; ILLEGAL COMMAND
0289 F09A 4C7BF0 JMP MONITR
0290 F09D ; JP2
0291 F09D 8A JP2
0292 F09E 0A TXA
0293 F09F AA ASL A
0294 F0A0 80C3F0 TAX
                           LDA JMPTAB,X
```

```

0295 F0A3 BD1C01 STA INDADR
0296 F0A6 BDC4F0 LDA JMPTAB+1,X
0297 F0A7 BD1D01 STA INDADH
0298 F0AC 20B2F0 JSR CMD
0299 F0AF 4C78F0 JMP MONITR
0300 F0B2 :
0301 F0B2 6C1C01 CMD JMP (INDADR)
0302 F0B5 :
0303 F0B5 : THIS TABLE CONTAINS THE COMMAND KEY-VALUES
0304 F0B5 : {THE COMMAND-CHAR CODES}.
0305 F0B5 :
0306 F0B5 CMDTAB
0307 F0B5 0C .BYT MEMCMD,MINCMD,ADRCMD,SSTCMD
0308 F0B9 47 .BYT GOCMD,INSCMD,DELCMD,LODCMD
0309 F0BD 57 .BYT DMPCMD,OFFCMD,TSTCMD,DISCMD
0310 F0C1 49 .BYT ASSCMD,REGCMD
0311 F0C3 :
0312 F0C3 NCMNDS=$-CMDTAB-1
0313 F0C3 :
0314 F0C3 : THIS TABLE CONTAINS THE ADDRESSES OF THE
0315 F0C3 : COMMAND FUNCTION ROUTINES.
0316 F0C3 : THE ROUTINES MUST RETURN WITH 'RTS'
0317 F0C3 :
0318 F0C3 25F1 JMPTAB .WDR PMEM,MMEM,INPADR,SSUSER
0319 F0CB 3CF2 .WDR GOUSER,INSDEL,INSDEL,LOAD
0320 F0D3 E8F6 .WDR DUMP,OFFSET,TESTER,DISA
0321 F0DB 96FB .WDR MNEENT,RSHOW
0322 F0DF :
0323 F0DF :
0324 F0DF : DATA TO SETUP THE MONITOR WORKSPACE
0325 F0DF :
0326 F0DF E3F5 VECTAB .WDR TERMINL,IRHDLL,NSUP,0
0327 F0E7 0002 .WDR $0200 ; PROGRAM COUNTER
0328 F0E9 00 .BYT 0,0,0,0 ; REGISTERS
0329 F0ED VECLEN =$-VECTAB-1
0330 F0ED :
0331 F0ED 5052 HDRMSG .BYT 'PROTON PC-1 V2.0',0
0332 F0FE :
0333 F0FE :

```

*** ENTRY FROM SINGLE-STEP ***

```

0335 F0FE :
0336 F0FE : EXIT THE USERPROGRAM BY AN 'IRQ' AND
0337 F0FE : ENTER THE MONITOR.
0338 F0FE :
0339 F0FE 85FA SSTINT: STA AC
0340 F100 86FB STX XR
0341 F102 84FC STY YR
0342 F104 68 PLA ; STATUS
0343 F105 85FD STA PS

```

1 PROGRAM COUNTER

0344	F107	68	PLA	
0345	F108	85F8	STA ADL	
0346	F10A	68	PLA	
0347	F10B	85F9	STA ADH	
0348	F10D	BA	TSX	; RP
0349	F10E	85FE	STX S PTR	
0350	F110	AD18E0	LDA T2LL1	; CLEAR SST-INT.
0351	F113	58	CLI	
0352	F114	D8	CLO	
0353	F115	2075F1	JSR RSHOW	; SHOW REGISTERSET
0354	F118	2087F2	JSR GETIUKY	; READ A NEW COMMAND
0355	F11B	C953	CMP #SSTCMD	
0356	F11D	D003	BNE \$+5	
0357	F11F	4C12F2	JMP SSUSER	; GO STEP AGAIN
0358	F122	4C82F0	JMP JUMPER	
0359	F125			;
0360	F125			; *** DIRECT TO +MEM & -MEM ***
0361	F125			;
0362	F125	A93E	P MEM	LDA \$'>
0363	F127	8D0501		STA CURCMD
0364	F12A	D005		BNE SEQADR
0365	F12C			;
0366	F12C	A93C	H MEM	LDA \$'<
0367	F12E	8D0501		STA CURCMD
0368	F131			;
0369	F131			; NOW SHOW NEXT LOCATION
0370	F131			;

*** INPUT DATA ON SEQ. ADDRESSES ***

0372	F131			;
0373	F131			; READ THE ADDRESS FROM THE DISPLAY
0374	F131			; AND WRITE THE MEMORYCONTENTS INTO
0375	F131			THE DATA-FIELD.
0376	F131			;
0377	F131	20A5F1	SEQADR	JSR WCOMND
0378	F134	A200		LDX #0
0379	F136	ASF9		LDA ADH
0380	F138	20E5F4		JSR WBYTE
0381	F13B	A5F8		LDA ADL
0382	F13D	20E5F4		JSR WBYTE
0383	F140	A200	S0A1	LDX #0
0384	F142	20D4F4		JSR RBYTE
0385	F145	85F9		STA ADH
0386	F147	20D4F4		JSR RBYTE
0387	F14A	85FB		STA ADL
0388	F14C	A200		LDX #0
0389	F14E	A1FB		LDA (ADL,X)
0390	F150	A206		LDX #5
0391	F152	20E5F4		JSR WBYTE
0392	F155	20B7F4		JSR GETBYT

0393	F158			
0394	F158	AE0501	LDX CURCMD	; CHECK FOR +MEM/-MEM
0395	F15B	E03C	CPX #'<	
0396	F15D	F009	BEQ SQA2	
0397	F15F	E6FB	INC ADL	
0398	F161	DOCE	BNE SEQADR	; NEXT LOCATION
0399	F163	E6F9	INC ADH	
0400	F165	4C31F1	JMP SEQADR	
0401	F168	C6F8	SQA2	DEC ADL ; PREVIOUS LOCATION
0402	F16A	A5F8	LDA ADL	
0403	F16C	C9FF	CMP #\$FF	
0404	F16E	DOC1	BNE SEQADR	
0405	F170	C6F9	DEC ADH	
0406	F172	4C31F1	JMP SEQADR	
0407	F175			
0408	F175			
0409	F175		: ##### SHOW REGISTER-CONTENTS #####	
0410	F175			
0411	F175		: DISPLAYFORMAT: PPPP AA XX YY PS	
0412	F175		: PPPP = PROGRAM COUNTER	
0413	F175		: AA, XX, YY = A X Y REGISTERS	
0414	F175		: PS = PROGRAM STATUS	
0415	F175			
0416	F175	20FBF4	RSHOW	JSR CLRDSP
0417	F178	A200		LDX #0
0418	F17A	A5F9		LDA ADH
0419	F17C	20E5F4		JSR WBYTE
0420	F17F	A5FB		LDA ADL
0421	F181	20E5F4		JSR WBYTE
0422	F184	E8		INX
0423	F185	A5FA		LDA AC
0424	F187	20E5F4		JSR WBYTE
0425	F18A	E8		INX
0426	F18B	A5FB		LDA XR
0427	F18D	20E5F4		JSR WBYTE
0428	F190	E8		INX
0429	F191	A5FC		LDA YR
0430	F193	20E5F4		JSR WBYTE
0431	F196	E8		INX
0432	F197	A5FD		LDA PS
0433	F199	4CE5F4		JMP WBYTE
0434	F19C			

INPUT AN ADDRESS

0436	F19C		; READ AN ADDRESS UNTIL A <RETURN>
0437	F19C		
0438	F19C	20A5F1	INPADR JSR WCOMMAND
0439	F19F	2091F4	JSR GETADR
0440	F1A2	4C40F1	JMP SQA1
0441	F1A5		

0442 F1A5 ; 0000 WRITE COMMANDCODE ONTO THE DISPLAY 0000
 0443 F1A5
 0444 F1A5 20FBF4 WCOMND JSR CLRDS^P
 0445 F1AB AD0501 WCOMI LDA CURCMD
 0446 F1AB BD2F01 STA D1BUFF+15
 0447 F1AE 60 RTS
 0448 F1AF ;
 0449 F1AF ;

*** INSERT & DELETE ***

0451 F1AF A000 INSDEL: LDY #0
 0452 F1B1 AD0201 ID0 LDA EAL
 0453 F1B4 CSFB CMP ADL ; TEST FOR END
 0454 F1B6 AD0301 LDA EAH
 0455 F1B9 ESF9 SBC ADH
 0456 F1BB B005 BCS ID1
 0457 F1BD F003 BEQ ID1
 0458 F1BF 4C40F1 JMP SQAI ; OUT
 0459 F1C2 AD0501 ID1 LDA CURCMD
 0460 F1C5 C912 CMP #DELCMD
 0461 F1C7 F00B BEQ ID0
 0462 F1C9 B1FB LDA (ADL),Y ; INSERT
 0463 F1CB 48 PHA
 0464 F1CC 8A TXA
 0465 F1CD 91FB STA (ADL),Y
 0466 F1CF 68 PLA
 0467 F1D0 AA TAX
 0468 F1D1 4CDAF1 JMP IDNEXT
 0469 F1D4 ;
 0470 F1D4 C8 ID0 INY
 0471 F1D5 B1FB LDA (ADL),Y
 0472 F1D7 B8 DEY
 0473 F1D8 91FB STA (ADL),Y
 0474 F1DA IDNEXT
 0475 F1DA E6FB INC ADL
 0476 F1DC D0D3 BNE ID0
 0477 F1DE E6F9 INC ADH
 0478 F1E0 D0CF BNE ID0
 0479 F1E2 ;

*** CALCULATE BRANCH OFFSET ***

0481 F1E2 ; WHEN THE OFFSET IS OUT OF RANGE, '##' IS
 0482 F1E2 ; WRITTEN INTO THE DATA-FIELD AND THE
 0483 F1E2 ; CORRECT ADDRESS IS TO BE TYPED IN.
 0484 F1E2 ;
 0485 F1E2 20A5F1 OFFSET JSR WCOMND
 0486 F1E5 2091F4 OFFS1 JSR GETADR
 0487 F1E8 A202 LDX #2

PC-1 HEX-MONITOR 0000 PROTON 650X ASSEMBLER V4.4 PAGE: 0011

0488	F1EA	20D4F4	JSR RBYTE				
0489	F1ED	18	CLC				
0490	F1EE	E5F8	SBC ADL				
0491	F1F0	08	PHP				
0492	F1F1	A200	LDX #0				
0493	F1F3	B1F8	STA (ADL,X)				
0494	F1F5	20D4F4	JSR RBYTE				
0495	F1F8	28	PLP				
0496	F1F9	E5F9	SBC ADH				
0497	F1FB	F012	BEQ OFFS9				
0498	F1FD	C9FF	CMP #\$FF				
0499	F1FF	F00E	BEQ OFFS9				
0500	F201	A206	LDX #6				
0501	F203	A913	LDA #\$13				
0502	F205	2007F5	JSR WCHAR				
0503	F208	EB	INX				
0504	F209	2007F5	JSR WCHAR				
0505	F20C	4CE5F1	JMP OFFS1				
0506	F20F	;					
0507	F20F	4C31F1	OFFS9	JMP SEQADR			
0508	F212	;					

*** SINGLE STEP THE USER ***

0510	F212	;	START THE SINGLE-STEP TIMER (T2) AND				
0511	F212	;	PERFORM A "GO"-COMMAND.				
0512	F212	;					
0513	F212	ADE1FO	SSUSER	LDA VECTAB+2	;	SET INT-VECTOR ON 'SYSTEM-INT'	
0514	F215	B5F2		STA INTVEC			
0515	F217	ADE2FO		LDA VECTAB+3			
0516	F21A	B5F3		STA INTVEC+1			
0517	F21C	2057F6		JSR BLKDIS			
0518	F21F	2062F6		JSR RELDIS			
0519	F222	AD1EE0		LDA IER1			
0520	F225	09A0		DRA #\$A0			
0521	F227	BD1EE0		STA IER1			
0522	F22A	AD1BEO		LDA ACRI			
0523	F22D	29DF		AND #\$DF			
0524	F22F	BD1BEO		STA ACRI			
0525	F232	A924		LDA #\$24			
0526	F234	BD1BEO		STA T2LL1			
0527	F237	A900		LDA #0			
0528	F239	BD19EO		STA T2DH1			
0529	F23C	;					

*** GO TO THE USER-PROGRAM ***

0531	F23C	;	RESTORE THE USER REGISTERS FROM RAM INTO THE				
0532	F23C	;	PROCESSOR. THEN CALL THE USERS PROGRAM BY 'RTI'.				
0533	F23C	;					

0534 F23C A6FE Bouser LDX SPTR ; DEFINE STACK
 0535 F23E 9A TXS
 0536 F23F A5F9 LDA ADH
 0537 F241 48 PHA
 0538 F242 A5FB LDA ADL
 0539 F244 48 PHA
 0540 F245 A5FD LDA PS
 0541 F247 48 PHA
 0542 F24B A5FA LDA AC
 0543 F24A A6FB LDX XR
 0544 F24C A4FC LDY YR
 0545 F24E 40 NSUP RTI
 0546 F24F I

888 ENTER MONITOR FROM "BRK" 888

0548 F24F ; WHEN A 'BRK'-INSTRUCTION IS EXECUTED, ALL REGISTERS
 0549 F24F ; MUST BE STORED. THE MONITOR-PROGRAM RESUMES EXECUTION
 0550 F24F ; TO ALLOW THE OPERATOR TO VIEW MEMORY & REGISTERS.

0551 F24F ;
 0552 F24F 85FA BKUSER: STA AC
 0553 F251 86FB STX XR
 0554 F253 84FC STY YR
 0555 F255 68 PLA
 0556 F256 85FD STA PS ; STATUS
 0557 F258 D8 CLD
 0558 F259 68 PLA
 0559 F25A 38 SEC
 0560 F25B E901 SBC #1 ; CORRECT I FROM 'BRK'
 0561 F25D 85FB STA ADL ; PC
 0562 F25F 68 PLA
 0563 F260 E900 SBC #0
 0564 F262 85F9 STA ADH
 0565 F264 BA TSX
 0566 F265 B6FE STX SPTR
 0567 F267 AD1BE0 LDA T2LL1 ; CLEAR TIMER2
 0568 F26A 5B CLI
 0569 F26B 4C31F1 JMP SEDADR ; SHOW NEXT LOCATION
 0570 F26E ;
 0571 F26E ; 8888 DISASSEMBLE A INSTRUCTION 8888
 0572 F26E ;
 0573 F26E 20FBF4 DISA JSR CLRDSIP
 0574 F271 2003F9 JSR DISASM
 0575 F274 38 SEC -
 0576 F275 65FB ADC ADL
 0577 F277 85FB STA ADL
 0578 F279 9002 BCC DISA1
 0579 F27B E6F9 INC ADH
 0580 F27D 2087F2 DISA1 JSR GETKEY
 0581 F280 C94B CMP #DISCMD
 0582 F282 F0EA BEQ DISA

PC-1 HEX-MONITOR 8888 PROTON 650X ASSEMBLER V4.4 PAGE 0013

```

0003 F284 4C82F0    JMP JUMPER      T113
0004 F287           ; 
0005 F287           ; 
0006 F287           ; 8888 GET UPPER-CASE KEY 8888
0007 F287           ; 
0008 F287           ; TRANSFORM ALL KEYBOARD INPUT FOR UPPER-CASE
0009 F287           ; CHARACTERS. THIS SIMPLIFIES THE MONITOR PROGRAM.
0010 F287           ;
0011 F287 2095F2 GETUKY: JSR GETKEY
0012 F288 C960        CMP #960          ; TRANSFORM LOWERCASE TO UPPERCASE
0013 F28C 9006        BCC GUKY9
0014 F28E C97B        CMP #97B
0015 F290 B002        BCS GUKY9
0016 F292 29DF        AND #9DF
0017 F294 60          GUKY9        RTS
0018 F295           ;
0019 F295           ;

```

III SET A KEY FROM KEYBOARD 888

```

0020 F295           ;
0021 F295           ;
0022 F295           ; THE GETKEY WILL PERFORM BOTH DOWN- & UP-DEBOUNCE
0023 F295           ; ON ANY CHAR-KEY. THE SHIFT, ALPHA-LOCK AND 'CTRL'
0024 F295           ; FUNCTIONS ARE PERFORMED LATER.
0025 F295           ;
0026 F295 BA         GETKEY: TXA          ; SAVE .X & .Y
0027 F296 48          PHA
0028 F297 98          TYA
0029 F298 48          PHA
0030 F299 20F1F2        JSR ROWER          ; ROLL-OVER & REPEAT
0031 F29C B01D        BCS GKEY9          ; HAVE A REPEAT.
0032 F29E 2020F3 GKEY1    JSR SETDBN          ; START DEBOUNCE-TIMER
0033 F2A1 202AF3 GKEY2    JSR DECODEKEY      ; DECODE THE KEY(S)
0034 F2A4 90FB        BCC GKEY1          ; NO KEY DEPRESSED
0035 F2A6 2019F3 GKEY3    JSR DHOLDER       ; STILL DOWN ???
0036 F2A9 D0F3        BNE GKEY1          ; NO. RELEASED PREMATURELY
0037 F2AB 2026F3        JSR DDEBNC          ; TIME-OUT ?
0038 F2BD D0F6        BNE GKEY3          ; NO ...
0039 F2B0 2020F3        JSR SETDBN          ; START TIMER FOR REPEAT
0040 F2B3 A91B        LDA #27            ; START REPEAT AFTER 0.8 SEC
0041 F2B5 B03401        STA REPTO          ; DECODE KEY WITH SHIFT & DNTL
0042 F2B8 2073F3        JSR KDECOD          ; RESTORE .Y & .X
0043 F2B9 68          GKEY9        PLA
0044 F2BC A8          TAX
0045 F2BD 68          PLA
0046 F2BE AA          TAX
0047 F2BF AD3401        LDA LSTKEY
0048 F2C2 60          RTS
0049 F2C3           ;

```

1 CHECK FOR KEY-ENTRY

0632	F2C3		; EXIT .C=0 --> NO VALID CHR, NO KEY		
0633	F2C3		; EXIT .C=1 --> VALID CHR IN .A		
0634	F2C3				
0635	F2C3	8A	SCNKEY: TXA	; SAVE .X & .Y	
0636	F2C4	48	PHA		
0637	F2C5	98	TYA		
0638	F2C6	48	PHA		
0639	F2C7	20F1F2	JSR ROVER	; ROLL-OVER & REPEAT	
0640	F2CA	B01D	BCS SCNk9	; HAVE A REPEAT.	
0641	F2CC	2020F3	SCNk1	JSR SETDBN	; START DEBOUNCE-TIMER
0642	F2CF	2024F3	SCNk2	JSR DECKEY	; DECODE THE KEY(S)
0643	F2D2	9015	BCC SCNk9	; NO KEY DEPRESSED, EXIT .C=0	
0644	F2D4	2019F3	SCNk3	JSR QHOLD	; STILL DOWN ??
0645	F2D7	D0F3	BNE SCNk1	; NO, RELEASED PREMATURELY	
0646	F2D9	2026F3	JSR QDEBNC	; TIME-OUT ?	
0647	F2DC	D0F6	BNE SCNk3	; NO .	
0648	F2DE	2020F3	JSR SETDBN	; START TIMER FOR REPEAT	
0649	F2E1	A91B	LDA #27	; START REPEAT AFTER 0.8 SEK	
0650	F2E3	B03601	STA REPTO		
0651	F2E6	2073F3	JSR KDECOD		
0652	F2E9	68	SCNk9 PLA	; DECODE KEY WITH SHIFT & CNTL	
0653	F2EA	A8	TAY	; RESTORE .Y & .X	
0654	F2EB	68	PLA		
0655	F2EC	A8	TAX		
0656	F2ED	A03401	LDA LSTKEY		
0657	F2F0	60	RTS		
0658	F2F1				
0659	F2F1		; WAIT TILL KEY IS RELEASED (WITH DEBOUNCE)		
0660	F2F1		; & GENERATE REPEAT WHEN HELDED (16 PER SEK).		
0661	F2F1				
0662	F2F1	2019F3	ROVER	JSR QHOLD	; STILL DOWN ??
0663	F2F4	D014	BNE ROVERS		
0664	F2F6	2026F3	JSR QDEBNC	; NO.	
0665	F2F9	D0F6	BNE ROVER	; TIME-OUT	
0666	F2FB	2020F3	JSR SETDBN	; NO. TRY AGAIN	
0667	F2FE	C03601	DEC REPTO	; RESTART TIMER	
0668	F301	D0EE	BNE ROVER	; COUNT INTERVALS (200MS)	
0669	F303	A903	LDA #3		
0670	F305	B03601	STA REPTO	; 50MS	
0671	F308	38	SEC		
0672	F309	60	RTS	; REPEAT	
0673	F30A				
0674	F30A	2020F3	ROVERS	JSR SETDBN	
0675	F30D	2019F3	ROVER6	JSR QHOLD	; START TIMER
0676	F310	F0DF	BNE ROVER		
0677	F312	2026F3	JSR QDEBNC	; NOW SEE IF THE KEY	
0678	F315	D0F6	BNE ROVERS	; NOT RELEASED.	
0679	F317	18	CLC	; TIME-OUT ??	
0680	F318	60	RTS		
0681	F319				
0682	F319	A03201	QHOLD	LDA IMASK	; KEY IS RELEASED
0683	F31C	2006E0	BIT INKB		

; .NE=RELEASED

PC-1 HEX-MONITOR 6000 PROTON 650X ASSEMBLER V4.4 PAGE: 0015

0084 F31F 60 RTS
 0085 F320 |
 0086 F320 | ; ##### START DEBOUNCE COUNTER #####
 0087 F320 |
 0088 F320 TDEBNC =3 ; 20-30 MS
 0089 F320 |
 0090 F320 A903 SETDBN LDA #TDEBNC
 0091 F322 BD1701 STA DBCNTR ; START DEBOUNCE-TIMER
 0092 F325 60 RTS
 0093 F326 |
 0094 F326 | ; ##### TEST FOR DEBOUNCE-TIME-OUT #####
 0095 F326 |
 0096 F326 AD1701 QDEBNC LDA DBCNTR
 0097 F329 60 RTS ; .E0 = TIME-OUT
 0098 F32A |
 0099 F32A |
 0100 F32A | ; ##### DECODE A KEY #####
 0101 F32A |
 0102 F32A | THIS ROUTINE DOES 2 THINGS:
 0103 F32A | 1. SCAN THE 'SPECIAL' KEYS:
 0104 F32A | <L-SHIFT>, <R-SHIFT>, <ALPHA-LOCK>, <CRTL>
 0105 F32A | AND SAVE THESE BIT IN 'CPATRN'.
 0106 F32A | BIT 4: <ALPHA-LOCK>
 0107 F32A | 5: <R-SHIFT>
 0108 F32A | 6: <L-SHIFT>
 0109 F32A | 7: <CRTL>
 0110 F32A | 2. SCAN THE REST OF THE MATRIX AND HOLD THE OUTPUT
 0111 F32A | WHEN A 'CLOSED-CONTACT' IS DETECTED, (.C=1)
 0112 F32A | WHEN NO KEY IS FOUND THE .C = 0
 0113 F32A |
 0114 F32A A901 DECKY LDA #1 ; ##### INIT MASKS #####
 0115 F32C BD3201 STA IMASK ; (POSITIVE LOGIC)
 0116 F32F BD3101 STA OMASK
 0117 F332 A204 LDX #4 ; WE HAVE 4 KEYS TO READ FIRST
 0118 F334 2053F3 DECK1 JSR QKB ; SETOUT OMASK & READ INPUTS & BUMP
 0119 F337 38 SEC
 0120 F338 D001 BNE #+3
 0121 F334 18 CLC
 0122 F339 6E3501 ROR CPATRN
 0123 F33E CA DEX
 0124 F33F D0F3 BNE DECK1 ; CHECK FOR END
 0125 F341 AD3201 DECK2 LDA IMASK
 0126 F344 2D3101 AND OMASK
 0127 F347 2901 AND #1
 0128 F349 18 CLC ; NO KEY DEPRESSED
 0129 F34A D006 BNE DECK9 ; READ A MATRIX-KNOB
 0130 F34C 2053F3 JSR QKB ; NOT CLOSED
 0131 F34F D0F0 BNE DECK2
 0132 F351 38 SEC
 0133 F352 60 DECK9 RTS
 0134 F353 | ; ##### READ A MATRIX-KNOB #####
 0135 F353 | ; ##### READ A MATRIX-KNOB #####

0736 F353 ;
 0737 F353 4E3101 0KB LSR DMASK ; INCR. MASKS
 0738 F356 D00B BNE 0KB1
 0739 F358 6E3101 RDR DMASK ; .C=1
 0740 F358 4E3201 LSR IMASK
 0741 F35E D003 BNE 0KB1
 0742 F360 6E3201 RDR IMASK ; .C=1
 0743 F363 AD3101 0KB1 LDA DMASK
 0744 F366 1-INV- EOR #\$FF ; <<< FOR NON-INV. OUTPUT
 0745 F366 BD05E0 STA OUTKB
 0746 F369 2072F3 JSR 0KB2 ; DELAY 12 USEK
 0747 F36C AD3201 LDA IMASK
 0748 F36F 2D06E0 AND INKB
 0749 F372 60 0KB2 RTS
 0750 F373 ;
 0751 F373 ;
 0752 F373 ; *** DECODE 'KEYNR' FROM MATRIX(X,Y) ***
 0753 F373 ;
 0754 F373 AD3201 KDECOD LDA IMASK
 0755 F376 A200 LDX #0
 0756 F378 20C2F3 JSR BINHEX ; CONVERT BIT-PATTERN TO HEX
 0757 F37B BA TXA
 0758 F37C 0A ASL A ; ADJUST FOR O-MASK
 0759 F37D 0A ASL A
 0760 F37E 0A ASL A
 0761 F37F AA TAX
 0762 F380 AD3101 LDA DMASK
 0763 F383 20C2F3 JSR BINHEX
 0764 F386 BD0C9F3 LDA KMATRIX,X ; GET LOGICAL NUMBER
 0765 F399 BD3301 STA KEYNR ; SAVE THE KEY'S NUMBER
 0766 F38C ;
 0767 F38C ; HAVE A SHIFT ??
 0768 F38C ;
 0769 F38C AD3501 LDA CPATRN
 0770 F38F 4960 EOR #\$60
 0771 F391 2960 AND #\$60
 0772 F393 F00B BEQ KDECS ; NO SHIFT
 0773 F395 A204 LDX #SHIFT
 0774 F397 2010F4 JSR SPROC ; CALL THE SHIFT-PROCESSOR
 0775 F39A 4CB1F3 JMP KDECS
 0776 F39D ;
 0777 F39D AD3501 KDECS LDA CPATRN
 0778 F3A0 2910 AND #\$10
 0779 F3A2 D00B BNE KDEC7 ; NO ALPHA-LOCK
 0780 F3A4 A200 LDX #LOCK
 0781 F3A6 2010F4 JSR SPROC ; CALL THE SHIFT-PROCESSOR
 0782 F3A9 4CB1F3 JMP KDEC8
 0783 F3AC ;
 0784 F3AC A900 KDEC7 LDA #0
 0785 F3AE 202DF4 JSR SPROC ; GET VALUE FROM KEYNR
 0786 F3B1 ;
 0787 F3B1 ; NOW MODIFY THE CODE ON 'CTRL'

PC-1 HEX-MONITOR 8888 PROTON 650X ASSEMBLER V4.4 PAGE: 0017

0788	F3B1		LDA CPATRN		0131	0131
0789	F3B1	AD3501	KDEC8	BMI KDEC9	0132	0132
0790	F3B4	3008		LDA LSTKEY	0133	0133
0791	F3B5	AD3401		AND #X00011111	0134	0134
0792	F3B9	291F		STA LSTKEY	0135	0135
0793	F3B8	BD3401		LDA LSTKEY	0136	0136
0794	F3B6	AD3401	KDEC9	RTS	0137	0137
0795	F3C1	60			0138	0138
0796	F3C2				0139	0139
0797	F3C2				0140	0140
0798	F3C2		1	8888 CONVERT BIT (.A) TO HEX ADD TO .X 8888	0141	0141
0799	F3C2				0142	0142
0800	F3C2	4A	BINHEX	LSR A	0143	0143
0801	F3C3	B003		BCS BHEX9	0144	0144
0802	F3C5	E8		INX	0145	0145
0803	F3C6	DOFA		BNE BINHEX	0146	0146
0804	F3C8	60	BHEX9	RTS	0147	0147
0805	F3C9				0148	0148
0806	F3C9				0149	0149
0807	F3C9		1	8888 CONVERT MATRIX TO KEY-NUMBERS 8888	0150	0150
0808	F3C9				0151	0151
0809	F3C9	01	KMATRIX	.BYT 1,46,56,35,3,45,1,53	0152	0152
0810	F3D1	2C		.BYT 44,19,41,23,5,4,37,42	0153	0153
0811	F3D9	15		.BYT 21,22,36,38,7,6,24,40	0154	0154
0812	F3E1	14		.BYT 20,25,43,39,9,8,26,32	0155	0155
0813	F3E9	1F		.BYT 31,28,27,33,11,10,29,14	0156	0156
0814	F3F1	10		.BYT 16,30,34,15,12,2,13,17	0157	0157
0815	F3F9	01		.BYT 1,52,55,0,18,48,49,51	0158	0158
0816	F401	01		.BYT 1,54,47,50	0159	0159
0817	F405			-- ALL UNASSIGNED KEYS ARE CODED AS THE SPACE-BAR --	0160	0160
0818	F405				0161	0161
0819	F405				0162	0162
0820	F405		1	8888 THIS TABLE CONTAINS INFORMATION OVER 8888	0163	0163
0821	F405			WHAT OPERATION MUST BE PERFORMED ONTO	0164	0164
0822	F405			WHITCH KEY.	0165	0165
0823	F405				0166	0166
0824	F405				0167	0167
0825	F405		SHIFTBL	=#	0168	0168
0826	F405				0169	0169
0827	F405	13	LOCK	=#-SHIFTBL	0170	0170
0828	F407	20		.BYT 19,44	0171	0171
0829	F408	00		.BYT \$20	0172	0172
0830	F409			.BYT 0	0173	0173
0831	F409				0174	0174
0832	F409	03	SHIFT	=#-SHIFTBL	0175	0175
0833	F409	10		.BYT 3,17	0176	0176
0834	F40B	12		.BYT \$10	0177	0177
0835	F40C	20		.BYT 18,48	0178	0178
0836	F40E	00		.BYT \$20	0179	0179
0837	F40F			.BYT 0	0180	0180
0838	F410				0181	0181
0839	F410		1	8888 SHIFT-PROCESSOR 8888	0182	0182
0840	F410				0183	0183

```

0840 F410      ; PERFORMS THE SCANNING OF THE LIST (.X)
0841 F410      ; CHECKS IF KEYNR IS IN RANGE, GETS THE
0842 F410      ; ASCII VALUE AND DOES THE OPERATION.
0843 F410      ;
0844 F410 8D05F4 SPROC    LDA SHFTBL,X
0845 F413 F018      BEQ SPROC7      ; END OF LIST, NO-OP
0846 F415 EB       INX
0847 F416 CD3301    CMP KEYNR
0848 F419 F002      BEQ SPROC3
0849 F41B B008      BCS SPROC1      ; NOT IN RANGE
0850 F41D 8D05F4 SPROC3    LDA SHFTBL,X
0851 F420 CD3301    CMP KEYNR
0852 F423 B004      BCS SPROC2      ; IN RANGE . .
0853 F425 EB       SPROC1      INX
0854 F426 EB       INX      ; TRY ON THE NEXT BLOCK
0855 F427 D0E7      BNE SPROC
0856 F429      ;
0857 F429 EB       SPROC2      INX      ; POINT TO THE OPERAND
0858 F42A 8D05F4    LDA SHFTBL,X
0859 F42D AC3301 SPROC7      LDY KEYNR
0860 F430 5937F4    EOR #SCVAL,Y
0861 F433 8D3401    STA LSTKEY      ; SAVE ASCII VALUE
0862 F436 60       RTS
0863 F437      ;
0864 F437      ; 0000 TRANSLATION OF KEYNR TO ASCII VALUE 0000
0865 F437      ;
0866 F437 11      ASCVAL    .BYT $11      ; CHAR INS
0867 F438 20      .BYT $20      ; SPACE
0868 F439 3031    .BYT '0123456789;,-./' ; KEY 2 - 17
0869 F449 4061    .BYT 'abcdefghijklmnopqrstuvwxyz' ; KEY 18 - 49
0870 F459 7071    .BYT 'pqrsuvwxyz[\]^'; KEY 50
0871 F468 12      .BYT $12      ; CHAR DEL
0872 F469 0D      .BYT $0D      ; KEY 55
0873 F46A 08      .BYT $08      ; KEY 56
0874 F46B 0A      .BYT $0A      ; KEY 57
0875 F46C 0B      .BYT $0B      ; KEY 58
0876 F46D 0C      .BYT $0C      ; KEY 59
0877 F46E 1E      .BYT $1E      ; KEY 60
0878 F46F 1B      .BYT $1B      ; HOME KEY 55
0879 F470          .BYT $1B      ; ESC
0880 F470          ;
0881 F470          ;

```

*** GET HEX KEY OR DO COMMAND ***

```

0883 F470 20B7F2 GETHEX   JSR GETKEY
0884 F473 C90D      CMP NENTER      ; UPPER-CASE KEY
0885 F475 F016      BEQ NENTER
0886 F477 C930      CMP #0
0887 F479 9013      BCC GHX19
0888 F47B C93A      CMP #''

```

; ERROR

PC-1 HEX-MONITOR 0000 PROTON 650X ASSEMBLER V4.4 PAGE: 0019

000 F47D 900A BCC GHEX1
001 F47F C941 CMP #'A
002 F481 900B BCC GHEX5
003 F483 C947 CMP #'6
004 F485 B007 BCS GHEX5
005 F487 E906 SBC #6
006 F489 290F GHEX1 AND #1F
007 F48B C910 CMP #10
008 F48D 60 GHEX9 RTS
009 F48E ;
010 F48E 4C82F0 GHEX5 JMP JUMPER
011 F491 ; : GO TO EXECUTE THE COMMAND

000 SET 4 DIGIT HEX ADDRESS ***

002 F491 ; GET AN 4-DIGIT ADDRESS INTO THE ADDRESS-FIELD.
003 F491 ;
004 F491 A900 GETADR LDA #0
005 F493 AA TAX
006 F494 2007F5 GA0 JSR WCHAR ; CLEAR ADR-FIELD
007 F497 EB INX
008 F498 E004 CPX #4
009 F49A 30F8 BMI GA0
010 F49C 2070F4 GA1 JSR GETHEX
011 F49F F015 BEQ GAEND
012 F4A1 A200 LDX #0
013 F4A3 48 PHA ; SAVE DATA
014 F4A4 B02101 GA2 LDA DIBUFF+1,X ; LEFT-SHIFT 4 DIGITS
015 F4A7 9D2001 STA DIBUFF,X
016 F4AA EB INX
017 F4AB E003 CPX #3
018 F4AD 30F5 BMI GA2
019 F4AF 68 PLA
020 F4B0 2007F5 JSR WCHAR
021 F4B3 4C9CF4 JMP GA1
022 F4B6 60 GAEND RTS
023 F4B7 ;

000 GET BYTE IN DATA-FIELD ***

024 F4B7 2070F4 GETBYT JSR GETHEX
025 F4B8 F00E BEQ GBEND ; LEFT-SHIFT ON DATA-FIELD
026 F4B9 AE2701 LDX DIBUFF+7
027 F4BF BE2601 STX DIBUFF+5
028 F4C2 A207 LDX #7
029 F4C4 2007F5 JSR WCHAR
030 F4C7 4CB7F4 JMP GETBYT
031 F4CA A206 GBEND LDX #6
032 F4CC 20D4F4 JSR RBYTE
033 F4CF A200 LDX #0

0935	F4D1	81FB	STA (ADL,X)
0936	F4D3	60	RTS
0937	F4D4		

*** READ A BYTE FROM DISPLAY ***

0939	F4D4		; THE POSITION OF THE NUMBER READ IS INDEXED BY .X
0940	F4D4		; THE NUMBER IS READ AND IS LEFT IN .A
0941	F4D4		
0942	F4D4	2049F5 RBYTE	JSR RCHAR
0943	F4D7	0A	ASL A
0944	F4D8	0A	ASL A
0945	F4D9	0A	ASL A
0946	F4DA	0A	ASL A
0947	F4DB	85F5	STA TEMP+1
0948	F4DD	E8	INX
0949	F4DE	2049F5	JSR RCHAR
0950	F4E1	05F5	DRA TEMP+1
0951	F4E3	E8	INX
0952	F4E4	60	RTS
0953	F4E5		

*** WRITE BYTE TO DISPL(X) ***

0955	F4E5		; WRITE A BYTE FROM .A TO THE DISPLAYPOSITION
0956	F4E5		; INDEXED BY .X
0957	F4E5		
0958	F4E5	85F5 WBYTE	STA TEMP+1
0959	F4E7	4A	LSR A
0960	F4E8	4A	LSR A
0961	F4E9	4A	LSR A
0962	F4EA	4A	LSR A
0963	F4EB	2007F5	JSR WCHAR
0964	F4EE	E8	INX
0965	F4EF	A5F5	LDA TEMP+1
0966	F4F1	290F	AND #\$0F
0967	F4F3	2007F5	JSR WCHAR
0968	F4F6	E8	INX
0969	F4F7	60	RTS
0970	F4FB		

*** CLEAR DISPLAY TO BLANKS ***

0972	F4FB	A920 CLRDSP	LDA #' '
0973	F4FA	A210 FILDSP	LDX #DIBUFL
0974	F4FC	9D2001 CD1	STA DIBUFF,X
0975	F4FF	CA	DEX
0976	F500	10FA	BPL CD1
0977	F502	E8	INX

0002 F503 BE3701 STX CURPOS
 0003 F506 60 RTS
 0004 F507 ;
 0005 F507 ;
 0006 F507 ;
 0007 F507 BC1B01 NCHAR STY YSAV
 0008 F50A AB TAY
 0009 F50B BA TXA
 0010 F50C 290F AND #\$0F
 0011 F50E AA TAX
 0012 F50F B95FF5 LDA CHASET,Y
 0013 F512 9D2001 STA DIBUFF,X
 0014 F515 98 TYA
 0015 F516 AC1B01 LDY YSAV
 0016 F519 60 RTS
 0017 F51A ;
 0018 F51A ;
 0019 F51A A920 BLANK LDA #\$20
 0020 F51C ;
 0021 F51C ; OUTPUT CHAR TO CURSOR ON DISPLAY
 0022 F51C ;
 0023 F51C BE3801 OUTALL STX OXSAV
 0024 F51F AE3701 LDX CURPOS
 0025 F522 C908 CMP #BACKSP
 0026 F524 F010 BEQ OUTAL5
 0027 F526 9D2001 STA DIBUFF,X
 0028 F529 48 PHA
 0029 F52A E8 INX
 0030 F52B 8A TXA
 0031 F52C 290F AND #\$0F
 0032 F52E 8D3701 STA CURPOS
 0033 F531 AE3801 LDX OXSAV
 0034 F534 68 PLA
 0035 F535 60 RTS
 0036 F536 ;
 0037 F536 CA OUTAL5 DEX
 0038 F537 1002 BPL *+4
 0039 F539 A20F LDX #DIBUFL-1
 0040 F538 A920 LDA #\$20
 0041 F53D 9D2001 STA DIBUFF,X
 0042 F540 BE3701 STX CURPOS
 0043 F543 AE3801 LDX OXSAV
 0044 F545 A908 LDA #BACKSP
 0045 F548 60 RTS
 0046 F549 ;
 0047 F549 ;

0000 PC-1 HEX-MONITOR 0000 PROTON 6501 ASSEMBLER V4.4

000 READ CHR FROM DISPLAY(X) 000

1028 F549 I READ A CHAR FROM DISPLAY POSITION .X
 1029 F549 I THE CHAR IS CONVERTED TO ITS BINARY VALUE.
 1030 F549 I
 1031 F549 BE1A01 RCHAR STX XSRV
 1032 F54C BD2001 LDA DIBUFF,X
 1033 F54F A214 LDX #NCHARS
 1034 F551 D05FF5 RC1 CMP CHASET,X
 1035 F554 F004 BEQ RC2
 1036 F556 CA DEX
 1037 F557 10FB BPL RC1
 1038 F559 EB INX
 1039 F55A 8A RC2 TIA
 1040 F55B AE1A01 LDY XSRV
 1041 F55E 60 RTS
 1042 F55F I
 1043 F55F 3031 CHASET .BYT '01234'
 1044 F564 3536 .BYT '56789'
 1045 F569 4142 .BYT 'ABCDEF'
 1046 F56F 202E .BYT ' .--'
 1047 F573 I
 1048 F573 I NCHARS1 = #CHASET
 1049 F573 I
 1050 F573 I 0000 PRINT A TEXT ON THE DISPLAY 0000
 1051 F573 I
 1052 F573 I THE TEXT IS ADDRESSED BY AN ADDRESS IN THE .Y/.A
 1053 F573 I REGISTER PAIR. .Y = HIGH-BYTE, .A = LOW-BYTE.
 1054 F573 I THE END OF THE TEXT IS MARKED BY A '00'.
 1055 F573 I
 1056 F573 B5F4 TPRINT STA TEMP ; TEXT-ADDRESS IN .YA
 1057 F575 B4F5 STY TEMP+1
 1058 F577 20FBF4 JSR CLRDISP
 1059 F57A A000 LDY #0
 1060 F57C B1F4 TPRNT1 LDA (TEMP)Y
 1061 F57E F006 BEQ TPRNT9 ; NEOTS
 1062 F580 992001 STA DIBUFF,Y
 1063 F583 C8 INY
 1064 F584 D0F6 BNE TPRNT1
 1065 F586 60 TPRNT9 RTS
 1066 F587 I
 1067 F587 I 0000 TEST THE KEYBOARD/DISPLAY & I/O 0000
 1068 F587 I
 1069 F587 A900 TESTER LDA #0 ; INIT VIA
 1070 F589 BD13E0 STA DDRB1
 1071 F58C A9FF LDA #FFF
 1072 F58E BD12E0 STA DDRB1
 1073 F591 A9EC LDA #SEC
 1074 F593 BD1CE0 STA POR1 ; CA2=0, CB2=1
 1075 F596 20FBF4 JSR CLRDISP

F599	A200	TEST1	LDX #0
F599	2095F2	TEST2	JSR GETKEY
F59E	C91B		CMP #ESC
F5A0	F012		BEQ TEST3
F5A2	9D2001		STA DIBUFF,X
F5A5	AD1CE0		LDA PDR1
F5A8	4922		EOR #\$22
F5AA	BD1CE0		STA PDR1
F5AD	E8		INX
F5AE	E010		CPX #DIBUFL
F5B0	B0E7		BCS TEST1
F5B2	90E7		BCC TEST2
F5B4	2015F7	TEST3	JSR VIAINI
F5B7	AD11E0	TEST4	LDA DRA1
F5B8	BD10E0		STA DRB1
F5B9	B0F8		BCS TEST4
FSBF	;		
FSBF	;		
FSBF	;		

000 INTERRUPT HANDLING 000

```

1096 F5BF      ; 000 HANDLER FOR IRQ 000
1097 F5BF      ; 000
1098 F5BF      ;
1099 F5BF 85F4 IRQHDL: STA TEMP
1100 F5C1 68 PLA
1101 F5C2 4B PHA
1102 F5C3 2910 AND #810
1103 F5C5 D00C BNE BRKINT
1104 F5C7 A01DE0 LDA IFR1
1105 F5C9 2920 AND #920
1106 F5CC D00A BNE STEPIN      ; SST-INTERRUPT TIMER T2
1107 F5CE A5F4 LDA TEMP
1108 F5D0 6CF600 JMP (IRQVEC)
1109 F5D3      ;
1110 F5D3 A5F4 BRKINT LDA TEMP
1111 F5D5 4C4FF2 JMP BRUSER
1112 F5D8      ;
1113 F5D9 A5F4 STEPIN LDA TEMP
1114 F5D9 4CFEF0 JMP SSTINT
1115 F5D0      ;
1116 F5D0      ; 000 HANDLER FOR NMI & IRQ 0000
1117 F5D0      ;
1118 F5D0 6CF000 NMI: JMP (NMIVEC)
1119 F5E0 6CF200 IRQ: JMP (INTVEC)
1120 F5E3      ;
1121 F5E3      ;

```

0000 DISPLAY ROUTINE 0000

```

1123 F5E3      ;
1124 F5E3      ; 0000 HANDLE THE INTERRUPTS 0000
1125 F5E3      ;
1126 F5E3 4B TERMINL: PHA
1127 F5E4 9A TIA
1128 F5E5 4B PHA
1129 F5E6 9B TYA
1130 F5E7 4B PHA
1131 F5EB 2021F6 JSR DISPL
1132 F5EB 20F4F5 JSR TIMUD      ; NEXT CHAR
1133 F5EE AB PLA      ; UPDATE TIMERS
1134 F5EF AB TAY
1135 F5F0 6B PLA
1136 F5F1 4A TAX
1137 F5F2 6B PLA
1138 F5F3 40 RTI
1139 F5F4      ;
1140 F5F4      ; UPDATE THE TIMERS

```

F5E4				
F5F4	CE0401	TIMUPD	DEC DIV1	
F5F7	1027		BPL TIMUP9	; EVERY 512 USEK
F5F9	A913		LDA #20-1	
F5FB	BD0401		STA DIV1	; 10.24 MS INTERVAL
F5FE	A212		LDX #18	
F600	D8	TIMUP1	CLD	; 8 TIMERS & DBCNTR
F601	38		SEC	
F602	BD0701		LDA TIMER,X	
F605	E901		SBC #1	
F607	9D0701		STA TIMER,X	
F60A	BD0801		LDA TIMER+1,X	
F60D	E900		SBC #0	
F60F	9D0801		STA TIMER+1,X	
F612	900B		BCS TIMUP2	; PAST '0000' ??
F614	A900		LDA #0	
F616	9D0701		STA TIMER,X	
F619	9D0801		STA TIMER+1,X	
F61C	CA	TIMUP2	DEX	
F61D	CA		DEX	
F61E	10E0		BPL TIMUP1	
F620	60	TIMUP9	RTS	
F621				
F621				
F621			1 **** REFRESH THE DISPLAY ****	
F621				
F621				
F621	AE3001	DISPL:	LDX DCPTR	; SHOW NEXT CHR.
F624	EB		INX	
F625	E010		CPX #DIBUFL	
F627	9002		BCC DISP1	; MODULO DIBUFL
F629	A200		LDX #0	
F62B	BE3001	DISP1	STX DCPTR	; SAVE THE PTR
F62E	A910		LDA #\$10	; BLANK CHR-SEL
F630	BD04E0		STA CHSEL	
F633	BD2001		LDA D1BUFF,X	
F636	D8		CLD	
F637	38		SEC	
F638	E920		SBC #\$20	
F63A	C940		CMP #\$40	
F63C	9002		BCC DISP2	
F63E	E920		SBC #\$20	; L-CASE --> U-CASE
F640	0A	DISP2	ASL A	; CHR. VAL ?!
F641	AA		TAX	
F642	BD6BF6		LDA CHSET,X	
F645	BD03E0		STA SEGMO	
F648	BD69F6		LDA CHSET+1,X	
F64B	BD02E0		STA SEGMI	
F64E	AD3001		LDA DCPTR	
F651	490F		EDR #\$0F	
F653	BD04E0		STA CHSEL	
F656	60		RTS	
F657				

1193 F657 ; DISABLE & ENABLE DISPLAY-INTERRUPTS (NMI)

1194 F657 ;

1195 F657 A920 BLXDIS LDA #920 ;

1196 F659 BD07E0 STA NMICTR ; CONTROL REGISTER

1197 F65C A910 LDA #910 ;

1198 F65E BD04E0 STA CHSEL ; BLANK THE DISPLAY

1199 F661 60 RTS ;

1200 F662 ;

1201 F662 A900 RELDIS LDA #900 ; RELEASE DISPLAY-INTERRUPTS

1202 F664 BD07E0 STA NMICTR ;

1203 F667 60 RTS ;

1204 F668 ;

1205 F668 ;

1206 F668 ; 0000 CHARACTER SET 0000

1207 F668 ;

1208 F668 ; FORMAT & HARDWARE : 1ST BYTE BIT 0-7 SEGMENT A-H

1209 F668 ; 2ND BYTE BIT 0-7 SEGMENT I-P

1210 F668 ;

1211 F668 ; A LOW BIT TURNS THE CORRESPONDING SEGMENT ON.

1212 F668 ;

1213 F668 FFFF CHSET .DBYTE \$FFFF ; SPACE

1214 F66A CEDB .DBYTE \$CEDB ;

1215 F66C DFFE .DBYTE \$DFFE ;

1216 F66E 31FC .DBYTE \$31FC ;

1217 F670 12FC .DBYTE \$12FC ;

1218 F672 1BC3 .DBYTE \$1BC3 ;

1219 F674 A2E6 .DBYTE \$A2E6 ;

1220 F676 FFFB .DBYTE \$FFFB ;

1221 F678 FFEB .DBYTE \$FFEB ; (

1222 F67A FFD7 .DBYTE \$FFD7 ;)

1223 F67C 3FC0 .DBYTE \$3FC0 ;

1224 F67E 3FFC .DBYTE \$3FFC ; *

1225 F680 FFD9 .DBYTE \$FFD9 ;

1226 F682 3FFF .DBYTE \$3FFF ;

1227 F684 FF7F .DBYTE \$FF7F ;

1228 F686 FFDB .DBYTE \$FFDB ; /

1229 F688 C00B .DBYTE \$C00B ; 0

1230 F68A FFFC .DBYTE \$FFFC ; 1

1231 F68C 24FF .DBYTE \$24FF ; 2

1232 F68E 70FF .DBYTE \$70FF ; 3

1233 F690 19FF .DBYTE \$19FF ; 4

1234 F692 96EF .DBYTE \$96EF ; 5

1235 F694 02FF .DBYTE \$02FF ; 6

1236 F696 FBFF .DBYTE \$FBFF ; 7

1237 F698 00FF .DBYTE \$00FF ; 8

1238 F69A 10FF .DBYTE \$10FF ; 9

1239 F69C F6FF .DBYTE \$F6FF ; :

1240 F69E BFDF .DBYTE \$BFDF ; 1

1241 F6A0 F7DB .DBYTE \$F7DB ; <

1242 F6A2 37FF .DBYTE \$37FF ; *

1243 F6A4 F7E7 .DBYTE \$F7E7 ; *

1244 F6A6 7CFD .DBYTE \$7CFD ; ?

FAD5	A0FD	.DBYTE \$A0FD	; A
FAD6	0BFF	.DBYTE \$0BFF	; A
FAD7	70FC	.DBYTE \$70FC	; B
FAD8	C6FF	.DBYTE \$C6FF	; C
FAD9	F0FC	.DBYTE \$F0FC	; D
FADA	06FF	.DBYTE \$06FF	; E
FADB	0EFF	.DBYTE \$0EFF	; F
FADC	42FF	.DBYTE \$42FF	; G
FADD	09FF	.DBYTE \$09FF	; H
FADE	F6FC	.DBYTE \$F6FC	; I
FADF	E1FF	.DBYTE \$E1FF	; J
FADG	BFE8	.DBYTE \$BFE8	; K
FADH	C7FF	.DBYTE \$C7FF	; L
FADJ	C9F3	.DBYTE \$C9F3	; M
FADK	C9E7	.DBYTE \$C9E7	; N
FADL	C0FF	.DBYTE \$C0FF	; O
FADM	0CFF	.DBYTE \$0CFF	; P
FADN	C0EF	.DBYTE \$C0EF	; Q
FADO	0CEF	.DBYTE \$0CEF	; R
FADP	12FF	.DBYTE \$12FF	; S
FADQ	FEFC	.DBYTE \$FEFC	; T
FADR	C1FF	.DBYTE \$C1FF	; U
FADS	CFDB	.DBYTE \$CFDB	; V
FADT	C9CF	.DBYTE \$C9CF	; W
FADU	FFC3	.DBYTE \$FFC3	; X
FADV	FFF1	.DBYTE \$FFF1	; Y
FADW	F6DB	.DBYTE \$F6DB	; Z
FADX	F9EB	.DBYTE \$F9EB	; [
FADY	FFE7	.DBYTE \$FFE7	;]
FADZ	CFD7	.DBYTE \$CFD7	; ^
FAD[FFDF	.DBYTE \$FFDF	; ~
FAD]	F7FF	.DBYTE \$F7FF	; -
FAD^			
FAD~			

*** CASSETTE DUMP ***

```

1280 F6EB      ;           *****
1281 F6EB      ;           *****
1282 F6EB      ;           *****
1283 F6EB      ;           **** DUMP MEMORY TO TAPE ****
1284 F6EB      ;           **** IN KIM-6502 FORMAT ****
1285 F6EB      ;           ****
1286 F6EB      ;           *****
1287 F6EB      ;
1288 F6EB      ;
1289 F6EB      F3700    =138          ; FOR 3700 HERTZ
1290 F6EB      F2400    =207          ; FOR 2400 HERTZ
1291 F6EB      ;
1292 F6EB      ;

```

*** DUMP MAIN ROUTINE ***

```

1294 F6EB 2057F6 DUMP:   JSR BLKDIS      ; BLOCK DISPLAY
1295 F6EB 20FAF6   JSR DMPOPN
1296 F6EE 2044F7   JSR DMPMEM
1297 F6F1 2062F7   JSR DMPCLS
1298 F6F4 2062F6   JSR RELDIS      ; RELEASE DISPLAY
1299 F6F7 4C31F1   JMP SEQADR
1300 F6FA      ;
1301 F6FA      ; *** OPEN FOR DUMP ***
1302 F6FA 2001F7 DMPOPN   JSR DMPINI
1303 F6FD 201BF7   JSR DMPHED
1304 F700 60       RTS
1305 F701      ;
1306 F701      ; *** INIT HARDWARE FOR DUMP ***
1307 F701 A9C0 DMPINI   LDA #9C0        ; T1 FREE-RUNNING, PB7 PULS
1308 F703 8D1BE0   STA ACRI
1309 F706 2015F7   JSR VIAINI
1310 F709 A900   LDA #0
1311 F70B 8D1E01   STA CHKSML
1312 F70E 8D1F01   STA CHKSML
1313 F711 8D15E0   STA T1CH1
1314 F714 60       RTS
1315 F715      ;
1316 F715 A9C0 VIAINI   LDA #9C0
1317 F717 8D1CE0   STA POR1      ; SET CB2 LOW
1318 F71A 60       RTS
1319 F71B      ;
1320 F71B      ; *** DUMP HEADER ***
1321 F71B A264 DMPHED   LDX #100
1322 F71D B6F4   STX TEMP
1323 F71F A916   LDA #916
1324 F721 209BF7 NXTSNIC JSR DMPACC

```

; PRE-AMBLE

F724	C6F4	DEC TEMP	RTN
F726	D0F9	BNE NXTSMC	
F728	A92A	LDA #'	
F729	209BF7	JSR DMPACC	
F72A	AD0001	LDA ID	; TRIGGER CHR
F730	208EF7	JSR DMPBYT	
F733	ASF8	LDA ADL	; ID. NUMBER
F735	207CF7	JSR ADDCK	
F738	208EF7	JSR DMPBYT	; START ADDRESS
F73B	ASF9	LDA ADH	
F73D	207CF7	JSR ADDCK	
F740	208EF7	JSR DMPBYT	
F743	60	RTS	
F744			
F744		; *** DUMP BODY (MEMORY) ***	
F744	A000	DMPMEM	LDY #0
F746	B1F8		LDA (ADL),Y
F748	207CF7		JSR ADDCK
F748	208EF7		JSR DMPBYT
F74E	E6F8		INC ADL
F750	D002		BNE DM1
F752	E6F9		INC ADH
F754	38	DM1	SEC
F755	AD0201		LDA EAL
F758	E5F8		SBC ADL
F75A	AD0301		LDA EAH
F75D	E5F9		SBC ADH
F75F	B0E3		BDS DMPMEM
F761	60		RTS
F762			
F762		; *** DUMP END-RECORD ***	
F762	A92F	DMPCLS	LDA #'/
F764	209BF7		JSR DMPACC
F767	AD1E01		LDA CHKSML
F76A	208EF7		JSR DMPBYT
F76D	AD1F01		LDA CHKSMH
F770	208EF7		JSR DMPBYT
F773	A904		LDA #04
F775	209BF7		JSR DMPACC
F778	209BF7		JSR DMPACC
F778	60		RTS
F77C			
F77C		; *** ADD BYTE TO CHECKSUM ***	
F77D	48	ADDCK	PHA
F77D	18		CLC
F77E	6D1E01		ADC CHKSML
F781	BD1E01		STA CHKSML
F784	AD1F01		LDA CHKSMH
F787	6900		ADC #0
F789	BD1F01		STA CHKSMH
F78C	68		PLA

1377	F780	60	RTS		
1378	F78E				
1379	F78E		DUMP ONE BYTE ***		
1380	F78E	20D1F7 DMPBYT	JSR HIASCH	; HEX --> ASCII HIGH-BYTE	
1381	F791	209BF7	JSR DMPACC		
1382	F794	20D9F7	JSR HIASOL	; HEX --> ASCII LOW-BYTE	
1383	F797	209BF7	JSR DMPACC		
1384	F79A	60	RTS		
1385	F79B				
1386	F79B		DUMP ACCUMULATOR ***		
1387	F79B	A208 DMPACC	LDX #8		
1388	F79D	48	PHA		
1389	F79E	48	PHA		
1390	F79F	18 NXTBIT	CLC		
1391	F7A0	20B3F7	JSR CPULSE	; 3700 Hz	
1392	F7A3	68	PLA		
1393	F7A4	48	LSR A		
1394	F7A5	48	PHA		
1395	F7A6	20B3F7	JSR CPULSE	; 3700 DR 2400 Hz	
1396	F7A9	38	SEC		
1397	F7AA	20B3F7	JSR CPULSE	; 2400 Hz	
1398	F7AD	CA	DEX		
1399	F7AE	D0EF	BNE NXTBIT		
1400	F7B0	68	PLA		
1401	F7B1	68	PLA		
1402	F7B2	60	RTS		
1403	F7B3				
1404	F7B3		MAKE A PULSE ***		
1405	F7B3	A900 CPULSE	LDA #0	; BIT IS IN CARRY	
1406	F7B5	2A	ROL A		
1407	F7B6	AB	TAY		
1408	F7B7	B9CDFF7	LDA FREQ,Y		
1409	F7B8	8D16E0	STA T1LL1		
1410	F7B9	B9C0FF7	LDA NPLS,Y		
1411	F7C0	AB	TAY		
1412	F7C1	2C1DE0 WAITPL	BIT IFRI		
1413	F7C4	50FB	BVC WAITPL		
1414	F7C6	AD14E0	LDA T1CL1	; CLEAR INT. FLAG	
1415	F7C9	88	DEY		
1416	F7CA	D0F5	BNE WAITPL		
1417	F7CC	60	RTS		
1418	F7CD				
1419	F7CD	88	FREQ	.BYTE F3700-2	
1420	F7CE	C0		.BYTE F2400-2	
1421	F7CF	12	NPLS	.BYTE 18	
1422	F7D0	0C		.BYTE 12	; NUMBER OF HALF-PULSES
1423	F7D1				
1424	F7D1		CONVERSION HEX --> ASCII ***		
1425	F7D1	85F5 HIASCH	STA TEMP+1		
1426	F7D3	4A	LSR A	; ENTRY FOR HIGH-BYTE	
1427	F7D4	4A	LSR A		
1428	F7D5	4A	LSR A		

F7D6	4A		LSR A
F7D7	1004		BPL HEXASC
F7D9	A5F5	HEXASC	LDA TEMP+1
F7DB	290F		AND #10F
F7DD	C90A		
F7DF	9002		
F7E1	6906	HA1	CMP #10A
F7E3	6930		BCC HA1
F7E5	60		ADC #6
F7E6	1		ADC #30
			RTS

I ENTRY FOR LOW-BYTE

888	CASSETTE LOAD	888	888	888
1441	F7E6			
1442	F7E6			
1443	F7E6			
1444	F7E6			
1445	F7E6		LOAD MEMORY FROM TAPY	
1446	F7E6		IN "KIM-6502" FORMAT	
1447	F7E6			
1448	F7E6			
1449	F7E6			
1450	F7E6			
1451	F7E6			
<hr/>				
888	LOAD	MAIN ROUTINE	888	888
1453	F7E6	2057FB LOAD:	JSR BLKDIS	; BLOCK DISPLAY
1454	F7E9	A9E0	LDA #9E0	; CR2 HIGH
1455	F7EB	BD1CEO	STA PCR1	
1456	F7EE	A97F	LDA #97F	; PB7 FOR INPUT
1457	F7F0	BD12E0	STA DDREB1	
1458	F7F3	A900	LDA #0	
1459	F7F5	BD1BEO	STA ACR1	; T2 ONE-SHOT
1460	F7FB	BD1E01	STA CKSM1	; CLEAR CHECKSUM
1461	F7FB	BD1F01	STA CKSMH	
1462	F7FE			
1463	F7FE	; 888 READ THE SYNC-BYTE'S 888		
1464	F7FE	20C9FB LO0SNC	JSR RD8BIT	
1465	FB01	6A	ROR A	
1466	FB02	C916	CMP #916	
1467	FB04	D0FB	BNE LO0SNC	
1468	FB06	A909	LDA #9	
1469	FB08	85F5	STA TEMP+1	
1470	FB0A	2080FB LS1	JSR LDACC	
1471	FB0D	C916	CMP #916	
1472	FB0F	D0ED	BNE LO0SNC	
1473	FB11	C6F5	DEC TEMP+1	
1474	FB13	I0F5	BPL LS1	
1475	FB15			
1476	FB15	; 888 FIND TRIGGER-WORD 888		
1477	FB15	2080FB LO0STR	JSR LDACC	
1478	FB18	C92A	CMP #'A	
1479	FB1A	F006	BED LS2	
1480	FB1C	C916	CMP #916	
1481	FB1E	F0F5	BED LO0STR	
1482	FB20	D0DC	BNE LO0SNC	
1483	FB22	2094FB LS2	JSR RD8BYTE	
1484	FB25	BD0101	STA TAPID	
1485	FB28	CD0001	CMP ID	; LEAVE CURRENT ID.

6800 PC-1 HEX-MONITOR 6800 PROTON 650X ASSEMBLER V4.4 PAGE: 0033

```

1486 FB2B F01A      BEQ LODSAD    ; LOAD EVERY FILE
1487 FB2D AD0001    LDA ID
1488 FB30 C900      CMP #0
1489 FB32 F013      BEQ LODSAD
1490 FB34 C9FF      CMP #1FF     ; ID=$FF 'RELOCATE & LOAD'
1491 FB36 D0C6      BNE LODSNC  ; FIND NEXT FILE
1492 FB38          I
1493 FB38 2094FB    JSR RDBYTE
1494 FB38 207CF7    JSR ADDCK
1495 FB3E 2094FB    JSR RDBYTE
1496 FB41 207CF7    JSR ADDCK
1497 FB44 4C57F8    JMP LODDAT
1498 FB47          ;
1499 FB47          ; *** GET START ADDRESS ***
1500 FB47 2094FB LODSAD  JSR RDBYTE
1501 FB4A 207CF7    JSR ADDCK
1502 FB4D 85F8      STA ADL
1503 FB4F 2094FB    JSR RDBYTE
1504 FB52 207CF7    JSR ADDCK
1505 FB55 85F9      STA ADH
1506 FB57          ;
1507 FB57          ; *** LOAD DATA TO MEM ***
1508 FB57 2094FB LODDAT  JSR RDBYTE
1509 FB5A B00F      BCS LODEND   ; #EOF#
1510 FB5C 207CF7    JSR ADDCK
1511 FB5F A000      LDY #0
1512 FB61 91FB      STA (ADL),Y
1513 FB63 E6FB      INC ADL
1514 FB65 D0F0      BNE LODDAT
1515 FB67 E6F9      INC ADH
1516 FB69 D0EC      BNE LODDAT
1517 FB6B          ;
1518 FB6B          ; *** TEST CHECKSUM ***
1519 FB6B 2094FB LODEND  JSR RDBYTE
1520 FB6E CD1E01    CMP CHKSML
1521 FB71 D008      BNE LE1
1522 FB73 2094FB    JSR RDBYTE
1523 FB76 CD1F01    CMP CHKSML
1524 FB79 F00D      BEQ LE2
1525 FB7B A9B0      LE1      LDA #$B0
1526 FB7D 0D0101    DRA TAPID   ; MARK TAPE-ERROR
1527 FB80 BD0101    STA TAPID
1528 FB83 A945      LDA #'E
1529 FB85 BD0501    STA CURCMD
1530 FB88 AD1BE0 LE2  LDA T2LL1   ; CLEAR T2 INT. FOR 'SST'
1531 FB88 2062F6    JSR RELDIS  ; RELEASE DISPLAY
1532 FB8E 2015F7    JSR VIAINI
1533 FB91 4C31F1    JMP SEQADR
1534 FB94          ;
1535 FB94          ; *** READ ONE BYTE ***
1536 FB94 20ABFB RDBYTE  JSR RDHEX
1537 FB97 B00E      BCS RD91   ; #EOF#

```

1538	F899	0A	ASL A	
1539	F89A	0A	ASL A	
1540	F89B	0A	ASL A	
1541	F89C	0A	STA TEMP	
1542	F89D	B5F4	JSR RDHEX	; EOF\$
1543	F89E	204BFB	BCS RD81	
1544	F8A2	B003	ORA TEMP	
1545	F8A4	05F4	CLC	
1546	F8A6	1B	RTS	
1547	F8A7	60	RD81	
1548	F8A8			
1549	F8A8		READ A HEX NUMBER \$00	
1550	F8A8	20BDFA	JSR LDACC	
1551	F8A8	C92F	CMP #1	
1552	F8AD	F00C	BED RH2	; EOF\$
1553	F8AF	3B	SEC	
1554	F8B0	E930	SBC #030	
1555	F8B2	C90A	CMP #00A	
1556	F8B4	3004	BMI RH1	
1557	F8B6	3B	SEC	
1558	F8B7	E907	SBC #007	
1559	F8B9	1B	CLC	; RETURN WITH HEX NUMBER
1560	F8BA	60	RH1	RTS
1561	F8BB	3B	RH2	SEC
1562	F8BC	60	RTS	; RETURN WITH EOF\$ MARKER
1563	F8BD			
1564	F8BD		READ 6 BIT'S TO ACC \$00	
1565	F8BD	A900	LDACC	LDA #0
1566	F8BF	A207		LDI #7
1567	F8C1	20C9FB	LDAA1	JSR RD81
1568	F8C4	6A	RDR A	
1569	F8C5	CA	DEX	
1570	F8C6	10F9	BPL LDA1	
1571	F8C8	60	RTS	
1572	F8C9			
1573	F8C9		READ ONE BIT \$00	
1574	F8C9	4B	RD81	PHA
1575	F8CA	2000FB	R80	JSR GETFRQ
1576	F8CD	F0FB		BED R80
1577	F8CF	A000		LDY #0
1578	F8D1	2000FB	R81	; WAIT FOR 2400 HZ
1579	F8D4	F003		JSR GETFRQ
1580	F8D6	C8		BED R82
1581	F8D7	10FB		INY
1582	F8D9	C009	R82	BPL R81
1583	F8DB	6B		CPY #9
1584	F8DC	60		PLA
1585	F8DD			RTS
1586	F8DD			; BIT IS IN CARRY
1587	F8DD			
1588	F8DD			
1589	F8DD			

PC-1 HEX-MONITOR PROTON 650X ASSEMBLER V4.4 PAGE: 0035

PROTON 650X ASSEMBLER V4.4

*** OPCODE CONVERSION ***

2409 015B | 0000 RESET & INTERRUPT VECTORS 0000
2410 015B | \$=ROMBAS-6+ROMSIZ
2411 015B |
2412 FFFA | .MOR NMI,RESET,IRQ
2413 FFFA DDF5 |
2414 0000 | .OPT INOBJECT
2415 0000 | 0000 PDS65 MONITOR LINKING 0000
2416 0000 |
2417 0000 | \$=C400
2418 0000 |
2419 C400 |
2420 C400 BFF5 | .MOR IRQHDL,NMI,IRQHDL
2421 C406 |
2422 C406 |
2423 C406 | .END PC249

ERRORS: 0000

<0000>