PHILIPS

MICROPROCESSOR ZELFSTUDIE-CURSUS

HANDLEIDING VOOR DE INSTRUCTOR 50

2

HANDLEIDING VOOR DE INSTRUCTOR 50

INHOUD

Deze handleiding bevat de gegevens die nodig zijn om met een "INSTRUCTOR 50" oefenprogramma's te kunnen testen en uitvoeren.

De INSTRUCTOR 50 is gebaseerd op de microprocessor 2650. Alle instructies van deze processor kunnen met behulp van de INSTRUCTOR 50 uitgevoerd worden; ook de instructies die de in- en uitvoer van gegevens bewerkstelligen. Voor dit laatste is geen additionele apparatuur vereist, aangezien het paneel de nodige toetsen bevat om data in te voeren, benevens een scherm met acht symboolplaatsen, om data zichtbaar te maken. Daarnaast zijn nog acht schakelaars met acht speciale lampen aanwezig om de communicatie met de buitenwereld te simuleren. Verder zijn er een aantal functietoetsen waarmee het mogelijk is, een proces dat onder invloed van een programma wordt uitgevoerd, te onderbreken, te wijzigen of bepaalde gegevens zichtbaar te maken. Via de in- en uitgang van de cassette-recorder kan informatie op cassetteband worden bewaard en weer teruggelezen in de INSTRUCTOR 50.

Ten behoeve van de lezers die zich willen verdiepen in de werking van de INSTRUCTOR 50, bevatten de bijlagen gedetailleerde gegevens over de opbouw van de INSTRUCTOR 50 en het MONITOR-programma.

INLEIDING

Stroomvoorziening

De INSTRUCTOR 50 wordt gevoed via een transformator, aangesloten op het 220 V/50 Hz wisselstroomnet. Het snoer aan de transformator is voorzien van een steker die aan de achterzijde van de INSTRUCTOR 50 in de stekerbus moet worden gestoken. Er is geen speciale schakelaar aanwezig voor het in- en uitschakelen van de INSTRUCTOR 50. Bij het inschakelen moet het woord "HELLO" verschijnen op het beeldschermpje. Indien dit niet het geval is, moet de toets "MON" ingedrukt worden.

Opbouw van het bedieningspaneel

Aan de bovenzijde van de INSTRUCTOR 50 (zie fig. 1) bevinden zich drie velden met schakelaars (toetsen), benevens het scherm waarop symbolen (zie fig. 2) zichtbaar kunnen worden gemaakt. Het rechtse toetsenbord bevat zestien toetsen, genummerd 0 t/m 9 en A t/m F. Deze toetsen worden gebruikt om hexadecimale symbolen aan de INSTRUCTOR 50 aan te bieden voor de invoer van programma's, data, het raadplegen van de toestand van de microprocessor, enz. Het middelste toetsenbord bevat een aantal functietoetsen, die het inbrengen en uitlezen van data moeten begeleiden. Tevens worden deze toetsen gebruikt om programma's te onderbreken en vast te stellen hoe de uitvoering van een programma verloopt. Met behulp van deze toetsen en het veld van de hexadecimale toetsen worden de opdrachten aan de instructor gegeven; de instructor zal via het scherm aan de gebruiker de gewenste informatie verstrekken. Het linkse veld van acht schakelaars en acht lampjes dient ter simulatie van een in- en uitvoer van een microprocessor 2650 systeem. De soort van in- en uitvoerapparatuur kan door middel van de middelste schakelaar worden gekozen. Hierop wordt nog nader teruggekomen bij de behandeling van de in- en uitvoer van data. Aan de onderzijde is de microprocessor voorzien van een aantal gaten. Door deze gaten kan men bepaalde snelheden van de machinewerking kiezen, de wijze van interruptie aanbrengen en speciale voorzieningen treffen voor de uitbreiding van de INSTRUCTOR 50. Aan de achterzijde is de montageplaat zodanig uitgevoerd dat hierin een steker kan worden gestoken, zodat men later desgewenst speciale in- en uitvoerapparatuur op de microprocessor kan aansluiten.

P.S. Bij het uitvoeren van de eerste oefeningen moet de schakelaar aan de onderzijde in de stand "KEYBOARD" staan.

Functiemodes

De instructor heeft twee functiemodes. De eerste wordt de "MONITOR-MODE" genoemd en de andere de "EXECUTION-MODE". De MONITOR-MODE wordt ingeschakeld zodra men de INSTRUCTOR 50 voor de eerste maal inschakelt of als men de "MON"-toets indrukt. Op het scherm verschijnt dan het woord "HELLO". Gedurende deze MONITOR-MODE kunnen de volgende handelingen verricht worden:

1. Het inbrengen, lezen en veranderen van een programma.
2. Het schrijven van een programma naar de band van een cassetterecorder.
3. Het lezen van een band van een cassetterecorder.
4. Op het scherm weergeven van de inhoud van de registers van de microprocessor 2650 resp. het veranderen van deze inhoud.

5. Idem voor het PROGRAM-STATUS-WORD (PSW).
6. Het onderzoeken en veranderen van de inhoud van het geheugen.
7. Het onderzoeken en veranderen van de inhoud van het instructie-adresregister (IAR).
8. Het specificeren en onderzoeken van een breekpunt in het programma.
9. Het stap-voor-stap laten uitvoeren van een programma.

De EXECUTION-MODE wordt gekozen door het indrukken van de "RUN"-toets, de "STEP"-toets of de "RESET"-toets (RST-toets). Als de RUN-toets wordt ingedrukt, begint de programma-uitvoer met die instructie waarvan het adres aanwezig is in het instructieadres-register (IAR). Drukt men de STEP-toets in, dan wordt een enkele instructie gelezen en uitgevoerd, waarna de INSTRUCTOR 50 weer in de MONITOR-MODE terugkeert. Het indrukken van de RST-toets heeft tot gevolg dat de instructor zijn activiteit beëindigt en dat het instructie-adresregister in de stand H'0000' wordt gezet (H duidt aan, dat de notatie in hexadecimale code geschiedt).

Het inbrengen van programma's, adressen, data, enz.

Zoals eerder vermeld, wordt voor het inbrengen van instructies het rechtse hexadecimale veld toetsen gebruikt. In fig. 3 wordt het verband weergegeven tussen de decimale, binaire en hexadecimale getallen. Op de instructor zijn alle instructies in hexadecimale code weergegeven, waarbij rekening is gehouden met registernummers, de zogenaamde pagina's, enz. Als data of een adres aan de INSTRUCTOR 50 kunnen worden toevertrouwd, verschijnt uiterst links op het scherm een punt. Ontbreekt deze punt, kan men geen gegevens inbrengen. Wenst men dit wel te doen, dan moet de functietoets (zie later) eerst ingedrukt worden.

Het corrigeren van fouten

Zodra men met behulp van de zestien hexadecimale toetsen gegevens in de machine brengt, verschijnen deze gegevens op het scherm. Elk nieuw aangeslagen element verschijnt rechts op het scherm terwijl het bestaande patroon van symbolen naar links verschuift. Op deze wijze kan men een eventuele fout corrigeren, door de gewenste waarde eenvoudig opnieuw aan te slaan. Het foutieve gegeven verdwijnt dan, geleidelijk naar links schuivend, van het scherm en de nieuwe, juiste data verschijnt in de plaats daarvan.

Oefening

Druk op de toets "MON". De INSTRUCTOR 50 zal antwoorden met het bericht "HELLO" op het scherm. Druk vervolgens op de toets "MEM". Nu verschijnt ".AD.=." De INSTRUCTOR 50 is nu gereed om een geheugenadres in hexadecimale code in ontvangst te nemen. Wil men nu b.v. het adres 0120 selecteren en slaat men abusievelijk 0121 aan, dan zal men op het scherm waarnemen ".Ad.= 0121".

Deze fout is eenvoudig te corrigeren door de toetsen 0, 1, 2, 0 achtereenvolgens in te drukken. Voer deze correctie zelf uit.

Het geheugen en de I/O-organisatie

Behalve in de microprocessor 2650, is ook in de INSTRUCTOR 50 een geheugen aanwezig. In dit geheugen is de functie van de INSTRUCTOR 50 door middel van een ROM vastgelegd. Voorts bevat dit geheugen een aantal plaatsen die zijn gereserveerd voor het verrichten van de functies van de INSTRUCTOR 50. Om deze reden kunnen niet alle geheugenplaatsen vrij gebruikt worden. De indeling van het geheugen is in fig. 4 weergegeven. De indeling van het geheugen zal niet direct duidelijk zijn. Voorlopig is het echter voldoende te weten dat het aan de gebruiker ter beschikking staande geheugen als eerste adres H'0000'. en als laatste adres H'01FF' heeft. Het kan worden voortgezet in het gebied H'0200' tot H'0FFF' en van H'2000' tot H'7FFF'. Dit moet geschieden via de stekeraansluiting aan de achterzijde van de INSTRUCTOR 50.

Het gebied tussen adres H'17C0' en H'1FFF' wordt gebruikt voor het programma van de INSTRUCTOR 50 zelf. Hierin bevindt zich het z.g. "MONITOR"-programma.

Opmerking

Het wordt ontraden in programma's gebruik te maken van de geheugencel H'OFFF', daar deze gereserveerd is voor I/O operaties ten behoeve van de programma's die men zelf schrijft. Voor dat doel kan men de cel dus wèl gebruiken.

De zogenaamde "EXTENDED I/O" beslaat de adressen H'00' tot H'F7'. In de INSTRUCTOR 50 is echter alleen adres H'07' bruikbaar.

TOETSEN, SCHAKELAARS EN INDICATORS

Het besturingstoetsenveld

Het toetsenbord met de besturingstoetsen heeft ten doel het inbrengen van programma's te begeleiden, het testen hiervan, het opvragen van informatie uit de INSTRUCTOR 50 en het schrijven, resp. lezen van het ingebrachte programma op een externe cassetterecorder.

De functies van de voor dit doel bestemde toetsen zijn hieronder opgesomd.

toets — *functie*

MON — De "MONITOR"-toets wordt gebruikt om een programma dat op een bepaald moment werkzaam is, te beëindigen, en veroorzaakt een sprong naar het programma van de monitor. De machine geeft dit aan door het woord "HELLO" te laten verschijnen. Voordat men andere commando's geeft, is het indrukken van deze toets gewenst en doorgaans zelfs vereist.

MEM — Door het indrukken van de "MEMORY"-toets krijgt men toegang tot het geheugen van de INSTRUCTOR 50 voor het inbrengen en opvragen van instructies en data. De INSTRUCTOR 50 geeft als reactie ".Ad.=" Dit betekent dat de INSTRUCTOR 50 om een adres vraagt, d.w.z. men moet nu het adres invoeren waarvan men de data wil onderzoeken, resp. nieuwe data wenst in te brengen of bestaande data wenst te veranderen.

REG — De "REGISTER"-toets geeft toegang tot de registers van de microprocessor 2650, de programmateller, of tot de inhoud van het z.g. PSW. De INSTRUCTOR 50 reageert met het geven van "r=". Om aan te geven welk register, resp. welk deel van het PSW, men na het indrukken van deze toets kiest, wordt een van de hexadecimale toetsen ingedrukt.

ENT NXT — Deze "ENTER NEXT"-toets dient om eerder ingebrachte informatie in het geheugen resp. de registers op te slaan. De informatie kan er betrekking op hebben of een adres dan wel data, of wel een wijziging van data, inclusief de instructies in het geheugen ingevoerd wordt. Voorts kan men met behulp van deze toets snel de inhoud van het geheugen of de registers nagaan. Op het gebruik van deze toets in samenwerking met de toetsen REG en MEM wordt nog nader ingegaan.

STEP — In de monitor-mode is het mogelijk het programma, instructie na instructie, uit te laten voeren door deze toets telkenmale in te drukken. Op het scherm verschijnt nu het adres van de volgende instructie, die daarna wordt uitgevoerd. Op deze wijze kan men in zijn eigen programma nagaan of de uitvoering aan de verwachtingen voldoet.

RUN — Als deze toets wordt ingedrukt, zal een programma dat om een of ander reden onderbroken is, b.v. door het overgaan in de MONITOR-MODE bij het drukken van de toets "MON", wederom verder worden uitgevoerd. De instructie, die als eerste wordt uitgevoerd, wordt gegeven door de inhoud van het instructie-adres register. De uitvoering van het programma wordt voortgezet tot een van de onderstaande gebeurtenissen plaats heeft:

1. Het BREEKPUNT is bereikt.
2. De toets "MON" of "RST" wordt ingedrukt.
3. Het programma voert een "HALT"- of een "WRTC"-instructie uit.

RST — Drukt men de "RESET"-toets in, dan wordt de instructie die in uitvoering is, beëindigd en een RESET-signaal gegeven naar de microprocessor 2650. Dit heeft tot gevolg dat het programma opnieuw wordt begonnen met het adres H'000'. Indien deze toets wordt ingedrukt, komt het systeem niet in de MONITOR-MODE.

SENSE — De "SENSE"-toets is een nabootsing van een extern signaal, dat aan de SENSE-pen van de 2650 kan worden toegevoerd. Dit signaal kan nu met de hand worden nagebootst. SENSE-input is in de normale toestand een logische "0". Het indrukken van de SENSE-toets zal tot gevolg hebben dat een logische "1" wordt gegeven.

INT — De microprocessor 2650 kan worden geïnterrumpeerd tijdens de uitvoering van een programma. Deze "INTERRUPT" kan met behulp van deze toets worden nagebootst. Het resultaat is een "VECTORED INTERRUPT" naar de plaats H'07' in het geheugen. Het hangt af van de stand van de "DIRECT/INDIRECT"-schakelaar op het bedieningspaneel, of het programma, dat op grond van deze interruptie moet worden uitgevoerd, begint op de locatie H'07',

dan wel dat deze plaats verwijst (indirecte adressering) naar een andere locatie. De INSTRUCTOR 50 is voorzien van een schakelaar die via een gat in de bodem toegankelijk is. Bij de normale problemen dient deze schakelaar in de stand "KEYBOARD" te staan. Alleen als men externe interrupts vanuit het wisselspanningsnet toestaat, dient de schakelaar in de stand "A.C.LINE" te staan.

BKPT — De "BKPT"-toets geeft de opdracht om een "BREEKPUNT" te specificeren. Slaat men het adres van dit breekpunt aan via het hexadecimale toetsenveld, dan wordt de inhoud van de betreffende geheugencel op het scherm weergegeven zodra dit breekpunt in het programma wordt bereikt.

WCAS — Het indrukken van deze "WRITE CASSETTE"-toets draagt er zorg voor dat het programma dat in het INSTRUCTOR 50 geheugen is geregistreerd, wordt overgedragen op de band van de cassetterecorder.

RCAS — Het indrukken van de "READ CASSETTE"-toets bewerkstelligt dat de data van de cassetterecorder wordt overgenomen in het geheugen van de INSTRUCTOR 50.

Schakelaars voor het invoeren van externe data

De microprocessor 2650 kan d.m.v. I/O apparatuur met zijn omgeving corresponderen. Een van de I/O apparaten is nagebootst met behulp van acht poortschakelaars. Deze dienen om data aan de 2650 te verstrekken en wel in de vorm van een z.g. "byte". Wordt een leesopdracht met een bijbehorend adres aan de 2650 verstrekt, dan zal de stand van deze schakelaars in de 2650 worden overgenomen.

Indicators voor het uitvoeren van externe data

Voor het uitvoeren van data naar zijn omgeving is eveneens I/O apparatuur noodzakelijk. De uitgifte is nagebootst m.b.v. acht poort-LED's, aangesloten op acht FLIP-FLOPS. Deze FLIP-FLOPS kunnen bij een uitvoeringshandeling met de daarbij behorende adressering in de stand '0' of '1' worden geplaatst, met als gevolg dat de LED uit, resp. aan is.

DIRECT/INDIRECT-schakelaar

Als de microprocessor 2650 geïnterrumpeerd wordt, zal deze een sprong maken naar een z.g. "SUBROUTINE". In het geval van de INSTRUCTOR 50 zal dit de plaats H'07' zijn. Op deze plaats begint het serviceprogramma van de subroutine, resp. bevindt zich de eerste byte van de verwijzing voor de indirecte adressering naar de plaats waar de subroutine begint. De gebruiker van de INSTRUCTOR 50 heeft dus niet de mogelijkheid om bij een interrupt er zonder meer voor te zorgen dat de interrupt-routine van een bepaald adres uit start. Of de adressering direct dan wel indirect is, wordt aangegeven door de stand van de "DIRECT/INDIRECT" schakelaar. Als externe I/O apparaten met de INSTRUCTOR verbonden worden d.m.v. de steker aan de achterzijde, dan kan een ander interrupt-adres gebruikt worden ("EXTENDED I/O").

Poort-adres selectieschakelaar

Deze schakelaar kiest de wijze, waarop de parallel geschakelde in- en uitvoerapparatuur wordt geadresseerd. De schakelaar heeft drie standen:
"MEMORY 0FFF"
"EXTENDED I/O PORT 07"
"NON EXTENDED DATA PORT"

BREEKPUNT-toets

Een hulpmiddel bij het controleren van programma's op de correcte uitvoering is het z.g. BREEKPUNT. Dit breekpunt wordt opgeroepen met behulp van de "BKPT"-toets op een wijze, die later nog zal worden behandeld. De gebruiker van de INSTRUCTOR 50 draagt met behulp van deze faciliteit het adres van het BREEKPUNT aan de INSTRUCTOR 50 over. Dit adres geeft aan, waar de programma-uitvoering moet worden onderbroken. Zodra het instructie-adresregister deze waarde heeft bereikt, wordt de EXECUTION MODE beëindigd en gaat de machine over tot de MONITOR MODE. Daarbij geeft het scherm het adres van de eerst volgende uit te voeren instructie aan. Nadat de uitvoering van het programma is gestopt, kan men weer continueren door de toets "RUN" in te drukken of herhaaldelijk de toets "STEP" in te drukken, waarna, zoals bekend, telkens één instructie wordt uitgevoerd.

Snelheid

De INSTRUCTOR 50 is voorzien van een 3,579545 MHz kristal voor het opwekken van de zeer nauwkeurige klokfrequentie.

BEDIENING VAN DE INSTRUCTOR 50

Inleiding

De INSTRUCTOR 50 kan diverse opdrachten uitvoeren. De eerste belangrijke opdracht is het laden van instructies en data in het geheugen. Een tweede opdracht is die van het laden van registers. Dit laden kan geschieden gedurende de MONITOR MODE. Dit betekent, dat als, tijdens de uitvoering van een programma, de INSTRUCTOR 50 door een druk op de toets "MON" in de MONITOR MODE wordt gebracht, men de inhoud van de registers kan raadplegen en ook wijzigen. Verder kan men het instructie-adresregister (IAR) zichtbaar maken en eventueel wijzigen. Andere commando's zijn:

- Zichtbaar maken en veranderen van het geheugen
- Zetten van het BREEKPUNT
- Stap-voor-stap uitvoeren van de instructies
- Starten van het programma
- Het geheel in de 0-stand terugzetten (RESET).

In het nu volgende deel zullen al deze functies behandeld worden aan de hand van de betreffende COMMANDO TOETS. In de tekst en op de bijbehorende tekeningen geven de kleine letters gevolgd door een haakje het verband aan tussen de tekst en de plaats op de tekening, waarop deze betrekking heeft.

Lezen en veranderen van CPU-registers en PSW

Functie

Dit commando maakt het mogelijk de inhoud van de registers en het PSW (PROGRAM STATUS WORD) te lezen en zo nodig te wijzigen.

Procedure

1. Deze functie wordt ingeleid door in de MONITOR MODE, dus na het indrukken van de "MON"-toets, de knop "REG" in te drukken. Zie fig. 5a). Dit heeft tot gevolg dat op het scherm verschijnt "r=".

Voer oefening 1 uit!

De INSTRUCTOR 50 wacht nu op een nummer dat bij een bepaald register of bij een deel van het PSW behoort. Deze nummers zijn:

nummer	*register*
0	R0
1	R1, BANK 0
2	R2, BANK 0
3	R3, BANK 0
4	R1, BANK 1
5	R2, BANK 1
6	R3, BANK 1
7	PSW UPPER
8	PSW LOWER

2. Zodra men een van deze getallen aanslaat, in ons voorbeeld b), dan verschijnt op het scherm .r4=A8. Dit is slechts een voorbeeld, en kan op een bepaald moment de inhoud zijn van, in dit geval, register 1 in bank 1. De A8 is hier de weergave van de inhoud van dit register, doch die inhoud is afhankelijk van b.v. het starten van de machine door op RUN te drukken of door het inschakelen, waarbij de inhoud van de registers en de geheugenplaatsen willekeurige waarden aannemen.

Verricht oefening 2!

3. Is men met deze inhoud tevreden, dan kan men de knop "ENT NXT" indrukken, zoals onder c). Nu verschijnt automatisch het nummer en de inhoud van het volgende register op het scherm. In ons geval dus register 5, d.w.z. register 2 in bank 1. De inhoud blijkt hier B6 te zijn. Drukt men wederom op ENT NXT d), dan verschijnt op het scherm .r6=A4 Zo kan men doorgaan: drukt men wederom e), dan verschijnt op het scherm .PU=0A Dit is de inhoud van het PSW UPPER. Drukt men vervolgens weer op het ENT NXT f) dan verschijnt .PL=21 Dit is de inhoud van het PSW LOWER. Hieruit blijkt dat het totale Program Status Word is: H'0A21'. Het hierboven behandelde heeft dus uitsluitend betrekking op het nagaan van de inhoud van de REG 4,5,6,7 en 8, die staan voor resp. R1 BANK 1, R2 BANK 1, R3 BANK 1, PSW UPPER en PSW LOWER.

4. Het veranderen van de inhoud van de registers, resp. het Program Status Word UPPER en LOWER geschiedt op een iets andere wijze. Zodra een register gekozen is of de "ENT NXT" toets is ingedrukt, blijkt links op het scherm een punt te verschijnen. Dit betekent dat de inhoud van het betreffende register, resp. PSW-deel, nog te wijzigen is, voordat de "ENT NXT" toets wordt ingedrukt. Deze punt treedt in fig. 5 op in de gevallen b) t/m f).

We zullen nu behandelen hoe men de inhoud van een register kan wijzigen. We beginnen weer (zie fig. 6) met het indrukken van de toets "REG", zoals onder a) is weergegeven. Vervolgens wordt toets 4 ingedrukt b). Op het scherm verschijnt nu .r4=A8. De inhoud A8 is, evenals in de vorige figuur, slechts een voorbeeld. Nu dienen we de inhoud van dit register (register 1 bank 1) te wijzigen. Dit gebeurt b.v. door het indrukken van de toets E, te weten het meest significante hexadecimale cijfer dat we in het register wensen te plaatsen. Dit is weergegeven onder c). Het scherm vertoont nu .r4=E.

De inhoud van het register is naar links geschoven en op de rechtse plaats op het scherm is de E verschenen. Als dan

de volgende hexadecimale digit wordt ingevoerd, en wel door toets 2 in te drukken, dan verschijnt op het scherm .r4=E2. Nu is de inhoud van register 1 bank 1 gelijk aan H'E2'. Door nu weer "ENT NXT" in te drukken, wordt de inhoud van R4 bevestigd en definitief gemaakt. Het scherm vertoont nu automatisch de inhoud van het volgende register, in ons geval .r5=B6 De inhoud hiervan kunnen we op overeenkomstige wijze veranderen.

Speciale aandacht wordt gevestigd op de mogelijkheid om de 2 bytes PSW U EN PSW L van het Program Status Word op deze wijze te wijzigen.

Verricht de boven omschreven handelingen zelf op de INSTRUCTOR 50!

Verricht oefening 3!

Verricht oefening 4!

Lezen en veranderen van geheugencellen

Functie

Door de "MEM" toets in te drukken, wordt toegang tot het geheugen verkregen. Bij het bedienen van deze toets (zie fig. 7 a) verschijnt op het scherm .Ad.=, d.w.z. de INSTRUCTOR 50 is bereid om het adres via het hexadecimale veld in ontvangst te nemen. Dit adres verwijst naar een cel in het geheugen, die 1 byte groot is. De bytes worden zoals bekend door twee hexadecimale cijfers weergegeven. Nu kan het meest significante hexadecimale cijfer ingevoerd worden. Indien deze (meest significante) cijfers 0 zijn, dan kan men ze eenvoudig weglaten.

Procedure

In figuur 7 wordt de geheugencel H'0042' opgeroepen. Daar er twee leidende nullen zijn, is het niet nodig deze aan te slaan en kunnen we als eerste cijfer met de 4 volstaan. Dit is weergegeven onder b). Vervolgens kan de minst significante digit, de 2, ingevoerd worden, als onder c) is weergegeven. Door de toets "ENT NXT" d) in te drukken, toont het scherm het volledige adres, voorafgegaan door een punt en de inhoud van de cel die met dit adres correspondeert. De inhoud van de cel is hier aangenomen als H'A5'. Door "ENT NXT" weer in te drukken, verschijnt het adres en de inhoud van de volgende geheugencel op het scherm, hier weergegeven als .0043 B7, zie e). Zo kan men doorgaan met het verifiëren van de inhoud van de geheugencellen.

Wenst men de inhoud van de geheugencellen niet alleen na te gaan maar ook te wijzigen, dan dient deze wijziging verricht te worden voordat "ENT NXT" ingedrukt wordt. Het begin van het wijzigen van de geheugencelinhoud is analoog aan die van het lezen ervan. In fig. 8 zijn de nodige stappen weergegeven.

De LAAD PROCEDURE begint op dezelfde wijze als het lezen, namelijk door het indrukken van de toets "MEM", zie a). Op het scherm verschijnt .Ad.=, hetgeen betekent dat de INSTRUCTOR 50 om een adres voor de selectie van een geheugencel vraagt. Stel dat weer geheugencel H'0042' wordt verlangd. Nu wordt toets 4 ingedrukt, zoals onder b) is weergegeven. Vervolgens kan, zoals vermeld onder c), de 2 ingevoerd worden. Door indrukken van "ENT NXT" worden deze aangeslagen gegevens in het geheugen opgenomen, zoals onder d) is weergegeven. De inhoud van de geheugencel is b.v. H'A5'. Laten we aannemen dat de inhoud van deze cel gewijzigd moet worden in H'64'. Het meest significante hexadecimale cijfer wordt nu aangeslagen, namelijk de 6, zie e). Vervolgens wordt de 4 aangeslagen, zie f). Heeft men een fout gemaakt, dan kan men deze eenvoudig herstellen door opnieuw de 6 en de 4 in te drukken. Men kan daarmee, indien men zich weer vergist, zolang doorgaan tot het scherm de juiste hexadecimale waarde H'64' toont. Om te bevestigen dat dit de definitieve waarde is die men in de geheugencel wenst te registreren, drukt men de toets "ENT NXT" in, zie g). Zodra men deze handeling heeft uitgevoerd, wordt het nummer van de volgende geheugencel en de inhoud van deze cel op het scherm zichtbaar gemaakt. Wenst men de byte van deze cel te veranderen, dan kan men dit volgens de hiervoor beschreven procedure verrichten.

Als men een correctie in het geheugen wenst uit te voeren, kan men dit dus eenvoudig doen door de toets "MEM" in te drukken, het adres te verstrekken, "ENT NXT" in te drukken, vervolgens de byte in hexadecimale code te verstrekken. Vervolgens kan men door "ENT NXT" weer in te drukken, bevestigen dat dit de juiste informatie is.

Verricht de boven omschreven handelingen zelf op de INSTRUCTOR 50!

Verricht de oefeningen 5, 6 en 7!

Snel laden van het geheugen

Functie

Als men veel informatie in het geheugen moet laden op opeenvolgende plaatsen, dan zou men telkens de "ENT NXT" toets kunnen indrukken. Het laden van het geheugen kan echter sneller geschieden zonder dat deze toets telkens wordt ingedrukt. Men moet daarbij wel bedenken dat het dan niet mogelijk is, tijdens het intoetsen gemaakte fouten te corrigeren. Men moet dan de snelle laadmethode onderbreken en voor de correctie terugvallen op de werkwijze zoals hiervoor omschreven.

Procedure

Het snelle laden wordt ingeleid door de toets "REG" in te drukken, zie fig. 9 a). Het scherm vertoont nu r=. De INSTRUCTOR 50 verlangt nu een hexadecimaal cijfer. Voor het snelle laden moet men nu de F-toets van het hexadecimale veld indrukken. Dit is een hexadecimaal cijfer. De machine reageert hierop door op het scherm .Ad.= zichtbaar te maken, zie b). Dit betekent dat de INSTRUCTOR 50 om een adres vraagt. Stel dat we het laden willen beginnen op het hexadecimale adres H'0012'. Om dit adres in te voeren, slaat men eerst de 1 aan, zie c), vervolgens de

2, zie d), en daarna ENT NXT, zie e). De machine reageert op deze serie handelingen door op het scherm .0012 zichtbaar te maken. De INSTRUCTOR 50 duidt daarmee aan dat het snelle laden begint in geheugencel H'0012', en wegens de uiterst links geplaatste punt dat men deze gegevens nu direct kan laden. Men kan nu beginnen met b.v. op deze plaats H'A1' te laden, zie f) en g). Slaat men nu vervolgens een F en een E aan, zie h) en i), dan wordt bij het aanslaan van de F het volgende adres aangegeven, namelijk .0013 en de aangeslagen F. Nadat de E aangeslagen is, verschijnt op het scherm het beeld .0013 FE, d.w.z. dat de volgende geheugenplaats automatisch wordt gekozen. Men bespaart zich aldus het indrukken van de "ENT NXT"-toets. Men kan hiermee doorgaan tot men alle data aan het geheugen heeft toevertrouwd. Tot slot kan men, zie j), de toets "MEM" indrukken, waarop de INSTRUCTOR 50 reageert met .Ad.=. Van dit moment af kan men eventueel gemaakte fouten corrigeren volgens de methode die hiervoor is aangegeven. Men kan het snelle laden desgewenst beëindigen door elke andere besturingstoets in te drukken, zoals b.v. "MON" of "REG".

Verschijnt tijdens het aanslaan op het scherm het woord "ERROR 3", dan duidt dit erop dat een fout gemaakt werd in de functie van de INSTRUCTOR 50. Het kan zijn, dat in dat geval het geheugen de informatie niet of foutief heeft geregistreerd. Men dient dan de toets "MEM" in te drukken om het gewenste adres opnieuw te verschaffen en de data in te voeren. Ook kan men in plaats hiervan weer de toets "REG" indrukken, vervolgens toets F, het adres, correcte data, ENT NXT, enz.

P.S. Snel laden kan ook geschieden door de informatie afkomstig van een cassetteband over te dragen.

Verricht deze handelingen zelf op de INSTRUCTOR 50!

Verricht de oefeningen 8 en 9!

Het zichtbaar maken en veranderen van de inhoud van het instructie-adresregister

Functie

Voor het elimineren van fouten is het van belang dat men de uitvoering van een programma kan onderbreken en dat men kan bepalen tot waar een programma op een bepaald moment is gekomen. Het instructie-adresregister wijst naar het adres van de *eerstvolgende* instructie die uitgevoerd zal worden. Door dit adres zichtbaar te maken, ziet men welk punt de uitvoering van het programma heeft bereikt.

Procedure

Men bepaalt dit door eerst op de toets "REG" te drukken; de INSTRUCTOR 50 toont dan op het scherm r=, zie fig. 10 a). De INSTRUCTOR 50 verwacht nu een hexadecimaal cijfer. Kiest men voor dit hexadecimale cijfer een C, dan is dit voor de INSTRUCTOR 50 een teken dat men de inhoud verlangt van het instructie-adresregister (*P*rogram *C*ounter, afgekort PC). Stel dat op dit moment het programma zo ver gevorderd is dat de inhoud van het instructie-adresregister H'0017' is. De INSTRUCTOR 50 reageert nu, zie b), door op het scherm .PC=0017 te tonen. Is dit voldoende, dan kan men eenvoudig op een van de andere commandotoetsen drukken. Als men echter de inhoud van de PC wil wijzigen, dan kan men nu de nieuwe inhoud presenteren. Stel men wenst de inhoud van het instructie-adresregister te wijzigen in H'0028'. Daar men de leidende nullen niet hoeft in te voeren, kan men volstaan met het aanslaan van 2 en 8, zie c) en d). In totaal kunnen vier hexadecimale digits in het instructie-adresregister worden geplaatst. Heeft men een fout gemaakt, dan kan men doorgaan met aanslaan tot de gewenste inhoud van het instructie-adresregister op het scherm verschijnt. Nu bevestigt men de juiste inhoud die definitief in het register moet worden geplaatst, door het indrukken van de toets "ENT NXT", zie e). Daar de INSTRUCTOR 50 aan het begin van deze serie handelingen eerst het indrukken van de toets "REG" heeft waargenomen, zal nu op het scherm r= verschijnen.

Verricht deze handelingen weer op de INSTRUCTOR 50!

Verricht de oefeningen 10 en 11!

Stap-voor-stap functie

Functie

Nadat men een programma heeft geladen, dan is het voor het opsporen van fouten nuttig om het programma instructie-na-instructie te doorlopen en zodoende telkens te kunnen nagaan of het resultaat aan de verwachtingen voldoet.

Procedure

Heeft men met behulp van het invoegen van een bepaald startpunt in het instructie-adres register het begin van het programma aan de INSTRUCTOR 50 meegedeeld, dan kan men met behulp van de "STEP"-toets het programma instructie-na-instructie doorlopen, zie fig. 11. Wenst men dit te doen, dan dient het instructie-adresregister een bepaalde waarde te bezitten, van waar uit men wil beginnen. Men drukt daartoe eerst op de toets "REG", zie a), waarop de INSTRUCTOR 50 reageert door op het scherm r= te tonen.

Vervolgens slaat men C aan, waarop de machine reageert door het in het instructie-adresregister aanwezige adres te laten verschijnen. In figuur 11 b) vertoont het scherm .PC=0017, aangevend dat de stand van het instructie-adresregister H'0017' is. Nu kan men het nieuwe adres invoeren. Als men in positie 0 wil beginnen, dan dient men een 0 aan te slaan. Hierdoor wordt het instructie-adresregister op 0 gezet. Wil men op een ander punt beginnen, dan dient men dat adres eerst in het instructie-adresregister te laden. Stel dat dit adres moet luiden: H'0122'. De leidende 0 behoeft men niet aan te slaan. Als eerste wordt dus een 1 ingevoerd, zie c) en vervolgens twee 2'en, zie d) en e). Om te bewerkstelligen dat het instructie-adresregister nu deze waarde aanneemt, drukt men de toets "ENT NXT" in. De instructie

die thans zal worden uitgevoerd bij het bedienen van de toets "STEP" is gelegen op plaats H'0122' van het geheugen. Als men de "STEP"-toets indrukt g), dan verschijnt het adres van de eerstvolgende instructie die uitgevoerd zal worden. Nemen we aan dat de instructie op H'0122' 2 bytes groot is, dan zal de INSTRUCTOR 50 nu 0124 als adres geven. Tevens wordt de operatiecode op het betreffende adres getoond. In fig. 11 g) is dit aangegeven als 0124 CA. CA is de instructiecode voor STORE RELATIVE REGISTER 2. Drukt men de "STEP"-toets weer in, dan zal het scherm het adres van de volgende instructie weergeven met de bijbehorende instructiecode. Dit zal dus zijn 0126 CC. Zo kan men doorgaan met het doorlopen van het gehele programma. Men kan met deze STEP-functie ophouden door op de toets "RUN" te drukken, waarna het programma geheel zelfstandig verder wordt uitgevoerd. Na elke stap is het mogelijk om de volgende details te verifiëren:

1. alle registers
2. het instructie-adresregister
3. het Program Status Word UPPER
4. het Program Status Word LOWER
5. elke geheugenplaats

Op deze wijze kan men na elke stap, d.w.z. na elke instructie die wordt uitgevoerd, vrijwel elk onderdeel van de machine op zijn inhoud onderzoeken en nagaan of dit aan de opzet voldoet. Bovendien kan men voor elke "STEP" controleren of de ingebrachte waarde inderdaad overeenkomt met die welke men heeft willen invoeren. Als men in plaats van de "STEP"-toets de "RUN"-toets indrukt, dan heeft dit tot gevolg dat het volledige programma wordt uitgevoerd.

Opmerking

Raadpleegt men het geheugen, de registers of het instructie-adresregister, dan kan men ook de inhoud hiervan gelijktijdig veranderen, een en ander volgens de hiervoor omschreven procedures.

Het aanbrengen van een BREEKPUNT in het programma

Functie

De zojuist beschreven STEP-methode is bijzonder waardevol als men inderdaad in staat is om het programma stap-voor-stap te doorlopen. Het kan echter voorkomen dat in het programma een lus aanwezig is, die verscheiden honderden malen achter elkaar wordt gebruikt. Bestaat deze lus bovendien uit een aantal instructies, dan moet men eerst enkele duizenden instructies uitvoeren voordat men uit de lus kan ontsnappen. Als men de lus een maal stap-voor-stap doorloopt, dan heeft men zijn functie reeds gecontroleerd en zou men het eigenlijk aan de INSTRUCTOR 50 willen overlaten om de overige lussen zelfstandig uit te voeren. Na afloop daarvan kan dan b.v. nog een instructie worden uitgevoerd. Vervolgens zou men dan de machine willen laten stoppen om daarna de rest van het programma weer stap-voor-stap te doorlopen.

Procedure

Deze faciliteit is aanwezig; hiertoe maakt men gebruik van het z.g. BREEKPUNT. Het breekpunt is een adres, dat men aan de INSTRUCTOR 50 moet verstrekken. Dit adres geeft aan waar de INSTRUCTOR 50 moet stoppen bij de uitvoering van een programma. Zodra dit adres door de INSTRUCTOR 50 bereikt wordt, zal de uitvoering van een programma ophouden en het adres met het bijbehorende operatiecodedeel op het scherm verschijnen. Zodra dit geschiedt, kan men dat waarnemen door het verschijnen van een – (streepje) vóór het adres, gevolgd door de instructiecode (–0154 C0) zoals in fig. 12 h) is aangegeven. Als men nu de"STEP"-toets indrukt, zal het programma weer stap-voor-stap uitgevoerd worden, terwijl het indrukken van de "RUN"-toets tot gevolg zal hebben dat het programma ononderbroken wordt uitgevoerd. Om een breekpunt te verwijderen, dient men de toets "BKPT" weer in te drukken, waarop de machine reageert met het tonen van .b.P=0154, te weten het adres dat zojuist werd ingevoegd, zie f). Drukt men thans de toets "BKPT" weer in, dan zal de machine hierop reageren door het tonen van .b.P=, echter zonder een adres weer te geven, zie g). Het breekpunt is nu verwijderd en men kan dus niet meer onderweg stoppen, tenzij men volgens de bovenstaande procedure een nieuw breekpunt invoegt.

Het breekpunt werkt uitsluitend indien men de "RUN"-toets heeft ingedrukt. Voert men een programma stap-voor-stap uit, dan wordt het breekpunt niet in acht genomen.

Meldingen van fouten bij het gebruik van een breekpunt

Als men het breekpunt heeft gespecificeerd, dan kan de INSTRUCTOR 50 reageren door één van de onderstaande fouten te melden:

ERROR 1. Dit betekent dat men een breekpunt wenst te specificeren in het ROM-gedeelte van het geheugen dat voor de INSTRUCTOR 50 zelf is gereserveerd. Om deze fout te elimineren kan men ermee volstaan, de toets BKPT weer in te drukken.

ERROR 2. Deze melding betekent dat men een nieuw breekpunt tracht te introduceren, direct na het inbrengen van een breekpunt door de "ENT NXT"-toets in te drukken. Men kan deze fout herstellen door een willekeurige functietoets in te drukken.

RUN-toets

Functie

Het indrukken van deze toets beëindigt de MONITOR MODE en heeft tot gevolg dat de uitvoering van het programma begint. De eerste instructie die wordt uitgevoerd, is die welke gespecificeerd is door het instructie-adresregister.

Procedure

De uitvoering blijft doorgaan totdat één van de onderstaande oorzaken het programma tot stoppen dwingt:

1. Een breekpunt is bereikt.
2. De "RST" of "MON"-toets wordt ingedrukt.
3. In het programma is een HALT- of een WRTC-instructie aanwezig. Deze WRTC-instructie wordt gebruikt door de MONITOR voor de bediening van de poorten en schakelaars op het frontpaneel.

RESET-toets

Functie

Als men de "RST" (RESET)-toets indrukt, wordt de activiteit van de INSTRUCTOR 50 onmiddellijk beëindigd en zal de processor opnieuw starten, uitgaande van de instructie op het adres H'000'. De initiële waarden van de registers zijn op dat moment onbekend.

P.S.: Indien een breekpunt is gespecificeerd, dan zal dit niet worden uitgevoerd.

Procedure

De programma-uitvoering gaat voort tot één van de onderstaande toestanden zich voordoet:

1. De RST-toets wordt opnieuw gedrukt.
2. Een HALT-instructie wordt uitgevoerd. Bij de uitvoering van deze HALT-instructie stopt de processor tot de RST-toets opnieuw wordt ingedrukt, of indien de INTERRUPT INHIBIT PSW-bit niet is gezet tot er een INTERRUPT verschijnt.
3. Een WRTC-instructie wordt uitgevoerd of de "MON" toets wordt ingedrukt. In dat geval wordt de besturing overgedragen aan het MONITOR-programma; op het scherm verschijnt dan het bericht "HELLO".
   Als de controle wordt overgedragen aan de MONITOR, dan wordt het adres van de laatste handeling geregistreerd in het instructie-adresregister en worden de registerwaarden gered in het RAM-deel van de MONITOR. Ze kunnen nu met behulp van de desbetreffende stuurtoetsen onderzocht worden.
4. Indien logische waarde van de PAUZE-ingang hoog is, zal de processor eveneens stoppen. Deze ingang bevindt zich aan de achterzijde (stekeraansluiting). Als dit gebeurt, dooft het RUN-licht. De uitvoering van het programma zal weer beginnen als de logische waarde van de PAUZE-ingang laag wordt.

Fout-meldingen

De MONITOR kan een aantal interne testen uitvoeren. Naast deze interne testen onderzoekt hij bovendien of bepaalde opdrachten wel toegestaan zijn. Treedt een fout op, hetzij in de machine, dan wel in de opdrachten, dan reageert de INSTRUCTOR 50 hierop door op het scherm de gedaante ERROR N te vermelden, waarbij N een cijfer kan zijn van 1 t/m 9. De betekenis van deze meldingen is als volgt:

ERROR 1. *Het breekpunt kan niet ingevoerd worden.* Dit is meestal het geval als men tracht een breekpunt aan te geven liggend in het gedeelte van het geheugen dat voor het MONITOR-programma is gereserveerd.

ERROR 2. *Niet toegestane opdracht* Dit bericht betekent dat de volgorde van de commando's niet juist is en dat men deze moet verbeteren. Dit kan o.a. gebeuren door de commandocyclus te beëindigen, de knop MONITOR in te drukken, enz.

ERROR 3. *Geheugeninhoud is niet op juiste wijze veranderd.* Dit bericht betekent dat er een poging is gedaan om data te schrijven in een geheugencel die b.v. niet toegankelijk is. Als men bij het invoeren van data dit bericht krijgt, dan kan het betekenen dat een fout is vastgesteld bij de controle of de data juist in het geheugen is geladen. Is dit het geval, dan moet men de toets "MEM" opnieuw indrukken en het laatste deel van de handelingen herhalen.

ERROR 4. *Cassette-controle fout.* Als data op een cassetterecorder wordt geschreven, dan wordt aan deze data aan het einde een controleteken toegevoegd; aldus kan men later bij het teruglezen van deze data nagaan, of er een fout is opgetreden. Als nu bij het teruglezen van de cassette wordt vastgesteld dat de gelezen informatie niet juist is, dan verschijnt de melding ERROR 4.
Men kan de cassette dan opnieuw laten inlezen. Indien het een tijdelijke fout was, b.v. stof, dan is het mogelijk dat de informatie nu foutloos ingelezen wordt. Is de band echter beschadigd, dan zal de informatie verloren zijn gegaan en is het desbetreffende deel van de band niet meer bruikbaar.

ERROR 5. *Een geheugenschrijffout tijdens het inlezen van een cassette.* Bij het inlezen van een cassette en het inschrijven in het geheugen van de INSTRUCTOR 50, worden de geregistreerde bytes gecontroleerd. Indien een fout optreedt, dan zal de INSTRUCTOR 50 ERROR 5 vermelden. Men dient de band dan opnieuw in te lezen.

ERROR 6. *De code van de magneetband wijkt af van de ASCII code.* De INSTRUCTOR 50 schrijft op de magnetische band in de z.g. ASCII code. Is een andere code op de band aanwezig, dan zal dit op een gegeven ogenblik blijken en gemeld worden. Voorts is het mogelijk dat de volumeregelaar van de cassetterecorder onjuist staat afgesteld. Men dient dit dan te verifiëren en de cassette, zo dit nodig is, opnieuw in te lezen.

ERROR 7. *STARTadres groter dan STOPadres.* Als men data op een cassette schrijft en men geeft een startadres aan dat groter is dan het stopadres, dan zal dit door middel van deze indicatie worden gemeld.

ERROR 8. *Er zijn twee toetsen in een kolom ingedrukt.* Dit betekent dat er twee toetsen tegelijk werden ingedrukt. De MONITOR kan dan niet vaststellen welke actie gewenst is.

ERROR 9. *Volgende instructie bevindt zich in het MONITOR-gebied.* Als de STEP-toets wordt ingedrukt en de volgende instructie zou in het MONITOR-gebied liggen, dan wordt dit gemeld door het verschijnen van de melding ERROR 9.

HET GEBRUIK VAN DE INSTRUCTOR 50

De schrijver van programma's die de INSTRUCTOR 50 wil gebruiken, heeft de volledige 2650 microprocessor-instructieverzameling tot zijn beschikking. Wegens de wisselwerking tussen de INSTRUCTOR 50 en de hardware en software van de gebruiker, moeten daarbij enkele beperkingen in acht worden genomen:

1. De MONITOR reserveert de instructie WRTC (Write Control). Tevens gebruikt hij de registerinstructies H'B0', H'B1', H'B2' en H'B3' om de plaats van een BREEKPUNT in een programma aan te geven. Als een dergelijke instructie wordt uitgevoerd in een programma van een gebruiker, dan zal de INSTRUCTOR 50 zelf terugkeren naar de MONITOR-mode en zal op het scherm het woord 'HELLO' verschijnen.
2. Indien een HALT-instructie (H'40') wordt uitgevoerd, dan zal de werking van de processor worden beëindigd. Dit wordt aangegeven door het doven van de LED "RUN". Er zijn twee wegen om de uitvoering opnieuw te starten. Dit kan geschieden door de RST-toets in te drukken, of – als interrupts niet zijn verboden – door het verrichten van het INTERRUPT. Dit kan b.v. worden bewerkstelligd door het indrukken van de toets INT.
   Als er een BREEKPUNT is gedefinieerd op de plaats van een HALT-instructie, dan zal de MONITOR ervoor zorgen dat deze HALT-instructie niet wordt uitgevoerd; de normale uitvoering van het programma zal dan doorgaan.
3. Van pagina 0 is, zoals bekend uit de instructiehandleiding, het onderste deel van deze pagina, evenals het bovenste deel, in wezen gereserveerd voor de ZBSR-instructies met positieve resp. negatieve displacements. Het MONITOR-programma gebruikt echter het bovenste deel van de pagina voor het springen naar MONITOR-programma's, zodat de gebruiker dit deel van de pagina niet moet gebruiken, tenzij hij de MONITOR wil binnengaan. Hetzelfde geldt voor INTERRUPT-vectors met negatieve displacements.
4. De MONITOR gebruikt voor zijn programma drie niveau's van de 2650-subroutine terugkeeradresstapel. Daar de terugkeeradresstapel is beperkt tot 8 niveau's, kan de gebruiker slechts over vijf niveau's van deze stapel beschikken voor het schrijven van zijn programma. Men dient zich te realiseren, dat tot deze 8 niveau's ook de INTERRUPTS behoren.

Het gebruik van INTERRUPTS

INTERRUPTS voorzien in een synchronisatie van onderling onafhankelijke processen, die echter op een bepaald ogenblik gegevens moeten uitwisselen. Als een extern proces, d.w.z. een proces buiten de microprocessor 2650, een bepaald stadium heeft bereikt, dan kan dit proces de aandacht van deze microprocessor vragen. Deze zal hierop reageren door naar een bepaalde serviceroutine te springen. Deze serviceroutine kan hetzij specifiek voor het interrumperende externe proces zijn, waarbij dit proces dan hetzij zijn identiteit moet meedelen (vector interrupt), dan wel een routine kan zijn; in het laatste geval gaat de microprocessor op zoek naar het externe proces dat de INTERRUPT veroorzaakt (z.g. polling). De identiteit van het externe proces dat interrumpeert, slaat niet zo zeer op het nummer van het betrokken proces, als wel op de fase waarin het proces zich bevindt. De INTERRUPT-vector geeft namelijk de plaats aan waar de vereiste gegevens voor de specifieke subroutine voor dit deel van het proces in het geheugen behorende bij de 2650 microprocessor aanwezig zijn. Het kan dus voorkomen dat één proces meer dan één INTERRUPT-vector kan genereren, die echter afhankelijk zijn van de voortgang van het betreffende proces.

De INSTRUCTOR 50 beschikt over drie methoden om de microprocessor 2650 te interrumperen.

De eerste methode is een INTERRUPT die de gebruiker van de INSTRUCTOR 50 teweeg kan brengen door de toets INT in te drukken.

De tweede methode maakt gebruik van een extern sig-

naal dat afgeleid wordt van de netfrequentie. In Nederland is deze 50 Hz. Hierdoor wordt elke 20 ms een INTERRUPT-verzoek ingediend. Met behulp van deze INTERRUPT zou men in de microprocessor 2650 een klokschakeling kunnen realiseren, in de Engelse literatuur een "real-time clock" genoemd. Wil men deze wijze van interrumperen toestaan, dan moet de INTERRUPT-schakelaar aan de onderzijde in de stand "A.C.LINE" geplaatst worden.

De derde methode van interrumperen is via de z.g. S100-bus interface. Deze S100-bus bevindt zich aan de achterzijde van de INSTRUCTOR 50 en kan m.b.v. de steker toegankelijk worden gemaakt.

Het interruptmechanisme kan onder "software control" worden in- of uitgeschakeld. Dit kan in wezen geschieden op elk willekeurig punt in het programma door de SET of RESET van het INTERRUPT INHIBIT-bit (II van het PSW).

Als het INTERRUPT INHIBIT-bit in stand 1 staat, dan accepteert de microprocessor 2650 geen verzoeken om te gaan interrumperen. Staat het INTERRUPT INHIBIT-bit daarentegen in de 0-stand, d.w.z. dat de INTERRUPT-aanvragen worden gehonoreerd, dan zal de microprocessor op de interne INTERRUPT reageren. Het INTERRUPT INHIBIT-bit zal in de volgende gevallen in de 0-stand worden gezet (toelaten van INTERRUPT-aanvragen):

1. Bij het indrukken van de RST-toets, d.w.z. het terugzetten van de processor.
2. Door het uitvoeren van de instructie CPSU (Clear Program Status Upper), welke instructie voorzien dient te zijn van het juiste masker.
3. Bij de uitvoering van de instructie RETE (Return from Subroutine and Enable Interrupt).
4. Bij het uitvoeren van de instructie LPSU (Load Program Status Upper), waarbij de II-bit in de 0-stand geplaatst wordt.

Het INTERRUPT-mechanisme van de microprocessor 2650 kan met een vectored INTERRUPT werken. Accepteert de microprocessor een "interrupt request" (INTREQ), dan verstrekt de microprocessor automatisch een "interrupt acknowledge" (INTACK)-signaal. Het proces dat de microprocessor 2650 interrumpeert, antwoordt hierop door het plaatsen van de INTERRUPT-vector op de 2650 data-bus. Deze vector wordt gebruikt als een adres relatief t.o.v. H'0000' op pagina 0, of als een sprong naar een subroutine. De INTERRUPT-vector kan een direct of een indirect adres afgeven aan de data-bus. Bij directe adressering start die subroutine op het aangegeven adres, d.w.z. in de eerste 64 bytes van pagina 0, hetgeen bij een enkel proces geoorloofd kan zijn. Indien meer processen zullen interrumperen, dan kunnen de 64 bytes niet voor het schrijven van een subroutine gebruikt worden en dienen ze elders in het geheugen te worden ondergebracht. Voor dat doel maakt men gebruik van het indirecte adresseren.

De geadresseerde byte op pagina 0 (in het gebied 0 t/m 63) is nu de eerste byte van een tweetal waar het adres van de subroutine zich bevindt. Op grond hiervan wordt een sprong gemaakt naar de desbetreffende subroutine. Deze kan ergens binnen de totale adresseerbare capaciteit van 32 K liggen.

Als het toegestaan was (II=0) te interrumperen, dan beantwoordt de INSTRUCTOR 50 een verzoek tot interruptie door de volgende serie handelingen:

1. Hij voltooit de instructie die op dat moment in uitvoering is.
2. De processor plaatst de INTERRUPT INHIBIT in de logische 1 (II=1).
3. De eerste byte van de ZBSR-instructie wordt door de microprocessor 2650 intern opgewekt en in het instructie-register geplaatst.
4. De processor geeft een INTERRUPT ACKNOWLEDGE (INTACK) uit en wacht op de INTERRUPT-vector, die door het interrumperende proces moet worden afgegeven op de data-bus.
5. De INSTRUCTOR 50 genereert zelf een INTERRUPT-vector, te weten H'07' of H'87' op de data-bus. De 7 duidt aan dat byte 7 van het geheugen wordt gebruikt als eerste byte voor het uitvoeren van de subroutine voor de INTERRUPT. De 0 of de 8 duidt op een directe (H'07') of indirecte (H'87') adressering. Welke van de beide wordt gekozen, hangt af van de stand van de schakelaar DIRECT/INDIRECT INTERRUPT op het bedieningspaneel van de INSTRUCTOR 50. De gebruiker van de INSTRUCTOR 50 dient ervoor te zorgen dat, ingeval de directe adressering (H'07') wordt gebruikt, de subroutine op deze plaats start. Ingeval dat van indirecte adressering gebruik wordt gemaakt, dient op de geheugenplaatsen 7 en 8 het absolute adres aanwezig te zijn van het adres waarop de subroutine elders in het geheugen begint.
6. De 2650 voert vervolgens de ZBSR-instructie uit, hetgeen tot gevolg heeft dat van plaats 7 af het INTERRUPT-programma wordt uitgevoerd, resp. dat er een sprong wordt gemaakt naar de subroutine elders in het geheugen. Het adres van de volgende instructie van het oorspronkelijke programma is in de adresstapel dan gered en de stapelwijzer wordt verhoogd.
7. Wanneer de INTERRUPT-routine in haar geheel is uitgevoerd, dan zal de laatste instructie zijn een RETE- of een RETC-instructie. Als gevolg hiervan verlaagt de microprocessor 2650 de stapelwijzer en vervangt hij de inhoud van het instructie-adresregister door de inhoud van de top van de stapel, zodat de uitvoering van het oorspronkelijke programma nu kan worden voortgezet.

Een externe INTERRUPT wordt gegenereerd door een onafhankelijk proces buiten de microprocessor 2650. Het is dus niet te voorspellen op welk moment deze INTERRUPT zal ontstaan. Een gevolg hiervan is dat een INTERRUPT kan plaats hebben bij elke willekeurige instructie van het

programma dat de microprocessor 2650 aan het uitvoeren is. Het programma kan evenwel een aantal registers in gebruik hebben. De INTERRUPT-subroutine kan echter ook behoefte hebben aan het gebruik van registers. Dit heeft tot gevolg dat de inhoud van deze registers op de een of andere wijze gered dient te worden. Dit kan worden gedaan op verschillende wijzen.

De eerste is ervoor zorg te dragen dat INTERRUPTS uitsluitend gedurende bepaalde instructies kunnen voorkomen. Zo kan men in het programma b.v. ervoor zorg dragen dat het bit II=1 is. Op een bepaald ogenblik kan men door middel van een instructie het bit II=0 maken en na de volgende instructie wederom II=1. Gedurende deze korte tijd kan een externe INTERRUPT het programma dan interrumperen. Voordat men echter deze interruptiemogelijkheid toestaat, kan men de inhoud van de registers in het geheugen redden. Na afloop van de subroutine dient het programma ervoor zorg te dragen dat de registers wederom geladen worden met de in het geheugen geredde informatie. De tweede methode is het redden van registers bij het betreden van de INTERRUPT-routine. Aan het einde van deze routine kan de geredde informatie dan . ederom in de registers geplaatst worden. Een derde mogelijkheid is ervoor te zorgen dat in het hoofdprogramma uitsluitend gebruik gemaakt wordt van de registers in bank 0, terwijl bij de INTERRUPT-routine gebruik gemaakt wordt van de registers in bank 1. Een bijzondere voorziening blijft echter noodzakelijk voor Registers 0 en Program Status Word Lower. De inhoud van deze registers dient altijd nog gered te worden voor zover deze voor het hoofdprogramma belangrijke informatie bevat.

De INSTRUCTOR 50 INTERRUPT-faciliteit kan worden gekozen door de combinatie van de stand van de INTERRUPT-schakelaar en/of een verbinding aan de onderzijde van de INSTRUCTOR 50. Zoals reeds vermeld, doet de DIRECT/INDIRECT-schakelaar op het frontpaneel van de INSTRUCTOR 50 dienst om te bepalen of de INTERRUPT-vector directe, resp. indirecte adressering tot gevolg heeft. De geheugencel H'07' wordt zowel gebruikt voor de INTERRUPT die veroorzaakt wordt door het drukken van de toets INT, als wel door het INTERRUPT dat mogelijk wordt door het omschakelen van de schakelaar aan de onderzijde van de INSTRUCTOR 50, namelijk in de stand "A.C.LINE". Naast deze beide mogelijkheden van interrumperen kan de INSTRUCTOR 50, zoals reeds vermeld, ook geïnterrumpeerd worden via de ingang van de S100-bus interface. Om dit te bewerkstelligen moet een verbinding aan de onderzijde van de INSTRUCTOR 50 worden gemaakt. Er wordt echter op gewezen dat dit uitsluitend zin heeft als externe apparatuur met de INSTRUCTOR 50 verbonden wordt. Bij de aflevering van de INSTRUCTOR 50 is de onderzijde zodanig bedraad dat de INTERRUPT hetzij geschiedt via het bedieningspaneel, dan wel door de 50 Hz voeding, een en ander afhankelijk van de stand van de schakelaar INTERRUPT SELECT SWITCH (zie fig. 13). In het vervolg zullen de verbindingen die bij de aflevering aanwezig zijn, in de tekst worden aangegeven d.m.v. een *. Het kiezen van een directe of een indirecte adressering blijft echter in alle gevallen bepaald door de stand DIRECT/INDIRECT van de INTERRUPT-schakelaar op het paneel.

Doorverbindingen in veld A

verbinding	beschrijving
1 – 2 *	Normale werking. De microprocessor 2650 herkent INTERRUPT-verzoeken van de INTERRUPT-toets of van de 50 Hz INTERRUPT-bron, afhankelijk van de stand van de INTERRUPT SELECT SWITCH.
2 – 4	Bus interface. De 2650 herkent het interrumperen van de interface (pen 4 op de S100-bus steker). Het herkennen van de INTERRUPT wordt gedaan gedurende de wisseling van laag naar hoog van het ingangssignaal.
2 – 39 3 – 4	Bus interface geïnverteerd. De situatie is identiek aan die geschetst onder 2 – 4, evenwel met de uitzondering dat de INTERRUPT wordt herkend bij de overgang van hoog naar laag van het ingangssignaal.

Het gebruik van de I/O schakelaars en LED's

De microprocessor 2650 kent een aantal methoden om de toestand en besturing van externe in- en uitvoerapparatuur vast te leggen. Eén van de methoden is de SERIËLE INPUT-poort, gevormd door de SENSE INPUT-pen en de SERIËLE OUTPUT-poort, die FLAG wordt genoemd. Voorts heeft de microprocessor 2650 voorzieningen voor 2 soorten parallelle in- en uitvoerinstructies, die men EXTENDED en NON-EXTENDED I/O noemt. De NON–EXTENDED I/O instructies zijn 1 byte groot en maken het mogelijk om data naar en uit twee 8 bit I/O-poorten te schrijven en te lezen, namelijk poort C en poort D. Deze beide poorten zijn via de data-bus en de controle-bus bereikbaar. De extended I/O-instructies zijn 2 bytes groot en kunnen de I/O capaciteiten van de microprocessor 2650 uitbreiden tot 256 bidirectionele I/O-poorten.

Behalve de genoemde I/O-instructies kan men ook een andere wijze van in- en uitvoer kiezen, de z.g. "MAPPED I/O". Deze methode wordt gerealiseerd door een geheugenadres aan te wijzen als een I/O-poort. In dat geval kan dus van de volledige geheugencapaciteit gebruik worden gemaakt. De I/O wordt echter wel benaderd vanuit het stand-

punt alsof het een geheugencel is, d.w.z. het wordt benaderd via de controlelijn, die onderscheid maakt tussen datastroom naar en van geheugen, resp. I/O. Bij "MAPPED I/O" wordt verondersteld dat deze I/O tot het geheugen behoort. Zie hiervoor verder de beschrijving van de microprocessor 2650 in een ander deel van deze handleidingen.

FLAG- en SENSE I/O-methodiek

De FLAG-output van de 2650 geeft in feite de toestand weer van een bit van het Program Status Word (PSW). Dit bit kan geschakeld worden met een PSW controleinstructie waaraan een masker is toegevoegd. Zodra de flip-flop in het PSW in de 1 staat, wordt de outputpen van de 2650 microprocessor hoog en als hij in de 0 staat, wordt deze pen laag. Op deze pen is een LED aangesloten, die op het bedieningspaneel zichtbaar is. Als deze LED licht geeft, dan staat de flip-flop in het PSW in de logische 1, als deze LED gedoofd is, dan bevindt zich de flip-flop in de logische 0. Seriële input wordt verkregen door middel van de SENSE-pen van de microprocessor. Deze pen is direct verbonden met een toets op het bedieningspaneel van de INSTRUCTOR 50, de SENSE-toets. Normaal is de spanning op de SENSE-pen laag. Zodra echter de SENSE-toets ingedrukt wordt, dan zal de spanning op een hoog niveau komen, d.w.z. een logische 1 worden.

NON-EXTENDED I/O-methodiek

De microprocessor 2650 kan twee externe bidirectionele I/O-poorten besturen m.b.v. vier verschillende 1 byte-instructies. De instructies zijn: WRTC, WRTD, REDC en REDD. Deze instructies bewerkstelligen de overdracht van één byte naar of van de poorten. De plaats van oorsprong, resp. van bestemming, is één van de zeven aanwezige registers. Het nummer van het register komt tot uitdrukking in de operatiecode, terwijl de bank van het betreffende register, zoals bekend, vooraf d.m.v. een instructie voor het PSL gedefinieerd dient te zijn. Het register R0 behoeft niet d.m.v. een dergelijke instructie gedefinieerd te worden, daar R0 gemeenschappelijk is voor de beide registerbanken. Eén van deze NON-EXTENDED poorten, en wel poort D, bevindt zich op het bedieningspaneel van de INSTRUCTOR 50. Zet men de poortadresschakelaar in de stand NON-EXTENDED, dan wordt deze poort automatisch gekozen. Uiteraard dient dan wel de instructie WRTD of REDD in het programma voor te komen. Door deze faciliteit is het mogelijk om met behulp van de acht schakelaars data aan de 2650 over te dragen. Op dezelfde wijze is het mogelijk data van de 2650 te betrekken d.m.v. de acht LED's die boven de schakelaars zijn geplaatst. Het minst significante bit van de overgedragen byte bevindt zich uiterst rechts, het meest significante bit uiterst links.

EXTENDED I/O-methodiek

De microprocessor 2650 kan maximaal 256 bidirectionele I/O-poorten bereiken m.b.v. de instructies WRTE (Write Extended) en REDE (Read Extended). Het tweede byte van deze instructie specificeert de desbetreffende I/O-poort. Deze I/O-poort wordt gekozen via de adresbus en d.m.v. de controledraad M/$\overline{IO}$. De extended I/O kan in de INSTRUCTOR 50 nagebootst worden m.b.v. dezelfde acht schakelaars en acht LED's, zoals bij de non-extended I/O. Men kan dit realiseren door de I/O-schakelaar in de positie EXTENDED I/O te zetten. Het adres van deze I/O-poort is in dit geval H'07'. Dit adres dient als tweede byte in de REDE- en WRTE-instructies aanwezig te zijn.

MEMORY MAPPED I/O-methodiek

Als men in de INSTRUCTOR 50 de "MEMORY MAPPED I/O" wenst na te bootsen, dan dient men de I/O-schakelaar in de bovenste stand te zetten. Nu wordt aan de I/O-schakelaars en indicators het adres H'OFFF' toegewezen. Hierdoor is het mogelijk data met de normale STORE- of LOAD-instructie naar, resp. van de I/O-schakelaars en LED's te voeren. De LOAD- en STORE-instructies bevatten een tweetal bits, die het register aangeven van waar, resp. waarheen data moet worden gevoerd. Deze data wordt betrokken, resp. overgedragen van de acht schakelaars, resp. acht LED's op het bedieningspaneel.

Keuze klokfrequentie – S100 bus

De bus-interface aan de onderzijde van de INSTRUCTOR 50 heeft een drietal aansluitingen voor het geven van bepaalde kloksignalen naar de omliggende logica. (Zie figuur 13 Veld B). De standaardfrequentie van de INSTRUCTOR 50 is 895 kHz. Tevens kan een tweede kloksignaal aan de onderzijde worden afgenomen ter grootte van ongeveer 303 kHz. Dit signaal is afgeleid van de pen OPREQ van de microprocessor 2650. Zoals blijkt uit de handleiding hiervan, behoeft dit signaal (de Operation Request) niet bij elke instructie aanwezig te zijn. De frequentie kan dus geringer zijn dan 303 kHz. Aan de onderzijde is op de pennummers 11 en 12 het systeemkloksignaal van 895 kHz van de INSTRUCTOR 50 aanwezig. Op de pennen 13 en 14 is het signaal OPREQ aanwezig.

De uitgangen van de S100-bus zijn verbonden met de punten 8, 9 en 10. Pen 8 is verbonden met uitgang 25, bekend als het bussignaal 01, pen 9 is verbonden met de uitgang 24, bekend als bussignaal 02 en eveneens met uitgang 49, bekend als bussignaal "klok". Bij aflevering van de INSTRUCTOR 50 is pen 9 verbonden met pen 12, zodat op uitgang 24 het 895 kHz signaal staat.

Keuze van de spanningsbron

De INSTRUCTOR 50 werkt normaal met zijn eigen interne spanningsstabilisatie, een en ander via de transformator die op het lichtnet kan worden aangesloten. Het is echter ook mogelijk om de INSTRUCTOR 50 te laten werken via de interface aansluiting van een geregelde 5 V spanningsvoorziening. Voor dat doel zijn de punten 18, 19 en 20 aanwezig (Zie figuur 13 Veld C). Zijn de punten 18 – 20 doorverbonden, zoals het geval is bij de aflevering van de INSTRUCTOR 50, dan werkt de INSTRUCTOR met de eigen interne spanningsvoorziening. Worden de punten 18 en 19 doorverbonden, dan moet de externe spanningsvoorziening op de achterzijde van de INSTRUCTOR 50 worden aangelegd.

INGEBOUWDE SUBROUTINES

De INSTRUCTOR 50 heeft in zijn geheugen een zestal subroutines die bijzonder nuttig zijn voor de communicatie met deze INSTRUCTOR en voor speciale bewerkingen die in een programma nodig kunnen zijn. De subroutines zijn: MOVE, DISPLAY, USER DISPLAY, NIBBLE, INPUT DATA en MODIFY DATA.

De subroutines worden opgeroepen door een ZBSR (Zero Branch to Subroutine Relative) instructie. De instructie dient voorzien te zijn van een extra byte. Deze byte is bij de hierna te behandelen instructie aangegeven in de vorm van twee hexadecimale cijfers. De byte begint altijd met een 1, waardoor dus een indirecte adressering wordt verkregen. De indirecte adressen zijn alle in de laatste 64 bytes van pagina 0 (8K bytes groot) gelokaliseerd. Wegens de indirecte adressering doet het voor de gebruiker niet ter zake waar de subroutines in feite in het geheugen aanwezig zijn. De subroutines maken veelal gebruik van een aantal registers; soms wordt ook vereist, dat deze registers, voordat de subroutine wordt aangeroepen, gevuld zijn met bepaalde informatie. Daar bij de aanroep van een subroutine op de adresstapel een adres wordt geplaatst, dient men van te voren zorgvuldig te overwegen of het nog mogelijk is, deze subroutine aan te roepen. Als er niet voldoende plaats meer op de stapel aanwezig is om het vertrekadres te redden, mag de subroutine niet worden aangeroepen. Het aantal niveau's dat de subroutine zelf oproept, is per subroutine aangegeven. Hiermee is dus niet bedoeld het niveau dat de subroutine zelf nodig heeft om aangeroepen te worden, doch uitsluitend het aantal malen dat de subroutine zèlf weer gebruik maakt van andere subroutines. Bij de aanroep van een subroutine dient dus ten minste voor één adres plaats te zijn op de stapel, en bovendien dient men nog zo veel plaats in reserve te hebben als het aantal aanroepen die bij de subroutine zelf aangegeven zijn. De maximale stapeling is acht; overschrijdt men dit getal, dan gaat het vertrekadres verloren.

MOVE-subroutine

Functie

MOVE verplaatst een bericht ter grootte van acht bytes van een gebruikersprogramma naar dat deel van het geheugen dat het scherm van informatie voorziet. Door deze subroutine te combineren met de subroutine DISPLAY, kunnen berichten op het scherm zichtbaar worden gemaakt.

Werking

Voordat deze subroutine wordt aangeroepen, dient een bericht van acht bytes in het programma aanwezig te zijn. De plaats waar dit bericht begint, dient in de registers R1 en R2 vóór het oproepen van de subroutine te worden geregistreerd. Het adres bestaat uit twee bytes, waarvan de eerste drie bits zeker bestaan uit nullen (32K geheugen). Het meest significante byte dient in register R1 geregistreerd te worden, het minst significante in register R2.

Aanroep-instructie:

mnemonic	*hexadecimale waarde*
ZBSR *MOV	BB, FE

Gebruikte registers: R1 = meest significante byte van het adres –1
R2 = minst significante byte van het adres –1

Vereiste aantal stapelplaatsen na de oproep: 0

Wegens de beperkte omvang van het scherm kunnen niet alle 256 mogelijkheden van het bericht van een byte worden gebruikt; er is slechts een beperkte weergave mogelijk. In onderstaande tabel zijn de hexadecimale waarden met de erbij behorende tekens op het scherm weergegeven.

karakter	*waarde*	*karakter*	*waarde*
*0 of O	H'00'	*P	H'10'
*1 of I	H'01'	*L	H'11'
*2	H'02'	*U	H'12'
*3	H'03'	*R	H'13'
*4	H'04'	*H	H'14'
*5 of S	H'05'	*O	H'15'
*6 of G	H'06'	*=	H'16'
*7	H'07'	*spatie	H'17'
*8	H'08'	*J	H'18'
*9	H'09'	* _	H'19'
*A	H'0A'	*	H'1A'
*B	H'0B'	*Y	H'1B'
*C	H'0C'	*N	H'1C'
*D	H'0D'		
*E	H'0E'		
*F	H'0F'		

Voorbeeld van een MOVE-subroutine aanroep

adres	*data*	*instructie mnemonic*	*commentaar*
.			
.			
0010	05,00	LODI,R1 H'00'	Laad adreswijzer
0012	06,FF	LODI,R2 H'FF'	–1 in R1 en R2
0014	BB,FE	ZBSR *MOV	(H'0100' –1 =
.			H'00FF') Spring
.			naar MOVE' Het
.			bericht op 0100-
			-0107 wordt ver-
			plaatst naar de
			buffer van het
			scherm
0100	17		= spatie
0101	14		= H
0102	0E		= E
0103	11		= L
0104	11		= L
0105	00		= O
0106	17		= spatie
0107	17		= spatie

DISPLAY-subroutine

Functie

Als deze subroutine wordt gebruikt in samenhang met de MOVE-subroutine, is het mogelijk op het scherm van de INSTRUCTOR 50 acht karakters weer te geven. DISPLAY leest het bericht dat is geregistreerd in de buffer van het scherm, b.v. door middel van de subroutine MOVE. Vervolgens plaatst deze subroutine dit bericht in dat deel van de INSTRUCTOR 50 dat de gegevens voor het scherm verzorgt. Deze routine verzorgt ook de weergave die plaats heeft na het indrukken van de besturingstoets of een hexadecimale toets.

Werking

De subroutine DISPLAY heeft drie operatiemodes. Deze worden geselecteerd door vóór het aanroepen van de subroutine in register R0 een opdracht-byte te schrijven. De waarde van deze byte en de erbij behorende functie zijn hieronder weergegeven.

waarde	*functie*
H'00'	Geef het bericht door aan het scherm tot een datatoets of een funktietoets wordt ingedrukt. Plaats de waarde van de ingedrukte toets in register R0.
H'01'	Voer de DISPLAY-subroutine één maal uit. Bepaal niet de waarde van een eventueel ingedrukte toets. Daar het eenmalig passeren van deze subroutine niet in zichtbare tekens op het scherm resulteert, moet de subroutine deel uitmaken van een lus waarin de DISPLAY subroutine een voldoend aantal malen wordt aangeroepen om het bericht zichtbaar te maken.

H'80' Deze opdracht is identiek aan die van H'00', met die uitzondering dat nu uiterst links op het scherm een punt verschijnt.

Aanroep-instructie:

mnemonic	*hexadecimale waarde*
ZBSR *DISPLY	BB, EC

Vereiste aantal stapelplaatsen na de oproep: 0
Gebruikte registers: R0, R1, R2 en R3.

R0= DISPLAY-opdracht. Na het uitvoeren van de subroutine is R0= toetswaarde. Hieronder zijn de toetsen en de bijbehorende waarde in het register R0 na afloop van de subroutine weergegeven.

toets	*hexadecimale waarde*	*toets*	*hex. waarde*
0	H'00'	C	H'0C'
1	H'01'	D	H'0D'
2	H'02'	E	H'0E'
3	H'03'	F	H'0F'
4	H'04'	WCAS	H'80'
5	H'05'	BKPT	H'81'
6	H'06'	RCAS	H'82'
7	H'07'	REG	H'83'
8	H'08'	STEP	H'84'
9	H'09'	MEM	H'85'
A	H'0A'	RUN	H'86'
B	H'0B'	ENT/NXT	H'87'

Voorbeeld van een MOVE en DISPLAY subroutine aanroep

adres	*data*	*instructie mnemonic*	*commentaar*
0010	05,00	LODI,R1 H'00'	Plaats berichtwijzer −1 in R1
0012	06,FF	LODI,R2 H'FF'	en R2 ('0100'−1 ='00FF')
0014	BB,FE	ZBSR *MOV	Roep de MOVE-subroutine aan.
0016	04,00	LODI,R0 H'00'	Plaats de opdracht in R0.
0018	BB,EC	ZBSR *DISPLY	Roep de DISPLAY-subroutine aan
001A	E4,86	COMI,R0 H'86'	Vergelijk teruggegeven toetscode met RUN-code.
001C	1C,XX,XX	BCTA,EQ H'XXXX'	Indien gelijk, spring naar dit adres: H'XXXX'
001F	1F,00,16	BCTA,UN H'0016'	Indien ongelijk, ga naar H'0016' en wacht op volgende toets
0100	17		Eerste byte van het bericht is een spatie
0101	14		= H
0102	0E		= E
0103	11		= L
0104	11		= L
0105	00		= O
0106	17		= spatie
0107	17		Laatste byte van het bericht is een spatie.

USER DISPLAY-subroutine

Functie

De USER DISPLAY-subroutine combineert de functies van de twee besproken subroutines MOVE en DISPLAY. Dit betekent dat USER DISPLAY een bericht van acht bytes verplaatst uit het programma van de gebruiker naar het gedeelte voor de buffer van het scherm en vervolgens dit bericht op het scherm zichtbaar maakt.

Werking

Voordat de USER DISPLAY wordt opgeroepen, dient het adres van het eerste byte van het bericht −1 gegeven te worden in de registers R1 en R2, zoals dit ook gebeurt bij de subroutine MOVE. Bovendien moet in register R3 het verlangde commando worden geladen, zoals dat gebeurt in de DISPLAY-subroutine in register R0. Na afloop van de subroutine treft men in R0 wederom de waarde van de ingedrukte toets aan, zoals dat ook gebeurt bij de DISPLAY-subroutine.

Aanroep-instructie:

mnemonic	*hexadecimale waarde*
ZBSR *USRDSP	BB,E6

Gebruikte registers: R0, R1, R2, R3.

R3= DISPLAY-commando.
R1= meest significante byte van het beginadres −1.
R2= minst significante byte van het beginadres −1.
R0= toetswaarde na afloop van de subroutine.

Vereiste aantal plaatsen op de adresstapel: 2

Hieronder volgt een voorbeeld voor het gebruik van de USER DISPLAY-subroutine.

Voorbeeld van een USER DISPLAY subroutine aanroep

adres	*data*	*instructie mnemonic*	*commentaar*
0010	05,00	LODI,R1 H'00'	Plaats berichten-
0012	06,FF	LODI,R2 H'FF'	tabelwijzer −1 en R1 en R2 ('0100' −1 = '00FF)
0014	07,00	LODI,R3 H'00'	Plaats opdracht in R3
0016	BB,E6	ZBSR * USRDSP	Roep USER DISPLAY aan
0018	E4,86	COMI,R0 H'86'	Vergelijk teruggegeven toetswaarde met RUN-waarde
001A	1C,XX,XX	BCTA,EQ H'XXXX'	Indien gelijk, spring naar H'XXXX'
001D	1F,00,10	BCTA,UN H'0010'	Indien ongelijk, ga terug naar H'0010' en druk andere toets
0100	17		Eerste byte van bericht is een spatie
0101	14		= H
0102	0E		= E
0103	11		= L
0104	11		= L
0105	00		= O
0106	17		= spatie
0107	17		Laatste byte is een spatie

NIBBLE-subroutine

Functie

Zoals reeds vermeld, kunnen niet alle 256 mogelijkheden, die een byte biedt, zichtbaar worden gemaakt op het scherm. Kunnen echter de 2x4 bits van een byte in afzonderlijke bytes worden geplaatst, zodat één byte in wezen twee bytes voortbrengt (elke byte met 4 leidende nullen), dan zou het mogelijk zijn deze byte met behulp van twee opeenvolgende plaatsen op het scherm zichtbaar te maken. De subroutine NIBBLE snijdt de byte daartoe in twee delen en plaatst deze in afzonderlijke bytes. Daar elke byte twee andere bytes produceert, kan men op deze wijze op het scherm maximaal vier bytes zichtbaar maken.

Werking

Aanroep-instructie:

mnemonic	*hexadecimale waarde*
ZBSR *DISLSD	BB,F4

Gebruikte registers: R0 en R1.

R0= byte, die gesplitst dient te worden
R0 na de subroutine = meest significante nibble
R1 na de subroutine = minst significante nibble

Vereiste aantal plaatsen op de stapel: 1

In het onderstaande programma is het gebruik van de NIBBLE-subroutine weergegeven. Het is duidelijk dat na afloop van de subroutine de inhoud van de registers R1 en R0 naar het geheugen moet worden overgebracht om te voorkomen dat deze inhoud verloren gaat.

Voorbeeld van NIBBLE

Bij betreden:	R0 = F3
Bij vertrek:	R0 = 0F
	R1 = 03

Voorbeeld van het gebruik van het NIBBLE-programma

adres	*data*	*instructie mnemonic*	*commentaar*
0000	70	REDD,R0	Lees I/O poort (non-extended in R0)
0001	BB,F4	ZBSR *DISLSD	Roep NIBBLE subroutine op
0003	CD,01,07	STRA,R1 H'0107'	Store low-order nibble in de berichtentabel
0006	CC,01,06	STRA,R0 H'0106'	Store high-order nibble in de berichtentabel
0009	05,00	LODI,R1 H'00'	Plaats berichten-
000B	06,FF	LODI,R2 H'FF'	wijzer −1 in R1 en R2
000D	04,00	LODI, R0 H'00'	Plaats opdracht in R0
000F	BB,E6	ZBSR * USRDSP	Roep USER DISPLAY-subroutine op Laat vorige poort D-waarde zien. Nieuwe I/0-waarde wordt op schakelaars gegeven en verschijnt op het scherm indien een van de toetsen wordt ingedrukt
0011	1B,6D	BCTR,UN H'6D'	Keer terug naar 0000 en verkrijg I/0 waarde
0100	13		= "R"(1e byte van bericht)
0101	0E		= "E"
0102	0D		= "D"
0103	0D		= "D"
0104	16		= "="
0105	17		= "spatie"
0106	17		= "spatie" (H-order nibble wordt hier opgeslagen.
0107	17		= "spatie" (L-order nibble wordt hier opgeslagen

INPUT DATA-subroutine

Functie

Deze subroutine wordt gebruikt om een tweetal bytes via het toetsenbord in te voeren en deze op het scherm zichtbaar te maken. Afhankelijk van de waarde van register R0 reageert de subroutine op data, resp. op het indrukken van een functietoets. Indien het register R0 H'00' bevat, kunnen een viertal hexadecimale cijfers worden ingevoerd via het hexadecimale toetsenveld. De ingevoerde cijfers worden zichtbaar gemaakt op het scherm, en wel zodanig dat de eerste nibble uiterst rechts verschijnt, bij het invoeren van het volgende cijfer één plaats naar links opschuift, enz. Het indrukken van een functietoets verzekert dat de subroutine wordt beëindigd en naar het hoofdprogramma wordt teruggekeerd. Dit kan b.v. het geval zijn bij het indrukken van de RUN- of STEP-toets. Indien het register R0 de waarde H'01' bevat, dan is de functie van de subroutine identiek aan de vorige, met die uitzondering echter, dat de ingedrukte functietoets ook zichtbaar wordt gemaakt op het scherm.

Werking

Aanroep-instructie:

mnemonic	*hexadecimale waarde*
ZBSR *GNP	BB,FA

Gebruikte registers: R0, R1, R2 en R3.

R0= opdracht bij het begin van het programma
R0= twee data toetswaarden na afloop van de subroutine
R1= twee data toetswaarden na afloop van de subroutine
R2= functie toetswaarde
R3= indicator dat data is ingevoerd

Vereiste aantal plaatsen op de stapel: 1

Voorbeeld van een INPUT DATA subroutine aanroep

adres	*data*	*instructie mnemonic*	*commentaar*
0050	05,00	LODI,R1 H'00'	Plaats wijzer −1
0052	06,FF	LODI,R2 H'FF'	in R1 en R2
0054	BB,FF	ZBSR *MOV	Roep MOVE subroutine aan
0056	04,00	LODI,R0 H'00'	Plaats opdracht in R0
0058	BB,FA	ZBSR *GNP	Roep INPUT DATA subroutine aan
0100	10		1e byte van het bericht = P
0101	11		= L
0102	12		= U
0103	05		= S
0104	17		= spatie
0105	17		= spatie
0106	17		= spatie
0107	17		Laatste byte van het bericht = spatie

Voorbeeld van een data entry- en register-inhoud bij terugkeer van de INPUT DATA-subroutine

toets	*scherm*	*commentaar*
	PLUS	Beginwaarde bij het binnengaan van de subroutine
[1] [2] [3] [4] [RUN]	PLUS 1234	Data-waarde wordt ingevoerd, de data-invoer wordt beëindigd en de besturing schakelt terug naar het hoofdprogramma

Register-inhoud bij terugkeer van de INPUT DATA-subroutine

register	*inhoud*	*commentaar*
R0	H'34'	minst-significante data waarden
R1	H'12'	meest-significante data waarden
R2	H'86'	waarde van de RUN-toets
R3	H'00'	indicatiewaarden van de data in R0 en R1

MODIFY DATA-subroutine

Functie

De MODIFY DATA-subroutine is vrijwel identiek aan de INPUT DATA-subroutine. Het belangrijkste verschil is dat het oorspronkelijke bericht op acht posities van het scherm kan worden waargenomen.

Werking

Aanroep-instructie:

mnemonic	*hexadecimale waarde*
ZBSR * GNPA	BB,FC

Gebruikte registers: R0, R1, R2 en R3.

R0= opdracht bij het begin van het programma
R0= twee data toetswaarden na afloop van de subroutine
R1= twee data toetswaarden na afloop van de subroutine
R2= functie toetswaarde
R3= indicator dat data is ingevoerd

Vereiste aantal plaatsen op de stapel: 1

Voorbeeld van een MODIFY DATA subroutine aanroep

adres	*data*	*instruction mnemonic*	*commentaar*
0034	05,00	LODI,R1 H'00'	Plaats berichten-
0036	06,FF	LODI,R2 H'FF'	wijzer in R1 en R2.
0038	BB,FE	ZBSR * MOV	Roep MOVE-routine aan om het bericht naar buffer van het scherm over te brengen

003A	04,01	LODI,R0 H'01	Plaats opdracht in R0 (H'01' = 2 digits)
003C	BB,FC	ZBSR *GNPA	Roep MODIFY DATA-subroutine aan.
0100	17		1e byte van het bericht = "spatie"
0101	18		= "J"
0102	15		= "O"
0103	0B		= "B"
0104	16		= "="
0105	17		= "spatie"
0106	00		= "0" (voorgaande ingevoerde data waarden)
0107	01		= "1" (voorgaande ingevoerde data waarden)

Registerinhoud bij een terugkeer uit de MODIFY DATA subroutine

register	*inhoud*	*commentaar*
R0	H'02'	Ingevoerde data is geplaatst in R0
R1	H'XX'	Data in R1 is betekenisloos
R2	H'86'	Waarde van de RUN-toets
R3	H'00'	Geeft aan dat de waarde in R0 geldig is

SYSTEEM-EXPANSIE

De INSTRUCTOR 50 biedt aan de achterzijde, zoals vermeld, de mogelijkheid om een contrasteker aan te brengen. Hierdoor kan de S100-interface worden benut. Dit schept de mogelijkheid om het systeem uit te breiden met extra geheugen en bovendien met een aantal in- en uitvoer-apparaten. Het aantal mogelijkheden is zo groot dat dit nauwelijks te beschrijven is. Behalve in de S100-bus zijn nog een aantal pennen nieuw benoemd. Deze reservepennen kunnen gebruikt worden om specifieke 2650 signalen, zoals b.v. OPREQ, R/W en M/IO uit te voeren. Hierdoor wordt aan de gebruiker de mogelijkheid geboden, alle speciale faciliteiten van de 2650 te gebruiken. De interface-signalen zijn onderstaand beschreven.

pen	*mnemonic*	*signaalbeschrijving*
1	+8	+8 V niet geregeld. Deze ingang kan een +8 V voorziening geven voor de INSTRUCTOR 50, afhankelijk van de positie van de verbinding voor de stroomvoorziening aan de onderzijde.
2	+16	Positieve 16 V spanningsvoorziening. Deze ingang wordt gereserveerd voor +16 V, eventueel nodig voor een S100 periferie printed circuit. De +16 V is niet noodzakelijk voor de INSTRUCTOR 50 en wordt ook niet door deze opgewekt.
3	XRDY	External Ready. XRDY is een signaal dat een I/O-apparaat geeft, als de data-overdracht met de 2650 is geschied. De INSTRUCTOR beschouwt dit als een OPACK voor de 2650.
4	VI0	Vectored Interrupt 0, VI0, is een externe INTERRUPT-lijn. Is hiervoor de verbinding aangebracht aan de onderzijde van de Instructor 50 voor externe INTERRUPTS, dan wordt de VI0 geregistreerd; deze veroorzaakt een directe of indirecte interrupt, afhankelijk van de stand van de schakelaar DIRECT/INDIRECT.
5 t/m 11		Niet gebruikt!
12	R/W*	Read/Write. Dit is een besturingssignaal van de microprocessor 2650 dat aangeeft of deze een lees- dan wel een schrijfoperatie met de externe apparatuur wenst uit te voeren. Dit signaal is uitsluitend geldig indien OPREQ "hoog" is.
13	WRP*	Write Pulse. De microprocessor 2650 wekt een signaal op gedurende een geheugen- of een I/O-schrijfoperatie.

* Dit is geen standaard S100-interface-signaal

		WRP kan worden gebruikt om data te laten overnemen door de externe apparatuur.
14	M/IO*	Memory/Input-Output. Dit is een signaal van de microprocessor 2650 dat aangeeft, of de adresbus een geheugen of een I/O-adres bevat gedurende de dataoverdracht.
15	RESET*	Reset. Als het signaal op deze pen hoog is wordt er een RESET-operatie uitgevoerd gelijk aan het indrukken van de RST-toets op het paneel van de INSTRUCTOR 50. Dit heeft tot gevolg dat de uitvoering van het programma op de plaats H'0000' begint.
16	RUN/WAIT*	Dit signaal van de microprocessor 2650 geeft aan of deze wacht dan wel een programma uitvoert.
17	PAUSE*	Dit is een 2650 controle-ingangssignaal, dat gebruikt kan worden voor een Direct Memory Access (DMA). Als het signaal "hoog" wordt gemaakt, stopt de microprocessor 2650 met de uitvoering van instructies, direct na het uitvoeren van de instructie waarmee hij bezig is.
18 t/m 23		Niet gebruikt!
24	Φ1	Fase 1-klok, kan worden bestuurd door het 895 kHz kloksignaal of het OPREQ-signaal, afhankelijk van de configuratie aan de onderzijde van de Instructor 50.
25	Φ2	Fase 2-klok, kan worden bestuurd door de systeemklok of door het OPREQ-signaal, afhankelijk van de configuratie, aangebracht aan de onderzijde van de Instructor 50.
26 t/m 28		Niet gebruikt!
29	A5	Adres-bit 5.
30	A4	Adres-bit 4.
31	A3	Adres-bit 3.
32	A15	Adres-bit 15. Aangezien de microprocessor 2650 slechts 32K kan adresseren, is deze verbonden met de aarde.
33	A12	Adres-bit 12.
34	A9	Adres-bit 9.
35	DO1	Data OUT bit 1.
36	DO0	Data OUT bit 0.
37	A10	Adres-bit 10.
38	DO4	Data OUT bit 4.
39	DO5	Data OUT bit 5.
40	DO6	Data OUT bit 6.
41	DI2	Data IN bit 2.
42	DI3	Data IN bit 3.
43	DI7	Data IN bit 7.
44		Niet gebruikt!
45	SOUT	SOUT geeft aan dat de adresbus een adres bevat voor een uitgangsapparaat. Het desbetreffende apparaat kan de waarde van de databus accepteren indien PWR (pen 77) actief is.
46	SINP	SINP geeft aan dat de adresbus een adres bevat voor een ingangsapparaat. Het geselecteerde apparaat kan zijn data overdragen indien PDBIN (pen 78) actief is.
47	SMEMR	Memory Read. Dit signaal geeft aan dat de adresbus het adres van een geheugenlokatie bevat en dat de microprocessor 2650 een geheugenleesoperatie uitvoert.
48		Niet gebruikt!
49	CLOCK	Systeemklok. Afhankelijk van de bedrading aan de onderkant van de Instructor 50, wordt deze pen aangedreven door de 895 kHz systeemklok dan wel door de 2650 OPREQ-output.
50	GND	Systeemaarde
51	+8 Volt	Een positieve 8 V-lijn, via welke +8 V aan de INSTRUCTOR 50 kan worden verstrekt als aan de onderzijde daartoe voorzieningen zijn getroffen.
52	−16 Volt	Een negatieve 16 V-lijn, die is gereserveerd voor de −16 V die noodzakelijk kan zijn voor een periferie-apparaat. De INSTRUCTOR 50 gebruikt deze spanning niet.
53		Niet in gebruik!
54		Niet in gebruik!
55	D0*	Data-bus-bit 0.
56	D1*	Data-bus-bit 1.
57	D2*	Data-bus-bit 2.
58	D3*	Data-bus-bit 3.
59	D4*	Data-bus-bit 4.
60		Niet in gebruik!
61	D5*	Data-bus-bit 5.
62	D6*	Data-bus-bit 6.
63	D7*	Data-bus-bit 7.
64	UOPREQ*	UOPREQ (User Operation Request) is een besturingssignaal voor de microprocessor 2650, dat aangeeft dat de adres- en de data-bus en andere controllijnen geldig zijn. UOPREQ kan worden gebruikt om de data-bus te copiëren voor de schrijfoperaties en activeert ingangsbusdrivers voor leesoperaties.

65	INTACK*	Interrupt Acknowledge. De microprocessor 2650 geeft een bevestiging in de vorm van INTACK als een interrupt is herkend en verwerkt. Na dit signaal kan een interrumperend apparaat via de data-bus een relatief adres sturen, waardoor m.b.v. de ZBSR-instructie een sprong kan worden gemaakt in de eerste 64 bytes van pagina 0.
66	FLAG*	Deze lijn presenteert het FLAG-signaal van de microprocessor 2650.
67	USENSE*	User Sense. Deze pen is verbonden met de 2650 SENSE-pen en kan gebruikt worden voor het invoeren van seriële data. FLAG en SENSE maken deel uit van het PSW.
68	MWRITE	Het Memory Write (MWRITE)-signaal geeft aan dat data in het geheugen kan worden geschreven op die plaats die de adresbus aangeeft.
69 t/m 71		Niet in gebruik!
72	PRDY	Het Processor Ready (PRDY)-signaal is met XRDY in een OF-schakeling samengevoegd, waarvan het resultaat het statussignaal OPACK (Operation Acknowledge) van de microprocessor 2650 is. PRDY wordt teruggezonden door een geadresseerd periferie-apparaat (hetzij geheugen, dan wel I/O), óf door een randapparaat, nadat de data-overdracht na een interruptie is geschied.
73	PINT	Processor Interrupt (PINT) is een S100 interface signaal dat overeenkomt met het Interrupt Request signaal van de microprocessor 2650. Deze bevestigt PINT, als de instructie, waarmee hij bezig was toen het PINT signaal "laag" werd, is voltooid. De microprocessor 2650 herkent PINT niet indien hij in de WACHT-toestand is óf indien er het INTERRUPT INHIBIT-bit (II) in het PSW in de 0-stand staat. PINT kan worden gebruikt om de microprocessor 2650 uit de HALT-toestand te brengen.
74 t/m 76		Niet in gebruik!
77	PWR	Processor Write (PWR) geeft aan dat de signalen op de data-bus geldig zijn en kunnen worden geaccepteerd door het geadresseerde geheugen of uitgangsapparaat.
78	PDBIN	Processor Data Bus In (PDBIN) geeft aan dat de microprocessor 2650 data leest van het geadresseerde geheugen óf van een INPUT-apparaat. PDBIN kan worden gebruikt om de busdrivers van een geselecteerd apparaat te activeren.
79	A0	Adres-bit 0.
80	A1	Adres-bit 1.
81	A2	Adres-bit 2.
82	A6	Adres-bit 6.
83	A7	Adres-bit 7.
84	A8	Adres-bit 8.
85	A13	Adres-bit 13.
86	A14	Adres-bit 14.
87	A11	Adres-bit 11.
88	DO2	Data OUT bit 2.
89	DO3	Data OUT bit 3.
90	DO7	Data OUT bit 7.
91	DI4	Data IN bit 4.
92	DI5	Data IN bit 5.
93	DI6	Data IN bit 6.
94	DI1	Data IN bit 1.
95	DI0	Data IN bit 0.
96 t/m 98		Niet in gebruik!
99	POR	Power On Reset (POR) is een uitgangssignaal dat aangeeft dat de stroomvoorziening is aangesloten op de INSTRUCTOR 50, en dat het systeem in de 0-stand is gebracht. Het uitgangssignaal POR kan worden gebruikt om de peroferie-apparatuur desgewenst eveneens in de 0-stand te brengen.
100	GND	Ground. Systeemaarde.

DE INTERFACE NAAR DE CASSETTE-RECORDER

Op de INSTRUCTOR 50 kan een cassette-recorder aangesloten worden om de in het geheugen geregistreerde data vast te leggen op een magnetische cassetteband. Deze mogelijkheid is van groot belang als men een wat groter programma heeft gemaakt en het programmeren op een bepaald moment wenst te beëindigen. Het nadeel is dan dat bij het wegvallen van de stroomvoorziening, ook de informatie uit de RAM geheugens verdwijnt. Indien men de informatie vóór het uitschakelen van de INSTRUCTOR 50 zou kunnen registreren, dan gaat deze niet verloren zodat ze dan later opnieuw aan de machine kan worden toevertrouwd. Voor dat doel kan een normale audio-cassette-recorder gebruikt worden.

Hiertoe worden twee snoeren meegeleverd, die gebruikt kunnen worden bij de in- en uitvoer van gegevens naar en van de cassette-recorder.

Op één cassette kunnen verscheiden programma's geregistreerd worden omdat de programma's ieder een eigen nummer krijgen, het z.g. "FILE" nummer. Voordat men begint met het registreren, resp. lezen, moet de cassetteband ongeveer op de juiste plaats worden gebracht. Men dient te voorkomen dat bij een reeds eerder geregistreerde file de data overschreven wordt, zodat men er verstandig aan doet, tussen de diverse files een behoorlijke ruimte open te laten.

Als men het ingangssignaal kan regelen d.m.v. een volumeregelaar, dan dient deze ongeveer de stand in te nemen waarbij men normaal een audio-signaal registreert. Er zijn enkele richtlijnen waarmee men rekening moet houden bij het schrijven, het afregelen en het lezen van de cassette.

Belangrijke opmerking

Sommige typen cassette-recorders hebben een zeer snel werkende automatische volumeregeling. Deze zijn minder geschikt om met de INSTRUCTOR 50 te worden gebruikt. Bij de afregelprocedure komt dit tot uiting door het uitblijven van een stabiele "–" indicatie op het scherm en zal men uitsluitend een "d" of een "U" op het scherm zien.

Voorts kan het voorkomen dat, bij gebruik van slechte soorten cassetteband, voortdurend problemen optreden bij het inlezen van programma's in de INSTRUCTOR 50. Doorgaans ziet men dan "ERROR 6" of "ERROR 4" op het scherm verschijnen. De oplossing voor dit probleem is erg eenvoudig; gebruik uitsluitend cassettes van een goede kwaliteit!

Schrijven op de cassette

De WCAS-toets (Write Cassette) biedt de mogelijkheid om een programma klaar te maken voor het schrijven op een cassetteband. Voor dat doel wordt eerst deze toets ingedrukt, zie fig. 14 a), zodat op het scherm L.Ad.= verschijnt. De INSTRUCTOR 50 wacht nu op het adres van het eerste byte dat moet worden geregistreerd. In vrijwel alle gevallen zal dit byte H'0000' zijn, zodat men kan volstaan met het indrukken van de hexadecimale toets 0, zie b). Vervolgens drukt men ENT/NXT in, zie c), om te bevestigen dat dit het adres is waar zich het eerste geregistreerde byte bevindt. Vervolgens moet men het adres aangeven van het laatste byte dat geregistreerd moet worden. Bij het indrukken van ENT/NXT, zie c), vermeldt het scherm U.Ad.=, d.w.z. dat de INSTRUCTOR 50 wacht op het adres van het laatste byte dat geregistreerd moet worden. Stel dat dit laatste byte zich op het adres H'76' bevindt. Men kan dit adres nu invoeren, zie d) en e), en bevestigen door ENT/NXT in te drukken. Op het scherm verschijnt dan S.Ad.=. Men kan nu aan de INSTRUCTOR 50 het startadres van het programma toevertrouwen. Dit startadres behoeft niet hetzelfde te zijn als het adres van het eerste byte van de te registreren informatie. Het kan immers voorkomen dat men in de eerste 64 bytes van het geheugen een aantal adressen heeft geplaatst in verband met de instructie ZBSR, die gebruik maakt van de eerste 64 bytes van het geheugen voor subroutines, resp. voor verwijzing naar subroutines. Het zou dus zeer wel mogelijk zijn dat H'10' (= adres 16 decimaal) het adres is waar zich de eerste instructie van het gekozen programma bevindt. Door dit adres aan de cassette-recorder toe te vertrouwen, wordt straks het instructie-adresregister automatisch in de juiste stand geplaatst. Na het invoeren van het startadres, zie g) en h), drukt men de toets ENT/NXT in, zie i), waarna .F= op het scherm verschijnt. Nu kan men een filenummer invoeren, zodat de geregistreerde informatie wordt voorzien van een nummer en het mogelijk wordt meer dan één file op één magnetische band te plaatsen. In ons geval nemen we aan, dat dit filenummer 1 is en we slaan dit cijfer dan ook aan, zie j).

Nu wordt de cassette-recorder gestart en de toets ENT/NXT *ingedrukt,* zie k). Het duurt dan nog ongeveer 5 seconden vóórdat het programma op de cassette wordt geschreven. Gedurende deze tijd gaat de FLAG LED aan en uit, en wel ongeveer eens per seconde. Is de registratie uit het geheugen volledig op de band overgebracht, dan geeft de INSTRUCTOR 50 dit aan met het woord "HELLO", zie l). Men kan de cassette-recorder nu stoppen. Het is verstandig om het nummer te noteren van de stand van het telwerk van de cassette-recorder, zodat men later bij het lezen van informatie, deze plaats bij benadering kent. Aldus wordt ook voorkomen dat de volgende file de nu geschreven informatie verdringt.

Het opnemen van de informatie uit het geheugen op de band kan worden waargenomen op de acht lampjes links op het bedieningspaneel. Hiertoe moet de 3-standen schakelaar in de stand "EXTENDED" worden geplaatst.

Afregelen van de cassette

Wil men de cassette gaan lezen, dan is het van belang dat het ontvangen signaal het juiste niveau heeft. Om de informatie uit de cassette wederom naar het geheugen over te dragen, moet men de cassette doorspoelen tot de stand bereikt is waar de over te dragen informatie staat. De uitgang van de cassette-recorder moet d.m.v. het bijgeleverde snoer op de "phone"-bus worden aangesloten. Om de cassette af te regelen, drukt men nu eerst op de toets REG, zie fig. 15 a). De INSTRUCTOR 50 wacht nu op het invoeren van nadere informatie. De opdracht tot het afregelen wordt gegeven door het indrukken van de toets A van het hexadecimale toetsenveld, zie b). Nu verschijnt bij het afspelen van de cassetterecorder op het scherm een letter. De letter U betekent dat het niveau te laag is en men het volume dus dient op te voeren. Verschijnt een d, dan wijst dat op een te hoog niveau van het signaal; men dient dat dan te verlagen. Het teken – betekent dat het juiste niveau is bereikt. Zo nodig moet men enkele keren terug naar het begin van de file om deze afregeling te herhalen, vooral ingeval betrekkelijk weinig informatie op de cassette is geplaatst. Nadat men het afregelen heeft verricht, d.w.z. dat een streepje op het scherm verschijnt, kan men op de knop MON drukken, zie e). Naast de informatie die op het scherm verschijnt, kan men ook nog d.m.v. de LED's van de I/O-poort waarnemen dat het niveau de juiste waarde heeft. Daartoe zet men de I/O-schakelaar in de middenstand (EXTENDED). De LED's hebben nu de volgende betekenis:

Alle LED's *uit:*	het niveau is *goed*
De meest linkse LED *aan* en enkele andere uit:	het niveau is te *laag*
De linkse LED *uit* en enkele LED's aan:	het niveau is te *hoog*

Lezen van de cassette

Om de informatie van de cassette weer over te dragen naar het geheugen, moet ze gelezen worden. Het lezen is al voorbereid door het afregelen van de cassette, en we nemen aan dat de juiste file vlak voor de leeskop aanwezig is.

Nu wordt, zie fig. 16, de toets RCAS ingedrukt, zie a). Op het scherm verschijnt nu .F=, hetgeen betekent dat de INSTRUCTOR 50 op het nummer van de file wacht. Men kan nu het filenummer aanslaan. In ons voorbeeld is dit 1, zie b). Op het scherm verschijnt dan .F=1. Men kan het filenummer bevestigen door de ENT/NXT-toets in te drukken, zie c).

Nu kan de cassette-recorder gestart worden. Na afloop van het lezen verschijnt op het scherm het woord "HELLO". De cassette-recorder kan dan gestopt worden.

Het lezen van informatie van de band naar het geheugen kan worden waargenomen op de lampjes links op het bedieningspaneel indien de 3-standen schakelaar in de middenstand "EXTENDED" wordt gezet.

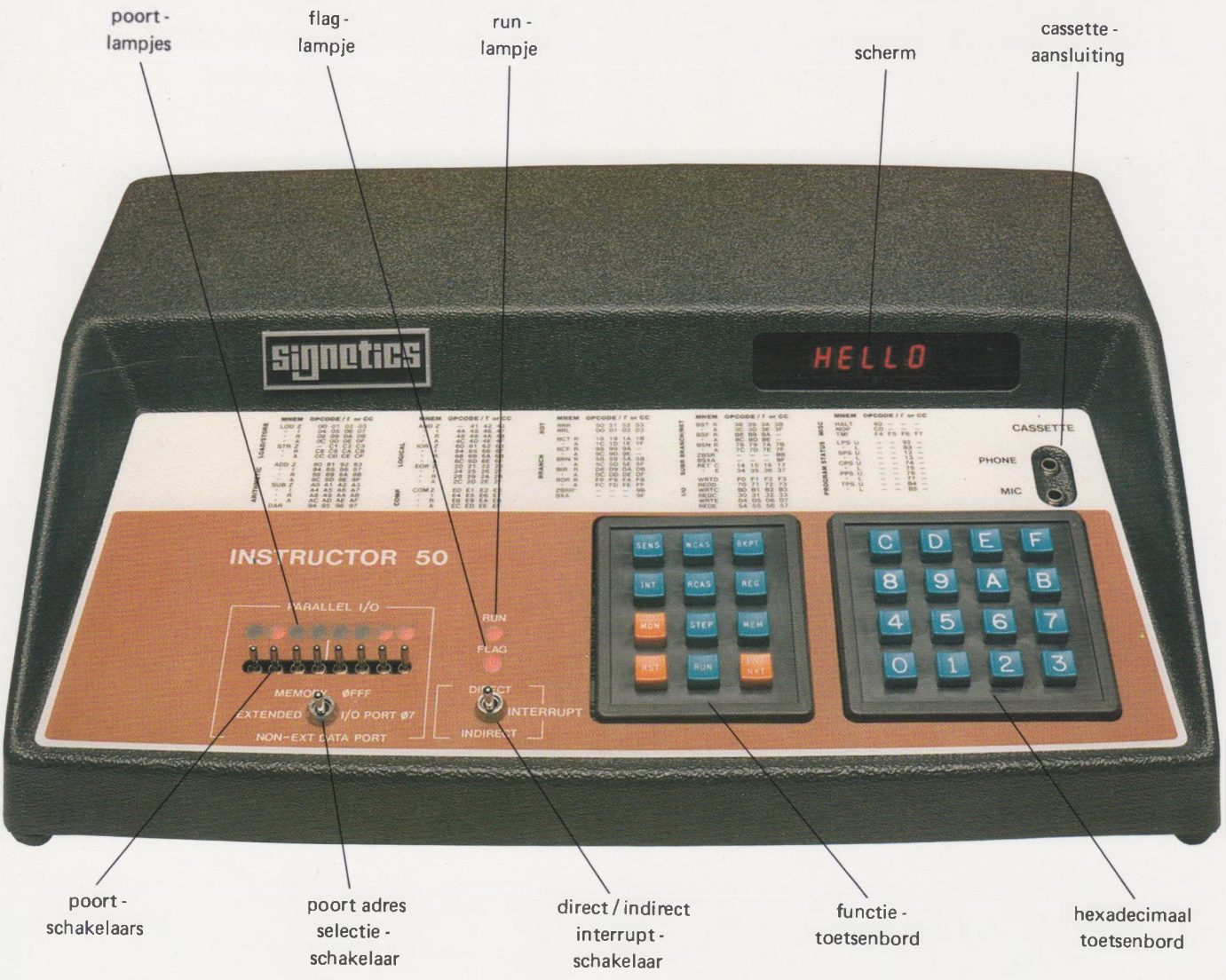


Fig. 1 De INSTRUCTOR 50.

SYMBOLEN VAN HET SCHERM

Fig. 2 Symbolen van het scherm.

TABEL DECIMALE–BINAIRE–HEXADECIMALE GETALLEN

DECIMAAL	BINAIR	HEXADEC.
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Fig. 3 Table Decimale-Binaire-Hexadecimale getallen.

GEHEUGEN EN I/O-ORGANISATIE

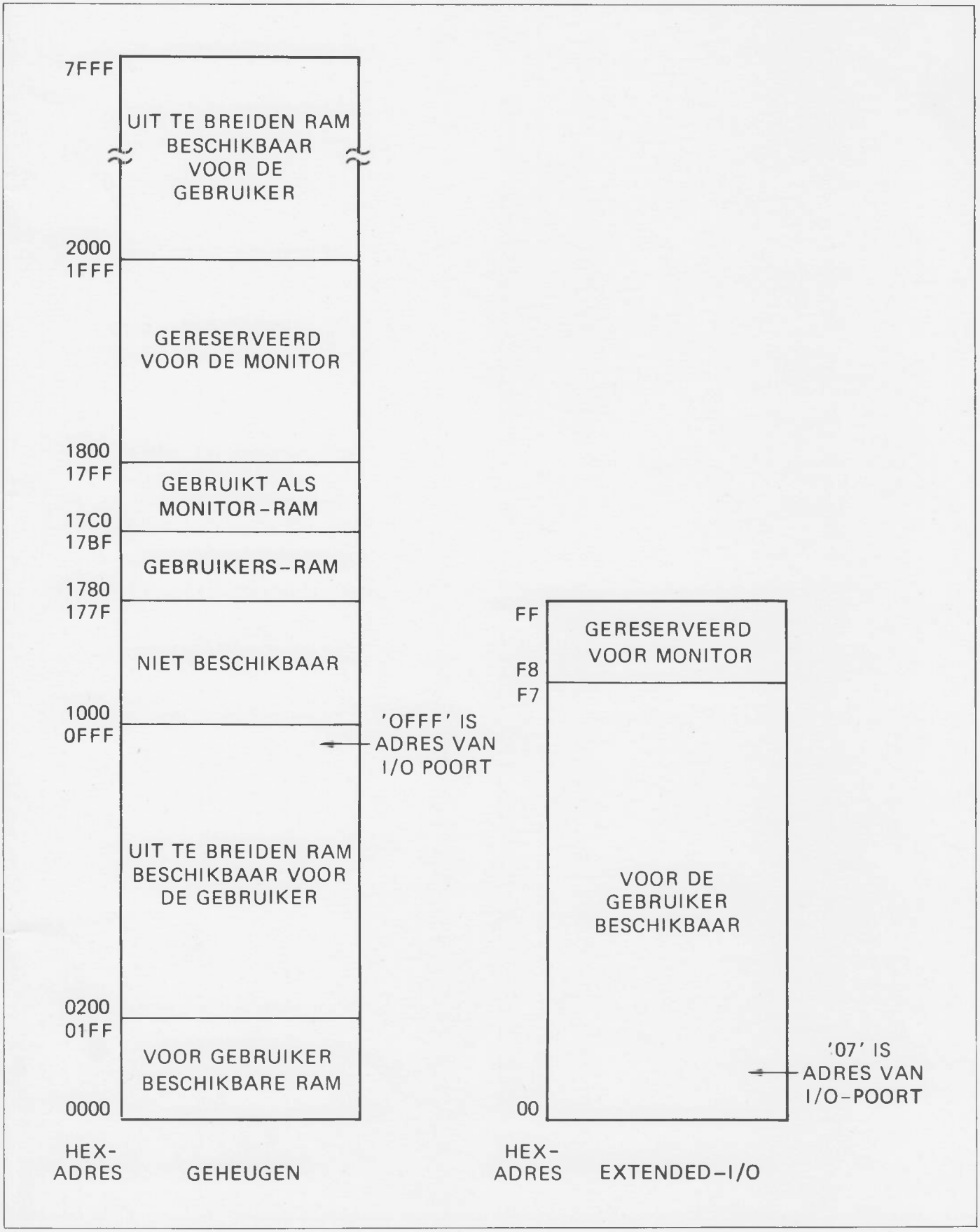


Fig. 4 Geheugen en I/O organisatie.

LEZEN VAN CPU-REGISTERS EN PSW

(Program Status Word)

	TOETS	SCHERM	COMMENTAAR
a)	REG	r =	DE INSTRUCTOR WACHT OP EEN NUMMER
b)	4	.r4 = A8	DE INHOUD VAN REGISTER 1 BANK 1 IS H'A8'
c)	ENT NXT	.r5 = B6	DE INHOUD VAN REGISTER 2 BANK 1 IS H'B6'
d)	ENT NXT	.r6 = A4	DE INHOUD VAN REGISTER 3 BANK 1 IS H'A4'
e)	ENT NXT	.PU = 0A	DE INHOUD VAN HET PROGRAM STATUS WORD UPPER IS H'0A'
f)	ENT NXT	.PL = 21	DE INHOUD VAN HET PROGRAM STATUS WORD LOWER IS H'21'

DE PSW-INHOUD IS DUS H'0A21'

Fig. 5 Lezen van CPU-registers en PSW.

LEZEN EN VERANDEREN VAN CPU-REGISTERS EN PSW

(Program Status Word)

	TOETS	SCHERM	COMMENTAAR
a)	REG	r =	DE INSTRUCTOR WACHT OP EEN NUMMER
b)	4	.r4 = A8	DE INHOUD VAN REGISTER 1 BANK 1 IS H'A8' DEZE A8 IS TOEVALLIG
c)	E	.r4 = E	DE INHOUD VAN REGISTER 1 BANK 1 WORDT GEWIJZIGD IN H'0E'
d)	2	.r4 = E2	DE INHOUD VAN REGISTER 1 BANK 1 WORDT GEWIJZIGD IN H'E2'
e)	ENT NXT	.r5 = B6	DE INHOUD VAN REGISTER 1 BANK 1 IS NU DEFINITIEF H'E2' GEWORDEN DE INHOUD VAN REGISTER 2 BANK 1 WORDT NU GETOOND EN ZOU TOEVALLIG H'B6' KUNNEN ZIJN

Fig. 6 Lezen en veranderen van CPU-registers en PSW.

LEZEN VAN GEHEUGENCELLEN

	TOETS	SCHERM	COMMENTAAR
a)	MEM	.Ad. =	DE INSTRUCTOR VRAAGT OM EEN ADRES VOOR DE SELECTIE VAN EEN GEHEUGENCEL
b)	4	.Ad. = 4	HET MEEST SIGNIFICANTE CIJFER VAN HET ADRES WORDT AANGESLAGEN
c)	2	.Ad. = 42	HET VOLGENDE CIJFER WORDT AANGESLAGEN
d)	ENT NXT	.0042 A5	DE INHOUD VAN DE GEHEUGENCEL H'0042' IS H'A5'
e)	ENT NXT	.0043 B7	DE INHOUD VAN DE VOLGENDE GEHEUGENCEL (HIER H'0043') IS H'B7'
f)	ENT NXT	.0044 05	DE INHOUD VAN DE VOLGENDE GEHEUGENCEL (HIER H'0044') IS H'05'

Fig. 7 Lezen van geheugencellen.

LEZEN EN VERANDEREN VAN GEHEUGENCELLEN

	TOETS	SCHERM	COMMENTAAR
a)	MEM	.Ad. =	DE INSTRUCTOR VRAAGT OM EEN ADRES VOOR DE SELECTIE VAN EEN GEHEUGENCEL
b)	4	.Ad. = 4	HET MEEST SIGNIFICANTE CIJFER VAN HET ADRES WORDT AANGESLAGEN EN IS 4
c)	2	.Ad. = 42	HET VOLGENDE CIJFER WORDT INGEVOERD
d)	ENT NXT	.0042 A5	DE INHOUD VAN GEHEUGENCEL H'0042' IS H'A5'
e)	6	.0042 6	OP DE INHOUD VAN GEHEUGENCEL H'0042' WORDT EEN CORRECTIE UITGEVOERD
f)	4	.0042 64	ER WORDT NOG EEN CORRECTIE UITGEVOERD
g)	ENT NXT	.0043 B7	DE INHOUD VAN H' 0042' IS NU DEFINITIEF H'64'. DE INHOUD VAN CEL H'0043' WORDT GETOOND EN IS H'B7'

Fig. 8 Lezen en veranderen van geheugencellen.

SNEL LADEN VAN HET GEHEUGEN

	TOETS	SCHERM	COMMENTAAR
a)	REG	r =	DE INSTRUCTOR WACHT OP EEN NUMMER
b)	F	.Ad. =	H'F' WORDT AANGESLAGEN. NU VRAAGT DE INSTRUCTOR OM EEN ADRES
c)	1	.Ad. = 1	HET MEEST SIGNIFICANTE ADRES-CIJFER (H'1') WORDT AANGESLAGEN
d)	2	.Ad. = 12	HET VOLGENDE ADRES-CIJFER WORDT AANGESLAGEN
e)	ENT NXT	.0012	DE GEKOZEN GEHEUGENCEL IS H'0012'
f)	A	.0012 A	HET EERSTE CIJFER WORDT IN DE H'0012'-CEL AANGESLAGEN
g)	1	.0012 A1	HET TWEEDE CIJFER WORDT AANGESLAGEN
h)	F	.0013 F	HET EERSTE CIJFER VAN DE VOLGENDE CEL (H'0013') WORDT AANGESLAGEN
i)	E	.0013 FE	HET TWEEDE CIJFER WORDT AANGESLAGEN IN CEL H'0013'
		enz.	
j)	MEM	.Ad. =	DE SNELLE INVOER WORDT BEEINDIGD EN DE STAP-VOOR-STAP INVOER (IN FIG. 8) WORDT NU UITGEVOERD

Fig. 9 Snel laden van het geheugen.

TONEN EN VERANDEREN VAN HET IAR

(Instructie Adres Register)

	TOETS	SCHERM	COMMENTAAR
a)	REG	r =	DE INSTRUCTOR 50 VRAAGT OM EEN NUMMER
b)	C	.PC = 0017	DE INHOUD VAN HET IAR (PC) IS H'0017'
c)	2	.PC = 2	HET IAR WORDT GEWIJZIGD
d)	8	.PC = 28	HET VOLGENDE CIJFER VOOR HET IAR WORDT AANGESLAGEN
e)	ENT NXT	r =	HET IAR IS DEFINITIEF GELADEN MET H'0028'

Fig. 10 Tonen en veranderen van het IAR.

STAP-VOOR-STAP FUNKTIE

	TOETS	SCHERM	COMMENTAAR
a)	REG	r =	DE INSTRUCTOR VRAAGT OM EEN NUMMER
b)	C	.PC = 0017	DE INHOUD VAN HET IAR WORDT GETOOND (H'0017')
c)	1	.PC = 1	HET IAR WORDT GEWIJZIGD
d)	2	.PC = 12	HET VOLGENDE CIJFER VAN HET IAR
e)	2	.PC = 122	HET VOLGENDE CIJFER VOOR HET IAR
f)	ENT NXT	r =	HET IAR IS DEFINITIEF MET H'0122' GELADEN
g)	STEP	0124 CA	DE INSTRUCTIE OP H'0122' WORDT UITGEVOERD EN IS 2 BYTES GROOT; DE VOLGENDE INSTRUCTIE DIE UITGEVOERD GAAT WORDEN IS OP H'0124' EN IS H'CA' (STRR,R2)
h)	STEP	0126 CC	DE INSTRUCTIE OP H'0124' WORDT UITGEVOERD EN IS 2 BYTES GROOT; DE VOLGENDE INSTRUCTIE BEGINT IN CEL H'0126' EN LUIDT H'CC' (STRA,R0)
i)	REG	r =	DE STAP-VOOR-STAP FUNCTIE WORDT BEËINDIGD

Fig. 11 Stap-voor-stap funktie.

HET AANBRENGEN VAN EEN BREEKPUNT IN HET PROGRAMMA

	TOETS	SCHERM	COMMENTAAR
a)	BKPT.	.b.P =	DE INSTRUCTOR 50 VRAAGT OM HET EERSTE CIJFER VAN HET BREEKPUNT
b)	1	.b.P = 1	HET EERSTE CIJFER WORDT AANGESLAGEN
c)	5	.b.P = 15	HET TWEEDE CIJFER WORDT AANGESLAGEN
d)	4	.b.P = 154	HET DERDE CIJFER WORDT AANGESLAGEN
e)	ENT NXT	b.P = 154	HET BREEKPUNT WORDT DEFINITIEF VASTGESTELD OP ADRES H'0154'
f)	BKPT	.b.P = 0154	DOOR HET INDRUKKEN VAN DE BKPT-TOETS WORDT HET BREEKPUNT GETOOND
g)	BKPT.	.b.P =	HET BREEKPUNT WORDT OPGEHEVEN
h)		-0154 C0	HET PROGRAMMA IS OP HET BREEKPUNT OP ADRES H'0154' AANGEKOMEN EN LAAT DE VOLGENDE INSTRUCTIE H'C0' (NOP) ZIEN

Fig. 12 Het aanbrengen van een breekpunt in het programma.

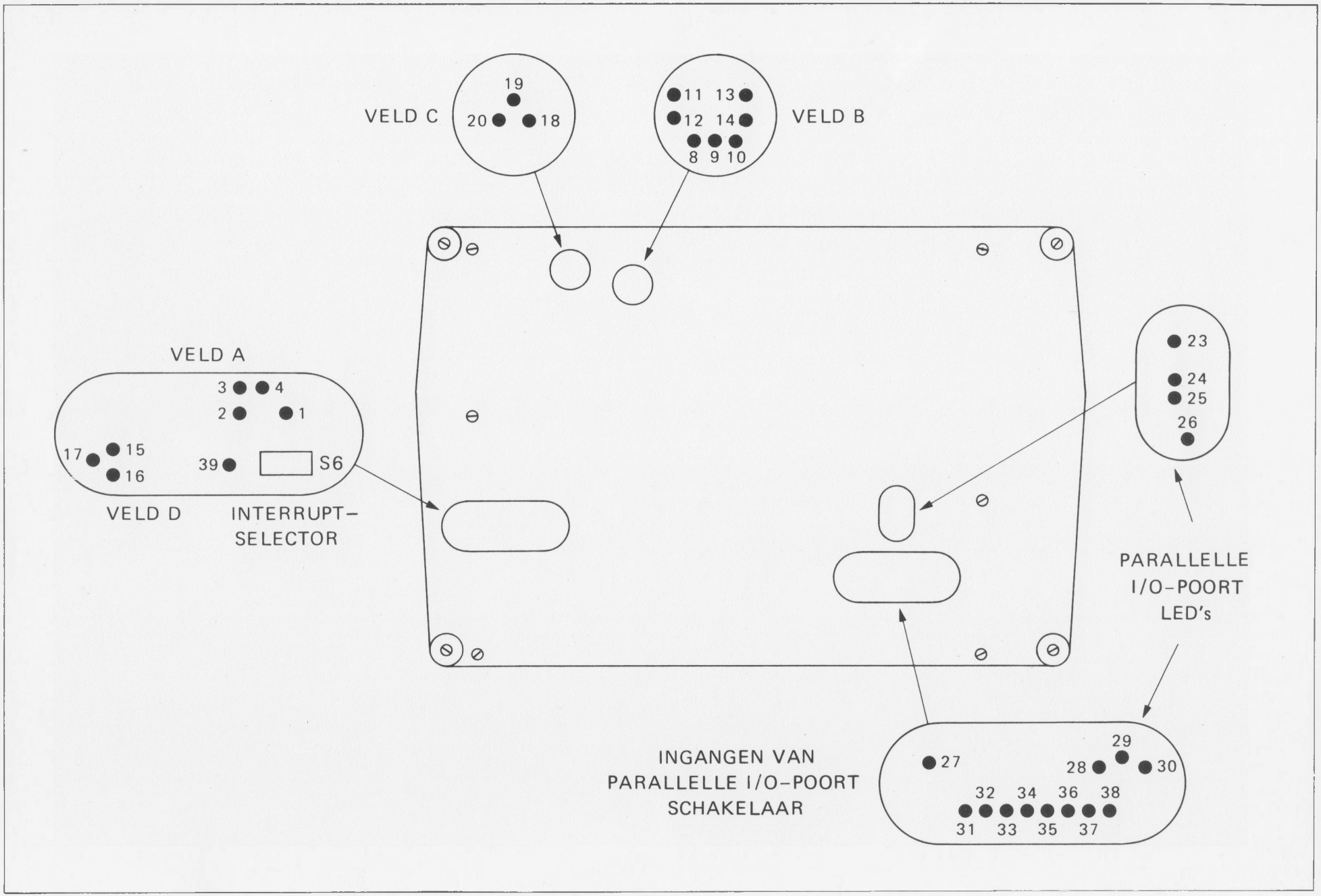


Fig. 13 Onderzijde van de INSTRUCTOR 50.

SCHRIJVEN OP DE CASSETTE

	TOETS	SCHERM	COMMENTAAR
a)	WCAS	L.Ad. =	DE INSTRUCTOR WACHT OP HET EERSTE CIJFER VAN HET MEEST SIGNIFICANTE DEEL VAN HET LAAGSTE ADRES VAN HET PROGRAMMA
b)	0	L.Ad. = 0	LAAGSTE ADRES WORDT AANGESLAGEN H'0000'
c)	ENT NXT	U.Ad. =	DE INSTRUCTOR WACHT OP HET EERSTE CIJFER VAN HET MEEST SIGNIFICANTE DEEL VAN HET HOOGSTE ADRES VAN HET PROGRAMMA
d)	7	U.Ad. = 7	HET HOOGSTE ADRES WORDT AANGESLAGEN H'0076'
e)	6	U.Ad. = 76	
f)	ENT NXT	S.Ad. =	DE INSTRUCTOR WACHT OP HET START-ADRES VAN HET PROGRAMMA
g)	1	S.Ad. = 1	HET START-ADRES WORDT AANGESLAGEN H'0010'
h)	0	S.Ad. = 10	
i)	ENT NXT	.F =	DE INSTRUCTOR WACHT OP EEN FILE-NUMMER
j)	1	.F = 1	FILE-NUMMER WORDT AANGESLAGEN H'01'
k)	ENT NXT		DATA WORDT OVERGEDRAGEN OP DE CASSETTEBAND
l)		HELLO	DATA IS OVERGEDRAGEN OP DE CASSETTEBAND

Fig. 14 Schrijven op de cassette.

AFREGELEN VAN HET SIGNAAL-NIVEAU VAN DE CASSETTE

	TOETS	SCHERM	COMMENTAAR
a)	REG	r =	DE INSTRUCTOR WACHT OP EEN NADERE SPECIFICATIE
b)	A	U	VERHOOG TERUGSPEEL-NIVEAU (VOLUMEREGELAAR OP CASSETTE-RECORDER)
c)		d.	VERMINDER TERUGSPEEL-NIVEAU (VOLUMEREGELAAR OP CASSETTE-RECORDER)
d)		–	HET AFSPEELNIVEAU IS JUIST; LAAT DE VOLUMEREGELAAR IN DEZE STAND STAAN
e)	MON	HELLO	BEËINDIG HET AFREGELEN

Fig. 15 Afregelen van het signaal-niveau van de cassette.

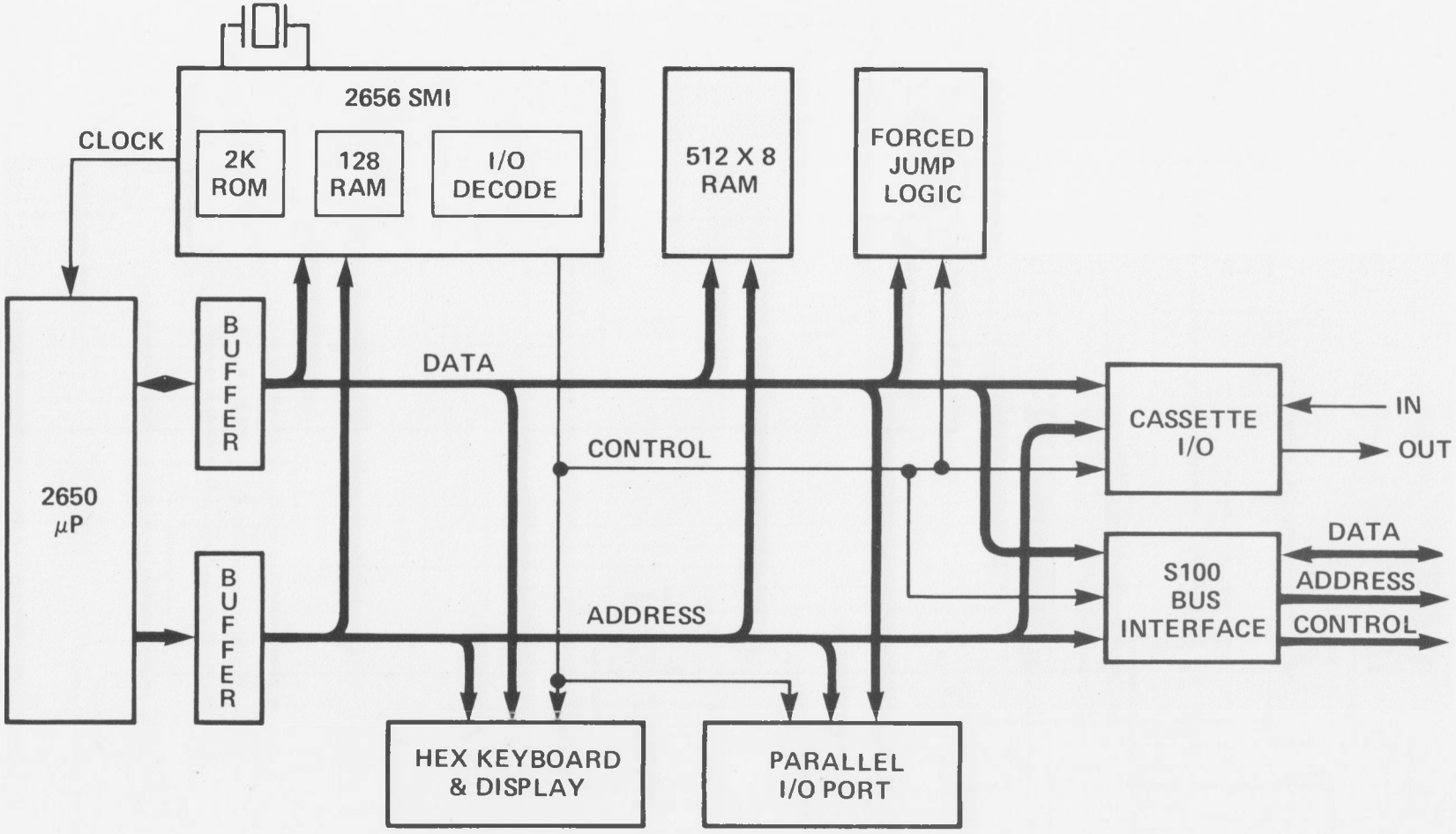
LEZEN VAN DE CASSETTE

	TOETS	SCHERM	COMMENTAAR
a)	RCAS	.F =	DE INSTRUCTOR WACHT OP EEN NADERE OPDRACHT
b)	1	.F = 1	HET FILE-NUMMER WORDT AANGESLAGEN H'01'
c)	ENT NXT		HET PROGRAMMA WORDT IN HET GEHEUGEN GELADEN
d)		HELLO	HET PROGRAMMA IS IN HET GEHEUGEN GELADEN

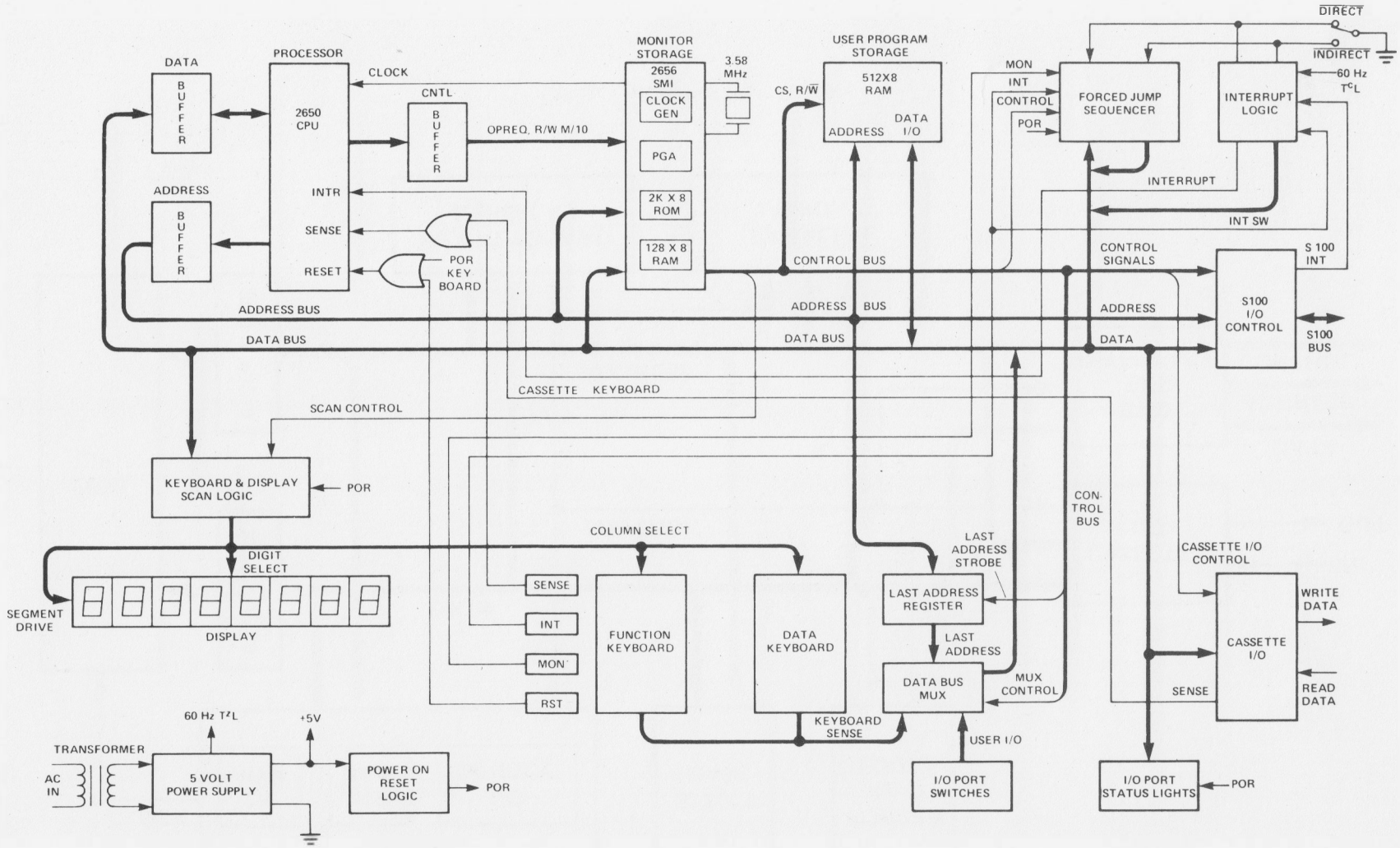
Fig. 16 Lezen van de cassette.

Appendix A

Instructor 50 blokschema's



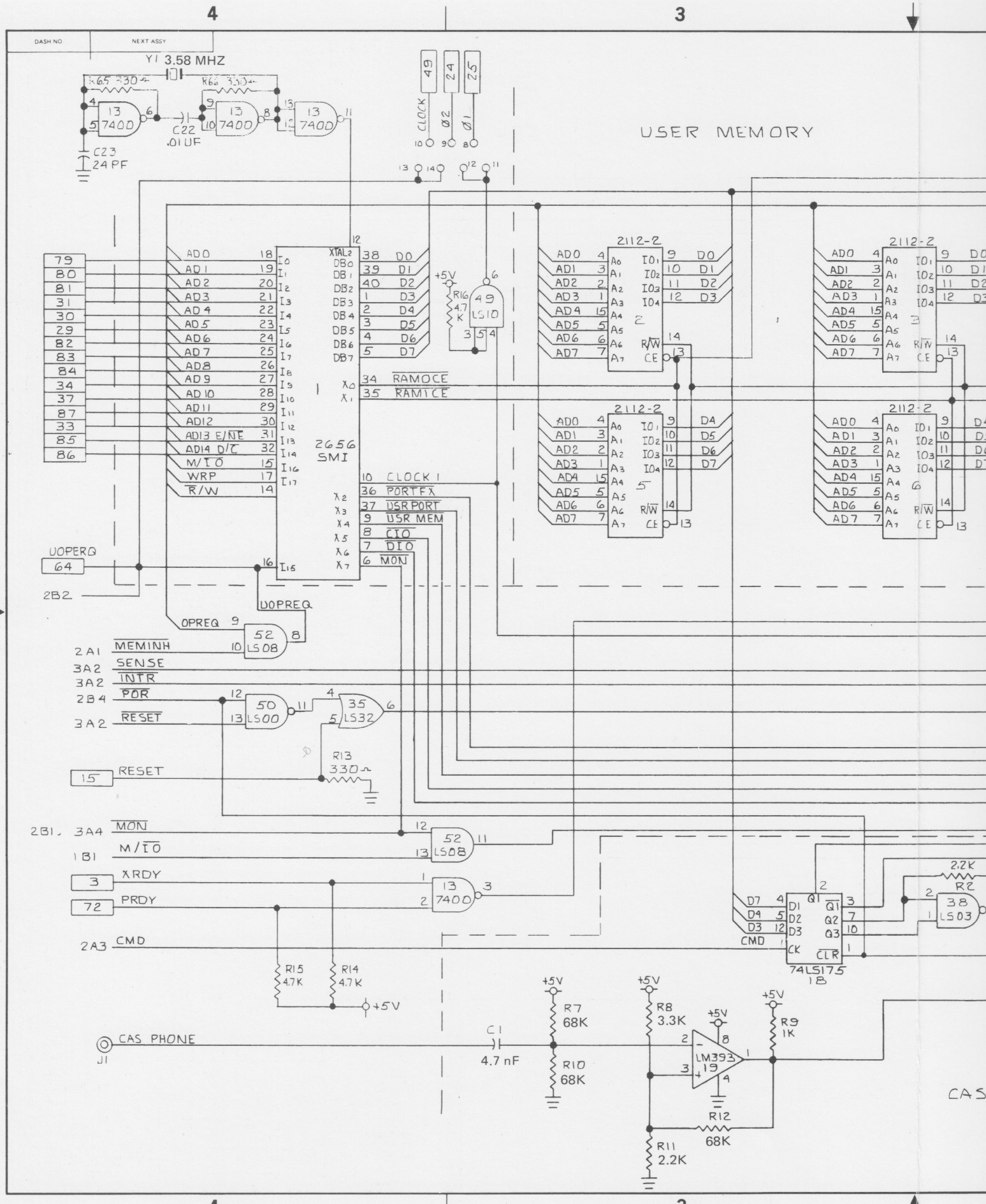
Opbouw van de INSTRUCTOR 50.

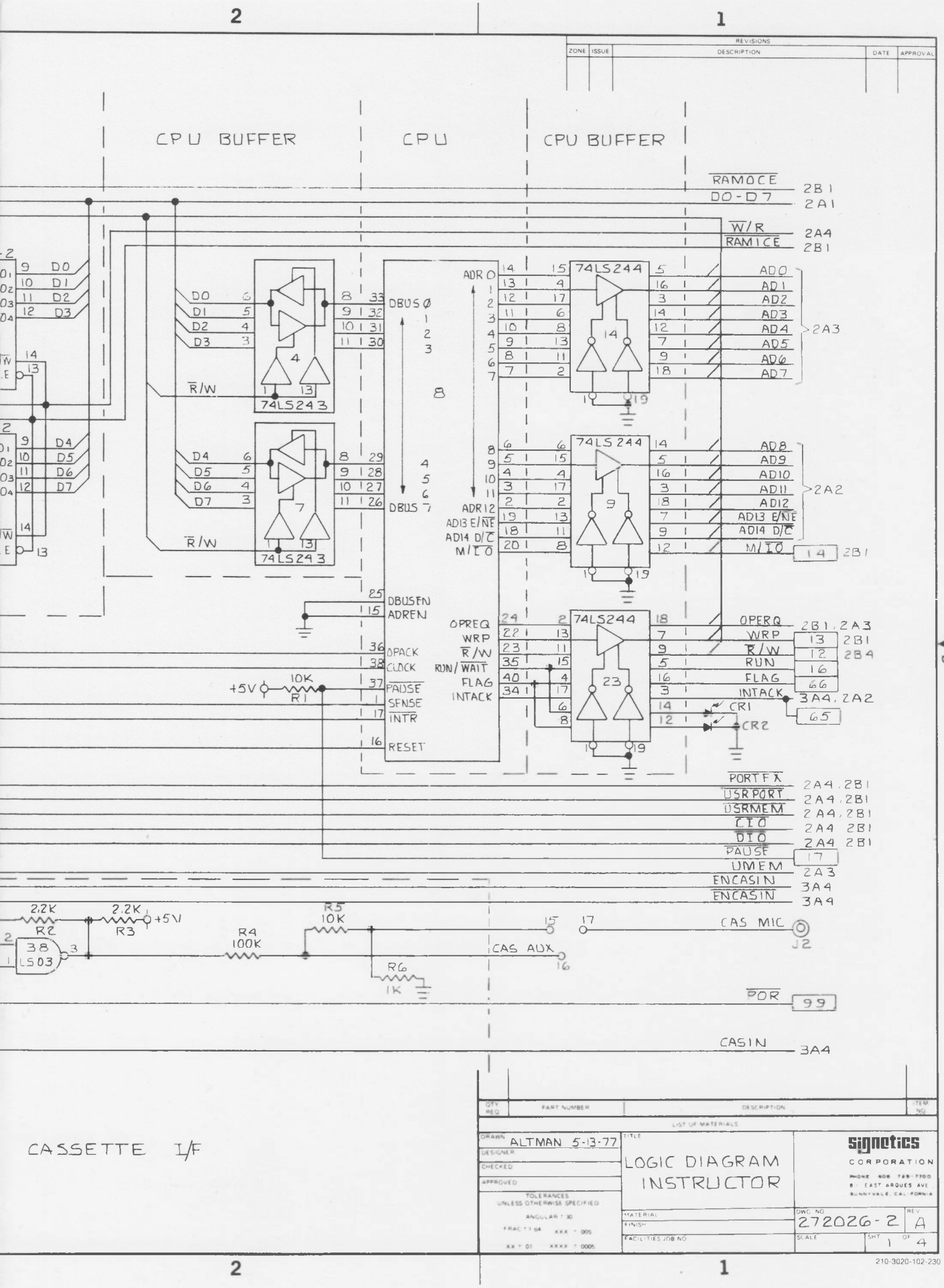


Gedetailleerde opbouw van de INSTRUCTOR 50.

Appendix B

Instructor 50 schema's


USER MEMORY
Y1 3.58 MHZ
C22 .01 UF
C23 24 PF
CLOCK
Ø2
Ø1
2656 SMI
2112-2
RAMOCE
RAMICE
CLOCK 1
PORTEX
USR PORT
USR MEM
CIO
DIO
MON
UOPERQ
UOPREQ
OPREQ
MEMINH
SENSE
INTR
POR
RESET
M/IO
XRDY
PRDY
CMD
CAS PHONE
74LS175
LM393
R7 68K
R8 3.3K
R9 1K
R10 68K
R11 2.2K
R12 68K
R13 330
R14 4.7K
R15 4.7K
R16 4.7K
C1 4.7 nF
2.2K
+5V



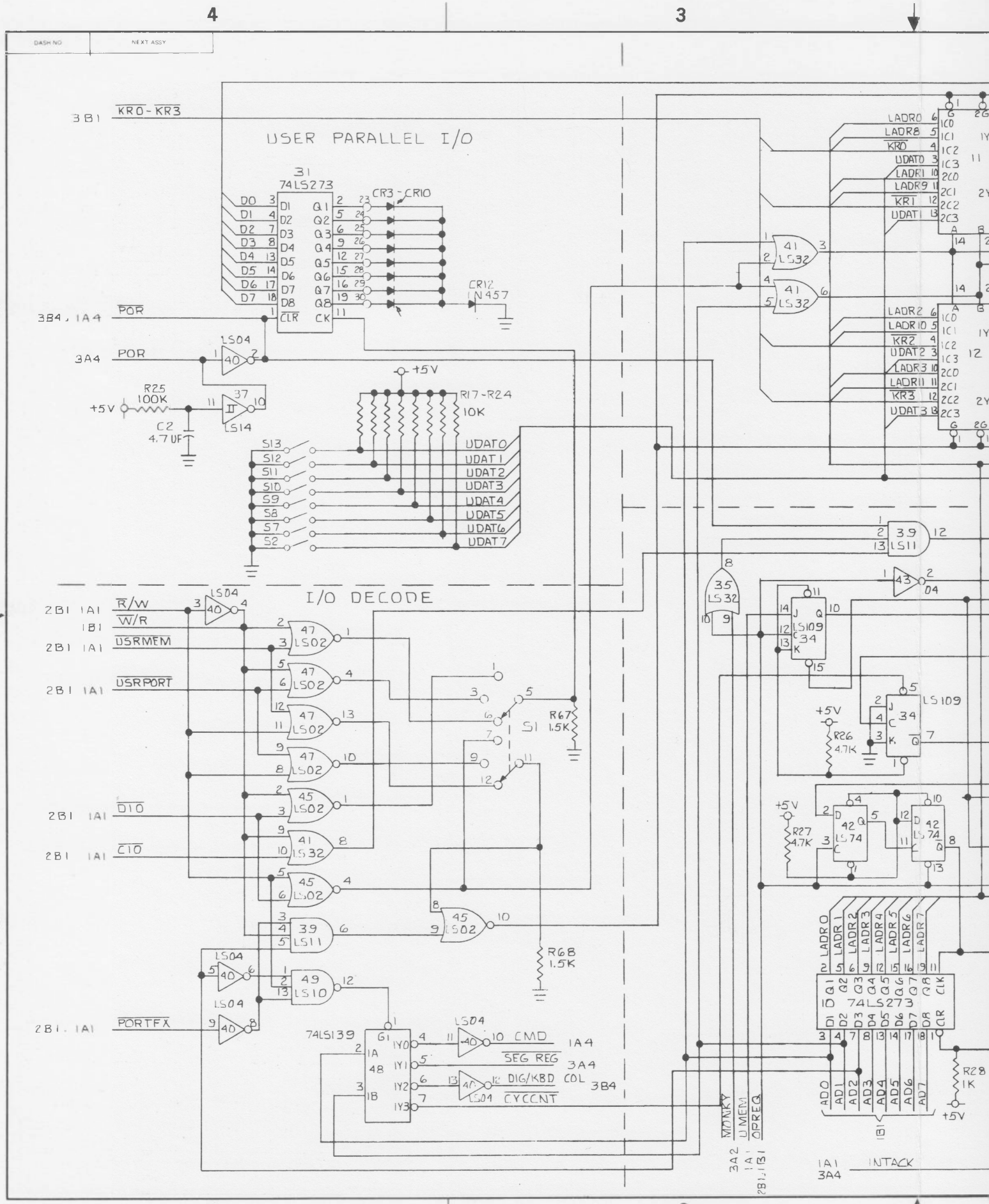
CASSETTE I/F

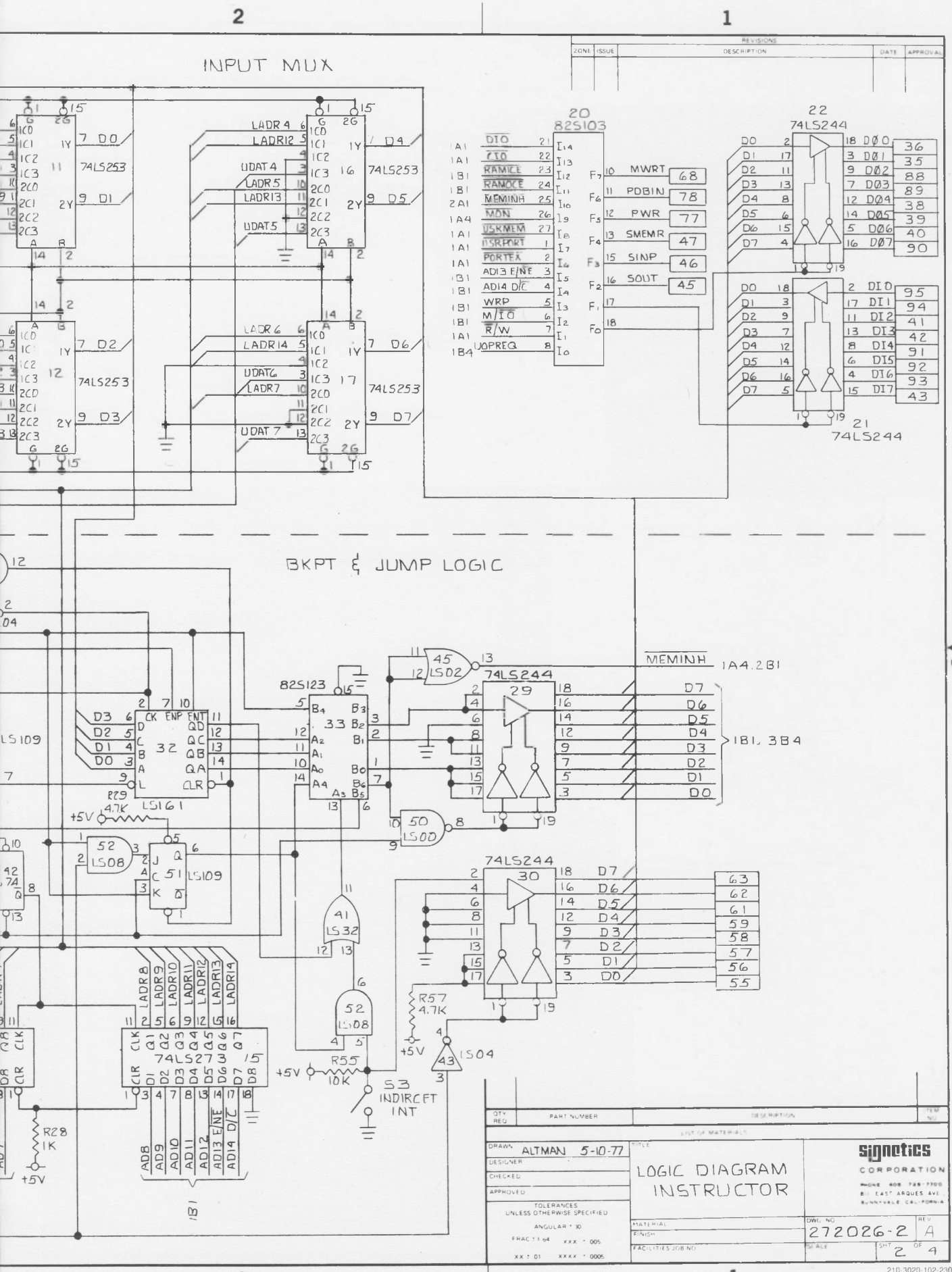
QTY REQ	PART NUMBER	DESCRIPTION	ITEM NO

LIST OF MATERIALS

DRAWN ALTMAN 5-13-77	TITLE LOGIC DIAGRAM INSTRUCTOR	signetics CORPORATION	
DESIGNER			
CHECKED			
APPROVED			
TOLERANCES UNLESS OTHERWISE SPECIFIED: ANGULAR ± 30 FRAC ± 1/64 XXX ± .005 XX ± .01 XXXX ± .0005	MATERIAL	DWG NO 272026-2	REV A
	FINISH	SCALE	SHT 1 OF 4
	FACILITIES JOB NO		

210-3020-102-230


4
3
DASH NO
NEXT ASSY
USER PARALLEL I/O
I/O DECODE
3B1 KR0-KR3
3B4, 1A4 POR
3A4 POR
74LS273
CR3-CR10
CR12 IN457
LS04
R25 100K
C2 4.7 UF
+5V
R17-R24 10K
S13
S12
S11
S10
S9
S8
S7
S2
UDAT0
UDAT1
UDAT2
UDAT3
UDAT4
UDAT5
UDAT6
UDAT7
2B1 1A1 R/W
1B1 W/R
2B1 1A1 USRMEM
2B1 1A1 USRPORT
2B1 1A1 IO
2B1 1A1 CIO
2B1 1A1 PORTFX
S1
R67 1.5K
R68 1.5K
74LS139
1Y0 CMD 1A4
1Y1 SEG REG 3A4
1Y2 DIG/KBD COL 3B4
1Y3 CYCCNT
LADR0
LADR8
KR0
UDAT0
LADR1
LADR9
KR1
UDAT1
LADR2
LADR10
KR2
UDAT2
LADR3
LADR11
KR3
UDAT3
LS109
R26 4.7K
R27 4.7K
LS74
R28 1K
MONKY
UMEM
OPREQ
3A2
1A1
2B1, 1B1
1A1 INTACK
3A4



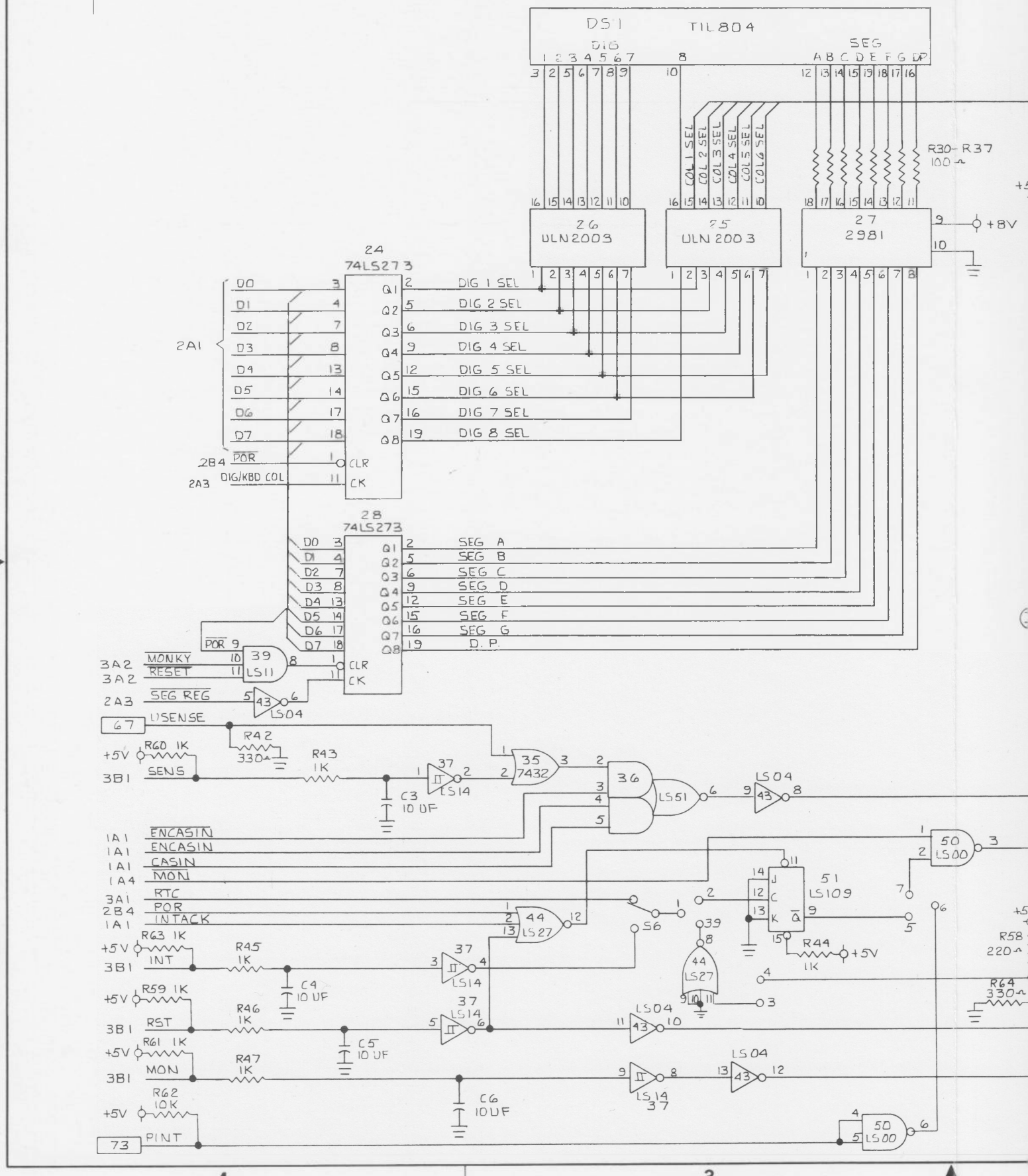
REVISIONS				
ZONE	ISSUE	DESCRIPTION	DATE	APPROVAL

QTY REQ	PART NUMBER	DESCRIPTION	ITEM NO

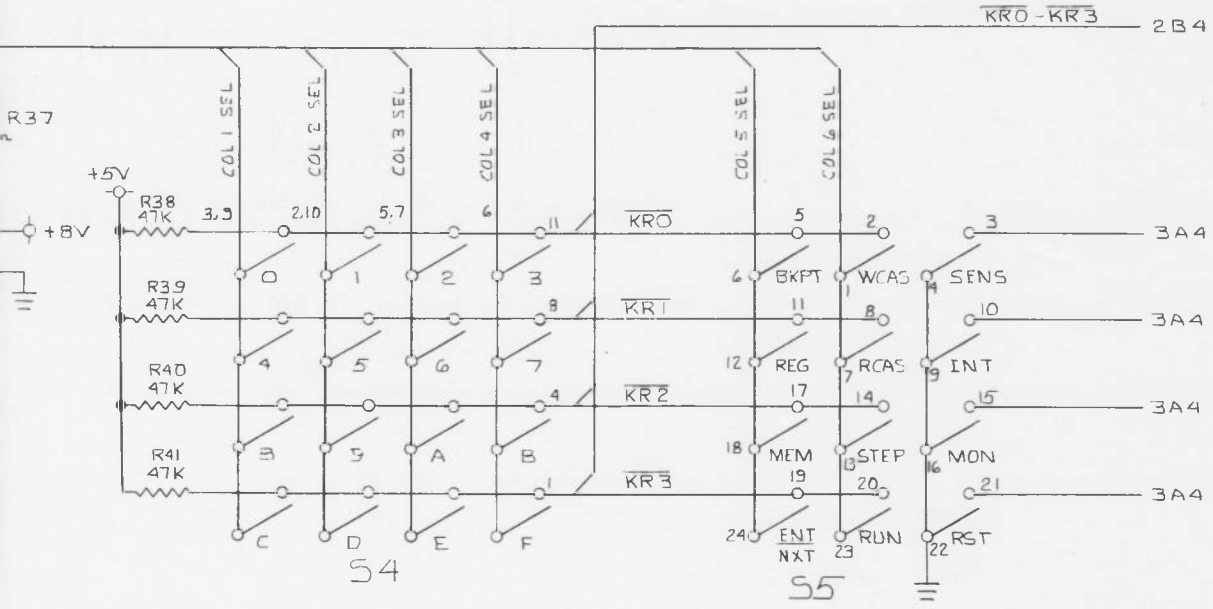
LIST OF MATERIALS

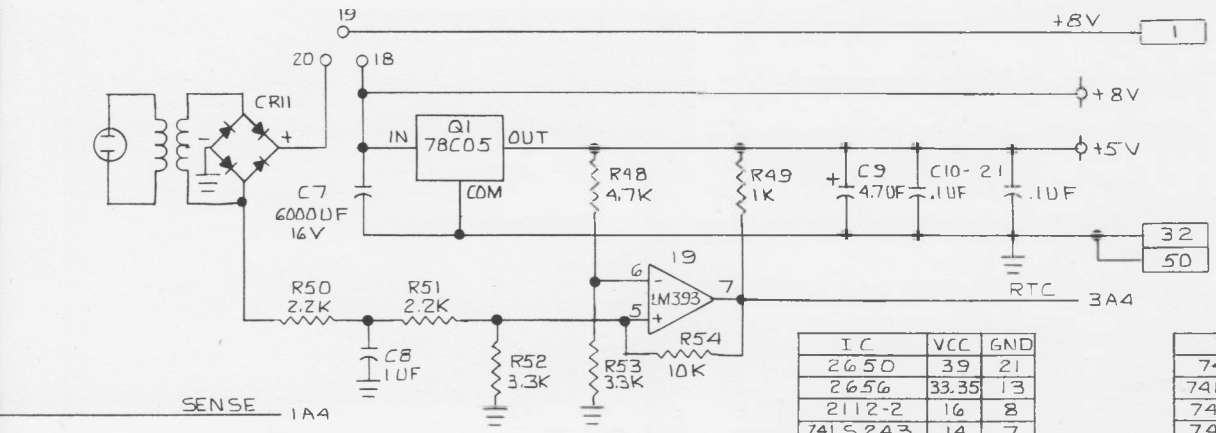
DRAWN	ALTMAN 5-10-77	TITLE	signetics CORPORATION
DESIGNER		LOGIC DIAGRAM INSTRUCTOR	PHONE 408 739-7700 811 EAST ARQUES AVE SUNNYVALE CALIFORNIA
CHECKED			
APPROVED			
TOLERANCES UNLESS OTHERWISE SPECIFIED ANGULAR ± 30 FRAC ± 1/64 XXX ± .005 XX ± .01 XXXX ± .0005		MATERIAL / FINISH / FACILITIES JOB NO	DWG NO 272026-2 REV A; SCALE; SHT 2 OF 4

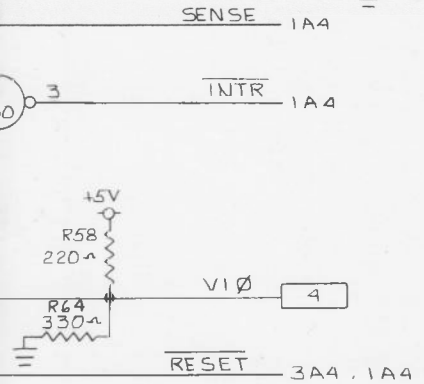
210-3020-102-230


DASH NO.
NEXT ASSY
4
3
DS1 TIL804
DIG
SEG
A B C D E F G DP
COL 1 SEL
COL 2 SEL
COL 3 SEL
COL 4 SEL
COL 5 SEL
COL 6 SEL
R30-R37
100Ω
+5
+8V
26 ULN2003
25 ULN2003
27 2981
24 74LS273
2A1
DO
D1
D2
D3
D4
D5
D6
D7
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
DIG 1 SEL
DIG 2 SEL
DIG 3 SEL
DIG 4 SEL
DIG 5 SEL
DIG 6 SEL
DIG 7 SEL
DIG 8 SEL
2B4 POR
CLR
2A3 DIG/KBD COL
CK
28 74LS273
SEG A
SEG B
SEG C
SEG D
SEG E
SEG F
SEG G
D. P.
POR 9
3A2 MONKY
3A2 RESET
39 LS11
2A3 SEG REG
43 LS04
67 USENSE
R42 330Ω
+5V R60 1K
3B1 SENS
R43 1K
37 LS14
C3 10 UF
35 7432
36 LS51
LS04
1A1 ENCASIN
1A1 ENCASIN
1A1 CASIN
1A4 MON
3A1 RTC
2B4 POR
1A1 INTACK
44 LS27
S6
51 LS109
50 LS00
R44 1K
+5V R63 1K
3B1 INT
R45 1K
C4 10 UF
+5V R59 1K
3B1 RST
R46 1K
C5 10 UF
+5V R61 1K
3B1 MON
R47 1K
C6 10UF
+5V R62 10K
73 PINT
R58 220Ω
R64 330Ω
B
A

REVISIONS				
ZONE	ISSUE	DESCRIPTION	DATE	APPROVAL







IC	VCC	GND
2650	39	21
2656	33.35	13
2112-2	16	8
74LS243	14	7
74LS244	20	10
74LS08	14	7
74LS00	14	7
74LS03	14	7
74LS175	16	8
LM393	8	4
74LS273	20	10
74LS253	16	8
74LS04	14	7
74LS02	14	7
74LS32	14	7
74LS11	14	7

IC	VCC	GND
74LS10	14	7
74LS139	16	8
74LS109	16	8
74LS161	16	8
82S123	16	8
74LS14	14	7
ULN2003	—	8
74LS27	14	7
74LS51	14	7
82S103	28	14
74LS74	14	7
7400	14	7

IC	+8V	GND
2981	9	10

QTY REQ	PART NUMBER	DESCRIPTION	ITEM NO
		LIST OF MATERIALS	

DRAWN ALTMAN 6-27-77	TITLE LOGIC DIAGRAM INSTRUCTOR	signetics CORPORATION PHONE 408 739-7700 811 EAST ARQUES AVE SUNNYVALE, CALIFORNIA	
DESIGNER			
CHECKED			
APPROVED			
TOLERANCES UNLESS OTHERWISE SPECIFIED: ANGULAR ± 30 FRAC ± 1/64 XX ± .005 XXX ± .001 XXXX ± .0005	MATERIAL	DWG NO 272026-2	REV A
	FINISH	SCALE	SHT 3 OF 4
	FACILITIES JOB NO		

DWG. NO. D

210-3020-102-230

DASH NO. | NEXT ASSY

2656 SMI PROGRAM CODING

FUNCTION SELECT	X0 E	X1 E	X2 E	X3 E	X4 E	X5 E	X6 E	X7 E			
	0	1	2	3	4	5	6	7	8	9	10
A0	X	X	X	1	1	X	X	X	1	X	X
A1	X	X	X	1	1	X	X	X	1	X	X
A2	X	X	X	1	1	X	X	X	1	X	X
A3	X	X	1	0	1	X	X	X	1	X	X
A4	X	X	1	0	1	X	X	X	1	X	X
A5	X	X	1	0	1	X	X	X	1	X	X
A6	X	X	1	0	1	X	X	X	1	X	X
A7	X	X	1	0	1	X	X	X	1	1	X
A8	0	1	X	X	1	X	X	X	1	1	X
A9	0	0	X	X	1	X	X	X	1	1	X
A10	0	0	X	X	1	X	X	X	1	1	X
A11	0	0	X	X	1	X	X	X	0	0	1
A12	0	0	X	X	0	X	X	1	1	1	1
A13	0	0	1	1	0	0	0	0	0	0	0
A14	0	0	X	X	0	0	1	0	0	0	0
OPREQ	1	1	1	1	1	1	1	X	1	1	1
M IO	1	1	0	0	1	0	0	X	1	1	1
WRP	1	1	1	1	1	1	1	X	1	1	1
	RAM 0 CE	RAM 1 CE	PORT FX	USR PORT	USR MEM	CTO	DTO	MONJ	SMI PORT	SMI RAM	SMI ROM

CLOCK SOURCE XTAL
CLOCK DIVIDE BY/4
NO DISABLE

82S123 PROGRAM TABLE

ADDRESS HEX	ADDRESS OCTAL	A4	A3	A2	A1	A0	OUTPUT B7	B6	B5	B4	B3	B2	B1	B0	
00	00	0	0	0	0	0	0	1	1	1	0	1	0	0	CØ
01	01	0	0	0	0	1	0	1	0	1	0	0	1	1	1F
02	02	0	0	0	1	0	0	1	0	1	0	0	1	0	18
03	03	0	0	0	1	1	0	1	0	1	0	0	0	0	ØØ
04	04	0	0	1	0	0	0	0	0	0	0	0	0	0	IDLE
05	05	0	0	1	0	1	0	0	0	0	0	0	0	0	IDLE
06	06	0	0	1	1	0	0	0	0	0	0	0	0	0	
07	07	0	0	1	1	1	0	0	0	0	0	0	0	0	IDLE
08	10	0	1	0	0	0	0	0	0	1	0	0	0	0	
09	11	0	1	0	0	1	0	0	0	1	0	0	0	0	
0A	12	0	1	0	1	0	0	0	0	1	0	0	0	0	
0B	13	0	1	0	1	1	0	0	0	1	0	0	0	0	
0C	14	0	1	1	0	0	0	0	0	1	0	0	0	0	
0D	15	0	1	1	0	1	0	0	0	1	0	0	0	0	
0E	16	0	1	1	1	0	0	0	0	1	0	0	0	0	
0F	17	0	1	1	1	1	0	0	0	1	0	0	0	0	
10	20	1	0	0	0	0	0	0	0	1	0	0	0	0	
11	21	1	0	0	0	1	0	0	0	1	0	0	0	0	DIRECT INT V
12	22	1	0	0	1	0	0	1	1	1	0	0	1	1	1F
13	23	1	0	0	1	1	0	1	0	1	0	0	1	0	18
14	24	1	0	1	0	0	0	1	0	1	0	0	0	0	ØØ
15	25	1	0	1	0	1	0	0	0	0	0	0	0	0	
16	26	1	0	1	1	0	0	0	0	0	0	0	0	0	
17	27	1	0	1	1	1	0	0	0	0	0	0	0	0	
18	30	1	1	0	0	0	0	0	0	0	0	0	0	0	
19	31	1	1	0	0	1	0	0	0	1	0	0	0	0	INDIRECT INT V
1A	32	1	1	0	1	0	0	0	0	1	0	0	0	0	HI ADDR
1B	33	1	1	0	1	1	0	0	0	1	0	0	0	0	LO ADDR
1C	34	1	1	1	0	0	0	1	1	1	0	0	1	1	1F
1D	35	1	1	1	0	1	0	1	0	1	0	0	1	0	18
1E	36	1	1	1	1	0	0	1	0	1	0	0	0	0	ØØ
1F	37	1	1	1	1	1	0	0	0	0	0	0	0	0	
							DATA BUS ENABLE	LAST ADDR LOAD	CNTR STOP			D6, D7	D3, D4	DØ, D1, D2	

SINGLE STEP

REVISIONS				
ZONE	ISSUE	DESCRIPTION	DATE	APPROVAL

82S1Ø3 PROGRAM TABLE

OUTPUT POLARITY		INPUT VARIABLE															
		IØ	I1	I2	I3	I4	I5	I6	I7	I8	I9	IA	IB	IC	ID	IE	IF	
FØ	L	H	H	–	–	–	–	H	H	H	H	H	H	H	H	H	–	ENABLE OUTPUT DATA
F1	L	H	L	–	–	–	–	H	H	H	H	H	H	H	H	H	–	ENABLE INPUT DATA
F2	H	–	H	L	–	–	–	H	H	H	H	H	H	H	H	H	–	$OUT
F3	H	–	L	L	–	–	–	H	H	H	H	H	H	H	H	H	–	$INP
F4	H	–	L	H	–	–	–	H	H	H	H	H	H	H	H	H	–	$MEMR
F5	L	H	H	–	H	–	–	H	H	H	H	H	H	H	H	H	–	PWR
F6	H	H	L	–	–	–	–	H	H	H	H	H	H	H	H	H	–	PDBIN
F7	H	H	H	H	–	–	–	H	H	H	H	H	H	H	H	H	–	MWRT
		OPREQ	R/W	M/IO	WRP	AD14D/C	AD13E/NE	PORTFX	USRPORT	USRMEM	MON	MEMINH	RAMØCE	RAM1CE	CIO	DIO		

ACTIVE HIGH = H
ACTIVE LOW = L

SINGLE STEP

CØ
IF
IB
ØØ
IDLE
IDLE

IDLE

ECT INT VECTOR
IF
IB
ØØ

DIRECT

INTERRUPT SPACE

RECT INT VECTOR
ADDR BYTE
ADDR BYTE
IF
IB
ØØ

INDIRECT

B

A

DWG. NO. D

QTY REQ	PART NUMBER	DESCRIPTION	ITEM NO
		LIST OF MATERIALS	

DRAWN	TITLE	signetics CORPORATION
DESIGNER	INSTRUCTOR PROGRAM TABLES	PHONE 408 739-7700 811 EAST ARQUES AVE. SUNNYVALE, CALIFORNIA
CHECKED		
APPROVED		
TOLERANCES UNLESS OTHERWISE SPECIFIED ANGULAR ± 30 FRAC ± 1/64 XXX ± .005 XX ± .01 XXXX ± .0005	MATERIAL / FINISH / FACILITIES JOB NO.	DWG NO. 272062-2 REV. A
		SCALE SHT 4 OF 4

210-3020-102-230

Appendix C

Instructor 50 monitor programma listings

```
TWIN ASSEMBLER VER 2.0    INSTRUCTOR 50 PRGM 11/1/78          PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0002               *PROGRAM WRITTEN BY DAVE WOTRING
0003               ********************************************************************
0004               *   EQUATE TABLES
0005               *
0006               *   REGISTER EQUATES
0007 0000          R0     EQU     0       REGISTER 0
0008 0001          R1     EQU     1       REGISTER 1
0009 0002          R2     EQU     2       REGISTER 2
0010 0003          R3     EQU     3       REGISTER 3
0011               *  CONDITION CODES
0012 0001          P      EQU     1       POSITIVE RESULT
0013 0000          Z      EQU     0       ZERO RESULT
0014 0002          NG     EQU     2       NEGATIVE RESULT
0015 0002          LT     EQU     2       LESS THAN
0016 0000          EQ     EQU     0       EQUAL TO
0017 0001          GT     EQU     1       GREATER THAN
0018 0003          UN     EQU     3       UNCONDITIONAL
0019               *  PSW LOWER EQUATES
0020 00C0          CC     EQU     H'C0'   CONDITIONAL CODES
0021 0020          IDC    EQU     H'20'   INTERDIGIT CARRY
0022 0010          RS     EQU     H'10'   REGISTER BANK
0023 0008          WC     EQU     H'08'   1=WITH 0=WITHOUT CARRY
0024 0004          OVF    EQU     H'04'   OVERFLOW
0025 0002          COM    EQU     H'02'   1=LOGIC 0=ARITHMETIC COMPARE
0026 0001          C      EQU     H'01'   CARRY/BORROW
0027               *  PSW UPPER EQUATES
0028 0080          SENS   EQU     H'80'   SENSE BIT
0029 0040          FLAG   EQU     H'40'   FLAG BIT
0030 0020          II     EQU     H'20'   INTERRUPT INHIBIT
0031 0007          SP     EQU     H'07'   STACK POINTER
0032               *  IO PORT DEFINITIONS
0033 0007          LEDS   EQU     H'07'   USER EXTENDED IO PORT
0034               *  INTERUPT VECTORS
0035 0007          UINTV  EQU     H'07'   USER DIRECT INTERUPT VECTOR
0036 0087          UINTVI EQU     H'87'   USER INDIRECT INTERUPT VECTOR
0037               * HARDWARE DEFINITIONS
0038 00FE          KBDIN  EQU     H'FE'   ADDRESS OF KBD IO PORT
0039 00F9          SEG    EQU     H'F9'   IO ADDRESS OF SEGMENT DRIVER
0040 00F9          DISP   EQU     SEG
0041 00FA          DIGIT  EQU     H'FA'   ADDRESS OF DIGIT ENABLE
0042 00F8          CTBYT  EQU     H'F8'   ADDRESS OF CONTROL BYTE
0043 00F8          CAS    EQU     CTBYT   ADDRESS OF CASSETTE INTERFACE
0044 00FB          OPRQCT EQU     H'FB'   ADDRESS OF OPREQ COUNTER
0045 00FD          LADRH  EQU     H'FD'   ADDRESS OF LAST ADDRESS REG HI BYTE
0046 00FC          LADRL  EQU     H'FC'   ADDRESS OF LAST ADDRESS REG LO BYTE
```

```
LINE ADDR  OBJECT  E SOURCE

0048               ************************************************
0049               *
0050 0000                ORG    H'1800'-64     TRAINING CARD RAM AREA
0051               *
0052 17C0          SCTCH RES    8       8 BYTE SCRATCH AREA
0053 17C6          TEMP  EQU    SCTCH+6 TEMP STORAGE
0054 17C8          EAD   RES    2       STOP ADDRESS FOR WCAS
0055 17CA          BAD   RES    2       BEGINNING ADDRESS FOR WCAS
0056 17CC          BPD   RES    1       DATA TO BE RESTORED IN BREAK LOC
0057 17CD          BPL   RES    2       ADDRESS OF BREAK POINT LOC
0058 17CF          BPF   RES    1       BREAK POINT SET FLAG
0059 17D0          SSF   RES    1       SINGLE STEP SET FLAG
0060 17D1          DISBUF RES   8       8 BYTE DISPLAY REGISTER
0061 17D9          SAVREG RES   4       A PLACE TO SAVE R0 THRU R3 OF ONE BANK
0062 17DD          MEM   RES    2       ADDRESS FOR ALTER OR PATCH COMMAND
0063 17DF          FID   RES    2       FILE ID FLAG AND FILE ID
0064 17E1          BCC   RES    1       BLOCK CHECK CHAR
0065 17E2          BSTT  RES    1       SAVE UNITS DIGIT
0066 17E3          T     RES    2       TEMP REGISTER
0067 17E5          T1    RES    1       TEMP REGISTER
0068 17E6          T2    RES    1       TEMP REGISTER
0069 17E7          T3    RES    1       TEMP REGISTER
0070 17E8          LADR  RES    2       COPY OF LAST ADDRESS REGISTER
0071 17EA          SLADR RES    2       SAVE LOCATION FOR LADR
0072 17EC          KFLG  RES    2       KBD SCAN FLAGS
0073 17EE                RES    1       KBD DEBOUNCE COUNT
0074 17EF          ALTF  RES    1       DISPLAY AND ALTER FLAG
0075 17F0          RESTF RES    1       RESTORE REGISTERS FLAG
0076 17F1          IFLG  RES    1       INTERUPT INHIBIT FLAG
0077 17F2          UREG  RES    12      STORAGE FOR USER REGISTERS
0078 17FE          PWRON RES    2       WHEN POWER ON THESE LOC CONTAIN H'5946'
0079               ************************************************
```

```
TWIN ASSEMBLER VER 2.0    INSTRUCTOR 50 PRGM 11/1/78          PAGE 0003

LINE ADDR  OBJECT  E SOURCE

0081                ************************************************************
0082 1800                 ORG H'1800'     BEGINING OF TRAINING CARD ROM AREA
0083                ************************************************************
0084                *
0085                *SAVE ALL REGISTERS UPON ENTRY TO PROGRAM
0086                *
0087                *REGISTERS USED
0088                *
0089                *R0 THRU R3' PSU PSL
0090                *
0091                *SUBROUTINES CALLED
0092                *
0093                *NONE
0094                *
0095                *RAM MEMORY USED
0096                *
0097                *UREG  = R0
0098                *UREG+1       = R1
0099                *UREG+2       = R2
0100                *UREG+3       = R3
0101                *UREG+4       = R1'
0102                *UREG+5       = R2'
0103                *UREG+6       = R3'
0104                *UREG+7       = PSU
0105                *UREG+8       = PSL
0106                *UREG+9       = PPSL INSTRUCTION OPCODE
0107                *UREG+10      = PSL
0108                *UREG+11      = RETC,UN       INSTRUCTION OPCODE
0109                *
0110                ************************************************************
0111                *
0112 1800 C870      SAVRG  STRR,R0 UREG    SAVE R0
0113 1802 13               SPSL           GET PSL
0114 1803 C875             STRR,R0 UREG+8  SAVE PSL
0115 1805 C875             STRR,R0 UREG+10 SAVE PSL        FOR RESTORE ROUTINE
0116 1807 12               SPSU           GET PSU
0117 1808 C86F             STRR,R0 UREG+7  SAVE PSU
0118 180A 7620             PPSU   II      SET INTERUPT INHIBIT
0119 180C 7510             CPSL   RS      CLEAR REGISTER SWITCH
0120 180E C963             STRR,R1 UREG+1  SAVE R1
0121 1810 CA62             STRR,R2 UREG+2  SAVE R2
0122 1812 CB61             STRR,R3 UREG+3  SAVE R3
0123 1814 7710             PPSL   RS      SET REGISTER SWITCH
0124 1816 C95E             STRR,R1 UREG+4  SAVE R1'
0125 1818 CA5D             STRR,R2 UREG+5  SAVE R2'
0126 181A CB5C             STRR,R3 UREG+6  SAVE R3'
0127 181C 75FF             CPSL   255     CLEAR PSL
0128 181E 7702             PPSL   COM     DO LOGICAL COMPARES
0129 1820 54FD             REDE,R0 LADRH   GET LAST ADDRESS HI BYTE
0130 1822 CC17E8           STRA,R0 LADR    SAVE IN MEMORY
0131 1825 54FC             REDE,R0 LADRL   GET LAST ADDRESS LO BYTE
0132 1827 CC17E9           STRA,R0 LADR+1  SAVE IT
0133 182A 20               EORZ   R0      GET A 0
0134 182B CC17F1           STRA,R0 IFLG    CLEAR INTERUPT INHIBIT FLAG
```

```
LINE ADDR  OBJECT  E SOURCE

0136               ****************************************************************
0137               *
0138               *
0139               *
0140               *PROGRAM ENTRY ROUTINE
0141               *
0142               *
0143               *DECIDES HOW REACHED ENTRY POINT OF PROGRAM
0144               *
0145               *1 POWER ON
0146               *2 SINGLE STEP
0147               *3 MONITOR PUSHBUTTON ON KEY BOARD
0148               *4 BREAKPOINT
0149               *
0150               *
0151               *
0152               *
0153               ****************************************************
0154               *
0155 182E 084E     BEG   LODR,R0 PWRON   CHECK POWER ON FLAG
0156 1830 E459           COMI,R0 H'59'   AFTER POWER VALUE OF FLAG IS
0157               *                     H'5946'
0158 1832 9813     BEG1  BCFR,EQ INIT    IF NOT CORRECT THIS IS POWER ON
0159               *                     GO INITIALIZE THE MONITOR FLAGS
0160 1834 0849     BEG3  LODR,R0 PWRON+1 CHECK LO BYTE OF POWER ON FLAG
0161 1836 E446           COMI,R0 H'46'   IS SECOND BYTE CORRECT
0162 1838 980D           BCFR,EQ INIT IF NOT INITIALIZE THE PROGRAM
0163 183A 0C17D0         LODA,R0 SSF     CHECK THE SINGLE STEP FLAG
0164 183D 9C19C9         BCFA,EQ SGLSTP  IF FLAG THEN GO SINGLE STEP
0165 1840 0899           LODR,R0 *MON+1     SEE IF BREAK POINT ENABLED
0166 1842 9C197A   BEG2  BCFA,EQ BRKPT  GO EXECUTE THE BREAK POINT ROUTINE
0167 1845 1B13           BCTR,UN MON     MUST BE MONITOR KEY
0168               *
```

```
LINE ADDR  OBJECT  E SOURCE

0170                ******************************************************
0171                *
0172                *
0173                *KEY BOARD MONITOR ROUTINE
0174                *
0175                *
0176                *REGISTERS USED
0177                *
0178                *R0 SCRATCH
0179                *R1 SCRATCH
0180                *R2 SCRATCH
0181                *R3 NOT USED
0182                *
0183                *SUBROUTINES USED
0184                *
0185                *MOV MOVE DATA TO DISPLAY BUFFER
0186                *DISPLY DISPLAY MESSAGE AND KEY BOARD SCAN
0187                *
0188                *
0189                *RAM MEMORY USED
0190                *
0191                *PWRON POWER ON FLAG
0192                *SSF   SINGLE STEP FLAG
0193                *BPF   BREAK POINT FLAG
0194                *BPL   BREAK POINT LOCATION
0195                *
0196                ***********************************************
0197                *
0198 1847 0459      INIT   LODI,R0 H'59'   SET THE POWER ON FLAG
0199 1849 CC17FE           STRA,R0 PWRON   TO POWER ON VALUE H'5946'
0200 184C 0446             LODI,R0 H'46'
0201 184E CC17FF           STRA,R0 PWRON+1
0202 1851 20               EORZ    R0      GET A 0
0203 1852 CC17DD           STRA,R0 MEM     PRESET INDIRECT ADDRESS MEM
0204 1855 CC17DE           STRA,R0 MEM+1
0205 1858 C881             STRR,R0 *MON+1  CLEAR BREAK POINT FLAG
0206 185A 0C17CF    MON    LODA,R0 BPF     GET BREAK POINT FLAG
0207 185D 180F             BCTR,EQ MON5    BREAK POINT NOT SET
0208 185F 0C17CC           LODA,R0 BPD     GET BREAK POINT DATA
0209 1862 CC97CD           STRA,R0 *BPL    CLEAR BREAK POINT
0210 1865 EC97CD           COMA,R0 *BPL    CHECK DATA STORED CORRECTLY
0211 1868 1804             BCTR,EQ MON5    BREAK POINT CLEARED OK
0212 186A 0701             LODI,R3 1       BREAK POINT DIDN'T CLEAR
0213 186C 9BE8             ZBRR    *ERR    GOTO ERROR
0214 186E 20        MON5   EORZ    R0      GET A 0
0215 186F D4F8             WRTE,R0 CTBYT   CLEAR CONTROL BYTE
0216 1871 CC17D0           STRA,R0 SSF     CLEAR SINGLE STEP FLAG
0217 1874 051F      MON3   LODI,R1 <HELLO-1        GET ADDRESS OF HELLO MESSAGE
0218 1876 0694             LODI,R2 >HELLO-1
0219 1878 BBFE      MON1   ZBSR *MOV      MOVE MESSAGE TO DISBUF
0220 187A 20        MON4   EORZ    R0      SET FLAG TO WAIT FOR ENTRY
0221 187B BBEC             ZBSR    *DISPLY DISPLAY MESSAGE AND SCAN KEYBOARD
0222 187D F680      MON2   TMI,R2  H'80'   CHECK COMMAND FLAG
0223 187F 9816             BCFR,EQ ERR2     IF FLAG NOT SET ERROR
0224 1881 460F             ANDI,R2 H'0F'   MASK COMMAND VALUE
```

```
LINE ADDR  OBJECT  E SOURCE

0225 1883 E607            COMI,R2 7       MAX COMMAND VALUE
0226 1885 1910            BCTR,GT ERR2     ERROR CODE VALUE TO LARGE
0227 1887 D2              RRL,R2          MULTIPLY INDEX BY 2
0228 1888 0E78A4          LODA,R0 CMD,R2  SET UP AN INDIRECT ADDRESS
0229 188B CC17E3          STRA,R0 T       TO THE FUNCTION WANTED
0230 188E 0E78A5          LODA,R0 CMD+1,R2
0231 1891 CC17E4          STRA,R0 T+1
0232 1894 1F97E3          BCTA,UN *T      EXECUTE A COMMAND
0233                 *
0234                 *
0235 1897 0702       ERR2 LODI,R3 2       INVALID COMMAND SEQUENCE
0236 1899 051F       ERRI LODI,R1 <ERROR-1  GET ADDRESS OF ERROR MESSAGE
0237 189B 0684            LODI,R2 >ERROR-1
0238 189D BBFE            ZBSR    *MOV    MOVE MESSAGE TO DISBUF
0239 189F CF17D8          STRA,R3 DISBUF+7       WRITE THE ERROR NUMBER
0240 18A2 1B56            BCTR,UN MON4    GO LOOK FOR NEW COMMAND
0241                 *
0242                 *COMMAND ADDRESS TABLE
0243                 *
0244 18A4 1C91       CMD  ACON    WCAS    WRITE CASSETTE COMMAND
0245 18A6 1D61            ACON    SCBP     BREAK POINT COMMAND
0246 18A8 1BAC            ACON    RCAS    READ CASSETTE COMMAND
0247 18AA 1A7E            ACON    REG     REGISTER DISPLAY AND ALTER COMMAND
0248 18AC 18B4            ACON    SSTEP      SINGLE STEP COMMAND
0249 18AE 1A0C            ACON    ALTER    DISPLAY AND ALTER MEMORY
0250 18B0 1E59            ACON    GO      GOTO COMMAND)
0251 18B2 187A            ACON    MON4    ENTR/NEXT KEY IS NOT COMMAND
0252                 *
```

```
LINE ADDR  OBJECT  E SOURCE

0254               ************************************************************
0255               *
0256               *
0257               *SINGLE STEP ROUTINES
0258               *THIS ROUTINE WRITTEN BY BBC
0259               *
0260               * PROCESSOR TRANSFERS CONTROL TO USER PROGRAM
0261               * AFTER COMPUTING THE NUMBER OF OPREQ'S TILL
0262               * THE NEXT INSTRUCTION FETCH.
0263               *
0264               *
0265               *REGISTERS USED
0266               *
0267               *R0 THRU R3 SCRATCH
0268               *
0269               *
0270               *SUBROUTINES CALLED
0271               *
0272               *RLADR RESTORE LAST ADDRESS REGISTER
0273               *
0274               *
0275               *RAM LOCATIONS USED
0276               *
0277               *LADR  LAST ADDRESS REGISTER
0278               *T3    TEMP REGISTER
0279               *TEMP  TEMP REGISTER
0280               *SCTCH SCRATCH REGISTER
0281               *
0282               ****************************************************************
0283 0000          OVHD   EQU     0      NEGATIVE NUMBER OF OPREQ'S
0284               *
0285               *
0286               *CHECK IF NEXT SINGLE STEP IS IN MONITOR AREA
0287               *
0288 18B4 0C17E8   SSTEP  LODA,R0 LADR   GET MSB OF LADR
0289 18B7 E410            COMI,R0 H'10'  IS ADDRESS LT H'1000'
0290 18B9 1A08            BCTR,LT SSTEP1 GO SINGLE STEP
0291 18BB E420            COMI,R0 H'20'  IS ADDRESS GT OR EQ H'2000'
0292 18BD 9A04            BCFR,LT SSTEP1 GO SINGLE STEP
0293 18BF 0709            LODI,R3 9      NEXT SINGLE STEP ENTERS MONITOR
0294 18C1 9BE8            ZBRR    *ERR   GOTO ERROR
0295 18C3 047F     SSTEP1 LODI,R0 127    SET SINGLE STEP FLAG
0296 18C5 CC17D0          STRA,R0 SSF    STORE IT
0297 18C8 0420            LODI,R0 H'20'  SET THE INTERUPT INHIBIT
0298 18CA CC17F1          STRA,R0 IFLG   SAVE IN INTERUPT INHIBIT FLAG
0299 18CD 7508            CPSL WC        CLEAR WITH CARRY IF SET
0300 18CF 0601     SSTEP2 LODI,R2 1      SET INDEX
0301 18D1 0EF7E8          LODA,R0 *LADR,R2      GET SECOND BYTE OF INSTRUCTION
0302 18D4 CC17E7          STRA,R0 T3     SAVE IT FOR LATER
0303 18D7 0F97E8          LODA,R3 *LADR  GET NEXT INSTRUCTION
0304 18DA 03              LODZ    R3     SAVE INSTRUCTION IN R0
0305 18DB 471C            ANDI,R3 H'1C'  EXTRACT INSTRUCTION CLASS
0306 18DD 0500            LODI,R1 OVHD   SET OVERHEAD OPREQ COUNT
0307 18DF 0605            LODI,R2 5      SHIFT OR MOVE COUNT
0308 18E1 F420            TMI,R0  H'20'  TEST FOR ODD OPCODE IN CLASS4
```

```
LINE ADDR  OBJECT  E SOURCE

0309 18E3 9F1902          BXA    CBRTB,R3       BRANCH TO CLASS PROCESSOR
0310                *
0311                * CLASS 5.     MIXED NUMBER OF OPREQ'S.
0312                *
0313 18E6 FA2F      CL5B  BDRR,R2 CL5A
0314 18E8 4707            ANDI,R3 H'07'  MASK TO OPCODE
0315 18EA 0F7967          LODA,R0 CL5TB,R3       GET NUMBER OF OPREQ'S FROM TABLE.
0316 18ED C1              STRZ   R1
0317                *
0318                * WRITE OPREQ COUNT AND EXIT TO USER
0319                *
0320 18EE 1F195E    EXIT  BCTA,UN EXIT4    GOTO USER
0321                *
0322                * RETURN FROM TEST BRANCH, IF BRANCH TAKEN
0323                *
0324 18F1 3F196F    BRCH  BSTA,UN RLADR    RESTORE LAST ADDRESS REGISTER
0325 18F4 0D17C6    BRCH1 LODA,R1 TEMP     GET OPREQ COUNT BACK AFTER TEST BRANCH
0326 18F7 7508            CPSL   WC       CLEAR PSL WC BIT
0327                *
0328                * ROUTINE TO ADD 2 OPREQ'S IF INDIRECT APPLIES.
0329                *
0330 18F9 0C17E7    CIND  LODA,R0 T3       GET SECOND BYTE OF INSTRUCTION
0331 18FC F480            TMI,R0  H'80'  TEST INDIRECT BIT
0332 18FE 1806            BCTR,0  PLS2   SET, ADD 2 OPREQ'S
0333 1900 1B6C            BCTR,UN EXIT   NOT SET, DO NOT ADD
0334                *
0335                * CLASS PROCESSOR TABLE.
0336                *
0337 1902           CBRTB EQU    $
0338                *
0339 1902 A501      PLS1  SUBI,R1 1        CLASS 0.        1 OPREQ
0340 1904 1B68            BCTR,UN EXIT
0341 1906 A502      PLS2  SUBI,R1 2        CLASS 1.        2 OPREQ'S
0342 1908 1B64            BCTR,UN EXIT
0343 190A A503            SUBI,R1 3        CLASS 2.        3 OPREQ'S + INDIRECT
0344 190C 1B6B            BCTR,UN CIND
0345 190E A504            SUBI,R1 4        CLASS 3.        4 OPREQ'S + INDIRECT
0346 1910 1B67            BCTR,UN CIND
0347 1912 1872            BCTR,EQ PLS2     CLASS4          2 OPREQS IF OPCODE ODD
0348 1914 1B6C            BCTR,UN PLS1                     1 OPREQ IF OPCODE EVEN
0349 1916 C3              STRZ   R3        CLASS 5.        MIXED NUMBER OF OPREQ'S
0350 1917 53        CL5A  RRR,R3           SHIFT OPCODE TO LOW BYTE
0351 1918 1B4C            BCTR,UN CL5B     AND LOOK UP IN TABLE
0352 191A 8501            ADDI,R1 1        CLASS 6.        2 OPREQ'S + IND IF BRANCH TAKEN
0353 191C 6404            IORI,R0 H'04'  CONVERT TO CLASS 7.
0354 191E A503            SUBI,R1 3        CLASS 7.        3 OPREQ'S + IND IF BRANCH TAKEN
0355                *
0356                * CLASS 6 AND 7.
0357                * ADD 2 OPREQ'S IF INDIRECT AND BRANCH IS TAKEN.
0358                *
0359 1920 C9D3            STRR,R1 *BRCH1+1 SAVE PRESENT NUMBER OF OPREQ'S IN TEMP
0360 1922 D5FB            WRTE,R1 OPRQCT   ALSO OUTPUT TO HARDWARE
0361 1924 F440            TMI,R0  H'40'  TEST FOR REGISTER CLASS
0362 1926 1804            BCTR,0  CL67B  IF SO, DO NOT TEST FOR UNCONDITIONAL
0363 1928 F403            TMI,R0  H'03'  IS BRANCH UNCONDITIONAL
0364 192A 184D            BCTR,0  CIND   IF SO, DO NOT TEST BRANCH
```

```
LINE ADDR  OBJECT  E SOURCE

0365 192C F4E0      CL67B  TMI,R0  H'E0'   TEST FOR BDR INST
0366 192E 1802             BCTR,0  CL67C   IF SO, DO NOT REMOVE 'SUBROUTINE' BIT
0367 1930 44DF             ANDI,R0 H'DF'   REMOVE SUBROUTINE BIT FROM OPCODE.
0368 1932 CC17C0    CL67C  STRA,R0 SCTCH   STORE IN TEST AREA
0369 1935 0E7951    MVCODE LODA,R0 BRCD,R2 GET ROM CODE
0370 1938 CE77C0           STRA,R0 SCTCH,R2  STORE IN RAM
0371 193B FA78             BDRR,R2 MVCODE  DO UNTILL ALL HAS BEEN MOVED.
0372                *
0373                *SAVE LADR
0374                *
0375 193D 0C17E8    SLAD1  LODA,R0 LADR    GET LAST ADDRESS REG
0376 1940 C8AE             STRR,R0 *RLADR+1   SAVE IT
0377 1942 0C17E9    SLAD2  LODA,R0 LADR+1
0378 1945 C8AE             STRR,R0 *RLADR1+1
0379 1947 0417             LODI,R0 <SCTCH  GET ADDRESS SCRATCH
0380 1949 C8F3             STRR,R0 *SLAD1+1
0381 194B 04C0             LODI,R0 >SCTCH
0382 194D CC9943           STRA,R0 *SLAD2+1
0383 1950 1B93             BCTR,UN *EXIT3+1      DO TEST BRANCH
0384                *
0385                *THIS IS CODE FOR TEST BRANCH
0386                *
0387 1951           BRCD   EQU     $-1
0388 1952 18F1             ACON    BRCH    ADDRESS FOR TEST BRANCH
0389 1954 1F1957           BCTA,UN EXIT2      RETURN IF BRANCH NOT TAKEN
0390                *
0391 1957 3B16      EXIT2  BSTR,UN RLADR   RESTORE LAST ADDRESS REG
0392 1959 0D17C6           LODA,R1 TEMP    GET OPREQ COUNT
0393 195C 7508             CPSL    WC      CLEAR WITH CARRY
0394 195E D5FB      EXIT4  WRTE,R1 OPRQCT  SET THE OPREQ COUNTER
0395 1960 20               EORZ    R0      CLEAR INTERUPT INHIBIT FLAG
0396 1961 CC17F1           STRA,R0 IFLG    SAVE IN INTERUPT INHIBIT FLAG
0397 1964 1F1E59    EXIT3  BCTA,UN GO
0398                *
0399                * CLASS 5 OPREQ TABLE
0400                *
0401 1967 FF        CL5TB  DATA    OVHD-1       RETC
0402 1968 FF               DATA    OVHD-1       RETE
0403 1969 FD               DATA    OVHD-3       REDE
0404 196A FE               DATA    OVHD-2       C-P PSW
0405 196B FF               DATA    OVHD-1       DAR
0406 196C FE               DATA    OVHD-2       TPSW
0407 196D FD               DATA    OVHD-3       WRTE
0408 196E FE               DATA    OVHD-2       TMI
0409                *
0410                *RLADR RESTORE LAST ADDRESS REG
0411                *
0412 196F 0C17EA    RLADR  LODA,R0 SLADR   GET SAVED LADR
0413 1972 C8CA             STRR,R0 *SLAD1+1
0414 1974 0C17EB    RLADR1 LODA,R0 SLADR+1
0415 1977 C8CA             STRR,R0 *SLAD2+1
0416 1979 17               RETC,UN
0417                *
```

```
LINE ADDR  OBJECT  E SOURCE

0419               ************************************************************************
0420               *
0421               *
0422               *BREAK POINT AND SINGLE STEP RUN TIME CODE
0423               *
0424               *
0425               *SINGLE STEP
0426               *
0427               *WHEN ENTERED AT SINGLE STEP.  SINGLE STEP FLAG IS CLEARED
0428               *AND DISPLAY IS ' ADDR DD'
0429               *
0430               *
0431               *WHEN ENTERED AT BREAK POINT AND BREAK POINT IS SET AND MATCHES
0432               *BREAK POINT REGISTER.  THE DISPLAY IS '-ADDR DD'
0433               *
0434               *
0435               *REGISTER USED
0436               *
0437               *R0
0438               *
0439               *SUBROUTINE CALLED
0440               *
0441               *DLSLD PREPARE BINARY DATA FOR DISPLAY
0442               *
0443               *
0444               *RAM MEMORY USED
0445               *
0446               *DISBUF DISPLAY BUFFER
0447               *BPF BREAK POINT FLAG
0448               *BPL BREAK POINT LOCATION
0449               *BPD DATA FOR BREAK POINT LOCATION
0450               *LADR COPY OF LAST ADDDRESS REGISTER
0451               *SSF SINGLE STEPFLAG
0452               *
0453               ************************************************************************
0454               *
0455 197A 0C17E8  BRKPT  LODA,R0 LADR    GET LAST ADDRESS REGISTER
0456 197D 0D17E9  BRK3   LODA,R1 LADR+1
0457 1980 7709           PPSL    C+WC    SET CARRY AND WITH CARRY
0458 1982 A501           SUBI,R1 1       DECREMENT LAST ADDRESS REG
0459 1984 A400           SUBI,R0 0       SO CAN COMPARE TO BREAK POINT REGISTER
0460 1986 447F           ANDI,R0 H'7F    MASK OFF UNUSED BIT
0461 1988 7509           CPSL    C+WC    CLEAR CARRY AND WITH CARRY
0462 198A E8AF    BRK2   COMR,R0 *BRKPT2+1 COMPARE WITH BPL
0463 198C 9C185A  BRK1   BCFA,EQ MON    NO COMPARE
0464 198F E9AF           COMR,R1 *BRKPT1+1 COMPARE WITH BPL+1
0465 1991 98FA           BCFR,EQ *BRK1+1   NO COMPARE
0466 1993 C8E6           STRR,R0 *BRKPT+1        IF COMPARE UP DATE PC
0467 1995 C9E7           STRR,R1 *BRK3+1
0468 1997 0C17CC         LODA,R0 BPD     IF COMAPRE CLEAR BREAK POINT
0469 199A CC97CD         STRA,R0 *BPL
0470 199D EC97CD         COMA,R0 *BPL    ERROR CHECK OF DATA WRITTEN
0471 19A0 1804           BCTR,EQ BRK0    DATA STORED OK
0472 19A2 0701           LODI,R3 1       BREAK POINT NOT CLEARED OK
0473 19A4 9BE8           ZBRR    *ERR
```

```
LINE ADDR  OBJECT  E SOURCE

0474 19A6 E440      BRK0   COMI,R0 H'40'   HALT INSTRUCTION OPCODE
0475 19A8 1808             BCTR,EQ BRKPT9  IF HALT DON'T DO HIDDEN SINGLE STEP
0476 19AA 0480             LODI,R0 H'80'   SET FLAG FOR HIDDEN SINGLE STEP
0477 19AC CC17CF    BRKPT3 STRA,R0 BPF     SET FLAG IN BREAK POINT
0478 19AF 1F18B4           BCTA,UN SSTEP   EXECUTE ONE USER INSTRUCTION
0479 19B2 047F      BRKPT9 LODI,R0 127     SET BREAK POINT FLAG
0480 19B4 C8F7             STRR,R0 *BRKPT3+1
0481 19B6 0419             LODI,R0 H'19'   DASH SYMBOL FOR BREAK POINT
0482 19B8 C89A             STRR,R0 *BRKPT8+1  SET THE DASH SYMBOL IN DISBUF
0483 19BA 0C17CD    BRKPT2 LODA,R0 BPL     GET BREAK POINT ADDRESS
0484 19BD 3B32             BSTR,UN BRKPT7  SET THE DISPLAY
0485 19BF 0C17CE    BRKPT1 LODA,R0 BPL+1
0486 19C2 3B36             BSTR,UN BRKPT6
0487 19C4 0C97CD           LODA,R0 *BPL    GET INSTRUCTION OPCODE
0488 19C7 1B1A             BCTR,UN BRKPT5
0489                *
0490                *ENTRY POINT FOR SINGLE STEP
0491                *
0492 19C9 20        SGLSTP EORZ    R0      GET A 0
0493 19CA CC17D0           STRA,R0 SSF     CLEAR SINGLE STEP FLAG
0494 19CD 08DE             LODR,R0 *BRKPT3+1       CHECK BREAK POINT FLAG
0495 19CF 1A61             BCTR,NG BRKPT9  DID A HIDDEN SINGLE STEP
0496                *                      DISPLAY THE BREAK POINT
0497 19D1 0417      SGLST9 LODI,R0 H'17'   BLANK SYMBOL
0498 19D3 CC17D1    BRKPT8 STRA,R0 DISBUF  SET DISPLAY BUFFER
0499 19D6 0C17E8           LODA,R0 LADR    GET ADDRESS
0500 19D9 3B16             BSTR,UN BRKPT7  SET THE DISPLAY
0501 19DB 0C17E9           LODA,R0 LADR+1
0502 19DE 3B1A             BSTR,UN BRKPT6  SET THE DISPLAY
0503 19E0 0C97E8           LODA,R0 *LADR   GET INSTRUCTION DATA
0504 19E3 3B03      BRKPT5  BSTR,UN BRKPTI  SET UP DISPLAY
0505 19E5 1F187A           BCTA,UN MON4     GOTO MONITOR
0506                *
0507                *SET UP DISBUF 6&7
0508                *
0509 19E8 BBF4      BRKPTI ZBSR    *DISLSD CONVERT TO BIN FOR DISPLAY
0510 19EA CC17D7           STRA,R0 DISBUF+6
0511 19ED CD17D8           STRA,R1 DISBUF+7
0512 19F0 17               RETC,UN
0513                *
0514                *SET UP DISBUF 1&2
0515                *
0516 19F1 BBF4      BRKPT7 ZBSR    *DISLSD CONVERT BIN TO DISPLAY
0517 19F3 CC17D2           STRA,R0 DISBUF+1
0518 19F6 CD17D3           STRA,R1 DISBUF+2
0519 19F9 17               RETC,UN
0520                *
0521                *SETUP DISBUF 3&4
0522                *
0523 19FA BBF4      BRKPT6 ZBSR    *DISLSD CONVERT BIN TO DISPLAY
0524 19FC CC17D4           STRA,R0 DISBUF+3        STORE DATA
0525 19FF CD17D5           STRA,R1 DISBUF+4
0526 1A02 0417             LODI,R0 H'17'   BLANK SYMBOL
0527 1A04 CC17D6           STRA,R0 DISBUF+5
0528 1A07 17               RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

0530               ********************************************************
0531               *
0532               *
0533               *DISPLAY AND ALTER MEMORY ROUTINE
0534               *PATCH MEMORY ROUTINE
0535               *
0536               *
0537               *REGISTERS USED
0538               *
0539               *R0 SCRATCH
0540               *R1 SCRATCH
0541               *R2 SCRATCH
0542               *R3 SCRATCH
0543               *
0544               *SUBROUTINES CALLED
0545               *
0546               *GAD GET ADDRESS PARAMETER
0547               *GNP GET NUMBER PARAMETER
0548               *ROT ROTATE R0 1 NIBBLE LEFT
0549               *BRKPT4        SETUP DISPLAY 6&7
0550               *BRKPT6        SETUP DISPLAY 3&4
0551               *BRKPT7        SETUP DISPLAY 1&2
0552               *
0553               *RAM MEMORY USED
0554               *
0555               *MEM INDIRECT ADDRESS MEMORY POINTER
0556               *ALTF ALTER FLAG = 1 FOR DISPLAY AND ALTER
0557               *                  3 OR 5 FOR PATCH
0558               *
0559               ********************************************************
0560               *
0561               *
0562               *ENTRY POINT FOR PATCH COMMAND
0563               *
0564 1A08 0403    PTCH   LODI,R0 3       SET ALTER FLAG TO PATCH
0565 1A0A 1B02           BCTR,UN ALTER5
0566               *
0567               *ENTRY POINT FOR DISPLAY AND ALTER COMMAND
0568               *
0569 1A0C 0401    ALTER  LODI,R0 1       SET ALTER FLAG TO ALTER
0570 1A0E C8A4    ALTER5 STRR,R0 *ALTER1+1       STORE IN ALTF
0571 1A10 3F1B04         BSTA,UN GAD     DISPLAY AD= AND WAIT TILL DIGITS ENTERED
0572 1A13 E687           COMI,R2 H'87'   ENTR/NXT?
0573 1A15 9C187D         BCFA,EQ MON2    NEW FUNCTION ABORT ALTER COMMAND
0574 1A18 5B0E           BRNR,R3 ALTER4  NO ADDRESS ENTERED CONTINUE FROM LAST LOCATION
0575 1A1A C88D           STRR,R0 *ALTER4+1 MEM+1   SAVE ADDRESS DATA
0576 1A1C C981           STRR,R1 *AL1+1
0577 1A1E 0E17DD  AL1    LODA,R2 MEM     GET DATA
0578 1A21 0717           LODI,R3 H'17'      BLANK
0579 1A23 CF17D8         STRA,R3 DISBUF+7       CLEAR DISPLAY
0580 1A26 1B05           BCTR,UN ALTER2  SET UP DISPLAY
0581               *
0582               *NO ADDRESS CONTINUE FROM LAST ADDRESS
0583               *
0584 1A28 0C17DE  ALTER4 LODA,R0 MEM+1   GET ADDRESS
```

```
LINE ADDR  OBJECT  E SOURCE

0585 1A2B 0AF2            LODR,R2 *AL1+1 MEM
0586                 *
0587                 *UPDATE THE DISPLAY
0588                 *
0589 1A2D 3B4B       ALTER2 BSTR,UN BRKPT6  SET UP ADDRESS DISPLAY
0590 1A2F 02              LODZ    R2       GET MSD
0591 1A30 3F19F1          BSTA,UN BRKPT7   SET UP DISPLAY
0592                 *
0593                 *
0594 1A33 0C17EF     ALTER1 LODA,R0 ALTF     CHECK ALTER FLAG
0595 1A36 E401            COMI,R0 1         PATCH COMMAND
0596 1A38 1807            BCTR,EQ ALTER8    NOT PATCH
0597 1A3A 0417            LODI,R0 H'17'     BLANK CHAR
0598 1A3C CC17D7          STRA,R0 DISBUF+6
0599 1A3F 1B05            BCTR,UN ALTER9    PATCH COMMAND
0600                 *
0601 1A41 0C97DD     ALTER8 LODA,R0 *MEM     GET THE DATA
0602 1A44 BBEA            ZBSR  *BRKPT4  SET UP DATA VALUE DISPLAY
0603 1A46 08EC       ALTER9 LODR,R0 *ALTER1+1    SET FLAG TO SINGLE BYTE DATA
0604 1A48 BBFC            ZBSR *GNPA    DISPLAY BUFFER AND WAIT FOR NEW ENTRY
0605 1A4A 5B0C            BRNR,R3 ALTER3   NO DATA
0606 1A4C CC97DD          STRA,R0 *MEM    CHANGE DATA IN LOCATION
0607 1A4F EC97DD          COMA,R0 *MEM    CHECK DATA STORED OK
0608 1A52 1804            BCTR,EQ ALTER3  DATA STORED OK
0609 1A54 0703            LODI,R3 3       ALTER OR PATCH WRITE ERROR
0610 1A56 9BE8            ZBRR    *ERR    GOTO ERROR
0611                 *
0612                 *EXIT FROM COMMAND
0613                 *
0614 1A58 08DA       ALTER3 LODR,R0 *ALTER1+1    EXIT FROM ALTER OR PATCH
0615 1A5A E401            COMI,R0 1       IS IT PATCH
0616 1A5C 9807            BCFR,EQ ALTER6  IF YES TAKE THIS BRANCH
0617 1A5E E687            COMI,R2 H'87'   IS IT ALTER NEXT KEY FUNCTION?
0618 1A60 9C187D     AL2    BCFA,EQ MON2    GO TO MONITOR NEW COMMAND
0619 1A63 1B0B            BCTR,UN ALTER7   GO UPDATE THE DISPLAY
0620                 *
0621                 *EXIT FROM PATCH
0622                 *
0623 1A65 E60F       ALTER6 COMI,R2 H'0F'   WAS LAST KEY FUNCTION KEY
0624 1A67 19F8            BCTR,GT *AL2+1 MON2    FUNCTION KEY WAS LAST GO TO MONITOR
0625 1A69 0405            LODI,R0 5       RETURN ON SECOND DIGIT FLAG
0626 1A6B C8C7            STRR,R0 *ALTER1+1       SAVE IN ALTF
0627 1A6D CE17D8          STRA,R2 DISBUF+7        SET DISPLAY
0628                 *
0629                 *INCREMENT INDIRECT ADDRESS
0630                 *
0631 1A70 3F1C55     ALTER7 BSTA,UN INK     INCREMENT THE ADDRESS
0632 1A73 1F1A2D          BCTA,UN ALTER2
0633                 *
0634                 *PREPARE BIN DATA FOR DISPLAY
0635                 *
0636 1A76 C1         DISLSI STRZ    R1       SAVE NUMBER IN R0
0637 1A77 44F0            ANDI,R0 H'F0'   MASK FOR MSD
0638 1A79 450F            ANDI,R1 H'0F'   MASK FOR LSD
0639 1A7B BBF6            ZBSR    *ROT    ROTATE A NIBBLE
0640 1A7D 17              RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

0642               ************************************************************
0643               *
0644               *
0645               *DISPLAY AND ALTER REGISTERS COMMAND
0646               *
0647               *THE DISPLAY AND ALTER REGISTERS COMMAND ALLOWS
0648               *THE USER TO EXAMINE AND ALTER R0,R1,R2,R3,R1',R2',R3',PSU,PSL,PC
0649               *
0650               *THIS COMMAND ALSO PROVIDES ENTRY POINT TO ALTERNATE FUNCTIONS
0651               *REG 9 NOT DEFINED
0652               *REG A ADJUST CASSETTE COMMAND
0653               *REG B NOT DEFINED
0654               *REG D NOT DEFINED
0655               *REG E NOT DEFINED
0656               *REG F ENTER THE FAST PATCH MODE
0657               *
0658               *REGISTERS USED
0659               *
0660               *R0 SCRATCH
0661               *R1 SCRATCH
0662               *R2 SCRATCH
0663               *R3 SCRATCH
0664               *
0665               *SUBROUTINES CALLED
0666               *
0667               *MOV   MOVE DATA TO DISBUF
0668               *GNP   GET NUMERIC PARAMETERS
0669               *ROT   ROTATE A NIBBLE
0670               *GNPA  DISPLAY AND GET NUMERIC PARAMETERS
0671               *BRKPT4        SET DISPLAY 6&7
0672               *SCBP2 SET DISPLAY 4&5
0673               *
0674               *RAM MEMORY USED
0675               *
0676               *DISBUF        DISPLAY BUFFER
0677               *UREG  USER REGISTERS
0678               *LADR  LAST ADDRESS REGISTER PC COUNTER
0679               *T2  TEMP REGISTER
0680               *
0681               ************************************************************
0682 1A7E 051F     REG    LODI,R1 <REQ-1  GET ADDRESS OF R= DISPLAY
0683 1A80 06A4            LODI,R2 >REQ-1
0684 1A82 BBFE            ZBSR *MOV     MOVE DATA TO DISBUF
0685 1A84 20              EORZ    R0      SET FLAG TO RETURN AFTER KEY PRESSED
0686 1A85 BBEC            ZBSR    *DISPLY
0687               *
0688 1A87 F480            TMI,R0  H'80'   SEE IF FUNCTION
0689 1A89 18B2            BCTR,EQ *REG14+1 MON2    GOTO MONITOR
0690 1A8B E409            COMI,R0 9       CHECK THE COMMAND
0691 1A8D 1E1AC2          BCTA,LT REG2    DISPLAY AND ALTER REGISTERS R0 THRU PSL
0692 1A90 E40A            COMI,R0 H'0A'   IS IT ADJUST CASSETTE COMMAND
0693 1A92 1C1F32          BCTA,EQ TCAS    TEST CASSETTE
0694 1A95 E40C            COMI,R0 H'0C'   IS IT DISPLAY AND ALTER PC
0695 1A97 1807            BCTR,EQ REG3    DISPLAY AND ALTER PC
0696 1A99 E40F            COMI,R0 H'0F'   IS IT THE PATCH COMMAND
```

```
LINE ADDR  OBJECT  E SOURCE

0697 1A98 1C1A08          BCTA,EQ PTCH    DO THE PATCH COMMAND
0698 1A9E 1B5E            BCTR,UN REG     NOT DEFINED TRY AGAIN
0699                 *
0700                 *DISPLAY AND ALTER PROGRAM COUNTER
0701                 *
0702 1AA0 051F       REG3   LODI,R1 <PCEQ-1 GET ADDRESS OF PC EQUALS DISPLAY
0703 1AA2 06AC              LODI,R2 >PCEQ-1
0704 1AA4 BBFE              ZBSR    *MOV    MOVE DATA TO DISBUF
0705 1AA6 088E              LODR,R0 *REG4+1  GET CURRENT PC ADDRESS
0706 1AA8 BBEA              ZBSR  *BRKPT4  SET UP DISPLAY
0707 1AAA 0C17E8     REG11  LODA,R0 LADR    GET MSB OF CURRENT PC
0708 1AAD 3F1D8A            BSTA,UN SCBP2   SET UP DISPLAY
0709 1AB0 20                EORZ    R0      SET FLAG TO DOUBLE BYTE
0710 1AB1 BBFC              ZBSR *GNPA    DISPLAY ADDRESS AND WAIT FOR ENTRY
0711 1AB3 5B05              BRNR,R3 REG5    DON'T CHANGE DATA
0712 1AB5 CC17E9     REG4   STRA,R0 LADR+1  UP DATE THE PC SAVE LSB
0713 1AB8 C9F1              STRR,R1 *REG11+1 SAVE MSB OF PC
0714 1ABA E687       REG5   COMI,R2 H'87'   ENTR/NXT TERMINATION
0715 1ABC 9C187D     REG14  BCFA,EQ MON2    IF NOT NEW FUNCTION EXIT
0716 1ABF 1F1A7E            BCTA,UN REG     GO ASK FOR NEW REGISTER
0717                 *
0718                 *DISPLAY AND ALTER REGISTERS
0719                 *
0720 1AC2 CC17E6     REG2   STRA,R0 T2      SAVE IT
0721 1AC5 C3                STRZ    R3      SAVE R0 TO USE AS INDEX
0722 1AC6 0510              LODI,R1 H'10'   P CHAR
0723 1AC8 E707              COMI,R3 7       IS IT PSU
0724 1ACA 1A0A              BCTR,LT REG8    NOT PSU PSL
0725 1ACC 1904              BCTR,GT REG10   NOT PSU
0726 1ACE 0412              LODI,R0 H'12'   CHAR U
0727 1AD0 1B06              BCTR,UN REG12    GO DISPLAY
0728                 *
0729 1AD2 0411       REG10  LODI,R0 H'11'   CHAR L
0730 1AD4 1B02              BCTR,UN REG12    GO DISPLAY
0731                 *
0732 1AD6 0513       REG8   LODI,R1 H'13'      CHAR R
0733 1AD8 CD17D3     REG12  STRA,R1 DISBUF+2       SET DISPLAY RN=
0734 1ADB CC17D4     REG9   STRA,R0 DISBUF+3       SET UP DISPLAY
0735 1ADE 0F77F2            LODA,R0 UREG,R3 GET REGISTER CONTENT
0736 1AE1 BBEA              ZBSR  *BRKPT4  SET UP DISPLAY
0737 1AE3 0401              LODI,R0 1       SET FLAG TO SINGLE BYTE
0738 1AE5 BBFC              ZBSR *GNPA    DISPLAY REG CONTENT AND WAIT FOR ENTRY
0739 1AE7 5B0C              BRNR,R3 REG7    NO DATA TERMINATE
0740 1AE9 0BD8       REG6   LODR,R3 *REG2+1       GET THE INDEX VALUE
0741 1AEB CF77F2            STRA,R0 UREG,R3 PUT NEW VALUE IN REGISTER
0742 1AEE E708              COMI,R3 8       IS IT PSL?
0743 1AF0 9803              BCFR,EQ REG7    NO CHECK TERMINATION
0744 1AF2 CC17FC            STRA,R0 UREG+10 SAVE FOR RESTORE OF PSL
0745 1AF5 E687       REG7   COMI,R2 H'87'   CHECK TERMINATION
0746 1AF7 98C4              BCFR,EQ *REG14+1 MON2    NEW FUNCTION
0747 1AF9 03                LODZ    R3      INCREMENT INDEX VALUE
0748 1AFA D800              BIRR,R0 $+2     INCREMENT REGISTER COUNT
0749 1AFC E408              COMI,R0 8       ROLL OVER?
0750 1AFE 9D1AC2     REG13  BCFA,GT REG2    NO UP DATE DISPLAY
0751 1B01 20                EORZ    R0      GET A 0 GO TO R0
0752 1B02 1BFB              BCTR,UN *REG13+1    UPDATE DISPLAY
```

```
LINE ADDR  OBJECT  E SOURCE

0754               ************************************************************
0755               *
0756               *
0757               *GET NUMERIC PARAMETERS
0758               *
0759               *
0760               *THIS ROUTINE GETS EITHER 2 OR 4 DIGIT NUMERIC PARAMETERS
0761               *
0762               *INPUT PARAMETERS
0763               *
0764               *R0 CONTAINS INPUT PARAMETER
0765               *
0766               *BIT0 = 0 DOUBLE BYTE
0767               *BIT0 = 1 SINGLE BYTE DATA TO BE RETURNED
0768               *BIT1 = 0 REQUIRES FUNCTION KEY DEPRESSION TO EXIT
0769               *BIT1 = 1 WHEN SET WITH BIT0 EXIT IS AFTER ENTRY OF THIRD DIGIT
0770               *         OF SINGLE BYTE DATA
0771               *BIT2 = 1 WHEN SET WITH BIT0 EXIT IS AFTER SECOND DIGIT
0772               *         OF SINGLE BYTE DATA
0773               *
0774               *SINGLE BYTE DATA USES DISPLAY BUFFER 5 THRU 7
0775               *DOUBLE BYTE DATA USES DISPLAY BUFFER 4 THRU 7
0776               *OTHER DIGITS OF DISBUF MUST BE INITIALIZED ON ENTRY
0777               *
0778               *RETURNS WHEN FUNCTION KEY DEPRESSED
0779               *
0780               *OUTPUT PARAMETERS
0781               *
0782               *R0 = LSB OF DOUBLE BYTE DATA OR SINGLE BYTE DATA
0783               *R1 = MSB OF DOUBLE BYTE DATA OR 0 FOR SINGLE BYTE DATA
0784               *R2 = FUNCTION KEY PRESSED CODE
0785               *R3 = 0 DATA RETURNED IN R0(LSB), R1(MSB)
0786               *R3 = NOT 0 NO DATA RETURNED R0,R1 = 0
0787               *
0788               *REGISTERS USED
0789               *
0790               *R0 SCRATCH
0791               *R1 SCRATCH
0792               *R2 SCRATCH
0793               *R3 SCRATCH
0794               *
0795               *SUBROUTINES CALLED
0796               *
0797               *DISPLY        DISPLAY AND READ KEY BOARD
0798               *CLR BLANK DIGIT DISPLAY
0799               *
0800               *RAM MEMORY USED
0801               *
0802               *T1 SAVE ENTRY FLAG
0803               *DISPLY 4 THRU 7
0804               *
0805               ************************************************************
0806               *
0807               *
0808               *DISPLAY AD= AND GET DATA
```

```
LINE ADDR  OBJECT  E SOURCE

0809                *
0810 1B04 051F      GAD    LODI,R1 <ADR-1  GET ADDRESS OF AD= DISPALY
0811 1B06 068C             LODI,R2 >ADR-1
0812 1B08 BBFE             ZBSR *MOV     MOVE DATA TO DISBUF
0813 1B0A 20               EORZ    R0    SET FLAG TO DOUBLE BYTE DATA
0814 1B0B 1B2E             BCTR,UN GNPI    GET THE ADDRESS DATA
0815                *
0816                *THIS ROUTINE CLEARS DIGIT DISPLAY
0817                *
0818 1B0D 0517      CLR    LODI,R1 H'17'  BLANK SYMBOL
0819 1B0F F401             TMI,R0  1      SINGLE BYTE?
0820 1B11 1803             BCTR,EQ CLR1   ONE BYTE DATA
0821 1B13 CD17D5           STRA,R1 DISBUF+4       INITIALIZE DISPLAY TO BLANK
0822 1B16 CD17D6    CLR1   STRA,R1 DISBUF+5       GET HERE FOR ONE BYTE DATA
0823 1B19 CD17D7           STRA,R1 DISBUF+6
0824 1B1C CD17D8           STRA,R1 DISBUF+7
0825 1B1F 17               RETC,UN
0826                *
0827                *THIS ENTRY POINT ALLOWS DISPLAY OF DATA IN DISPLAY
0828                *BUFFER 4 THRU 7
0829                *
0830 1B20 C8A6      GNPAI  STRR,R0 *GNP12+1       SAVE INPUT FLAG IN T1
0831 1B22 0480             LODI,R0 H'80'  TURN ON DECIMAL POINT FOR ENTRY
0832 1B24 BBEC             ZBSR    *DISPLY DISPLAY MESSAGE AND READ KEY BOARD
0833 1B26 08A0             LODR,R0 *GNP12+1     GET INPUT PARAMETER
0834 1B28 F680             TMI,R2  H'80'  FUNCTION KEY?
0835 1B2A 9809             BCFR,EQ GNP13  FIRST CHAR IS COMMAND TERMINATE
0836 1B2C E687             COMI,R2 H'87'  ENTR/NXT?
0837 1B2E 1802             BCTR,EQ $+4
0838 1B30 3B5B             BSTR,UN CLR    CLEAR DISPLAY
0839 1B32 1F1B74           BCTA,UN GNP4
0840 1B35 F404      GNP13  TMI,R0  4      PATCH COMMAND RETURN SECOND DIGIT?
0841 1B37 3A54             BSTR,NG CLR    CLEAR DISPLAY
0842 1B39 1B0C             BCTR,UN GNP12
0843                *
0844                *THIS ENTRY POINT CLEARS DISPLAY AND WAITS FOR ENTRY
0845                *
0846 1B3B C88B      GNPI   STRR,R0 *GNP12+1       SAVE INPUT FLAG IN T1
0847 1B3D 3B4E      GNP11  BSTR,UN CLR    CLEAR DISPLAY
0848 1B3F 0480      GNP2   LODI,R0 H'80'  TURN ON DECIMAL POINT FOR ENTRY
0849 1B41 BBEC             ZBSR *DISPLY  DISPLAY MESSAGE AND READ KEY BOARD
0850 1B43 F680      GNP5   TMI,R2  H'80'  FUNCTION KEY PRESSED?
0851 1B45 182D             BCTR,EQ GNP4   GO TERMINATE
0852                *
0853                *MOVE DISPLAY 1 DIGIT LEFT
0854                *
0855 1B47 0C17E5    GNP12  LODA,R0 T1     GET INPUT PARAMETER
0856 1B4A F483             TMI,R0  H'83'  THIRD DIGIT *EXIT ON THIRD ENTRY*SINGLE BYTE
0857 1B4C 1826             BCTR,EQ GNP4   GO TERMINATE
0858 1B4E F425             TMI,R0  H'25'  2ND DIGIT*EXIT ON 2ND DIGIT*SINGLE BYTE
0859 1B50 1822             BCTR,EQ GNP4   GO TERMINATE
0860 1B52 F401             TMI,R0  1      SINGLE BYTE DATA?
0861 1B54 1806             BCTR,EQ GNP3   ONLY TWO DIGITS
0862 1B56 0D17D6    GN1    LODA,R1  DISBUF+5      GET DIGIT
0863 1B59 CD17D5    GN2    STRA,R1 DISBUF+4       SHIFT IT
0864 1B5C 0D17D7    GN3    LODA,R1 DISBUF+6       GET DIGIT
```

```
LINE ADDR  OBJECT  E SOURCE

0865 1B5F C9F6           STRR,R1 *GN1+1       SHIFT IT
0866 1B61 0D17D8    GNP3 LODA,R1 DISBUF+7      GET DIGIT
0867 1B64 C9F7           STRR,R1 *GN3+1       SHIFT IT
0868 1B66 CAFA           STRR,R2 *GNP3+1       ENTER NEW DIGIT
0869 1B68 08DE           LODR,R0 *GNP12+1     GET INPUT PARAMETER
0870 1B6A 7508           CPSL    WC      CLEAR WITH CARRY
0871 1B6C 8440           ADDI,R0 H'40'   SET BEEN HERE ONCE FLAG
0872 1B6E 6420           IORI,R0 H'20'   SET SECOND DIGIT FLAG
0873 1B70 C8D6      GN4  STRR,R0 *GNP12+1     RESTORE THE FLAG
0874 1B72 1B4B           BCTR,UN GNP2    GET NEXT ENTRY
0875                *
0876                *SET UP DATA TO BE RETURNED
0877                *
0878 1B74 20        GNP4 EORZ    R0      GET A 0
0879 1B75 C1             STRZ    R1      CLEAR R1 DATA
0880 1B76 C3             STRZ    R3      CLEAR R3
0881 1B77 08CF           LODR,R0 *GNP12+1     GET INPUT PARAMETER
0882 1B79 F401           TMI,R0  1       CHECK FOR SINGLE BYTE
0883 1B7B 1812           BCTR,EQ GNP7    IF EQ ONLY 1 DIGIT
0884 1B7D 0C9B5A         LODA,R0 *GN2+1 DISBUF+4       GET MSD OF MSB
0885 1B80 E410           COMI,R0 H'10'   SEE IF HEX DIGIT
0886 1B82 9A03           BCFR,LT GNP6    IF NOT SKIP TO NEXT DIGIT
0887 1B84 3B1F           BSTR,UN ROTI     ROTATE NIBBLE
0888 1B86 C1             STRZ    R1      SAVE IN R1
0889 1B87 08CE      GNP6 LODR,R0 *GN1+1 DISBUF+5       GET LSD OF MSB
0890 1B89 E410           COMI,R0 H'10'   SEE IF HEX DIGIT
0891 1B8B 9A02           BCFR,LT GNP7    IF NOT SKIP TO NEXT DIGIT
0892 1B8D 61             IORZ    R1      INCLUSIVE OR MSD AND LSD OF MSB
0893 1B8E C1             STRZ    R1      SAVE IN R1
0894 1B8F 08CC      GNP7 LODR,R0 *GN3+1 DISBUF+6       GET MSD OF LSB
0895 1B91 E410           COMI,R0 H'10'   SEE IF HEX DIGIT
0896 1B93 9A03           BCFR,LT GNP8    IF NOT SKIP TO NEXT DIGIT
0897 1B95 3B0E           BSTR,UN ROTI     ROTATE THE NIBBLE
0898 1B97 C3             STRZ    R3      SAVE IN R3
0899 1B98 08C8      GNP8 LODR,R0 *GNP3+1 DISBUF+7       GET LSD OF LSB
0900 1B9A E410           COMI,R0 H'10'   SEE IF HEX DIGIT
0901 1B9C 9A04           BCFR,LT GNP9    IF NOT RETURN
0902 1B9E 63             IORZ    R3      INCLUSIVE OR MSD WITH LSD OF LSB
0903 1B9F 0700           LODI,R3 0     SET DATA IN R0,R1 FLAG
0904 1BA1 17             RETC,UN
0905 1BA2 077F      GNP9 LODI,R3 127     NO DATA
0906 1BA4 17             RETC,UN
0907                *
0908                *THIS ROUTINE ROTATES A NIBBLE 4 BITS LEFT
0909                *
0910 1BA5 7508      ROTI  CPSL    WC      CLEAR WITH CARRY
0911 1BA7 D0             RRL,R0
0912 1BA8 D0             RRL,R0
0913 1BA9 D0             RRL,R0
0914 1BAA D0             RRL,R0
0915 1BAB 17             RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

0917               ************************************************************
0918               *
0919               *
0920               *READ CASSETTE COMMAND
0921               *
0922               *
0923               *
0924               *THIS IS THE HEX OBJECT LOADER
0925               *
0926               *THIS ROUTINE REQUESTS A FILE ID AND THEN LOADS 2650 HEX OBJECT MODULES
0927               *INTO MEMORY
0928               *
0929               *REGISTERS USED
0930               *
0931               *ALL
0932               *
0933               *SUBROUTINES CALLED
0934               *
0935               *IN CASSETTE INPUT ROUTINE
0936               *MOV   MOVE DATA TO DISPLAY BUFFER
0937               *GNP   GET NUMERIC PARAMETERS
0938               *
0939               ************************************************************************
0940               *
0941               *
0942 1BAC 051F    RCAS   LODI,R1 <FE0-1  GET ADDRESS OF F= DISPLAY
0943 1BAE 06B4           LODI,R2 >FE0-1
0944 1BB0 B8FE           ZBSR *MOV     MOVE DATA TO DISBUF
0945 1BB2 0401           LODI,R0 1       SET FLAG FOR SINGLE BYTE
0946 1BB4 B8FA           ZBSR    *GNP    GET THE FILE ID
0947 1BB6 180A           BCTR,EQ RCAS1   FILE ID SPECIFIED
0948 1BB8 E687           COMI,R2 H'87'   ENTR/NXT KEY?
0949 1BBA 988C           BCFR,EQ *RCAS4+1        GODO NEW FUNCTION
0950 1BBC 047F           LODI,R0 127     SET FILE ID FLAG TO FILE ID FOUND
0951 1BBE C8A4           STRR,R0 *RCAS5+1        STORE IN FILE ID FLAG
0952 1BC0 1B24           BCTR,UN LOAD
0953               *
0954               *
0955               *FILE ID SPECIFIED
0956               *
0957 1BC2 CC17E0  RCAS1  STRA,R0 FID+1   SAVE FILE ID
0958 1BC5 E687           COMI,R2 H'87'   ENTR/NXT KEY?
0959 1BC7 9C187D  RCAS4  BCFA,EQ MON2    GO DO NEW FUNCTION
0960 1BCA 20             EORZ    R0      SET FILE ID TO ID NOT FOUND
0961 1BCB C897           STRR,R0 *RCAS5+1        STORE IN FILE ID FLAG
0962 1BCD 75FD           CPSL    H'FD'     CLEAR PSL
0963 1BCF BBEE    RCAS2  ZBSR    *IN     LOOK FOR BEGINNING OF FILE
0964 1BD1 E416           COMI,R0 H'16'   BEGINNING OF FILE CHAR?
0965 1BD3 987A           BCFR,EQ RCAS2   LOOP TILL FIND BEGIN OF FILE
0966 1BD5 3F1C28         BSTA,UN BIN     GET THE FILE ID
0967 1BD8 E9E9           COMR,R1 *RCAS1+1   CHECK FILE ID FOR MATCH
0968 1BDA 1805           BCTR,EQ RCAS3   FOUND A MATCH
0969 1BDC 20             EORZ    R0      GET A 0
0970 1BDD C885           STRR,R0 *RCAS5+1 NO MATCH SAVE IN FID FLAG
0971 1BDF 1B05           BCTR,UN LOAD
```

```
LINE ADDR  OBJECT  E SOURCE

0972 1BE1 047F      RCAS3  LODI,R0 127    SET FLAG TO FILE IS MATCH
0973 1BE3 CC17DF    RCAS5  STRA,R0 FID    FILE ID FOUND
0974 1BE6 75FD      LOAD   CPSL   H'FD'    CLEAR PSL
0975 1BE8 BBEE             ZBSR   *IN     GET A CHAR
0976 1BEA E43A             COMI,R0 A':'   START OF LINE CHAR?
0977 1BEC 9878             BCFR,EQ LOAD   LOOP TILL FIND START FO RECORD
0978 1BEE 20               EORZ   R0      GET A 0
0979 1BEF CC17E1           STRA,R0 BCC    PRESET BCC
0980 1BF2 3B34             BSTR,UN BIN    INPUT A BYTE OF DATA
0981 1BF4 CD17DD           STRA,R1 MEM HI ADDR
0982 1BF7 3B2F             BSTR,UN BIN INPUT A BYTE OF DATA
0983 1BF9 CD17DE           STRA,R1 MEM+1 LO ADDR
0984 1BFC 3B2A             BSTR,UN BIN INPUT A BYTE OF DATA
0985 1BFE 01               LODZ   R1
0986 1BFF 1C1C42           BCTA,EQ LOAD1  GO TO START OF PROGRAM IF BYTE COUNT 0
0987 1C02 C3               STRZ   R3      SAVE BYTE COUNT
0988 1C03 08DF             LODR,R0 *LOAD-2 GET FILE ID FLAG
0989 1C05 185F             BCTR,EQ LOAD   FILE ID NOT FOUND SKIP TO END OF FILE
0990 1C07 3B1F             BSTR,UN BIN INPUT A BYTE OF DATA
0991 1C09 1804             BCTR,EQ BLOA   BCC OK READ THE RECORD
0992 1C0B 0704      BLOA1  LODI,R3 4      BCC ERROR
0993 1C0D 9BE8             ZBRR   *ERR    GOTO ERROR
0994 1C0F 3B17      BLOA   BSTR,UN BIN INPUT A BYTE OF DATA
0995 1C11 CD97DD           STRA,R1 *MEM STORE DATA IN MEMORY
0996 1C14 ED97DD           COMA,R1 *MEM   DO THE ERROR CHECK
0997 1C17 1804             BCTR,EQ BLOA2  DATA STORED OK
0998 1C19 0705             LODI,R3 5      READ CASSETTE MEMORY WRITE ERROR
0999 1C1B 9BE8             ZBRR   *ERR    GOTO ERROR
1000 1C1D 3B36      BLOA2  BSTR,UN INK INCREMENT POINTER MEM
1001 1C1F FB6E             BDRR,R3 BLOA LOOP TILL DONE
1002 1C21 3B05             BSTR,UN BIN INPUT A BYTE OF DATA
1003 1C23 9866             BCFR,EQ BLOA1 BCC ERROR
1004 1C25 1F1BE6           BCTA,UN LOAD
1005                *
1006                * INPUT A PAIR OF HEX ASCII CHAR
1007                *CONVERT TO BINARY
1008                *OUTPUT IS IN R1
1009                *CALCULATE BCC ON DATA
1010                *
1011 1C28 BBEE      BIN    ZBSR   *IN INPUT A CHAR
1012 1C2A 7509             CPSL   C+WC    CLEAR CARRY AND WITH CARRY
1013 1C2C 3B36      BIN1   BSTR,UN AHO3    LOOK UP VALUE
1014 1C2E 02               LODZ   R2      PUT VALUE IN R0
1015 1C2F BBF6             ZBSR   *ROT    ROTATE VALUE
1016 1C31 C1               STRZ   R1      SAVE VALUE IN R1
1017 1C32 BBEE             ZBSR   *IN     GET A CHAR
1018 1C34 7509             CPSL   C+WC    CLEAR CARRY AND WITH CARRY
1019 1C36 3B2C             BSTR,UN AHO3    LOOK UP VALUE
1020 1C38 01               LODZ   R1      GET SAVED VALUE
1021 1C39 62               IORZ   R2      MAKE THE BINARY BYTE
1022                *
1023                *CALCULATE BCC
1024                *
1025 1C3A C1        CBCC   STRZ   R1      SAVE VALUE
1026 1C3B 2C17E1           EORA,R0 BCC    XOR WITH CURRENT BCC
1027 1C3E D0               RRL,R0         ROTATE LEFT
```

```
LINE ADDR  OBJECT  E SOURCE

1028 1C3F C8FB            STRR,R0 *CBCC+2 UPDATE THE BCC
1029 1C41 17              RETC,UN
1030                *
1031                *
1032                *FINISHED READING FILE
1033                *
1034 1C42 0C17DF    LOAD1 LODA,R0 FID     CHECK FILE ID FLAG FOR FILE ID FOUND
1035 1C45 1C1BCF          BCTA,EQ RCAS2   NO LOOK FOR START OF NEXT FILE
1036 1C48 088F            LODR,R0 *INK2+1 GET VALUE FROM MEM     PLACE START ADDRESS IN PC
1037 1C4A CC17E8          STRA,R0 LADR
1038 1C4D 0887            LODR,R0 *INK+1  GET VALUE FROM MEM+1
1039 1C4F CC17E9          STRA,R0 LADR+1
1040 1C52 1F1874          BCTA,UN MON3     GO TO THE MONITOR
1041                *
1042                *INCREMENT ADDRESS MEM
1043                *
1044 1C55 0C17DE    INK   LODA,R0 MEM+1   GET ADDRESS
1045 1C58 0E17DD    INK2  LODA,R2 MEM
1046 1C5B D802            BIRR,R0 INK1    INCREMENT IT
1047 1C5D DA00            BIRR,R2 INK1
1048 1C5F C8F5      INK1  STRR,R0 *INK+1   SAVE  IN MEM+1
1049 1C61 CAF6            STRR,R2 *INK2+1 SAVE IN MEM
1050 1C63 17              RETC,UN
1051                *
1052                *LOOK UP ASCII HEX TO CONVERT TO BINARY
1053                *
1054 1C64 06FF      AHO3  LODI,R2 255     PRESET INDEX
1055 1C66 EE3FC5          COMA,R0 ASCII,R2,+      CHECK THE VALUE
1056 1C69 14              RETC,EQ RETURN IF EQUAL
1057 1C6A E610            COMI,R2 H'10'   CHECK FOR MAX COUNT
1058 1C6C 9878            BCFR,EQ AHO3+2  LOOP
1059 1C6E 0706            LODI,R3 6       CHAR NOT ASCII HEX
1060 1C70 9BE8            ZBRR    *ERR    GOTO ERROR
1061                *
1062                *CARRAGE RETURN AND LINE FEED
1063                *
1064 1C72 040D      CRLFF LODI,R0 13      CARRAGE RETURN
1065 1C74 BBF0            ZBSR *OUT       PRINT
1066 1C76 040A            LODI,R0 10      LINE FEED
1067 1C78 BBF0            ZBSR *OUT       PRINT
1068 1C7A 17              RETC,UN
1069                *
1070                *CONVERT BINARY TO ASCII HEX AND PRINT
1071                *
1072 1C7B 7508      HOUTT  CPSL    WC
1073 1C7D BBF4             ZBSR    *DISLSD CONVERT BIN TO NIBBLE
1074 1C7F C2               STRZ    R2      SAVE IN R2
1075 1C80 0E7FC5           LODA,R0 ASCII,R2        TENS DIGIT
1076 1C83 BBF0             ZBSR    *OUT    PRINT TENS DIGIT
1077 1C85 0D7FC5           LODA,R0 ASCII,R1        GET UNITS DIGIT
1078 1C88 BBF0             ZBSR *OUT       PRINT UNITS DIGIT
1079 1C8A 17               RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1081               ************************************************************
1082               *
1083               *
1084               *WRITE CASSETTE COMMAND
1085               *
1086               *THIS ROUTINE WRITES 2650 HEX FORMAT TO CASSETTE TAPE
1087               *
1088               *REGISTERS USED
1089               *
1090               *R0 SCRATCH
1091               *R1 SCRATCH
1092               *R2 SCRATCH
1093               *R3 SCRATCH
1094               *
1095               *SUBROUTINES CALLED
1096               *
1097               *OUT WRITE CHAR TO TAPE
1098               *HOUT CONVERT BINARY TO ASCII HEX AND WRITE TO TAPE
1099               *INK INCREMENT POINTER MEM
1100               *
1101               *RAM USED
1102               *
1103               *BCC   BLOCK CHECK CHAR
1104               *MEM   POINTER
1105               *BAD   PROGRAM START ADDRESS
1106               *SAD   DUMP STOP ADDRESS
1107               *FID   FILE ID FLAG AND STORAGE
1108               *
1109               *THIS ROUTINE PUNCHES A HEX FORMAT TAPE
1110               *
1111               *
1112               *   LEADER16ID:ADDRCTBCAADDCCRR.....BC
1113               *
1114               ************************************************************
1115               *
1116               *
1117 1C8B 20      WCAS4  EORZ    R0      GET A 0
1118 1C8C BBFA           ZBSR    *GNP    GET NUMBER
1119 1C8E E687           COMI,R2 H'87'   ENTR/NXT KEY
1120 1C90 17             RETC,UN
1121               *
1122               *
1123 1C91 051F    WCAS   LODI,R1 <LADEQ-1        GET ADDRESS OF LAD= DISPLAY
1124 1C93 06BC           LODI,R2 >LADEQ-1
1125 1C95 B8FE           ZBSR    *MOV    MOVE TO DISPLAY BUFFER
1126 1C97 3B72           BSTR,UN WCAS4   GET ADDRESS DATA
1127 1C99 988E           BCFR,EQ *WCAS6+1 MON2    IF NOT EXIT
1128 1C9B CD17DD         STRA,R1 MEM     SAVE START ADDRESS
1129 1C9E CC17DE         STRA,R0 MEM+1
1130 1CA1 0412           LODI,R0 H'12'   CHANGE DISPLAY
1131 1CA3 CC17D1         STRA,R0 DISBUF  DISPLAY 'LAD=    '
1132 1CA6 3B63           BSTR,UN WCAS4   GET ADDRESS DATA
1133 1CA8 9C187D  WCAS6  BCFA,EQ MON2    NOT ENTR/NXT MUST BE NEW COMMAND
1134               *
1135               *CHECK FOR START ADDRESS GT THAN STOP
```

```
LINE ADDR  OBJECT  E SOURCE

1136                *
1137 1CAB ED17DD           COMA,R1 MEM     CHECK HI BYTE
1138 1CAE 1806             BCTR,EQ WCAS7
1139 1CB0 1909             BCTR,GT WCAS9
1140 1CB2 0707      WCAS8  LODI,R3 7       SET THE ERROR NUMBER
1141 1CB4 9BE8             ZBRR    *ERR    GOTO ERROR
1142 1CB6 EC17DE    WCAS7  COMA,R0 MEM+1   CHECK LO BYTE
1143 1CB9 1A77             BCTR,LT WCAS8
1144                *
1145 1CBB D802      WCAS9  BIRR,R0 WCASA   INCREMENT STOP ADDRESS
1146 1CBD D900             BIRR,R1 WCASA   SO DUMP IS INCLUSIVE
1147 1CBF CD17C8    WCASA  STRA,R1 EAD     SAVE END ADDRESS
1148 1CC2 CC17C9           STRA,R0 EAD+1
1149 1CC5 0405             LODI,R0 H'05'   CHANGE DISPLAY
1150 1CC7 CC17D1           STRA,R0 DISBUF  DISPLAY 'SAD=    '
1151 1CCA 3F1C8B           BSTA,UN WCAS4   GET PROGRAM START ADDRESS
1152 1CCD 9C9CA9    WCAS3  BCFA,EQ *WCAS6+1 MON2   GOTO MONITOR NEW FUNCTION
1153 1CD0 CD17CA           STRA,R1 BAD     SAVE START ADDRESS
1154 1CD3 CC17CB           STRA,R0 BAD+1
1155 1CD6 051F             LODI,R1 <FEQ-1  GET ADDRESS OF F= DISPLAY
1156 1CD8 06B4             LODI,R2 >FEQ-1
1157 1CDA BBFE             ZBSR *MOV     MOVE DATA TO DISBUF
1158 1CDC 0401             LODI,R0 1       SET FLAG TO SINGLE BYTE
1159 1CDE BBFA             ZBSR *GNP     GET THE FILE ID
1160 1CE0 E687             COMI,R2 H'87'   ENTR/NXT KEY
1161 1CE2 98C5             BCFR,EQ *WCAS6+1 MON2    EXIT NEW COMMAND
1162 1CE4 C894             STRR,R0 *WCAS5+1   SAVE FILE ID
1163 1CE6 060A             LODI,R2 10      SET THE DELAY
1164 1CE8 0719      PUN10  LODI,R3 25
1165 1CEA 20                EORZ    R0      GET A 0
1166 1CEB BBF0             ZBSR    *OUT    OUTPUT A LEADER
1167 1CED FB7B             BDRR,R3 PUN10+2
1168 1CEF 12               SPSU        GET FLAG
1169 1CF0 2440             EORI,R0 H'40'   COMPLEMENT IT
1170 1CF2 92               LPSU        RESTORE IT
1171 1CF3 FA73             BDRR,R2 PUN10    DECREASE THE COUNT
1172 1CF5 0416             LODI,R0 H'16' START OF FILE CHAR
1173 1CF7 BBF0             ZBSR *OUT       PRINT
1174 1CF9 0C17E0    WCAS5  LODA,R0 FID+1   GET FILE ID
1175 1CFC BBF2             ZBSR *HOUT      CONVERT TO ASCII HEX AND PRINT
1176 1CFE BBF8      PUN2   ZBSR    *CRLF   OUTPUT CARRAGE RETURN AND LINE FEED
1177 1D00 043A             LODI,R0 A':'    START OF BLOCK CHAR
1178 1D02 BBF0             ZBSR    *OUT    PRINT
1179 1D04 20               EORZ    R0      GET A 0
1180 1D05 C8A3             STRR,R0 *PUN3+1     PRESET BCC
1181 1D07 0C17C8           LODA,R0 EAD CALCULATE NO OF BYTES TO OUTPUT
1182 1D0A 7709             PPSL    WC+C    SET CARRY AND WITH CARRY
1183 1D0C 0F17C9           LODA,R3 EAD+1   GET END ADDRESS
1184 1D0F ABAC             SUBR,R3 *BDUM1+1 MEM+1  SUBTRACT START ADDRESS FROM STOP ADDRESS
1185 1D11 A8A5             SUBR,R0 *BDUM+1 MEM
1186 1D13 7508             CPSL    WC      CLEAR WITH CARRY
1187 1D15 1E1CB2           BCTA,NG WCAS8   START > STOP
1188                *
1189                *
1190 1D18 581B      PUN4   BRNR,R0 ADUM    START ADDRESS GT THAN 256 AWAY FROM STOP
1191 1D1A 5B15             BRNR,R3 GDUM    START ADDRESS LT 256 AWAY FROM STOP
```

```
LINE ADDR  OBJECT  E SOURCE

1192 1D1C 0C17CA          LODA,R0 BAD THIS IS END OF FILE BLOCK
1193 1D1F 3B39            BSTR,UN EDUM SO OUTPUT START ADDRESS OF PROGRAM
1194 1D21 0C17CB          LODA,R0 BAD+1
1195 1D24 3B34            BSTR,UN EDUM    OUTPUT A BYTE AS 2 ASCII HEX CHARS
1196 1D26 20              EORZ    R0      END OF FILE BLOCK
1197 1D27 3B31            BSTR,UN EDUM OUTPUT BYTE COUNT
1198 1D29 0C17E1    PUN3  LODA,R0 BCC     GET BCC
1199 1D2C 3B2C            BSTR,UN EDUM OUTPUT BCC
1200 1D2E 1F1874          BCTA,UN       MON3      GOTO MONITOR
1201                *
1202                *
1203 1D31 E71E      GDUM  COMI,R3 H'1E'   IS START LT 30 AWAY FROM STOP
1204 1D33 1A02            BCTR,LT BDUM    OUTPUT LAST BYTES
1205 1D35 071E      ADUM  LODI,R3 H'1E'   NO OF BYTES THIS RECORD IS 30
1206 1D37 0C17DD    BDUM  LODA,R0 MEM     OUT ADDR HI
1207 1D3A 3B1E            BSTR,UN EDUM    OUTPUT BYTE AS 2 ASCII HEX CHARS
1208 1D3C 0C17DE    BDUM1 LODA,R0 MEM+1   OUT ADDR LO
1209 1D3F 3B19            BSTR,UN EDUM    OUTPUT BYTE AS 2 ASCII HEX CHARS
1210 1D41 03              LODZ    R3      OUT BYTE COUNT
1211 1D42 3B16            BSTR,UN EDUM    OUTPUT BYTE AS 2 ASCII HEX CHARS
1212 1D44 08E4            LODR,R0 *PUN3+1     OUT BCC FOR ADDR AND BYTE COUNT
1213 1D46 3B12            BSTR,UN EDUM    OUTPUT BYTE AS 2 ASCII HEX CHARS
1214 1D48 0C97DD    DDUM  LODA,R0 *MEM    OUTPUT DATA FROM MEM
1215 1D4B 3B0D            BSTR,UN EDUM    OUTPUT BYTE AS 2 ASCII HEX CHARS
1216 1D4D 3F1C55          BSTA,UN INK     INCREMEMT POINTER MEM
1217 1D50 FB76            BDRR,R3 DDUM    LOOP TILL DONE
1218 1D52 0C17E1          LODA,R0 BCC     GET BCC
1219 1D55 3B03            BSTR,UN EDUM    OUTPUT BCC FOR DATA
1220 1D57 1F1CFE          BCTA,UN PUN2
1221                *
1222 1D5A 3F1C3A    EDUM  BSTA,UN CBCC    CALCULATE BCC
1223 1D5D 01              LODZ    R1      GET VALUE TO OUTPUT
1224 1D5E BBF2            ZBSR    *HOUT   PRINT AS 2 ASCII HEX CHARS
1225 1D60 17              RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1227                ********************************************************
1228                *
1229                *
1230                *SET OR CLEAR BREAK POINT
1231                *
1232                *
1233                *TO SET BREAK POINT ENTR ADDRESS AND DEPRESS FUNCTION KEY
1234                *TO CLEAR BREAK POINT DEPRESS FUNCTION KEY
1235                *
1236                *SUBROUTINES CALLED
1237                *
1238                *MOV MOVE DATA TO DISBUF
1239                *GNPA DISPLAY AND GET ADDRESS DATA
1240                *ROT ROTATE A NIBBLE
1241                *SCBP2 SET DISBUF 4&5
1242                *BRKPT4 SET DISBUF 6&7
1243                *DSLSD CONVERT TO BINARY FOR DISPLAY
1244                *
1245                *RAM MEMORY USED
1246                *
1247                *BPF BREAK POINT FLAG
1248                *BPL LOCATION OF BREAK POINT
1249                *BPD DATA TO BE RESTORED IN BREAK POINT LOCATION
1250                *
1251                *
1252                *
1253                ********************************************************************
1254                *
1255 1D61 051F      SCBP   LODI,R1 <BPEQ-1  GET ADDRESS OF BP= DISPALY
1256 1D63 069C             LODI,R2 >BPEQ-1
1257 1D65 BBFE             ZBSR *MOV     MOVE DATA TO DISBUF
1258 1D67 0C17CF           LODA,R0 BPF     BREAK POINT SET?
1259 1D6A 180A             BCTR,EQ SCBP1   NOT SET GET ADDRESS
1260                *
1261                *BREAK POINT SET SET UP ADDRESS DISPLAY
1262                *
1263 1D6C 0C17CE           LODA,R0 BPL+1   PREPARE THE ADDRESS
1264 1D6F BBEA             ZBSR  *BRKPT4  SET UP DISPLAY
1265 1D71 0C17CD           LODA,R0 BPL     GET MSB
1266 1D74 3B14             BSTR,UN SCBP2   SETUP DISPLAY
1267 1D76 20        SCBP1  EORZ    R0      SET UP GET NUMBER PARAMETER TO 4 DIGIT
1268 1D77 BBFC             ZBSR *GNPA    GET THE ADDESS IF ANY
1269 1D79 1818             BCTR,EQ SCBP4   SET THE BREAK POINT
1270                *
1271                *THIS SECTION CLEARS THE BREAK POINT
1272                *
1273 1D7B 0B88             LODR,R3 *SCBP6+1     CHECK BREAK POINT FLAG
1274 1D7D 1889             BCTR,EQ *SCBP5+1    BREAK POINT NOT SET GO TO MONITOR
1275 1D7F E681             COMI,R2 H'81'   IS TERMINATION BKP?
1276 1D81 9885             BCFR,EQ *SCBP5+1    NO LEAVE BREAK POINT SET GO TO MONITOR
1277 1D83 20               EORZ    R0      GET A 0
1278 1D84 CC17CF    SCBP6  STRA,R0 BPF     CLEAR BREAK POINT FLAG
1279 1D87 1F167D    SCBP5  BCTA,UN MON2    GO TO MONITOR
1280                *
1281 1D8A BBF4      SCBP2  ZBSR    *DISLSD CONVERT TO BIN FOR DISPLAY
```

```
LINE ADDR  OBJECT  E SOURCE

1282 1D8C CD17D6           STRA,R1 DISBUF+5
1283 1D8F CC17D5           STRA,R0 DISBUF+4
1284 1D92 17               RETC,UN
1285                *
1286                *THIS SECTION SETS THE BREAK POINT
1287                *
1288 1D93 CC17CE    SCBP4  STRA,R0 BPL+1   SET BREAK POINT ADDRESS
1289 1D96 CD17CD           STRA,R1 BPL
1290 1D99 20               EORZ   R0      CLEAR BREAK POINT FLAG
1291 1D9A C8E3             STRR,R0 *SCBP6+1       CHECK THE BREAK POINT CAN BE SET
1292 1D9C 0C37CD           LODA,R0 *BPL   GET DATA FROM BREAK POINT LOCATION
1293 1D9F 05B0             LODI,R1 H'B0'  BREAK POINT INSTRUCTION..WRTC,R0
1294 1DA1 CD97CD           STRA,R1 *BPL   TRY TO SET BREAK POINT
1295 1DA4 ED97CD           COMA,R1 *BPL   DID IT SET OK?
1296 1DA7 1804             BCTR,EQ SCBP7  BREAK POINT CAN BE SET
1297 1DA9 0701             LODI,R3 1      CANT SET BREAK POINT ERROR
1298 1DAB 3BE8             ZBRR   *ERR    GOTO ERROR
1299 1DAD CC97CD    SCBP7  STRA,R0 *BPL   RESTORE USER DATA
1300 1DB0 047F             LODI,R0 127    SET THE BREAK POINT FLAG
1301 1DB2 C8D1             STRR,R0 *SCBP6+1       SET IT
1302 1DB4 1BD2             BCTR,UN *SCBP5+1   GOTO MONITOR
```

```
LINE ADDR  OBJECT  E SOURCE

1304                ********************************************************
1305                *
1306                *
1307                *MOVE 8 BYTES OF DATA POINTED TO IN R1 AND R2 TO DISBUF
1308                *
1309                *
1310                *REGISTERS USED
1311                *
1312                *R0 SCRATCH
1313                *R1 HI ADDRESS BYTE OF DATA ADDRESS-1
1314                *R2 LO ADDRESS BYTE OF DATA ADDRESS-1
1315                *R3 NOT USED
1316                *
1317                *SUBROUTINES CALLED
1318                *
1319                *NONE
1320                *
1321                *RAM MEMORY USED
1322                *
1323                *T TEMP INDIRECT ADDRESS
1324                *
1325                ********************************************************
1326                *
1327 1DB6 CD17E3    MOVI   STRA,R1 T       SET INDIRECT ADDRESS
1328 1DB9 CE17E4           STRA,R2 T+1
1329 1DBC 0608             LODI,R2 8       SET INDEX TO MOVE 8 BYTES
1330 1DBE 0EF7E3    MOV1   LODA,R0 *T,R2   GET A BYTE
1331 1DC1 CE77D0           STRA,R0 DISBUF-1,R2       MOVE TO BUFFER
1332 1DC4 FA78             BDRR,R2 MOV1
1333 1DC6 17               RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1335                ************************************************************
1336                *
1337                *
1338                *KEY BOARD SCAN AND DISPLAY ROUTINE
1339                *
1340                *THIS ROUTINE WRITTEN BY ALEX GOLDBURGER
1341                *
1342                *
1343                *TO USE THIS ROUTINE PLACE DATA TO BE DISPLAYED
1344                *IN DISBUF (SEE CODES AT BEGINNING OF PROGRAM)
1345                *
1346                *ON ENTRY R0 CONTAINS A FLAG
1347                *
1348                *R0 = 0 NORMAL OPERATION
1349                *       ON EXIT R0 = KEY PRESSED CODE
1350                *R0 = 1-127 GO THRU SCAN ONCE AND EXIT
1351                *           ON EXIT R0 = KEY PRESSED CODE
1352                *R0 = H'80' TURN ON DECIMAL POINT FOR ENTRY MODE
1353                *           ON EXIT R0 = KEY PRESSED CODE
1354                *
1355                *SEE KEY PRESSED CODES AT BEGINNING OF PROGRAM
1356                *
1357                *REGISTERS USED IN BANK ON ENTRY
1358                *
1359                *R0    SCRATCH
1360                *R1    KEYBOARD FLAGS
1361                *R2    DIGIT SELECT
1362                *R3    DIGIT POINTER
1363                *
1364                *SUBROUTINES CALLED
1365                *
1366                *NONE
1367                *
1368                *RAM MEMORY USED
1369                *
1370                *DISBUF        DISPLAY BUFFER
1371                *KFLG  KEY BOARD FLAG
1372                *
1373                ************************************************************
1374                *
1375 1DC7 0406        DLOOP  LODI,R0 6       DELAY TO MAKE LOOPS EQUAL
1376 1DC9 F87E               BDRR,R0 $
1377 1DCB 52         DLOOP1 RRR,R2  ROTATE DIGIT SELECT
1378 1DCC 0F77D0             LODA,R0 DISBUF-1,R3       GET DATA TO BE DISPLAYED
1379 1DCF C1                 STRZ    R1      SAVE DISPLAY CODE
1380 1DD0 4580               ANDI,R1 H'80'   MASK FOR DECIMAL POINT
1381 1DD2 447F               ANDI,R0 H'7F'   MASK OFF DECIMAL POINT
1382 1DD4 0C7F68             LODA,R0 SEGTBL,R0       CONVERT TO SEGMENT DATA
1383 1DD7 61                 IORZ    R1      SET THE DECIMAL POINT IF NEEDED
1384 1DD8 F601               TMI,R2  H'01'   COL 7?
1385 1DDA 1A08               BCTR,NG DLOOP3  DONT PUT DECIMAL POINT HERE
1386 1DDC 0D17ED             LODA,R1 KFLG+1  GET FLAG
1387 1DDF 9A03               BCFR,NG DLOOP3  IF FLAG NOT NEG NO DECIMAL POINT
1388 1DE1 4580               ANDI,R1 H'80'   MASK DECIMAL POINT
1389 1DE3 61                 IORZ    R1      SET DECIMAL POINT
```

```
LINE ADDR  OBJECT  E SOURCE

1390 1DE4 0500      DLOOP3 LODI,R1 0       GET A 0
1391 1DE6 D5F9             WRTE,R1 SEG     TURN OFF SEGMENTS
1392 1DE8 D6FA             WRTE,R2 DIGIT   ENABLE NEXT DIGIT
1393 1DEA D4F9             WRTE,R0 SEG     AND DISPLAY IT
1394 1DEC 0C17EC           LODA,R0 KFLG    SEE IF KEY IS DOWN?
1395 1DEF 980B             BCFR,EQ DLOOP4  KEY UP DEBOUNCE
1396 1DF1 1B1A             BCTR,UN DLOOP5  IS KEY DOWN?
1397                *
1398 1DF3 FB52      DLOOP2 BDRR,R3 DLOOP   DECREMENT DIGIT PTR
1399                *                      TEST IF ONE SCAN IS DONE.
1400                *                      IF ONE SCAN DONE INITIALIZE SCAN
1401                *                      PARAMETERS AND KEY FLAGS
1402 1DF5 0C17ED           LODA,R0  KFLG+1 CHECK FOR ONE PASS THEN EXIT MODE
1403 1DF8 1933             BCTR,GT DISP3   IF ONE PASS EXIT
1404 1DFA 1B23             BCTR,UN DISP4   RESET THE FLAGS
1405                *
1406                *
1407 1DFC 3B28      DLOOP4 BSTR,UN GETKEY  GET A KEY
1408 1DFE 9806             BCFR,EQ DLP0    KEY IS DOWN RESET DEBOUNCE
1409 1E00 0887             LODR,R0 *DLP1+1 KFLG+2  GET COUNTER VALUE
1410 1E02 F804             BDRR,R0 DLP1
1411 1E04 1B14             BCTR,UN DISP1   SET FLAG TO ACCEPT KEY
1412 1E06 0460      DLP0   LODI,R0 H'60'   SET THE DELAY COUNT
1413 1E08 CC17EE    DLP1   STRA,R0 KFLG+2  SAVE DELAY COUNT
1414 1E0B 1B66             BCTR,UN DLOOP2  DO THE NEXT SCAN
1415                *
1416                *
1417 1E0D 3B17      DLOOP5 BSTR,UN GETKEY  IS A KEY DOWN?
1418 1E0F 1962             BCTR,EQ DLOOP2    NO
1419 1E11 1B24             BCTR,UN CODE
1420                *
1421                *ENTRY TO DISPLAY ROUTINE HERE
1422                *
1423 1E13 CC17ED    DISPLI STRA,R0 KFLG+1  SAVE INPUT PARAMETER
1424 1E16 0460      DISP2  LODI,R0 H'60'   KEY WAS DOWN - SET KFLG
1425                *                      NOT TO ACCEPT KEY NEXT SCAN
1426 1E18 C8EF             STRR,R0 *DLP1+1 KFLG+2  SET KEY DEBOUNCE DELAY
1427 1E1A CC17EC    DISP1  STRA,R0 KFLG    SAVE KFLG
1428 1E1D 7509             CPSL    C+WC    CLEAR CARRY AND WITH CARRY
1429 1E1F 0708      DISP4  LODI,R3 H'08'   INITIALIZE DIGIT POINTER
1430 1E21 0601             LODI,R2 H'01'   AND DIGIT SELECT
1431 1E23 1F1DCB           BCTA,UN DLOOP1  GO DISPLAY
1432                *
1433                *GET KEY CODE
1434                *
1435 1E26 55FE      GETKEY REDE,R1 KBDIN   READ KEYBOARD
1436 1E28 450F             ANDI,R1 H'0F'   MASK OFF UNUSED BITS
1437 1E2A 250F             EORI,R1 H'0F'   INVERT THE INPUT
1438 1E2C 17               RETC,UN
1439                *
1440                *SINGLE PASS EXIT
1441                *
1442 1E2D 040A      DISP3  LODI,R0 10
1443 1E2F F87E             BDRR,R0 $       DELAY
1444 1E31 D4F9             WRTE,R0 SEG     TURN OFF SEGMENTS
1445 1E33 0488             LODI,R0 H'88'   NO KEY PRESSED CODE
```

```
LINE ADDR  OBJECT  E SOURCE

1446 1E35 C2              STRZ    R2      SAVE IN R2
1447 1E36 17              RETC,UN
1448                *
1449                *CONVERT KEY LINE DATA TO KEY CODE
1450                *
1451 1E37 20        CODE  EORZ    R0      GET A 0
1452 1E38 D4F9            WRTE,R0 SEG     TURN OFF SEGMENTS
1453 1E3A A701            SUBI,R3 1       DECREMENT COLUMN COUNTER
1454 1E3C D4FA            WRTE,R0 DIGIT   TURN OFF COLUMNS
1455 1E3E 0604      CODE1 LODI,R2 4       LOOP COUNT
1456 1E40 51        CODE4 RRR,R1          GET WEIGHT OF KEY LINE
1457 1E41 E580            COMI,R1 H'80'   CHECK FOR 1 KEY DOWN
1458 1E43 1808            BCTR,EQ CODE2   R0 = 0,4,8, OR H'C'
1459 1E45 3404            ADDI,R0 H'04'
1460 1E47 FA77            BDRR,R2 CODE4   CHECK FOR ONLY 1 KEY
1461 1E49 0708            LODI,R3 8       MORE THAN 1 KEY DOWN OR NO KEY DOWN
1462 1E4B 9BE8            ZBRR    *ERR    GOTO ERROR
1463 1E4D E704      CODE2   COMI,R3 H'04'   NUMBER OR FUNCTION KEY?
1464 1E4F 1A05            BCTR,LT CODE3   # KEY
1465 1E51 50              RRR,R0          DIVIDE KEYLINE WEIGHT BY 2
1466 1E52 6480            IORI,R0 H'80'   FUNCTION KEY DESIGNATOR
1467 1E54 4701            ANDI,R3 H'01'   RETAIN LSB ONLY
1468 1E56 83        CODE3 ADDZ    R3      TO GET WHOLE KEYCODE
1469 1E57 C2              STRZ    R2      SAVE KEY CODE IN R2
1470 1E58 17              RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1472               ************************************************************
1473               *
1474               *
1475               *GOTO ROUTINE
1476               *
1477               *
1478               *REGISTERS USED
1479               *
1480               *R0 SCRATCH
1481               *R1 SCRATCH
1482               *R2 SCRATCH
1483               *R3 SCRATCH
1484               *R1' RESTORED
1485               *R2' RESTORED
1486               *R3' RESTORED
1487               *PSU RESTORED
1488               *PSL RESTORED
1489               *
1490               *SUBROUTINES USED
1491               *
1492               *NONE
1493               *
1494               *RAM MEMORY USED
1495               *
1496               *SSF   SINGLE STEP FLAG
1497               *BPF   BREAK POINT FLAG
1498               *BPL   BREAK POINT LOCATION
1499               *BPD   BREAK POINT DATA
1500               *LADR INDIRECT ADDRESS TO JUMP THRU
1501               *
1502               ************************************************************
1503               *
1504 1E59 0C17D0   GO     LODA,R0 SSF     GET SINGLE STEP FLAG
1505 1E5C 9819           BCFR,EQ G01     NO SINGLE STEP GOTO USER
1506 1E5E 0C17CF         LODA,R0 BPF     GET BREAK POINT FLAG
1507 1E61 1814           BCTR,EQ G01     BREAK POINT GO TO USER NO BREAK POINT
1508 1E63 0C97CD         LODA,R0 *BPL    GET USER DATA
1509 1E66 CC17CC         STRA,R0 BPD     SAVE USER DATA
1510 1E69 04B0           LODI,R0 H'B0'   WRTC,R0 BREAK POINT INSTRUCTION
1511 1E6B CC97CD         STRA,R0 *BPL    SET THE BPEAK POINT
1512 1E6E EC97CD         COMA,R0 *BPL    CHECK BREAK POINT SET OK
1513 1E71 1804           BCTR,EQ G01     GOTO USER
1514 1E73 0701           LODI,R3 1       ERROR BREAK POINT NOT SET OK
1515 1E75 9BE8           ZBRR    *ERR    GOTO ERROR
1516 1E77          G01   EQU     $
```

```
LINE ADDR  OBJECT  E SOURCE

1518                ************************************************************
1519                *
1520                *
1521                *RESTORE REGISTERS BEFORE GOING TO USER PROGRAM
1522                *
1523                *
1524                *
1525                *
1526                *REGISTERS USED
1527                *
1528                *R0 THRU R3' PSU PSL
1529                *
1530                *SUBROUTINES CALLED
1531                *
1532                *UREG+9        RESTORE PSL
1533                *
1534                *RAM MEMORY USED
1535                *
1536                *UREG          = R0
1537                *UREG+1        = R1
1538                *UREG+2        = R2
1539                *UREG+3        = R3
1540                *UREG+4        = R1'
1541                *UREG+5        = R2'
1542                *UREG+6        = R3'
1543                *UREG+7        = PSU
1544                *UREG+8        = PSL
1545                *UREG+9        = PPSL INSTRUCTION OPCODE
1546                *UREG+10       = PSL
1547                *UREG+11       = RETC,UN       INSTRUCTION OPCODE
1548                *
1549                ************************************************************
1550 1E77 0577      RESTRG LODI,R1 H'77'   PPSL INSTRUCTION OPCODE
1551 1E79 CD17FB          STRA,R1 UREG+9  CREATE A SUBROUTINE TO RESTORE PSL
1552 1E7C 0517          LODI,R1 H'17'   RETC,UN INSTRUCTION OPCODE
1553 1E7E CD17FD          STRA,R1 UREG+11
1554 1E81 7510          CPSL    RS      CLEAR REGISTER SWITCH
1555 1E83 0D17F3          LODA,R1 UREG+1  RESTORE R1
1556 1E86 0E17F4          LODA,R2 UREG+2  RESTORE R2
1557 1E89 0F17F5          LODA,R3 UREG+3  RESTORE R3
1558 1E8C 7710          PPSL    RS      SET THE REGISTER SWITCH
1559 1E8E 0D17F6          LODA,R1 UREG+4  RESTORE R1'
1560 1E91 0E17F7          LODA,R2 UREG+5  RESTORE R2'
1561 1E94 0F17F8          LODA,R3 UREG+6  RESTORE R3'
1562 1E97 0C17F9      RESTR1 LODA,R0 UREG+7  GET PSU DATA
1563 1E9A 6C17F1          IORA,R0 IFLG    SET INTERUPT INHIBIT IF REQUIRED
1564 1E9D 92              LPSU            RESTORE PSU
1565 1E9E 0C17F2          LODA,R0 UREG    RESTORE R0
1566 1EA1 75FF          CPSL    255     CLEAR PSL
1567 1EA3 3F17FB          BSTA,UN UREG+9  RESTORE PSL
1568 1EA6 1F97E8          BCTA,UN *LADR   GOTO USER
1569                *
```

```
LINE ADDR  OBJECT  E SOURCE

1571               ************************************************************
1572               *
1573               *
1574               *SUBROUTINE TO SAVE R1,R2,R3
1575               *
1576               *REGISTERS USED IN BANK ON ENTRY
1577               *
1578               *R1 SAVED IN SAVREG+1
1579               *R2 SAVED IN SAVREG+2
1580               *R3 SAVED IN SAVREG+3
1581               *
1582               *SUBROUTINES CALLED
1583               *
1584               *NONE
1585               *
1586               *RAM MEMORY USED
1587               *
1588               *SAVREG+1
1589               *SAVREG+2
1590               *SAVREG+3
1591               ************************************************
1592 1EA9 CD17DA  SAVR0  STRA,R1 SAVREG+1
1593 1EAC CE17DB  SAVR01 STRA,R2 SAVREG+2
1594 1EAF CF17DC  SAVR02 STRA,R3 SAVREG+3
1595 1EB2 17            RETC,UN
1596               ************************************************************
1597               *
1598               *
1599               *SUBROUTINE TO RESTORE R1,R2,R3
1600               *
1601               *
1602               *REGISTERS USED IN BANK ON ENTRY
1603               *
1604               *R1 RESTORED TO VALUE IN SAVREG+1
1605               *R2 RESTORED TO VALUE IN SAVREG+2
1606               *R3 RESTORED TO VALUE IN SAVERG+3
1607               *
1608               *SUBROUTINES CALLED
1609               *
1610               *NONE
1611               *
1612               *RAM MEMORY USED
1613               *
1614               *SAVREG+1
1615               *SAVREG+2
1616               *SAVREG+3
1617               ************************************************
1618 1EB3 09F5  RESTR0 LODR,R1 *SAVR0+1
1619 1EB5 0AF6       LODR,R2 *SAVR01+1
1620 1EB7 0BF7       LODR,R3 *SAVR02+1
1621 1EB9 17         RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1623                ********************************************************
1624                *
1625                *
1626                *CASSETTE IO ROUTINES
1627                *PROGRAM WRITTEN BY BBC
1628                *
1629                * 04-27-77
1630                *
1631                * THESE ROUTINES WRITES OR READS ONE BYTE TO OR FROM
1632                * THE CASSETTE IN SIMCA FORMAT.
1633                *
1634                * THE FREQUENCY IS DETERMINED BY FREQ
1635                * (CYCLE TIME IS 3.333 MICRO-SEC.)
1636                *
1637                *
1638                *ROUTINES SAVE AND REESTORE R1,R2,R3 OF CURRENT BANK
1639                *
1640                *IN RETURNS WITH DATA BYTE IN R0
1641                *OUT REQUIRES BYTE TO BE OUTPUT TO BE IN R0
1642                *
1643                *TCAS IS THE CASSETTE READ TEST USED TO SET LEVELS ON PLAY BACK
1644                *
1645                *SEE FRONT OF PROGRAM FOR DISPLAYS AND INSTRUCTIONS
1646                *
1647                *
1648                *REGISTERS USED
1649                *
1650                *R0,R1,R2,R3 ARE SCRATCH
1651                *
1652                *SUBROUTINES CALLED
1653                *
1654                *SAVR0 SAVES R1,R2,R3
1655                *RESTR0 RESTORES R1,R2,R3
1656                *
1657                *RAM MEMORY USED
1658                *
1659                *TEMP  TEMPORARY STORAGE
1660                *
1661                ********************************************************************************
1662 0011           FREQ   EQU     17      PULSE TIME ( 0.2 MSEC. )
1663 0088           SPDLY  EQU     8*FREQ  INTER-BIT SPACE
1664 0013           TMDLY  EQU     19      TIME-OUT FOR INTER-BIT DETECTION
1665 0003           PULS1  EQU     3       NUMBER OF PULSES FOR A ONE
1666 0006           PULS0  EQU     2*PULS1 NUMBER OF PULSES FOR A ZERO
1667 0009           THRES  EQU     3*PULS1 TRANSITION THRESHOLD FOR DETECTION
1668 000F           EBIT   EQU     5*PULS1 TRANSITION THRESHOLD FOR END BIT
1669                *
1670                *
1671                * SUBROUTINE OUT
1672                * WRITES ONE BYTE FROM R0 TO CASSETTE
1673                *
1674 1EBA 3B6D       OUTT   BSTR,UN SAVR0   SAVE R1-R3
1675 1EBC D407              WRTE,R0 LEDS    WRITE BYTE TO LEDS FOR DISPLAY
1676 1EBE 0708              LODI,R3 8       BIT COUNT
1677 1EC0 C8A8       OUT1   STRR,R0 *OUT5+1 TEMP    SAVE BYTE IN TEMP
```

```
LINE ADDR  OBJECT  E SOURCE

1678 1EC2 CBAA            STRR,R3 *OUT6+1 TEMP+1  SAVE BIT COUNT IN TEMP+1
1679 1EC4 0506            LODI,R1 PULS0   GET NUMBER OF PULSES FOR A ZERO
1680 1EC6 F401            TMI,R0  H'01'   TEST FOR A ONE
1681 1EC8 9801            BCFR,0  OUT2
1682 1ECA 51              RRR,R1          DIVIDE COUNT IF A ONE
1683 1ECB FB02      OUT2  BDRR,R3 OUT3    CHECK FOR LAST BIT
1684 1ECD 8506            ADDI,R1 PULS0   YES, ADD LAST BIT PULSES
1685 1ECF 0611      OUT3  LODI,R2 FREQ    LENGTH OF PULSE
1686 1ED1 0718            LODI,R3 H'18'   SET ENV AND FREQ
1687 1ED3 D7F8            WRTE,R3 CAS
1688 1ED5 FA7E            BDRR,R2 $       DELAY 10 MICRO-SEC PER ITERATION
1689 1ED7 0611            LODI,R2 FREQ    LENGTH OF PULSE
1690 1ED9 0710            LODI,R3 H'10'   RESET FREQ
1691 1EDB D7F8            WRTE,R3 CAS
1692 1EDD FA7E            BDRR,R2 $       DELAY 10 MICRO-SEC PER ITERATION
1693 1EDF F96E            BDRR,R1 OUT3    DO NEXT PULSE
1694 1EE1 0688            LODI,R2 SPDLY   INTER-BIT SPACE
1695 1EE3 0700            LODI,R3 H'00'   TURN OFF ENV AND FREQ
1696 1EE5 D7F8            WRTE,R3 CAS
1697 1EE7 FA7E            BDRR,R2 $       DELAY 10 MICRO-SEC PER ITERATION
1698                *
1699 1EE9 0C17C6    OUT5  LODA,R0 TEMP    GET CHARACTER BACK
1700 1EEC 50              RRR,R0          ROTATE RIGHT ONE PLACE
1701 1EED 0F17C7    OUT6  LODA,R3 TEMP+1  GET BIT COUNT
1702 1EF0 FB4E            BDRR,R3 OUT1    CONTINUE IF COUNT NON-ZERO
1703 1EF2 3F1EB3    OUT4  BSTA,UN RESTR0  RESTORE R1-R3
1704 1EF5 17              RETC,UN         ELSE, RETURN
1705                *
1706                * SUBROUTINE IN
1707                * READS ONE BYTE FROM CASSETTE TO R0
1708                *
1709 1EF6 3F1EA9    INN   BSTA,UN SAVR0   SAVE R1-R3
1710 1EF9 20              EORZ    R0      SET R0 TO ZERO
1711 1EFA 44FE      IN1   ANDI,R0 H'FE'   MASK OUT LOW BIT
1712 1EFC C8EC            STRR,R0 *OUT5+1 TEMP    SAVE PARTIAL BYTE
1713 1EFE 3B0A            BSTR,UN GBIT    GET NEXT BIT
1714 1F00 88E8            ADDR,R0 *OUT5+1 TEMP    ADD IN PARTIAL BYTE
1715 1F02 50              RRR,R0          MOVE NEW BIT TO HIGH POSITION
1716 1F03 5975            BRNR,R1 IN1     TEST LAST BIT FLAG
1717 1F05 3BEC            BSTR,UN *OUT4+1  YES, RESTORE R1-R3
1718 1F07 D407            WRTE,R0 LEDS    WRITE BYTE TO LEDS FOR DISPLAY
1719 1F09 17              RETC,UN         RETURN
1720                *
1721                * SUBROUTINE TO GET THE NEXT BIT FROM CASSETTE
1722                * BIT IS RETURNED AS LEAST SIGNIFICANT BIT OF R0
1723                *
1724 1F0A 0580      GBIT  LODI,R1 H'80'
1725 1F0C D5F8            WRTE,R1 CAS     SET SENSE TO CASSETTE
1726 1F0E 12              SPSU            GET PSU
1727 1F0F 07FF            LODI,R3 -1      SET TRANSITION COUNT TO -1
1728 1F11 06FF            LODI,R2 H'FF'   SET TIME-OUT TO MAX FOR FIRST TRANSITION
1729 1F13 1B02            BCTR,UN GBT3
1730 1F15 0613      GBT2  LODI,R2 TMDLY   SET END-OF-BIT DETECTION DELAY
1731 1F17 C1        GBT3  STRZ    R1      SAVE LAST COPY OF PSU IN R1
1732 1F18 8701            ADDI,R3 1       INCREMENT TRANSITION COUNTER
1733 1F1A 12        GBT4  SPSU            LOOK FOR TRANSITION
```

```
LINE ADDR  OBJECT  E SOURCE

1734 1F1B E1              COMZ    R1
1735 1F1C 9877            BCFR,EQ GBT2    IF NOT EQUAL NEW TRANSITION
1736 1F1E FA7A            BDRR,R2 GBT4    IF EQUAL, TEST TIME-OUT
1737 1F20 20              EORZ    R0      SET R0 TO ZERO
1738 1F21 D4F8            WRTE,R0 CAS     SET SENSE BACK TO USER
1739 1F23 0501            LODI,R1 1       PRESET END FLAG TO 1
1740 1F25 E70F            COMI,R3 EBIT    ENDBIT THRESHOLD
1741 1F27 9903            BCFR,GT GBT5
1742 1F29 A70C            SUBI,R3 2*PULS0 LAST BIT, SUB ENDBIT PULSES
1743 1F2B C1              STRZ    R1      AND SET END FLAG
1744 1F2C E709      GBT5  COMI,R3 THRES   IS COUNT GREATER THAN THRESHOLD
1745 1F2E 15              RETC,GT         RETURN IF TRUE
1746 1F2F 0401            LODI,R0 1       NO, SET BIT TO ONE
1747 1F31 17              RETC,UN RETURN
1748                *
1749                *
1750                * SUBROUTINE TEST CASSETTE READS
1751                *
1752 1F32 0580      TCAS  LODI,R1 H'80'   SELECT LEAST SIGNIFICANT DIGIT
1753 1F34 D5FA            WRTE,R1 DIGIT
1754 1F36 0740      TCS0  LODI,R3 H'40'   OUTPUT '-' TO DISPLAY
1755 1F38 D407      TCS1  WRTE,R0 LEDS    OUTPUT VALUE TO LED'S
1756 1F3A D7F9            WRTE,R3 DISP    OUTPUT TO DISPLAY
1757 1F3C CF17C7    TCS10 STRA,R3 TEMP+1  SAVE UD CONDITION
1758 1F3F 060A            LODI,R2 10      RETURN AFTER 10 EXACT READS
1759 1F41 CE17C6    TCS2  STRA,R2 TEMP    SAVE R2
1760 1F44 3B44            BSTR,UN GBIT    GET A BIT
1761 1F46 0AFA            LODR,R2 *TCS2+1 TEMP    RESTORE R2
1762 1F48 050C            LODI,R1 2*PULS0 NUMBER OF TRANSITIONS FOR A ZERO
1763 1F4A 60              IORZ    R0      GET CONDITION CODE FOR R0
1764 1F4B 1801            BCTR,EQ TCS3    BRANCH IF A ZERO
1765 1F4D 51              RRR,R1          DIVIDE NOMINAL TRANSITION COUNT BY 2
1766 1F4E 03        TCS3  LODZ    R3      GET COUNT IN R0
1767 1F4F 1867            BCTR,EQ TCS1    BLANK DISPLAY IF 0
1768 1F51 A1              SUBZ    R1      TEST COUNT
1769 1F52 9804            BCFR,EQ TCS4    IF NOT EQUAL, RETURN
1770 1F54 FA6B      TCS35 BDRR,R2 TCS2    IF EQUAL AND COUNT NOT UP, GET NEW BIT
1771 1F56 1B5E            BCTR,UN TCS0
1772 1F58 190A      TCS4  BCTR,GT TCS5    DETERMINE POLARITY
1773 1F5A 08E1            LODR,R0 *TCS10+1        TEMP+1 GET UD CONDITION
1774 1F5C E4DE            COMI,R0 H'DE'   DOWN CONDITION
1775 1F5E 1874            BCTR,EQ TCS35    CANT GO DIRECT FROM DOWN TO UP
1776 1F60 073E            LODI,R3 H'3E'   OUTPUT 'U' TO DISPLAY
1777 1F62 1B54            BCTR,UN TCS1
1778 1F64 07DE      TCS5  LODI,R3 H'DE'   OUTPUT 'D' TO DISPLAY
1779 1F66 1B50            BCTR,UN TCS1
```

```
LINE ADDR  OBJECT  E SOURCE

1781               ************************************************************************
1782               *
1783               *HEXTAB LOOKUP TABLE FOR HEX TO SEVEN SEGMENT
1784               *
1785               *THIS TABLE CONTAINS THE VALUES FOR LIGHTING THE
1786               *SEGMENTS FOR THE DIGITS 0 THRU 9 AND LETTERS A TO F
1787               *
1788 1F68 3F065B4F  SEGTBL DATA    H'3F,06,5B,4F,66,6D,7D,07,7F,67,77,FC,39,DE,79,71'
     1F6C 666D7D07
     1F70 7F6777FC
     1F74 39DE7971
1789               *
1790               *SEGMENT DATA FOR SYMBOLS P L U R H O = BLANK J - . Y N
1791               *
1792 1F78 73383E50         DATA    H'73,38,3E,50,76,5C,48,00,0E,40,80,6E,54'
     1F7C 765C4800
     1F80 0E40806E
     1F84 54
1793               *
1794               *THIS TABLE CONTAINS THE DISPLAY ERROR
1795               *
1796 1F85 170E1313  ERROR  DATA    H'17,0E,13,13,15,13,17,17'
     1F89 15131717
1797               *
1798               *THIS TABLE CONTAINS THE DISPLAY AD=
1799               *
1800 1F8D 170A0D16  ADR    DATA    H'17,0A,0D,16,17,17,17,17'
     1F91 17171717
1801               *
1802               *THIS TABLE CONTAINS THE DISPLAY HELLO
1803               *
1804 1F95 17140E11  HELLO  DATA    H'17,14,0E,11,11,00,17,17'
     1F99 11001717
1805               *
1806               *THIS TABLE CONTAINS THE DISPLAY BP=
1807               *
1808 1F9D 170B1016  BPEQ   DATA    H'17,0B,10,16,17,17,17,17'
     1FA1 17171717
1809               *
1810               *THIS TABLE CONTAINS THE DISPLAY R=
1811               *
1812 1FA5 17171317  REQ    DATA    H'17,17,13,17,16,17,17,17'
     1FA9 16171717
1813               *
1814               *THIS TABLE CONTAINS THE DISPLAY PC=
1815               *
1816 1FAD 17100C16  PCEQ   DATA    H'17,10,0C,16,17,17,17,17'
     1FB1 17171717
1817               *
1818               *THIS TABLE CONTAINS THE DISPLAY F=
1819               *
1820 1FB5 17170F16  FEQ    DATA    H'17,17,0F,16,17,17,17,17'
     1FB9 17171717
1821               *
1822               *THIS TABLE CONTAINS THE DISPLAY LAD=
```

```
LINE ADDR  OBJECT  E SOURCE

1823                *
1824 1FBD 110A0D16  LADEQ  DATA    H'11,0A,0D,16,17,17,17,17'
     1FC1 17171717
1825                *
1826                *THIS TABLE IS THE ASCII LOOK UP TABLE
1827                *
1828 1FC5 30313233  ASCII  DATA    A'0123456789ABCDEF'
     1FC9 34353637
     1FCD 38394142
     1FD1 43444546
1829                *
```

```
TWIN ASSEMBLER VER 2.0    INSTRUCTOR 50 PRGM 11/1/78          PAGE 0039

LINE ADDR  OBJECT  E SOURCE

1831               ************************************************************
1832               *
1833               *
1834               *USER ENTRY TO DISPLAY ROUTINES
1835               *
1836               *
1837 1FD5 BBFE      USRDSI ZBSR    *MOV    SET UP DISPLAY
1838 1FD7 03              LODZ    R3      GET DISPLAY FLAG
1839 1FD8 BBEC            ZBSR    *DISPLY GO TO DISPLAY ROUTINE
1840 1FDA 17              RETC,UN
```

```
TWIN ASSEMBLER VER 2.0    INSTRUCTOR 50 PRGM 11/1/78          PAGE 0040

LINE ADDR  OBJECT  E SOURCE

1842                ********************************************************
1843 1FD8                ORG 8192-26 THE ZBSR OR ZBRR VECTORS ARE HERE
1844                ********************************************************
1845 1FE6 1FD5      USRDSP ACON    USRDSI  USER ENTRY TO DISPLAY ROUTINES
1846 1FE8 1899      ERR    ACON    ERRI    ERROR MESSAGE
1847 1FEA 19E8      BRKPT4  ACON    BRKPTI SET DISBUF6,7 WITH CONTENTS OF R0
1848 1FEC 1E13      DISPLY ACON    DISPLI DISPLAY AND KEYBOARD ROUTINE
1849 1FEE 1EF6      IN     ACON    INN     CASSETTE INPUT ROUTINE
1850 1FF0 1EBA      OUT    ACON    OUTT    CASETTE OUT PUT
1851 1FF2 1C7B      HOUT   ACON    HOUTT   CASSETTE BINARY TO ASCII HEX OUTPUT
1852 1FF4 1A76      DISLSD ACON    DISLSI  CONVERT BYTE TO NIBBLE
1853 1FF6 1BA5      ROT    ACON    ROTI    ROTATE A  NIBBLE
1854 1FF8 1C72      CRLF   ACON    CRLFF   CARRAGE RETURN AND LINE FEED
1855 1FFA 1B3B      GNP    ACON    GNPI    GET NUMBERS
1856 1FFC 1B20      GNPA   ACON    GNPAI   GET NUMBERS AND DISPLAY
1857 1FFE 1DB6      MOV     ACON    MOVI    MOVE DATA TO DISBUF
1858                ********************************************************
1859 1800                END    SAVRG

 TOTAL ASSEMBLY ERRORS = 0000
```

Electronic components and materials

for professional, industrial and consumer uses

from the world-wide Philips Group of Companies

Argentina: FAPESA I.y.C., Av. Crovara 2550, Tablada, Prov. de BUENOS AIRES, Tel. 652-7438/7478.
Australia: PHILIPS INDUSTRIES HOLDINGS LTD., Elcoma Division, 67 Mars Road, LANE COVE, 2066, N.S.W., Tel. 427 08 88.
Austria: ÖSTERREICHISCHE PHILIPS BAUELEMENTE Industrie G.m.b.H., Triester Str. 64, A-1101 WIEN, Tel. 62 91 11.
Belgium: M.B.L.E., 80, rue des Deux Gares, B-1070 BRUXELLES, Tel. 523 00 00.
Brazil: IBRAPE, Caixa Postal 7383, Av. Brigadeiro Faria Lima, 1735 SAO PAULO, SP, Tel. (011) 211-2600.
Canada: PHILIPS ELECTRONICS LTD., Electron Devices Div., 601 Milner Ave., SCARBOROUGH, Ontario, M1B 1M8, Tel. 292-5161.
Chile: PHILIPS CHILENA S.A., Av. Santa Maria 0760, SANTIAGO, Tel. 39-40 01.
Colombia: SADAPE S.A., P.O. Box 9805, Calle 13, No. 51 + 39, BOGOTA D.E. 1., Tel. 600 600.
Denmark: MINIWATT A/S, Emdrupvej 115A, DK-2400 KØBENHAVN NV., Tel. (01) 69 16 22.
Finland: OY PHILIPS AB, Elcoma Division, Kaivokatu 8, SF-00100 HELSINKI 10, Tel. 1 72 71.
France: R.T.C. LA RADIOTECHNIQUE-COMPELEC, 130 Avenue Ledru Rollin, F-75540 PARIS 11, Tel. 355-44-99.
Germany: VALVO, UB Bauelemente der Philips G.m.b.H., Valvo Haus, Burchardstrasse 19, D-2 HAMBURG 1, Tel. (040) 3296-1.
Greece: PHILIPS S.A. HELLENIQUE, Elcoma Division, 52, Av. Syngrou, ATHENS, Tel. 915 311.
Hong Kong: PHILIPS HONG KONG LTD., Elcoma Div., 15/F Philips Ind. Bldg., 24-28 Kung Yip St., KWAI CHUNG, Tel. NT 24 51 21.
India: PEICO ELECTRONICS & ELECTRICALS LTD., Ramon House, 169 Backbay Reclamation, BOMBAY 400020, Tel. 295144.
Indonesia: P.T. PHILIPS-RALIN ELECTRONICS, Elcoma Division, 'Timah' Building, Jl. Jen. Gatot Subroto, P.O. Box 220, JAKARTA, Tel. 44 163.
Ireland: PHILIPS ELECTRICAL (IRELAND) LTD., Newstead, Clonskeagh, DUBLIN 14, Tel. 69 33 55.
Italy: PHILIPS S.p.A., Sezione Elcoma, Piazza IV Novembre 3, I-20124 MILANO, Tel. 2-6994.
Japan: NIHON PHILIPS CORP., Shuwa Shinagawa Bldg., 26-33 Takanawa 3-chome, Minato-ku, TOKYO (108), Tel. 448-5611.
(IC Products) SIGNETICS JAPAN, LTD, TOKYO, Tel. (03)230-1521.
Korea: PHILIPS ELECTRONICS (KOREA) LTD., Elcoma Div., Philips House, 260-199 Itaewon-dong, Yongsan-ku, C.P.O. Box 3680, SEOUL, Tel. 794-4202.
Malaysia: PHILIPS MALAYSIA SDN. BERHAD, Lot 2, Jalan 222, Section 14, Petaling Jaya, P.O.B. 2163, KUALA LUMPUR, Selangor, Tel. 77 44 11.
Mexico: ELECTRONICA S.A. de C.V., Varsovia No. 36, MEXICO 6, D.F., Tel. 533-11-80.
Netherlands: PHILIPS NEDERLAND B.V., Afd. Elonco, Boschdijk 525, 5600 PB EINDHOVEN, Tel. (040) 79 33 33.
New Zealand: PHILIPS ELECTRICAL IND. LTD., Elcoma Division, 2 Wagener Place, St. Lukes, AUCKLAND, Tel. 867 119.
Norway: NORSK A/S PHILIPS, Electronica, Sørkedalsveien 6, OSLO 3, Tel. 46 38 90.
Peru: CADESA, Rocca de Vergallo 247, LIMA 17, Tel. 62 85 99.
Philippines: PHILIPS INDUSTRIAL DEV. INC., 2246 Pasong Tamo, P.O. Box 911, Makati Comm. Centre, MAKATI-RIZAL 3116, Tel. 86-89-51 to 59.
Portugal: PHILIPS PORTUGESA S.A.R.L., Av. Eng. Duharte Pacheco 6, LISBOA 1, Tel. 68 31 21.
Singapore: PHILIPS PROJECT DEV. (Singapore) PTE LTD., Elcoma Div., P.O.B. 340, Toa Payoh CPO, Lorong 1, Toa Payoh, SINGAPORE 12, Tel. 53 88 11.
South Africa: EDAC (Pty.) Ltd., 3rd Floor Rainer House, Upper Railway Rd. & Ove St., New Doornfontein, JOHANNESBURG 2001, Tel. 614-2362/9.
Spain: COPRESA S.A., Balmes 22, BARCELONA 7, Tel. 301 63 12.
Sweden: A.B. ELCOMA, Lidingövägen 50, S-115 84 STOCKHOLM 27, Tel. 08/67 97 80.
Switzerland: PHILIPS A.G., Elcoma Dept., Allmendstrasse 140-142, CH-8027 ZÜRICH, Tel. 01/43 22 11.
Taiwan: PHILIPS TAIWAN LTD., 3rd Fl., San Min Building, 57-1, Chung Shan N. Rd, Section 2, P.O. Box 22978, TAIPEI, Tel. 5513101-5.
Thailand: PHILIPS ELECTRICAL CO. OF THAILAND LTD., 283 Silom Road, P.O. Box 961, BANGKOK, Tel. 233-6330-9.
Turkey: TÜRK PHILIPS TICARET A.S., EMET Department, Inonu Cad. No. 78-80, ISTANBUL, Tel. 43 59 10.
United Kingdom: MULLARD LTD., Mullard House, Torrington Place, LONDON WC1E 7HD, Tel. 01-580 6633.
United States: (Active devices & Materials) AMPEREX SALES CORP., Providence Pike, SLATERSVILLE, R.I. 02876, Tel. (401) 762-9000.
(Passive devices) MEPCO/ELECTRA INC., Columbia Rd., MORRISTOWN, N.J. 07960, Tel. (201) 539-2000.
(IC Products) SIGNETICS CORPORATION, 811 East Arques Avenue, SUNNYVALE, California 94086, Tel. (408) 739-7700.
Uruguay: LUZILECTRON S.A., Rondeau 1567, piso 5, MONTEVIDEO, Tel. 9 43 21.
Venezuela: IND. VENEZOLANAS PHILIPS S.A., Elcoma Dept., A. Ppal de los Ruices, Edif. Centro Colgate, CARACAS, Tel. 36 05 11.

A15

Printed in The Netherlands

9398 6076 0221