CODEN:LUTFD2/(TFRT-5287)/0-44/(1982)

UPPBYGGNAD AV EN DIGITAL BILHANTERING I ETT SVEPELEKTRONMIKROSKOP

KAJ ANDERSSON

INSTITUTIONEN FÖR REGLERTEKNIK
LUNDS TEKNISKA HÖGSKOLA

OKTOBER 1982

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AUTOMATIC CONTROL
Box 725
S 220 07 Lund 7 Sweden

Document name
Report

Date of issue
October 1982

Document number
CODEN:LUTFD2/(TFRT-5287)/0-44/(1982)

Author(s)
Kaj Andersson

Supervisor
Björn Wittenmark

Sponsoring organization

Title and subtitle
Uppbyggnad av en digital bilhantering i ett svepelektronmikroskop. (Subroutines for taking up and displaying pictures from a computer controlled scanning-electronmicroscope (SEM)).

Abstract

The programs are made to decide testingpoints in an integrated circuit. The measurements on the IC are performed in a SEM using digital electronbeamdeflection and digital secondary-electrondetection. A PDP11/23 computer and a QRGB-256 video-interface are used. Programs are written in MACRO-assembler and FORTRAN. Pictures may be taken up and stored on a massstorage (floppy-disk) for later retrieval. The picture image may be changed (color and displayingarea). Co-ordinates in the displayed pictures may be estimated with a joystickmovable cross. Also the picturearea may be moved with the joystick.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
0-44

Recipient's notes

Security classification

Subroutines for taking up and displaying pictures from a computer controlled scanning-electronmicroscope (SEM).

The programs are made to decide testingpoints in an integrated circuit. The measurements on the IC are performed in a SEM using digital electronbeamdeflection and digital secondary-electrondetection. A PDP11/23 computer and a QRGB-256 video-interface are used. Programs are written in MACRO-assembler and FORTRAN. Pictures may be taken up and stored on a mass-storage (floppy-disk) for later retrieval. The picture image may be changed (color and displayingarea). Coordinates in the displayed pictures may be estimated with a joystickmovable cross. Also the picturearea may be moved with the joystick.

UPPBYGGNAD AV EN DIGITAL BILDHANTERING I ETT SVEPELEKTRONMIKROSKOP

Kaj Andersson

oktober 1982

INNEHÅLLSFÖRTECKNING

APPENDIXFÖRTECKNING

Appendix i separat del

SAMMANFATTNING

Rapporten beskriver programvara för ett digitalt styrt svep-elektronmikroskop (SEM).

Integrerade kretsars ökande komplexitet skapar behov av medel och metoder att testa ett stort antal noder i samma integrerade krets. En här beskriven metod är att använda en elektronstråle som prob. Elektronstrålen riktas då mot testnoden och de från noden exciterade sekundärelektronerna detekteras. Härigenom fås en uppfattning om spänningsnivån i noden. Genom att avlänka elektronstrålen digitalt fås en hög punktupplösning samtidigt som ett stort antal testnoder kan mätas. Sekundärelektrondetektorvärdet analog-digitalomvandlas för att behandlas i en dator.

Programvaran används för att bestämma testnodernas läge på den testade kretsen. Den består av rutiner som visar bilder av kretsen och rutiner som bestämmer koordinater för punkter i den presenterade bilden. Bilder av områden på kretsen kan tas upp on-line och off-line.

On-line innebär att den upptagna bilden presenteras samtidigt som avsökningen av kretsen sker (funktion som i ett vanligt SEM). Området kan bestämmas med joystick eller genom parametrar.

Off-line innebär att den upptagna bilden lagras i en buffert (i dator eller på diskett) för att senare kunna presenteras och analyseras.

Presentationsprogrammen utnyttjar resurserna parallellt och låter elektronstrålen stabiliseras samtidigt som sekundärelektrondetektorns värde ad-omvandlas. I on-line programmet sker även färgkodning och presentation parallellt.

Presentationen sker på en färgvideomonitor. Området på monitorn, där bilden presenteras, kan väljas så att flera kretsområden kan presenteras samtidigt. Områdenas storlek kan varieras. På monitorn visade helbilder kan lagras undan för senare presentation.

Färgkoden skapas genom att detektorns ad-omvandlade värden jämföres med olika gränser. Intervallet mellan varje gräns motsvarar en färg. Färgerna och gränserna kan anpassas till presentationsobjektet.

För att definiera bildpunkter finns rutiner som lägger ut ett hårkors i den presenterade bilden och låter detta flyttas med en joystick. Hårkorsets storlek och färg kan anpassas till bakgrunden.

(I appendix finns ett flödesschema över de använda rutinernas beroende) ./.

1 TESTNING AV LSI / VLSI-KRETSAR MED ELEKTRONSTRÅLE

1.1 Bakgrund

Med ökad komplexitet på integrerade kretsar blir det allt svårare och dyrare att genomföra testning. Det interna antalet noder i en krets är idag stort relativt det antal noder som finns tillgängliga för in- och ut-signaler. Automatisk testning med dagens metodik av ett minne om 64 kbit tar upp till 1 minut och kostar ca 10 kronor, vilket är ungefär lika mycket som kretsen kostat att processa fram till testningen.

År 1985 kommer om utvecklingstrenden håller i sig ytterligare fem år, antalet transistorer per integrerad krets att kunna uppgå till 1 - 10 miljoner. Ett viktigt villkor för denna utveckling är att det finns möjligheter att analysera och testa dessa kretsar.

En ny metod för testning av LSI och VLSI -kretsar är att nyttja en elektronstråle för datorstyrd testning. Med en elektronstråle kan spänningen i en intern nod i en krets snabbt mätas och jämföras med ett förväntat värde. Upplösningen i spänning kan uppgå till 100 mV eller bättre. Med denna upplösning kan per sekund ca 100 000 noder avläsas. Vidare kan pulser med korta stigtider (< 1 ns) registreras med samplingsteknik.

Testning med elektronstråle är av speciellt intresse vid analys av nyutvecklad krets. För närvarande används mekaniska prober med spetsradier på 1 - 10 mikrometer. Dessa är mycket svåra att positionera på en ledare med 1 - 3 mikrometers bredd och sliter ofta sönder metalliseringen. Därtill lastar de ner kretsen kraftigt med sin kapacitans. (se figur 1). Alternativet att använda en elektronstråle i stället för mekaniska prober innebär väsentliga fördelar som liten kretspåverkan samt en möjlighet att snabbt och precist flytta proben mellan mätpunkter. Denna spänningsmätande elektronstråleprobe kan väsentligt förbättra möjligheten att lokalisera och analysera fel i integrerade kretsar speciellt i kretsar med hög komplexitet och i kretsar tillverkade med komponentdimensioner i mikron eller submikronområdet.

Elektornstrålen kan liksom i ett svepelektronmikroskop (SEM) också användas för andra mätningar som t ex analys av kretsens geometriska toleranser och atomära sammansättning. Dessa mätningar är av intresse i samband med t ex analys av orsak till kretsfel och kvalitetskarakterisering av en krets.

För att realisera ett testsystem som utnyttjar en elektronstråleprobe utgår man ifrån ett konventionellt svepelektronmikroskop. Man får därigenom möjlighet att vid stora förstoringar rent visuellt studera kiselskivan samtidigt som man kan dra nytta av elektronstråleprobens fördelar vid spänningsmätningar.

Sammanfattningsvis så har elektronstråleproben följande fördelar:

1 Elektronstrålen kan fokuseras med elektronoptiska linser till en stråldiameter på mindre än 1 mikrometer.

2 Elektronstrålen kan snabbt och exakt avböjas av elektriska eller magnetiska fält så att den kan positioneras på en önskad punkt.

3 Elektronernas massa är så liten att de inte vid de accelerationsspänningar som används får en sådan stor rörelseenergi att de kan förstöra objektet vid testning.

4 Den elektriska belastningen av kretsen beror av accelerationsspänningen och strålströmmen. De kan väljas på ett sådant sätt att testobjektets funktion ej äventyras och så att mätresultatet ej påverkas.

En nackdel är att mätningarna måste genomföras i vakuum.

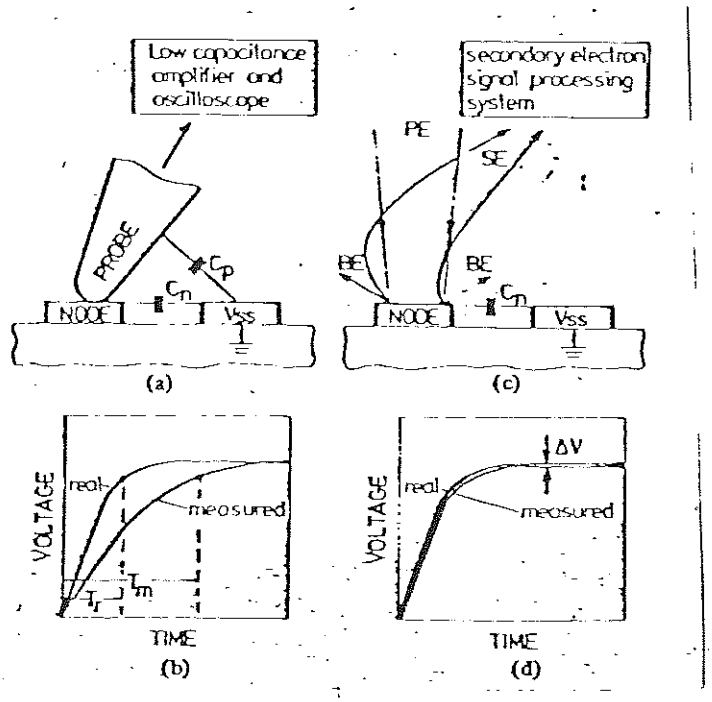
1.2 Inledning

I samband med tidigare projekt inom området elektronstrålelitografi har inom MIKROELEKTRONIKGRUPPEN en bred kunskap byggts upp kring att generera, avlänka och digitalt styra elektronstrålar samt att detektera vad som sker i strålens träffpunkt. Därtill har inom CAD-gruppen vid Institutionen för Tillämpad Elektronik (KTH) en erfarenhet byggts upp inom området Testning av digitala kretsar och system. Metoder för logisk simulering av kretsar samt verkan av logiska fel har utvecklats liksom metoder för att generera effektiva testsekvenser.

Med denna bakgrund har inom MIKROELEKTRONIKGRUPPENS ram startats ett projekt (lett av tekn dr Göran Stille) med syftet att utveckla en metod för testning av LSI och VLSI-kretsar baserad på användning av en datorstyrd elektronstråle. Efter inledande förstudier har man valt att under den närmaste tiden koncentrera insatserna på spänningsmätning för analys av IC-kretsars inre noder. Denna inriktning innebär att man nu arbetar med att:

- utveckla en spänningskänslig detektor
- utveckla en metod för kalibrering av elektronstråleavlänkningen
- installera ett snabbt buffertminne för lagring av de testsekvenser som skall påföras kretsen
- utveckla den programvara som behövs för att en användare (kretskonstruktör) lätt skall kunna hantera mätsystemet t ex som en flerkanaligt oscilloskop/logikanalysator
- bygga en enhet för numeriskt styrd sampling

figur 1



De olika belastningsfallen som uppstår vid användning av metallprobe (a) respektive elektronstråleprobe (c) kan jämföras vid mätningar på snabba pulser. Antag att en puls har en viss stigtid T_r. Metallproben med kapacitansen C_p ger vid mätningen en ökad stigtid. Elektronstråleproben ger däremot oförändrad stigtid men orsakar istället ett litet mätfel V. Mätfelet uppstår p g a att en liten ström flyter till eller från mätpunkten (noden) därefter att antalet primärelektroner (PE) inte överensstämmer med antalet sekundärelektroner (SE) plus antalet reflekterade elektroner (BE). Mätfelet ΔV är av storleksordningen 1 mV och är därför vid de flesta tillämpningar försumbar.

2 DEFINITION AV ARBETET

2.1 Specifikation av uppdraget

Uppbyggnad av en digital bildhantering i ett svepelektronmikroskop.

Arbetet innebär att:

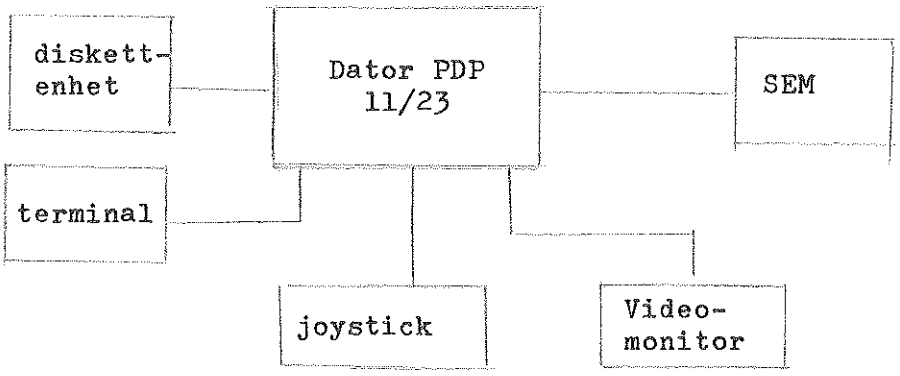
- ta upp en SEM-bild från ett svepelektronmikroskop och lagra undan denna
- presentera denna bild på en videomonitor
- att med hjälp av en "joystick" och ett hårkors superponerat på SEM-bilden genomföra dimensionsmätningar i bilden och manövrering av bildfältets läge och storlek

Arbetet innebär i väsentlig del analys av funktionen, specificering och utveckling av programvara samt test och demonstration av den önskade funktionen.

Programvaran utvecklas väsentligen i FORTRAN med några subrutiner i assembler. Vid arbetets slut skall programvaran vara väl dokumenterad.

2.2 Hårdvarusystemet

Systemets uppbyggnad framgår nedan.



Datorn skall styras med joystick eller terminal att avlänka och ta upp bildområden i SEMet (svepelektronmikroskopet). De upptagna bilderna presenteras på videomonitorn och/eller lagras undan.

2.2.1 Datorsystemet

Den dator som används är PDP 11/23. Denna har extended memory och floating-pointenhet.

Processorn har åtta styck 16-bitars GPR (general purpose register). Dessa register kan arbeta som accumulatorer, indexregister, autoinkrement- och autodekrement-register eller som stackpekare. Aritmetiska operationer kan vara från ett GPR till ett annat, från en minnesposition eller yttre enhetsregister till ett annat eller mellan minnespositioner eller yttre enhetsregister och GPR.

General purpose register:

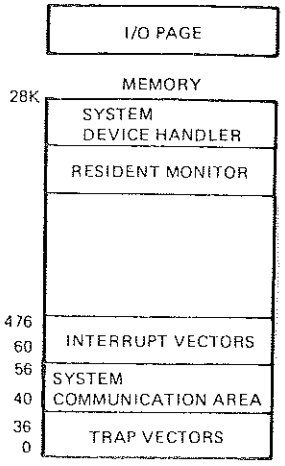
	R0
	R1
	R2
	R3
	R4
	R5
stackpekare	R6
programräknare	R7

Genom att använda GPR fås en kortare exekveringstid än om primärminnet adresseras.
Instruktionsuppsättningen innefattar byte- och ord-instruktioner. PDP11 använder dubbeloperandinstruktioner, vilket möjliggör många operationer med en instruktion, t ex ADD A,B (B=B+A) och MOV A,B (B=A). Därtill kommer en uppsättning villkorliga hopp. Man kan använda sju styck adresseringsmoder.
Registermode INC R3, register deffered mode INC (R3)(R3 används som adress), autoincrement mode INC (R3)+ (som föregående därefter R3=R3+2), autoincrement deferred mode INC @(R2)+ (R2 används som adress till adressen av operanden), autodecrement mode INCB-(R0) (R0 dekrementeras och används som adress), autodecrement deferred mode CLR @-(R0) (R0=R0-2, därefter nollställs adressen given av R0), index mode CLR 200(R4) (200 adderas till R4 och används som adress), index deferred mode Add @1000(R2),R1 (1000 och innehållet i R2 adderas och utgör adressen).

Typiska instruktionstider är 4,6 us för MOV A,B.

Det direkt adresserbara minnets storlek är 64 kB. De övre 4096 adresserna används till kommunikation med yttre enheter (interface). Detta system har 128 kB som adresseras genom en extended memorymonitor.
Ett 16-bitars ord är indelat i en hög och en låg byte. Ordadresser är alltid jämna.

Minnets utnyttjande framgår nedan.



Massminnet utgörs av RX02 diskettenhet. Enheten består av två flexskiveinkast med vardera 512 kB (double density), Den kontrolleras av RXV21 diskkontrollenhet. Varje diskett har 77 spår med vardera 26 sektorer. Varje sektor på spåret är 256 ord (ett block). Medelaccesstiden är 262 ms (se appendix, RXV21). ./.

Systemet arbetar under operativsystemet RT-11. Under RT-11 finns bl a Fortran kompilator och MACRO-assembler liksom systembibliotek med systemrutiner.

När ett FORTRAN-program kallar på en rutin skriven i MACRO (CALL SUBR(A,B)) så får MACRO-rutinen argumenten på följande sätt

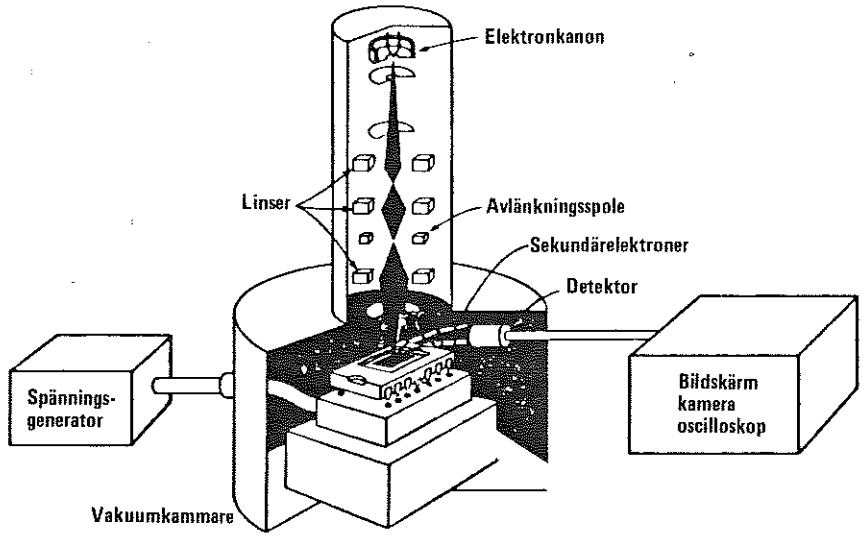
```
R5 ⟹    0          antal argument
        adress till argument 1 (A)
          -"-                2 (B)

        o s v
```

MACRO/SYSTEM interface vid programmed request.
De vid anropet utnyttjade parametrarna läggs i en EMT AREA genom assemblern eller genom instruktioner i programkoden. Assemblern lägger vidare ut instruktioner för den programmerade begäran. Programflödet blir enligt nedan.

```
USER PROGRAM:      Programmed Request

SYSTEM TRAP AREA:  Points to EMT Processor Code   30
                   PSW                            32

RMON:              EMT Processor
                   RTI Instruction

SYSCOM AREA:       EMT Error                      52

                   User Program
```

2.2.2 Enheter för bildupptagning

Bildupptagningssystemet är uppbyggt kring ett Mini-Sem svep-elektronmikroskop. I detta används vakumarbetskammare, xy-bord samt elektronkanon och scintillator. Mini-Semet fungerar normalt då det digitalt styrda systemet ej används. Normalt innebär att objektet sveps av en elektronstråle och detektorns utslag presenteras som en gråskala på en monitor. Man kan i Semet ställa in funktioner som elektronstrålens storlek (spotsize), svephastighet av bilden, bildstorlek, kontrast och ljus i bilden. Man kan också justera spänningsnivåer på filtergaller.



För att kunna avlänka elektronstrålen digitalt finns två avlänkningsförstärkare. Dessa får avlänkningsadressen från två 16-bitars da-omvandlare (DAC 1136K). DA-omvandlarna får adressinformationen från en parallellport DRV11-J. DRV11-J kan ta fyra ord om 16 bitar. Den kan sättas i olika moder (se appendix). DA-omvandlaren har en typisk omvandlingstid på 6 us då den arbetar i strömmode. Till avlänkningsförstärkarna är anslutet två avlänkningsspolar som avlänkar elektronstrålen över en yta. (Noggrannhet o,1 um)

./.

De genom elektronstrålen utslagna och exciteradeelektronerna detekteras med en detektor. Detektorn består av ett extraktionsgaller som drar ut dessa elektroner från objektet, ett filtergaller som repellerar de elektroner som ej har tillräcklig energi. De elektroner som släppts igenom filtret får träffa en scintillator och orsakar där ljuspulser. Dessa ljuspulser detekteras och förstärks sedan i en fotomultiplikator. Den från fotomultiplikatorn erhållna signalen omvandlas till digital form med ad-omvandlaren ADV11-A, för att kunna behandlas i datorn. ADV11-A är en 12-bitars ad-omvandlare, som arbetar med successiv approximation. Omvandlaren har 8 kanaler (kvasi-

differentiella). Genom att skriva ett kontrollord på en given adress, kan man få ADV11-A-enheten att följa en valfri kanal och att sampla. En omvandling tar ca 32 us. (se appendix). ./.

2.2.3 Delar för operatörskommunikation

Terminalen som används är Datamedia DT80/1.

Joystickens utslag styr två potentiometrar som fungerar som spänningsdelare. Spänningsvärdet läses av med ad-omvandlaren ADV11-A (kanal 6, 7).

För att kunna presentera den digitalt svepta bilden används i systemet ett videointerfacekort QRGB-256 från Matrox. Interfacet kan definiera 16 färger (4 bitar) i 256x256 punkter. Inskrivning av en färg i en punkt sker genom att först läsa in adressen till punkten på adressen och därefter färgbyten till färgadressen. Läsning går till på motsvarande sätt. Man kan genom kontrollbyten bl a slå på och av signalen ut från interfacet liksom man kan styra var bilden på skärmen skall börja presenteras (scrol) (se appendix). För att trimma in interfacet har ett intrimningsprogram skrivits. Detta lägger ut färgbalkar. ./.
Interfacet är anslutet till en färgmonitor.

3. ARBETETS GENOMFÖRANDE

3.1 Analys och ansatser

Programvarans uppgift är att göra det möjligt för operatören att hitta och definiera intressanta punkter och områden på en integrerad krets samt att utgöra grund för analys.

Ansats: Vid koordinatbestämning presenteras först en SEM-bild på videomonitorn. Därefter bestäms bildpunktskoordinaterna utifrån den presenterade bilden.

3.1.1 Bildhantering

Upptagning av SEM-bilder

För att ta upp en bildpunkt riktas elektronstrålen mot den önskade koordinaten. Koordinaten anges med ett 16-bitars ord (horisontellt och vertikalt) och skrivs in i parallellporten för avlänkningsförstärkaren. Efter att strålen har stabiliserats samplas sekundärelektrondetektorns värde. Detta värde ad-omvandlas därefter till 12 bitar. Omvandlingen tar ca 30 us.

För att ta upp bilder måste man specificera vilket område man vill ta upp och vilken skala man vill använda. Med skala avses avståndet mellan de punkter som strålen riktas mot. Området kan anges av antal punkter åt olika håll utgående från en referenspunkt. Ur programmeringssynpunkt torde det vara lättast om man angav något hörn som referenspunkt och att bilden byggs upp linjevis med samma antal punkter per linje.

Ansats: Området specifieeras med övre vänstra hörnpunktskoordinaten och antalet punkter per linje och antalet linjer. Linjerna sveps från vänster till höger. Olika skala kan användas horisontellt och vertikalt.

Presentation av SEM-bilder

Vid presentation skall värdet från detektorn omvandlas till bildpunkter på videomonitorn. Värdet har 12 bitar (4096 nivåer). Detta värde skall omvandlas till 16-färger (eller gråniväer) (se appendix QRGB-256). Omvandlingen kan ske t ex genom ./.

1. matematisk skalning
F (detektorvärdet) = färgen
F kan vara t ex en linjär, kvadratisk eller exponentiell skalfunktion.
2. jämförelseskalning
Det från detektorn erhållna värdet jämföres med gränser. Intervallet mellan varje gräns motsvarar en färg. Denna metod innefattar den matematiska skalningen som specialfall (den är dessutom snabbare för datorn).

Bilden presenteras linjevis på videomonitorn på samma sätt som bildupptagningen. Om man kan definiera var monitorbilden skall börja och dess storlek, kan flera bilder presenteras samtidigt.

Ansats: Omvandlingen från detektorvärde till färgkod sker genom jämförelseskalning. Monitorbildens övre vänstra hörn anges som startpunkt och bildytan definieras som antal punkter per linje och antal linjer.

Undanlagring av SEM-bilder

Bilden skall lagras undan för att senare kunna analyseras och användas som referens. Antalet bildpunkter i videointerfacet är 256x256=65536. För att kunna lagra undan en bilds detektorvärden (12-bitars ord) fordras 128 kB. Detta motsvarar hela det nuvarande primärminnet eller 1/4 av innehållet i en diskett (512 kB). Minnesbehovet kan minskas om man kodar bort redundant information. För att skriva ut 128 kB på diskett åtgår ca 10 ms per block (256 ord) + en medelaccesstid per block på 262 ms. Detta ger en total tid av ca 1 minut att skriva ut en bild på diskett om man använder systemrutiner (programmed request). Genom att i stället använda DMA (direkt minnesaccess) kan skriv/lästiden reduceras till några sekunder. DMA läser in och skriver ut stora minnesblock i primärminnet.

Ansats: Skapa en rutin som avlänkar och tar upp detektorvärden och lägger dessa i en vektor i primärminnet. Vektorn kan därefter skrivas ut på diskett (inledningsvis med systemrutiner). Om rutinen även håller reda på detektor max- och detektor min-värde fås gränsparametrar till färgskalningen. Rutinerna kan då nyttjas både om man använder DMA, systemrutiner eller väljer att expandera primärminnet och lagra bilddetektorvärdena där.

Alternativ undanlagring av bilder

I ett vanligt SEM lagras bilden ej undan utan bilden kodas och presenteras på en monitor. Om denna lösning implementerades med en snabb rutin skulle operatören kunna arbeta med stor interaktivitet. Avlänkning och detektering av en hel monitorbild 256x256pktr tar vid optimalitet ca 2s (256x256x30 us) Den på monitorn genom videointerfacet presenterade bilden kan också utgöra grund för analys och referens. Om videointerfaceinnehållet för en bild (256x256x4 bits) kan packas och lagras på diskett finns tillsammans med SEM-bildupptagning utan undanlagring ett minnesbesparande alternativ till undanlagring av detektorvärden. Minnesbehovet blir då 64 block för en fullbild.

Ansats: Skapa en snabb rutin som avkodar och presenterar på videomonitorn, som ett"vanligt SEM". Implementera därtill en rutin som packar och lagrar undan videointerfacets innehåll på diskett samt en rutin som packar upp och presenterar det undanlagrade.

3.1.2 Användaraspekter

Definition av bilden

Om den fulla monitorbilden byggs upp av bildsegment kan flera bildområden visas på samma gång. Om färger och färggränser vid presentation kan ändras fås en anpassning till objektet.

Ansats: Monitorbildytan byggs upp av 16 kvadrater (bildsegment) av 64x64 bildpunkter. Den bild som tas upp måste överensstämma med segmentindelningen. Monitorområdet definieras av startpunktskoordinat (11..44) och bildstorlek (11..44) på matrisvis.
Färgkodningsparametrarna kan ändras.

Definition av bildområden med joystick

Genom att flytta bildområdet med hjälp av joystick fås en möjlighet till snabb definition av detta. Upptagning av en helt ny bild tar förhållandevis lång tid jämfört med att i videointerfacet flytta tidigare visade bilddelar (förhållandet 4/1).
Om joystickutslaget utgör villkor för upptagning fås en möjlighet att ta upp nya bilder efter t ex att ha ändrat någon nod på den integrerade kretsen.

Ansats: Vid definition av område med joystick kan först den bilddel flyttas som skall vara kvar inom videointerfacet, därefter fylls bildytan ut med ny upptagning. Joystickens utslag kan utgöra upptagningsvillkor.

Undanlagring av bilder

Om bilder som skall undanlagras tas upp som små rektanglar kan man vid senare analys utnyttja lokaliteten. Man kan då också på ett enkelt sätt presentera delar av de undanlagrade bilderna. Väldefinierade filtyper underlättar.

Ansats: Bilder som skall undanlagras tas upp som vertikala kolonner 32 punkter breda (1/2 bildsegmentsbredd) med början uppe till vänster. Man kan då definiera bildområde på en upptagen bild på ett halvt segments format. Strikta filbeteckningar används.

3.1.3 Koordinatbestämning med joystick

Joystickens utslag kan tolkas som riktning, tid, hastighet och steglängd. Joysticken skall reglera flyttning av hårkors för bestämning av koordinater samt styra bestämning av bildområde vid flyttning (se ovan).

Ansats: Joystickutslagets riktning ger åt vilket håll hårkors eller bild skall flyttas. Om utslaget är ungefär 0 skall inget ske. Utslaget reglerar tiden mellan flyttningar då utslaget är litet, vilket ger hög precision. Stora utslag flyttas stegvis och ger snabbt rätt område.

Hårkorset måste utformas så att kontrast mot bakgrunden blir god. Detta kan ske genom:
1.korset färganpassas automatiskt till bakgrunden
2.operatören kan reglera färg och storlek
Alternativ två ger enklare programmering.

Ansats: Operatören bestämmer färg och storlek på hårkorset.

Koordinaterna fås genom omvandling av videomonitorkoordinater till verkliga avlänkningskoordinater. Avstånd och dimensioner fås genom kända avståndsformler.

Ansats: Korset förflyttas med joystick till en intressant punkt. Punktens videomonitorkoordinat erhålles. Utgående från monitorkoordinaten räknas den verkliga avlänkningskoordinaten ut. Kända avståndsformler används för att få avstånd och dimensioner.

3.2 Assembler eller högnivåspråk

Då avlänkning, ad-enhet och videointerface styrs på bitnivå måste rutiner skrivas i assembler som hanterar dessa enheter. Det stora antalet bildpunkter gör att assembler-skrivna rutiner ger snabbast resultat vid upptagning och presentation. Bildundanlagringsrutinen skrivs också i assembler för att kunna packa bildinformationen. Korset implementeras också i assembler för att få tillräcklig interaktivitet. Administrativa rutiner skrivs i FORTRAN.

3.3 Resultat

Beskrivning av programpaket för digitalt styrt SEM. Programpaketet ger följande möjligheter:

- komposition av bilder genom bestämning av bildformat och position på monitorn

- ta upp bilder med samtidig presentation (on-line)

- flytta runt bilden över ett kretsområde med "joystick"

- låta joysticken utgöra villkor för upptagning

- lagra undan och presentera bilder över ett kretsområde (off-line)

- lagra undan och presentera de komponerade monitorbilderna

- modifiera färgtolkning vid presentation av kretsbilder

- flytta runt ett kors över bildytan med joystick

- använda korset för att bestämma koordinater

(I appendix finns ett flödesschema som visar beroendet mellan rutinerna liksom programlistningar) ./.

3.3.1 Rutiner som arbetar mot svepelektronmikroskopet och presenterar bilder.

Funktionsbeskrivning

Avlänkning sker linjevis från höger till vänster uppifrån och ner med start i övre vänstra hörnet. Den horisontella koordinaten ökar åt vänster och den vertikala nedåt. Adressering i SAVDEF och SEMPIC (flödesschema finns i appendix) sker genom att parallellporten DRV11-J sätts i skrivmod. Detta innebär att porten fungerar som en adress i minnet. Portens utgång styr da-omvandlaren i avlänkningsförstärkaren. Kanal A används till horisontell och kanal B till vertikal avlänkning.

Efter att den avlänkade elektronstrålen har stabiliserat sig, vilket tar ca 30 us, samplas sekundärelektrondetektorns värde. Värdets da-omvandling tar ca 30 us. Genom att låta elektronstrålen stabiliseras samtidigt som ad-omvandlingen sker sparas tid. I SEMPIC sker även färgkodningen samtidigt. Detektorvärdet läggs i SAVDEF i den vid anropet givna vektorn.

(ad-omvandlaren beskrivs i systembeskrivningsdelen)

Kodningen från värde till färg i SEMPIC och DISPLY sker genom att värdet jämföres med gränser. Intervallet mellan varje gräns motsvaras av en färg. Genom att låta gränserna och färgerna vara parametrar fås en stor flexibilitet. En ytterligare fördel är att värden utanför undre och övre gräns ej medför någon förändning, vilket möjliggör multipelkodning av samma bild. En snabb kodning fås genom att

a) värdena jämförs med gränserna i form av ett binärt träd. Detta ger 4 eller 5 jämförelser per punkt.

				L1	ingen färgändring på monitorn
			L2		färg C1
		L3			färg C2
			L4		färg C3
	L5				färg C4
			L6		färg C5
		L7			färg C6
			L8		färg C7
L9					färg C8
			L10		färg C9
		L11			färg C10
			L12		färg C11
	L13				färg C12
			L14		färg C13
		L15			färg C14
			L16		färg C15
				L17	färg C16
					ingen färgändring på monitorn

vid jämförelse mellan värde och färggräns gäller att strikt mindre än medför hopp nedåt i schemat

b) gränserna och färgerna lagras vid initieringen direkt efter en absolut mod instruktion. Absolut mod instruktionen medför att GPR (general purpose registren) kan utnyttjas till annat än vektorpekare, men har samma instruktionstid.

c) SOB instruktionen (subtrahera och hoppa om ej 0) används så mycket som möjligt (detta är orsak till den något ologiska ordningen i färgkodningsprogramdelen)

För att få en punkt på monitorn skrivs först punktadressen in i videointerfacets punktadressregister. Den undre byten motsvarar horisontell och den övre byten vertikal koordinat. Därefter skrivs färgbyten in i färgbyteadressen. Färgkoden finns i de 4 lägre bitarna (0-15).

(Videointerfacet QRGB-256 beskrivs i hårdvarudelen)

Upptagning och presentation med mellanlagring

SAVDEF

SAVDEF rutinen tar upp sekundär-elektrondetektorvärden från en punkt, linje eller yta av vad som finns i svep-elektronmikroskopet (SEM) och lägger detta/dessa i en vektor. Därtill jämförs detektor max- och minvärden. Rutinen används till att ta upp SEM-bilder så att detektorvärden kan behandlas (lagras undan, filtreras m m)

anrop:CALL SAVDEF (ISTARTX, ISTARTY, IDISTX , IDISTY , NUMADP, NUMLIN, MAXV, MINV, IVECT)

alla parametrar INTEGER

ISTARTX, ISTARTY (kan vara INTEGER*4) första avlänknings-punkts X, Y kordinat (övre vänstra hörnet) (0.. 65535)

IDISTX, IDISTY	avlänkningsavstånd mellan punkter
NUMADP	antal punkter per horisontell linje
NUMLIN	antal horisontella linjer
MAXV	vid anrop det detektorvärde som skall anses vara det största (0 ..4095)
	innehåller efter anrop det detektor-värde som var störst inberäknat an-ropsvärdet (0 ..4095)
MINV	vid anrop det detektorvärde som skall anses vara det lägsta (0 ..4095)
	innehåller efter anrop det detektor-värde som var lägst inberäknat an-ropsvärdet (0 ..4095)
IVECT	vektor att lägga detektorvärdena i av minst storlek NUMADP x NUMLIN
SAVDEF	kontrollerar ej några parametrar

Följande exempel tar upp ett område med start i horisontell avlänkningspunkt 0 och vertikal avlänkningspunkt 10 med det horisontella avståndet 50 och vertikala avståndet 30. Ytan har 15 punkter per horisontell linje och 20 linjer. Efter an-rop skrivs områdets detektormaxvärde i MAXV och detektormin-värde i MINV, Dessa värden skrivs ut.

```
INTEGER IVECT (300)
MAXV = 0
MINV== 4095
CALL SAVDEF (0, 10, 50, 30, 15, 20, MAXV, MINV, IVECT)
WRITE (7, *) MAXV, MINV
END
```

DISPLY

DISPLY kodar värden från en vektor i elementordning till färgkod och lägger ut som punkter och linjer på monitorn. Presentationen sker linjevis från vänster till höger uppifrån och ned. Kodningen sker genom att vektorvärdet jämföres med gränser. Varje område mellan två gränser motsvarar en färg.

övre gräns
L 1 L 2 L 3 L 4 L 5 L 6 L 7
färg C1 färg C2 färg C3 färg C4 färg C5 färg C6 färg C7

L 8 L 9 L 10 L 11 L 12 L 13 L 14
färg C8 färg C9 färg C10 färg C11 färg C12 färg C13 färg 14
undre gräns
L 15 L 16 L 17
färg C15 färg C16

strikt mindre än medför färgen till höger.

värden över/lika med den övre gränsen L1 och strikt mindre än den undre gränsen L 17 medför att punkten ej skrivs ut på monitorn. Detta möjliggör topografiska bilder.

anrop: CALL DISPLY (ISTARTMONY , ISTARTMONY, NUMADP, NUMLIN, IVECT)

alla parametrar INTEGER

ISTARTMONX	horisontell monitorstartpunkt (0 vänster ..255 höger)
ISTARTMONY	vertikal monitorstartpunkt (0 överst .. 255 nederst)
NUMADP	antal punkter per horisontell linje
NUMLIN	antal linjer
IVECT	värdevektor av minst storlek NUMADPx NUMLIN

DISPLY kontrollerar ej några parametrar

DISLIM

Dislim initierar färgkodningsrutinen DISPLY genom att lägga in färggränsvektorns 17 element i DISPLYS programkod. Första elementet motsvarar den övre gränsen L 1 osv. Gränserna förblir desamma tills nästa DISLIM anrop.

anrop: CALL DISLIM (LIMITS)

LIMITS färggränsvektor med 17 element (lämpligen i intervallet 0 ..4095 (12 bitar)

DISCOL

DISCOL initierar färgkodningsrutinen DISPLY genom att lägga in färgkodvektorns 16 element i DISPLYs programkod. Första elementet motsvarar färgen mellan gräns L1 och L2 osv. Färgerna förblir desamma tills nästa DISCOL-anrop.

färgkoden är:

0	svart
1	mörkgrön
2	mörkblå
3	ljusblå
4	röd
5	brun
6	magenta
7	ljus magenta
8	grön
9	ljus grön
10	mörk cyan
11	ljus cyan
12	brun-grön
13	gulgrön
14	ljusröd
15	vit

anrop: CALL DISCOL (ICOLCODE)

ICOLCODE	färgkodvektor med 16 element färgkod i intervallet 0 ..15.

Följande exempel visar hur en bildyta tas upp över ett område i SEMet med start i horisontell punkt (-32767 (motsvarar punkt 32769 i intervallet (0-65535)(tvåkomplement)). Det horisontella punktavståndet är 30 och det vertikala 40. Bildområdet har 64 punkter per linje och 128 linjer.

Värdesintervallet mellan detektor max och detektor min indelas i två intervall. För att vid presentationen få en topografisk bild anpassas färggränserna till nytt intervall efter genomgång av 16 färger. Bilden presenteras med början i horisontal monitorposition 192 och vertikal monitorposition 0. (192,0)

Efter den topografiska bilden presenteras en kontrastbild i svart och vitt för att urskilja de båda topografiska delbilderna. Delbilden med lägre värden motsvarar svart, den med högre värden vitt. Bilden presenteras med början i position (0,0) (överst till vänster).

```
      PROGRAM TOP
      DIMENSION IVECT (8192), LIM (17), ICOL (16)
      DATA ICOL (0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)
      MAXV=0
      MINV=4095
      CALL SAVDEF (-32767,0,30,40,64,128,MAXV,
      MINV, IVECI)
      MXVD=(MAXV-MINV)/32
      DO IO K=1,17
10    LIM (18-K)=MINV+(K-1)+MXVD
      CALL DISLIM (LIM)
      CALL DISCOL (ICOL)
      CALL DISPLY(192,0,64,128,IVECT
      MIDDLE=LIM(1)
      LIM(1)=MAXV+1
      DO 15 L=1,16
15    LIM(18-L)=MIDDLE+(L-1)-MXVD
      CALL DISLIM (LIM)
      CALL DISPLY (192,0,64,128,IVECT)
      DO 20 M=1,16
      LIM(18-M)=MINV+(M-1)-2-MXVD
      LIM(1) =MAXV+1
      LIM(9) =MIDDLE
      DO 25 N=1,8
      ICOL(N)=15
      ICOL (N+8)=0
      CALL DISCOL (ICOL)
      CALL DISPLY (0,0, 64,128,IVECT)
      END
```

Upptagning med samtidig presentation

SEMPIC

SEMPIC rutinen tar upp sekundärelektrondetektorvärden från en punkt, linje eller yta. De upptagna detektorvärdena kodas samtidigt till färg och presenteras på videomonitorn. Upptagningen sker linjevis från vänster till höger, uppifrån och ner.

Kodningen sker genom att detektorvärdet jämföres med gränser. Gränsterna finns i en vektor med största värdet i element 1 osv. Varje område mellan två gränser motsvaras av ett färgvektorelement. Värden över/lika med L1 och strikt mindre än L17 skriv ej ut.

övre gräns

L 1	L 2	L 3	L 4	L 5	L 6	L 7
färg C1	färg C2	färg C3	färg C4	färg C 5	färg C6	färg C7

L 8	L 9	L 10	L 11	L 12	L 13	L 14
färg C8	färg C9	färg C10	färg C11	färg C12	färg C13	färg C14

undre gräns

L 15	L 16	L 17
färg C15	färg C16	

Detta möjliggör topografiska bilder

Färgkoden är:

0	svart
1	mörkgrön
2	mörkblå
3	ljusblå
4	röd
5	brun
6	magenta
7	ljus magenta
8	grön
9	ljus grön
10	mörk cyan
11	ljus cyan
12	brun-grön
13	gul-grön
14	ljusröd
15	vit

Färggränserna väljes.

anrop: CALL SEMPIC (ISTARTX, ISTARTY, DISTX, DISTY, NUMADP, NUMLIN, LIMITS, ICOLCODE, ISTARTMONX, ISTARTMONY)

alla parametrar INTEGER

ISTARTX ISTARTY (kan vara INTEGER *4) första avlänkningspunkts X, Y kordinat (övre vänstra hörnet) (0 ..65535)

IDISTX, IDISTY	avlänkningsavstånd mellan punkter
NUMADP	antal punkter per horisontell linje
NUMLIN	antal horisontella linjer
LIMITS	färggränsvektor med 17 element (lämpligen i intervallet 0 ..4095) (12 bitar)
ICOLCODE	färgkodvektor med 16 element, färgkod i intervallet 0 ..15.
ISTARTMONX	horisontell monitorstartpunkt (0 vänster ..255 höger)
ISTARTMONY	vertikal monitorstartpunkt (0 överst .. 255 nederst)

SEMPIC kontrollerar ej några parametrar.

Följande exempel visar en kontinuerlig bildupptagning och presentation över ett område.

```
    Program TAKSEM
    INTEGER C(16), LIM (17)
    DATA C/0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15/
    DO 100 K= 1,17
    A=(K-1)/16
100 LIM (18-K)=A+2000
150 CALL SEMPIC (0,0,45,256,256,LIM,C,0,0)
    GOTO 150
    END
```

3.3.2 Rutiner som till huvuddelen arbetar mot videointerfacet

Funktionsbeskrivning

Inskrivning och läsning av en punkt i videointerfacet sker genom att först adressera adressordet och därefter skriva eller läsa in de 4 färgbitarna i färgbyten (se QRGB beskrivning).

I SVISUB och VIDSUB (flödesschema finns i appendix) skrives respektive läses de 4 färgbitarna i packad form in i eller ut ur ett 16 bitars ord. Färgbitarna startar i de mest signifikanta positionerna. Bilden söks av linjevis från vänster till höger uppifrån och ner. För att snabbt kunna skriva och läsa används dubbelbuffert. Dubbelbuffert innebär att två alternerande buffertar turas om att vara buffert för det som skall skrivas eller läsas från yttre fil. Varje buffert är ett block (256 ord). Filen anges genom anropet som ett kanalnummer. .WAIT, .WRITE, .READ är systemrutiner.

MOVPIC innehåller 4 delar för flyttning åt olika riktningar. Programmet väljer själv vilken del som skall arbeta utgående från flyttningsriktningen.

SVISUB

SVISUB lagrar undan hela den på monitorn presenterade bilden. Den undanlagrade bilden kan sedan presenteras med rutinen VIDSUB.

anrop: CALL SVISUB (ICHAN, BLOSTR, MES)

alla parametrar INTEGER

ICHAN	numret på en öppnad kanal där video-innehållet (64 block) skall skrivas
BLOSTR	numret på kanalens första block där videoinnehållet kan skrivas
MES	numret på det sist skrivna blocket efter anrop utan fel, annars fel-meddelande

Felmeddelande MES

0 = försökte skriva förbi filslut
-1 = hårdvarufel
-2 = kanalen ej öppnad
-3 = -"-
-4 = hårdvarufel
-5 = fel antal ord skrivna

VIDSUB

VIDSUB visar en hel monitorbild undanlagrad med SVISUB

anrop: CALL VIDSUB (ICHAN, BLOSTR, MES)

alla parametrar INTEGER

ICHAN	numret på en öppnad kanal där video-innehållet (64 block) skall läsas
BLOSTR	numret på kanalens första block där videoinnehållet kan läsas
MES	numret på det sist lästa blocket efter anrop utan fel, annars fel-meddelande

Felmeddelande MES

0 = försökte läsa förbi filslut
-1 = hårdvarufel
-2 = kanalen ej öppnad
-3 = -"-
-4 = hårdvarufel
-5 = fel antal ord lästa

Följande program visar hur man lagrar undan, presenterar och deletar monitorbilder. Subrutinen ICHAIO öppnar, stänger och deletar en fil. Den beskrivs i avsnitt 3.3.3.

CLOSEC, IFREEC är systemrutiner.

```
            PROGRAM TRIFT
            LOGICAL+1 ITYPE(3)
     C      INTEGER+2 IBUF (256)
            DATA ITYPE/'P','V','D'/
    10      WRITE(7,+)'SAVE (1), DISPLAY (2), DELETE(3)'
            READ (5,+)IN
            IF(IN.LT.1.OR.IN.GT.3)GOTO 10
            GOTO (20,30,40), IN
    20      I=ICHAIO(ITYPE, 64
            IF(I.LT.0) GOTO 10
            CALL SVISUB (I,0,0EM)
            IC=CLOSEC(I)
            CALL IFREEC(I)
            IF (IEM.GT.0)GOTO 10
            WRITE (7,-)'SAVIDERROR!', IABS(IEM)
            GOTO 10
    30      L=0
            I=ICHAIO(ITYPE,L)
            IF(I.LT.0)GOTO 10
            CALL VIDSUB (I,0, IEM)
            IC=CLOSEC(I)
            CALL IFREEC(I)
            IF(IEM.GT.0)GOTO 10
            WRITE(7,*)'VIDISERROR !', IABS(IEM)
            GOTO 10
    40      I=ICHAIO(ITYPE,-1)
     C      DELETE A FILE
            GOTO 10
            END
```

MOVPIC

MOVPIC flyttar på monitorn presenterade bildområden åt godtyckligt håll. Rutinen används för att inte behöva ta upp helt nya bilder när bildområdet flyttas. Programmet räknar utgående från anrops parametrarna ut antalet punkter att flytta i horisontell och vertikal led.

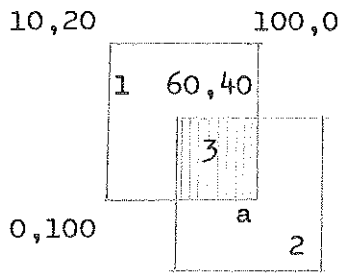
anrop: CALL MOVPIC (ICORNX, ICORNY, IXOFF, IYOFF, NDOT, NLIN)

ICORNX, ICORNY	den flyttade bildens nya horisontella och vertikala hörnkoordinat i flyttningsriktningen (om någon riktning är noll så väljes hörnet med högsta koordinatvärdet (koordinat(0,0)är överst till vänster och (255,255) är nederst till höger)
IXOFF	antal bildpunktssteg att flytta bilden i horisontell riktning (-255 ..0..255) negativt åt vänster, positivt åt höger
IYOFF	antal bildpunktssteg att flytta bilden i vertikal riktning (-255 ..0..255) negativt uppåt, positivt nedåt
NDOT	antal punkter per linje som bildytan har före flyttningen med referens i (ICORNX, ICORNY)
NLIN	antal linjer som bildytan har före flyttningen med referens i (ICORNX, ICORNY)

Om IXOFF och IYOFF = 0 samtidigt görs RETURN

i övrigt kontrollerar MOVPIC inga parametrar

Exempel:



bildytan 1 skall flyttas till 2 så att det markerade området 3 i den flyttade bilden sparas

följande anrop genomför detta

CALL MOVPIC (100, 100, 50, 20, 90, 80)

koordinaten för a är 100, 100 (ICORNX, ICORNY)
xoffset är 60-10 = 50 (IXOFF)
yoffset är 40-10 = 30 (IYOFF)
antal punkter före 100-10 = 90 (NDOT)
antal linjer före 100-20 = 80 (NLIN)

GETCRS

GETCRS lagrar undan bildinformation bakom ett kors i en i anropet given vektor och lägger därefter ut ett kors (utan centrumpunkt). Bakgrunden återställs med RECCRS.

anrop: CALL GETCRS (MONX, MONY, IWING, ICOL, ISAVEC)

Anropsparametrarna beskrivs nedan.

PUTCRS

PUTCRS lägger ut ett kors (utan centrumpunkt) på monitorn

anrop: CALL PUTCRS (MONX, MONY, IWING, KOL)

Anropsparametrarna beskrivs nedan.

RECCRS

RECCRS återställer bakgrunden bakom ett utlagt kors undanlagrat med GETCRS.

anrop: CALL RECCRS (MONX, MONY, IWING, ISAVEC)

Anropsparametrarna beskrivs nedan.

alla parametrar INTEGER

MONX, MONY	monitorkoordinal att placera korset på (0 ..255) (0,0 överst till vänster, 255,255 nederst till höger)
IWING	korsarmarnas längd (trunkeras om de hamnar "utanför" monitorn
ICOL	korsets färg (0-svart, 1-mörkgrön, 2-mörkblå, 3-ljusblå, 4-röd, 5-brun, 6-magenta, 7-ljus magenta, 8-grön, 9-ljusgrön, 10-mörk cyan, 11-ljus cyan, 12-brun/grön, 13-gulgrön, 14-ljusröd, 15-vit)
ISAVEC	vektor som lagrar bakgrundsinformation av minst storlek IWING x 4

3.3.3 Kommunikations- och hjälprutiner

Rutiner som används tillsammans med joystick.

Joysticken finns beskriven i hårdvarudelen.

ADIN

ADIN läser av en kanal i ad-omvandlaren ADV11-A. Används till att läsa av joystickutslag.

anrop:CALL ADIN (ICMAN, IRESULT)

ICHAN	ad-omvandlarkanal
IRESULT	innehåller efter anrop avläst värde (i intervallet 0 ..4095) (12 bitar)

INIJOY

INIJOY låter operatören föra joysticken till 0, max, min utslag för att dessa skall kunna användas som parametrar i LIMITS, CRSLIM. Värdena skrivs ut på terminal. Återhopp sker genom terminaltangent.

anrop: CALL INIJOY (MZN6, MZN7)

Efter anrop innehåller treelementvektorerna joysticks-utslagsvärdena från ad-kanal 6 och ad-kanal 7.

LIMITS, CRSLIM

CRSLIM är identisk med LIMITS

LIMITS delar utslagsintervallet för joysticken i fem intervall. För de fyra yttersta intervallen skapas två normeringskonstanter. Intervallgränserna och skalkonstanter utnyttjas av JYSTEP för att bestämma hur joystick-utslaget skall tolkas.

anrop: CALL LIMITS (MZN, L, CDIST)

MZN	treelementvektor innehållande MZN(1)max, MZN(2) noll och MZN(3) min från tex INIJOY)
L	6-element gränsvektor av intervallet mellan joystickutslagets max och min L(1) är max och L(6) min
CDIST	två-elementvektor för normerings-konstanter CDIST(1)=yttre intervallen CDIST(2)=inre intervallen runt noll

JYSTEP

JYSTEP tolkar joystickens utslag och ger utgående från utslaget, steglängd (i horisontell och vertikal riktning) en "väntetid". Joystickens utslag avlässs med subrutinen ADIN (vertikalt, horisontalt). Utslaget passar in i ett av fem intervall. Intervallen bestäms av tolkningsgränsparametrarna. Dessa skapas med LIMITS, CRSLIM. I ytterintervallen skalar joystickens utslag den maximala steglängden. Väntetiden blir en i anropet given konstant. I intervallen innanför ytterintervallen (på båda sidor) är steglängden ett och joystickens utslag reglerar väntetiden. I mitt intervallet är steglängden 0 och väntetiden konstant.

anrop: CALL JYSTEP (ISLX, ISLY, ITIME, MAXTIM, MNX, ISTP, LIM6, LIM7, CDIST6, CDIST7, ITX, ITY)

ISLX	steg x
ISLY	steg y
ITIME	väntetid
MAXTIM	längsta väntetid (vid enkelsteg)
MNX	tidsdifferens mellan största och minsta väntetid (vid enkelsteg)
ISTP	fast väntetid om ej enkelsteg
LIM6, LIM7	6 elementvektorer med tolkningsgränser (övre gräns i vektorposition ett)
CDIST6, CDIST7	2 elementvektorer med normeringskonstanter (det yttre intervallets i position 1)
ITX	längsta horisontella steglängd
ITY	längsta vertikala steglängd

Hjälprutiner

ICHAIO

ICHAIO skapar, öppnar eller deletar en fil av given typ (och längd). Filen bestäms av device(DYO:,DY1:)+filnamn (6 bokstäver max).Tidigare existerande fil deletas med /DOLD kommandot. /DOLD används också vid deletning av filer.

anrop: I = ICHAIO (ITYPE, ISIZE)

I är kanalnummer på den öppnade filen annars -1

ITYPE(3) LOGICAL*1	anger filtyp
ISIZE	önskad filstorlek om större än 0 öppning av fil om lika med 0 deletning av fil om mindre än 0

GETPAR

GETPAR ändrar elementvärdet i en vektor. De nya värdena skrivs in genom terminal.

anrop: CALL GETPAR (IVEC, IELVEC)

IVEC	vektornamn
IELVEC	antal vektorelement i IVEC

GLOCOR konverterar och skriver ut en videomonitorpunkts avlänkningskoordinat på terminalen

anrop: CALL GLOCOR (IDP, ISX, ISY, IDX, IDY, ISCRX, ISCRY)

IDP	start standardposition för bilden på monitorn (11..44)
ISX, ISY INTEGER*4	den upptagna bildens startposition
IDX, IDY	horisontalt och vertikalt avstånd
ISCRX, ISCRY	monitorpunkt att konvertera

3.3.4 Överordnade program

Funktionsbeskrivning

TAKEUP tar upp en bild med SAVDEF. Bilden byggs upp av vertikala kolonner 32 bildpunkter breda med start uppe till vänster. De erhållna detektorvärdena kodas och presenteras eventuellt med DISPLY. Därefter skrivs de ut på en öppnad kanal med systemrutinerna IWAIT och IWRITE.

EDDIT öppnar en fil och läser med systemrutinerna IWAIT, IREAD, de med TAKEUP undanlagrade detektorvärdena som skall presenteras. Dessa kodas och presenteras av DISPLY.

EDON räknar med utgångspunkt från bildens startpunkt ut antalet möjliga bildsteg att flytta i olika riktningar. (med INTEGERx4 systemrutiner). Genom att kontrollera och anpassa dessa kontrolleras att bilden inte "går runt" (bild från båda sidor). Steglängden att flytta bilden fås från JYSTEP vars utslag tolkas enligt anropsparametrarna.
För att kunna återvända då rutinen går i "loop" används systemrutinen ITTNR.

EDOST kombinerar flyttning av tidigare upptagna bilddelar med nyupptagning. Nya bilder och bilddelar tas upp med macroprogrammet SEMPIC. Flyttning av en visad bild sker med MOVPIC. Bilden flyttas för att snabbare finna det sökta området.
Bildens uppbyggnadsmönster då delar flyttas är:

1. flytta det bildsegment som skall behållas till bildhörnet
2. den VERTIKALA DELEN av det utrymme som skall fyllas ut med ny bildinformation görs så hög som möjligt
3. om den gamla bilden skiftats ut till vänster (upp eller ner) tas först den VERTIKALA DELEN upp. Därefter den återstående horisontella.
4. om den gamla bilden skiftats ut till höger (upp eller ner) tas först den norisontella delen upp därefter den höga VERTIKALA DELEN.

Vissa gemensamma anropsparametrar

IDF — presentations och upptagningsformat (11 .. 44). Tiotalet anger storlek i antal fjärdedelar av full bildyta i vertikal led, entalet anger antalet fjärdedelar i horisontell led

IDP — startposition att presentera bilden på fjärdedelar av fullbild (11 .. 44) (positionen som matriselement, 11 överst till vänster - 44 nederst till höger)

LL — 17-elements färgvektor (övre gräns i vektorposition 1 o s v)

ICC — 16-elementsfärgkodvektor (färgkod för intervallen LL (1) .. LL (2) i CC (1) o s v)
(0-svart, 1-mörkgrön, 2-mörkblå 3-ljusblå, 4-röd, 5-brun, 6-magenta, 7-ljus magenta, 8-grön, 9-ljusgrön, 10- mörk cyan, 11-ljus cyan, 12-brun/grön, 13-gulgrön, 14-ljusröd, 15-vit)

Värden över eller lika med gränsen medför nästa intervall med lägre gränsvektorelement. Värden över eller lika med den övre gränsen LL(1) och strikt mindre än den undre gränsen LL(17) medför ej någon färgändring)

IEM — felmeddelande

1. bildformatfel
2. positionsfel på monitorn
3. format och position överensstämmer ej
4. avlänkningssteg utanför intervallet (1 .. 256)
5. delar av de önskade avlänkningspositionerna existerar ej
6. felaktig joystickmod
7.
8. försök att skriva förbi filslut
9. hårdvarufel
10. kanalen ej öppnad
11. den öppnade kanalen var skyddad
12. försök att läsa förbi filslut
13. storleken på den öppnade filen felaktig
14. format på upptagen bild felaktig
15. position på upptagen bild felaktig
16. format och position på upptagen bild överensstämmer ej

TAKEUP

TAKEUP sveper en SEM-bild och lagrar undan motsvarande detektorvärden på en fil (diskett). Filen öppnas med programmet ICHAIO fråntermínal. De upptagna detektorvärdena kan färgkodas och presenteras i samband med upptagningen. Efter upptagning av varje bildkolonn (en åttondels fullbild bred) skrivs dittills varande detektormax och detektormin ut på terminal. De sist utskrivna värdena gäller hela upptagningen. TAKEUP använder rutinerna SAVDEF, DISPLY, ICHAIO och några systemrutiner. Filtypen blir .SAD.

anrop: TAKEUP (IDF, IDP, ISX, ISY, IDX, IDY, LL, ICC, IEM, ICODE)

IDF	upptagningspresentation - format (11 ..44, se avsnittsinledning)
IDP	presentationsplats (11 ..44, se avsnittsinledning; 0 om ingen presentation)
ISX, ISY (INTEGER x 4)	startposition horisontellt, vertikalt avlänkningskoordinat i bildens övre vänstra hörn
IDX, IDY	avlänkningsavstånd horisontellt, vertikalt (1 .. 256)
LL	färggränsvektor (se avsnittsinledning)
ICC	färgkodvektor (se avsnittsinledning)
IEM	felmeddelande (1-5, 8-11)
ICODE	används ej (dummyparameter)

EDDIS

EDDIS presenterar bilder undanlagrade med TAKEUP. Bilden kan presenteras i ett standardformat mindre eller lika med upptagningsformatet. Man kan då välja vilken del av bilden man vill presentera. Filen med värden att presentera öppnas med ICHAIO från terminal (typ .SAD). Eventuella felmeddelanden skrivs ut på denna. EDDIS använder rutinerna DISPLY, ICHAIO, och några systemrutiner

anrop: CALL EDDIS (IDF, IDP, IEM, ITFO, INSTD, LL, ICC)

IDF	presentationsformat (11 ..44, se avsnittsinledning)
IDP	presentationsplats (11 .44. se avsnittsinledning)
IEM	felmeddelande (1-3, 9-16, se avsnittsinledning)
ITFO	upptagningsformat (på öppnad fil) (11..44, jämför IDF i avsnittsinledningen)
INSTD	startposition för önskat bildområde på den öppnade filens bild (11 ..44, jämför IDP i avsnittsinledningen, 1100, 1150, 1105, 1155 ..3355) om man vill få bildytan att börja ett halvt standardformat (en åttondels bild) ytterligare nedåt eller /och till höger lägger man till 50, 05, 55 (tiotalssiffran nedåt, entalssiffran till höger)
LL	färggränsvektor (se avsnittsinledning)
ICC	färgkodsvektor (se avsnittsinledning)

EDON

EDON tar upp en enstaka SEM-bild med samtidig presentation (on-line) eller tar upp nya SEM-bilder kontinuerligt. Bildområdet kan definieras med anropsparametrar eller genom att flyttas med joystick. Återhopp till anropande program görs med godtycklig tangentnedtryckning på terminal (ej enkelbild). EDON använder EDOST, JYSTEP och systemrutiner.

anrop: CALL EDON (IDF, IDP, ISX, ISY, IDX, IDY,ISLX, ISLY, LL, ICC, JOYMOD, IWPIC, IEM, TXT, LIM6, LIM7, CDIST6, CDIST7,MAXTIM, MNX, ISTP)

IDF	upptagningspresentation - format (11 ..44, se avsnittsinledning)
IDP	presentationsplats (11 ..44, se avsnittsinledning, 0 om ingen presentation)
ISX, ISY (INTEGER x 4)	startposition horisontellt, vertikalt avlänkningskoordinat i bildens övre vänstra hörn
IDX, IDY	avlänkningsavstånd horisontellt, vertikalt (1 .. 256)
ISLX	antal steg att flytta bildområdet horisontellt. Trunkeras om man kommer utanför det definierade bildupptagningsområdet(kan ändras).
ISLY	antal steg att flytta bildområdet vertikalt.(Trunkeras liksom ovan).
LL	färggränsvektor (se avsnittsinledning)
ICC	färgkodvektor (se avsnittsinledning)
JOYMOD	anger hur bilden skall tas upp (1=enkelbild, 2=kontinuerligt bildupptagning utan joystick, 3=ta upp definierad bild när joystick ej är i centrum, 4=låt joysticken definiera nya bilder genom att flytta redan visade bildområden och ta upp de nya, 5= som 4 men ingen flyttning av gamla bilder (nyupptagning), 6=bildområden tas upp flyttade av joystick när joystickutslaget är skilt från centrum.
IWPIC	väntetid mellan bildsvep i ticks (1 tick = 20 ms)

IEM	felmeddelande (1-6)
TXT	används ej (dummyparameter)

Parametrar som används av JYSTEP (1 tick=20ms)

LIM6, LIM7	tolkningsgränser för joystick (jämför LIMITS)
CDIST6, CDIST7	normeringskonstanter för joystick
MAXTIM	maximal väntetid i ticks mellan enstegsflyttningar
MNX	tidsskillnad mellan maximal och minimal väntetid i ticks vid enstegsflyttningar
ISTP	väntetid vid stegning i ticks

EDOST

EDOST tar upp och presenterar en helt ny SEM-bild eller kombinerar flyttning av en på videomonitorn tidigare presenterad bild med upptagning av nya SEM-bildsområden. Flyttningsrutinen används till att snabbt definiera och visa en SEM-bild över ett nytt område. Flyttning tar kortare tid än vad en ny SEM-bildupptagning tar. EDOST använder sig av MOVPIC och SEMPIC

anrop: CALL EDOST (ISX, ISY, IDX, IDY, ISLEDX, ISLEDY, ITX, ITY, IMX, IMY, LL, ICC)

ISX, ISY(INTEGER x 4)	startpunkt (horisontell, vertikal avlänkningskoordinat i bildens övre vänstra hörn) (vid flyttning; koordinat före flyttningen. Innehåller efter anrop den nya koordinaten.)
IDX, IDY	avlänkningsavstånd mellan punkter
ISLEDX	antal bildpunktssteg att flytta bildområdet (horisontellt) (negativt om bildområdet skall utökas till vänster, positivt till höger) 0 om helt ny bild skall tas upp
ISLEDY	antal bildpunktssteg att flytta bildområdet (vertikalt) (negativt för att öka bildområdet uppåt, positivt för att öka bildområdet nedåt) 0 om helt ny bild tas upp
ITX	den presenterade bildytans antal punkter horisontellt
ITY	den presenterade bildytans antal punkter vertikalt
IMX	horisontell monitorstartpunkt (0 vänster ..255 höger)
IMY	vertikal monitorstartpunkt (0 överst .. 255 nederst)
LL	färggränsvektor med 17 element (lämpligen i intervallet 0 ..4095) (12 bitar)
ICC	färgkodvektor med 16 element, färgkod i intervallet 0 ..15.

CRSJOY

CRSJOY lägger ut ett kors på videomonitorn och låter detta flyttas med joysticken. Om korset inte rör sig skrivs monitorkoordinaten för korset ut på terminalen.Operatören kan ändra färgen på korset genom att trycka ned en terminaltangent (0-9, A-F, 0-svart, 1-mörkgrön, 2-mörkblå, 3-ljusblå, 4-röd, 5-brun, 6-magata, 7-ljus magenta, 8-grön, 9-ljusgrön, A -mörk cyan, B -ljus cyan, C -brun/grön, D -gulgrön, E -ljusröd, F -vit) Korset fås att börja blinka med terminaltangenten T och sluta blinka med tangent N. H-tangenten låter korset vara kvar på videomonitorn vid återhopp från rutinen. En annan tangent än 0-9, A-F, N, T, H orsakar återhopp från subrutinen med återläggning av korsets bakgrundsinformation.

CRSJOY använder JYSTEP, GETCRS, PUTCRS, RECCRS och systemrutiner

anrop: CALL CRSJOY (IDF, IDP, ICRSX, ICRSY, IWING, ICOL, IVEC, IDV, IWPIC, LIM6, LIM 7, CDIST6, CDIST7, ITWINK, IEM)

IDF	format (11 ..44) jämför avsnittsinledning) anger storleken på området inom vilket korset får röra sig
IDP	position (11 ..44, jämför avsnittsinledning) anger området IDF:s läge
ICRSX	horisontell videomonitorkoordinat (0 till vänster, 255 till höger) som korset är placerat i (IDF och IDP måste ange området)
ICRSY	vertikal videomonitorkoordinat (0 överst, 255 nederst) som korset skall eller är placerat i (IDF, IDP måste ange området)
IWING	vinglängder på korset
ICOL	utgångsfärg på korset
IVEC	vektor att lagra bakgrundsinformation i av minst storlek 4 x IWING
IDV	DIMENSION av IVEC
IWPIC	anger aktiv och passiv tid då korset blinkar i ticks (20 ms)
LIM6, LIM7	tolkningsgränser för JYSTEP (jämför CRSLIM)
CDIST6, CDIST7	normeringskonstanter för JYSTEP

ITWINK (LOGICAL)	anger om korset skall blinka eller ej
IEM	felmeddelande (1 - 3)

Överordnade testrutiner

TKESUB

TKESUB handhar operatörs kommunikation för EDDIS, TAKEUP. Inparametrar för rutinen är färgtabell och gränstabell. Bildparametrarna tas upp från terminalen. Dessa används för upptagning och presentation. Man kan också ändra gränser i gränstabellen linjärt. Filer kan deletas med ICHAIO. (se programlistning i appendix)

TRISUB

TRISUB skapar, öppnar eller deletar en .PVD-fil (packad video) med ICHAIO. Den på videomonitorn visade bilden lagras undan med macrorutinen SVISUB. Den undanlagrade bilden presenteras med VIDSUB. (se programlistning i appendix).

EDOMAN (huvudtestprogram)

EDOMAN provar de olika rutinerna tillsammans. Användar-beskrivning finns i appendix.

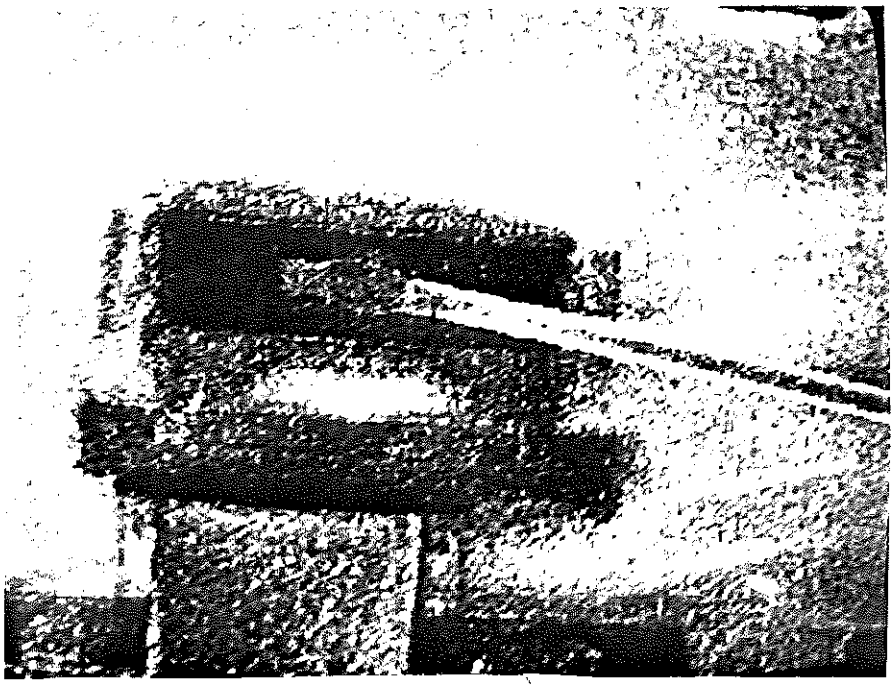
4 KÖRNING OCH TEST AV PROGRAM

4.1 Körning av programmet

Först initierades joysticken tillsammans med övriga parametrar (skala, område, presentationsplats, färgkod, färggränsparametrar, hur joysticken skulle arbeta; beskrivs i appendix).

För att finna det intressanta bildområdet flyttades bilden med joysticken. Intressanta bildpunkters koordinater hittades med hjälp av korset. Den intressanta bilden lagrades sedan undan på diskett. Genom att ändra skalan och ta upp en ny bild med undanlagring av bilddata fick man information om sampelvärdens max, min. Man kunde därefter justera färgskala och kodningsrutin för bästa presentation genom att visa den undanlagda bilden på nytt. Nedan visas en bild av en bondingyta upptagen av systemet.

Bildupptagningsprogrammet höll de tider som var ansatsen (on-linebild på ca 4 sek och off-linebild på under 1 minut)



4.2 Programmets begränsningar

Bildupptagningsparametrarna lagras ej undan på massminne.
Effektiv skalningsrutin och färgbestämningsrutin saknas.
I övrigt finns mer att önska av operatörskommunikationen.

Referenslista

1
E Wolfgang, R Lindner, P Fazekas, H P Feuerbaum
Electron-beam testing of integrated circuits
IEEE J Solid-state circuits, Vol sc-14, p 471-481, april 1979

2
Lennart Haggård
Elektronstråle ersätter mekanisk sond?
Elteknik med aktuell elektronik, nummer 16, sid 10-15, okt 1982

3
Mats Brisegård
Konstruktion, tillverkning och test av en spänningskänslig
detektor till ett svepelektronmikroskop
Examensarbete vid Institutionen för Tillämpad Elektronik, KTH 1981

4
Microcomputer processor handbook
Digital Equipment Corporation, 1979

5
Microcomputer interfaces handbook
Digital Equipment Corporation, 1980

6
Systemmanualer för PDP11
RT-11 V 4,0
Volume 3 A Programmers Reference
Volume 3 B Software Support SSM

UPPBYGGNAD AV EN DIGITAL BILDHANTERING I ETT SVEPELEKTRONMIKROSKOP

Appendixdel

Kaj Andersson

oktober 1982

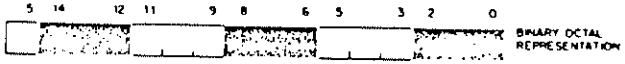
APPENDIXFÖRTECKNING

digital pdp-11

PROGRAMMING CARD

FOR FAMILY OF PDP-11 COMPUTERS

WORD FORMAT:



MODE	R

Mode	Name	Symbolic	Description
0	register	R	(R) is operand [ex. R2=%2]
1	register deferred	(R)	(R) is address
2	auto-increment	(R)+	(R) is adrs; (R) + (1 or 2)
3	auto-incr deferred	@(R)+	(R) is adrs of adrs; (R) + 2
4	auto-decrement	-(R)	(R) - (1 or 2); (R) is adrs
5	auto-decr deferred	@-(R)	(R) - 2; (R) is adrs of adrs
6	index	X(R)	(R) + X is adrs
7	index deferred	@X(R)	(R) + X is adrs of adrs

PROGRAM COUNTER ADDRESSING: Reg = 7

MODE	7

2	immediate	#n	operand n follows instr
3	absolute	@#A	address A follows instr
6	relative	A	instr adrs + 4 + X is adrs
7	relative deferred	@A	instr adrs + 4 + X is adrs of adrs

LEGEND:

Op Codes

- . = 0 for word/1 for byte
- SS = source field (6 bits)
- DD = destination field (6 bits)
- R = gen register (3 bits), 0 to 7
- XXX = offset (8 bits), +127 to -128
- N = number (3 bits)
- NN = number (6 bits)

Operations

- ( ) = contents of
- s = contents of source
- d = contents of destination
- r = contents of register
- ← = becomes
- X = relative address
- % = register definition

Boolean

- ∧ = AND
- ∨ = Inclusive OR
- ⊻ = exclusive OR
- ~ = NOT

Condition Codes

- * = conditionally set/cleared
- - = not affected
- 0 = cleared
- 1 = set

NOTE:

- ▲ = Applies to the 11/35, 11/40, 11/45 & 11/70 computers
- ● = Applies to the 11/45 & 11/70 computers

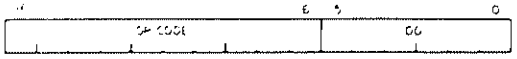
digital equipment corporation

MAYNARD, MASSACHUSETTS July 1975

7-BIT ASCII CODE:

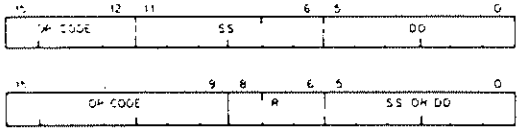
Octal Code	Char	Octal Code	Char	Octal Code	Char	Octal Code	Char
000	NUL	040	SP	100	@	140	`
001	SOH	041	!	101	A	141	a
002	STX	042	"	102	B	142	b
003	ETX	043	#	103	C	143	c
004	EOT	044	$	104	D	144	d
005	ENQ	045	%	105	E	145	e
006	ACK	046	&	106	F	146	f
007	BEL	047	'	107	G	147	g
010	BS	050	(	110	H	150	h
011	HT	051	)	111	I	151	i
012	LF	052	*	112	J	152	j
013	VT	053	+	113	K	153	k
014	FF	054	,	114	L	154	l
015	CR	055	-	115	M	155	m
016	SO	056	.	116	N	156	n
017	SI	057	/	117	O	157	o
020	DLE	060	0	120	P	160	p
021	DC1	061	1	121	Q	161	q
022	DC2	062	2	122	R	162	r
023	DC3	063	3	123	S	163	s
024	DC4	064	4	124	T	164	t
025	NAK	065	5	125	U	165	u
026	SYN	066	6	126	V	166	v
027	ETB	067	7	127	W	167	w
030	CAN	070	8	130	X	170	x
031	EM	071	9	131	Y	171	y
032	SUB	072	:	132	Z	172	z
033	ESC	073	;	133	[	173	{
034	FS	074	<	134	\	174	|
035	GS	075	=	135	]	175	}
036	RS	076	>	136	^	176	~
037	US	077	?	137	_	177	DEL

SINGLE OPERAND: OPR dst



Mnemonic	Op Code	Instruction	dst Result	N	Z	V	C
General							
CLR(B)	■ 0500DD	clear	0	0	1	0	0
COM(B)	■ 0510DD	complement (1's)	~d	*	*	0	1
INC(B)	■ 0520DD	increment	d + 1	*	*	*	–
DEC(B)	■ 0530DD	decrement	d − 1	*	*	*	–
NEG(B)	■ 0540DD	negate (2's compl)	−d	*	*	*	*
TST(B)	■ 0570DD	test	d	*	*	0	0
Rotate & Shift							
ROR(B)	■ 0600DD	rotate right	→C, d	*	*	*	*
ROL(B)	■ 0610DD	rotate left	C, d ←	*	*	*	*
ASR(B)	■ 0620DD	arith shift right	d/2	*	*	*	*
ASL(B)	■ 0630DD	arith shift left	2d	*	*	*	*
SWAB	0003DD	swap bytes		*	*	*	0
Multiple Precision							
ADC(B)	■ 0550DD	add carry	d + C	*	*	*	*
SBC(B)	■ 0560DD	subtract carry	d − C	*	*	*	*
▲SXT	0067DD	sign extend	0 or −1	–	*	0	–

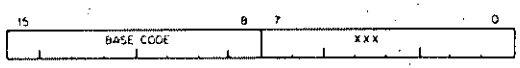
DOUBLE OPERAND: OPR src, dst OPR src, R or OPR R, dst



Mnemonic	Op Code	Instruction	Operation	N	Z	V	C
General							
MOV(B)	■ 1SSDD	move	d ← s	*	*	0	–
CMP(B)	■ 2SSDD	compare	s − d	*	*	*	*
ADD	06SSDD	add	d ← s + d	*	*	*	*
SUB	16SSDD	subtract	d ← d − s	*	*	*	*
Logical							
BIT(B)	■ 3SSDD	bit test (AND)	s ∧ d	*	*	0	–
BIC(B)	■ 4SSDD	bit clear	d ← (~s) ∧ d	*	*	0	–
BIS(B)	■ 5SSDD	bit set (OR)	d ← s ∨ d	*	*	0	–
▲Register							
MUL	070RSS	multiply	r ← r x s	*	*	0	*
DIV	071RSS	divide	r ← r/s	*	*	*	*
ASH	072RSS	shift arithmetically		*	*	*	*
ASHC	073RSS	arith shift combined		*	*	*	*
XOR	074RDD	exclusive OR	d ← r ⊻ d	*	*	0	–

BRANCH: B – – location

If condition is satisfied:
Branch to location,
New PC ← Updated PC + (2 x offset)
adrs of br instr + 2



Op Code = Base Code + XXX

Mnemonic	Base Code	Instruction	Branch Condition	
Branches				
BR	000400	branch (unconditional)	(always)	
BNE	001000	br if not equal (to 0)	≠ 0	Z = 0
BEQ	001400	br if equal (to 0)	= 0	Z = 1
BPL	100000	branch if plus	+	N = 0
BMI	100400	branch if minus	−	N = 1
BVC	102000	br if overflow is clear		V = 0
BVS	102400	br if overflow is set		V = 1
BCC	103000	br if carry is clear		C = 0
BCS	103400	br if carry is set		C = 1
Signed Conditional Branches				
BGE	002000	br if greater or eq (to 0)	≥0	N ⊻ V = 0
BLT	002400	br if less than (0)	<0	N ⊻ V = 1
BGT	003000	br if greater than (0)	>0	Z v (N ⊻ V) = 0
BLE	003400	br if less or equal (to 0)	≤0	Z v (N ⊻ V) = 1
Unsigned Conditional Branches				
BHI	101000	branch if higher	>	C v Z = 0
BLOS	101400	branch if lower or same	≤	C v Z = 1
BHIS	103000	branch if higher or same	≥	C = 0
BLO	103400	branch if lower	<	C = 1

JUMP & SUBROUTINE:

Mnemonic	Op Code	Instruction	Notes
JMP	0001DD	jump	PC ← dst
JSR	004RDD	jump to subroutine	use same R
RTS	00020R	return from subroutine	use same R
▲MARK	0064NN	mark	aid in subr return
▲SOB	077RNN	subtract 1 & br (if ≠ 0)	(R) − 1, then if (R) ≠ 0: PC ← Updated PC − (2 x NN)

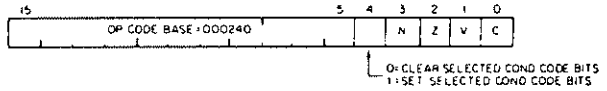
TRAP & INTERRUPT:

Mnemonic	Op Code	Instruction	Notes
EMT	104000 to 104377	emulator trap (not for general use)	PC at 30, PS at 32
TRAP	104400 to 104777	trap	PC at 34, PS at 36
BPT	000003	breakpoint trap	PC at 14, PS at 16
IOT	000004	input/output trap	PC at 20, PS at 22
RTI	000002	return from interrupt	
▲RTT	000006	return from interrupt	inhibit T bit trap

MISCELLANEOUS:

Mnemonic	Op Code	Instruction
HALT	000000	halt
WAIT	000001	wait for interrupt
RESET	000005	reset external bus
NOP	000240	(no operation)
●SPL	00023N	set priority level (to N)
▲MFPI	0065SS	move from previous instr space
▲MTPI	0066DD	move to previous instr space
●MFPD	1065SS	move from previous data space
●MTPD	1066DD	move to previous data space

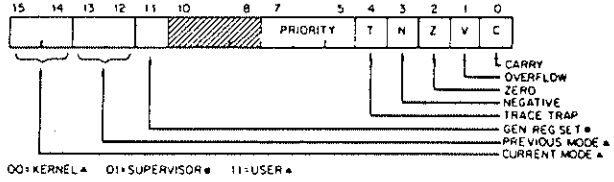
CONDITION CODE OPERATORS:



Mnemonic	Op Code	Instruction	N	Z	V	C
CLC	000241	clear C	–	–	–	0
CLV	000242	clear V	–	–	0	–
CLZ	000244	clear Z	–	0	–	–
CLN	000250	clear N	0	–	–	–
CCC	000257	clear all cc bits	0	0	0	0
SEC	000261	set C	–	–	–	1
SEV	000262	set V	–	–	1	–
SEZ	000264	set Z	–	1	–	–
SEN	000270	set N	1	–	–	–
SCC	000277	set all cc bits	1	1	1	1

PROCESSOR REGISTER ADDRESSES:

Processor Status Word
PS - 777 776



▲Stack Limit Register — 777 774

●Program Interrupt Request — 777 772

General Registers (console use only)
R0 — 777 700
R1 — 777 701
R2 — 777 702
R3 — 777 703
R4 — 777 704
R5 — 777 705
R6 — 777 706
R7 — 777 707
(not for 11/45)

Console Switches & Display Register — 777 570

RXV21 FLOPPY DISK OPTION

GENERAL

The RXV21 floppy disk option is a random access mass memory device that stores data in fixed-length blocks on a preformatted, flexible diskette. Each diskette can store and retrieve up to 512K 8-bit bytes of data. The RXV21 system is rack-mountable and consists of an interface module, an interface cable, and either a single or dual RX02 floppy disk drive.

FEATURES

- Compact disk system
- Stores/retrieves 512K 8-bit bytes of data
- Rack mountable
- Available with either single or dual disk drive
- Available for 115 or 230 Vac, 50 or 60 Hz
- Can be converted (50 Hz version) for 105, 115, 220 or 240 Vac operation
- Direct Memory Access data tranfer
- Industry-compatible mode under software selection

SPECIFICATIONS

Module	
Identification	M8029
Size	Double
Power	+5V ±5% at 1.8A typically
Bus Loads	
AC	3
DC	1
Drive	
Identification	RX02
Size	46.3 cm w × 28.7 cm h × 53.3 cm d (19 in. w × 10.5 in. h × 21 in. d)
Recommended Service Clearance (front and rear)	55 cm (22 in.)

AC Power	4A at 115 Vac; 2A at 230 Vac (dual drive)
Cable Included	BC05L-15 (15 ft.)
Drive Performance	
Capacity (8-bit bytes)	
Per diskette	512,512 bytes
Per track	6,656 bytes
Per sector	256 bytes
Data Transfer Rate	
Diskette to controller buffer	2 μsec/data bit (500K bits/sec)
Buffer to RXV21 interface	1.2 μsec/bit (500K bits/sec)
RXV21 interface to LSI-11 I/O bus	23 μsec/16-bit word
Track-to-track move	6 msec/track maximum
Head settle time	25 msec maximum
Rotational speed	360 rpm ±2.5%; 166 msec/rev nominal
Recording surfaces per disk	1
Tracks per disk	77 (0-76) or ($0\text{-}114_8$)
Sectors per track	26 (1-26) or ($0\text{-}32_8$)
Sectors per disk	2002
Recording technique	Double frequency (FM) or modified (MFM)
Bit density	3200 bpi (FM); 6400 bpi (modified MFM)
Track density	48 tracks/in.
Average access	262 msec, computed as follows:

Seek	Settle	Rotate	Total

(77 tks/3) × 6 msec + 25 msec + (166 msec/2) = 262 msec

DRV11-J

(jfr ref 5)

CHAPTER 4
PROGRAMMING

4.1 GENERAL

The software control of the DRV11-J is performed by eight addressable bus registers. The bus registers are divided into two groups of four control status registers (CSRA, B, C, and D) and four data buffer registers (DBRA, B, C, and D). These registers are assigned contiguous addresses starting with the jumper selectable base device address of CSRA. These registers can be individually read or loaded by the program using the assigned addresses. The DIGITAL software requires that device addresses be located in the I/O page of memory (760000_8 through 777776_8). The DRV11-J is assigned a range of addresses from 760000_8 through 777760_8 which falls within the I/O page.

4.2 CONTROL STATUS REGISTERS

The control status registers (CSRA, CSRB, CSRC and CSRD) are read/write byte addressable registers with bit assignments as shown in Figures 4-1, 4-2, 4-3, and 4-4. Descriptions of the control status register bits are contained in Tables 4-1, 4-2, 4-3, and 4-4.

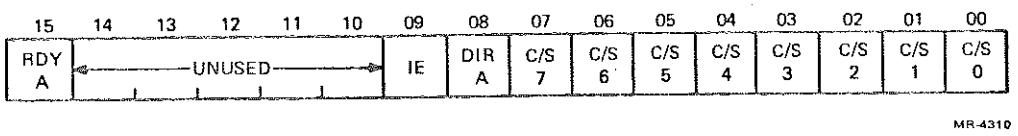


Figure 4-1 CSRA Bit Assignments

Table 4-1 CSRA Bit Function and Description

Bit	Name	Function	Description
07:00	C/S7-C/S0	Read/Write	These bits are used in conjunction with CSRD bits <07:00> to program interrupt control group 1. They contain status information when read and command words when written. Unaffected by BINIT. (See Paragraphs 5.8.5 and 5.8.6 for status and command definitions.)
08	DIR A	Read/Write	DIRECTION A. Used for controlling DBRA. This bit, in conjunction with the USER RDY signal, controls the direction of data transfer. When the DIR bit is cleared, the DRV11J RDY output signal is asserted and the DRV11J is the input device. When set and the USER RDY signal is asserted, the DRV11J is the output device. The negation of either DIR or USER RDY will cause the DRV11J outputs to remain in their high impedance state. Cleared by BINIT.
09	IE	Read/Write	Interrupt Enable. Enables the DRV11J to generate processor interrupts when set. Used to enable both group 1 and group 2 interrupts. Cleared by BINIT.
14:10			Unused. Read as zeros.
15	RDY A	Read Only	USER READY A. Used for controlling DBRA. When read, yields the state of the USER RDY signal. A zero equals negated (0) and a one equals asserted (1). It is used in conjunction with the DIR bit to enable DRV11J output operations. The user device asserts this signal when it desires the DRV11J to output data. Unaffected by BINIT.

Speciell for CSRA

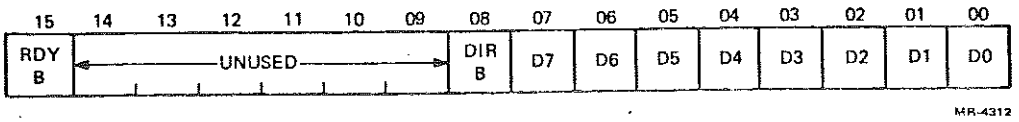


Figure 4-2 CSRB Bit Assignments

Table 4-2 CSRB Bit Function and Description

Bit	Name	Function	Description
07:00	D7-DO	Read/Write	These bits are used in conjunction with CSRA Bits <07:00> to program interrupt control group 1. They contain information selected by the command word loaded through CSRA. The registers available are the IRR, ISR, ACR, IMR and the vector address memory. Unaffected by BINIT. (See Paragraph 5.8 for a detailed description of the registers and their functions.)
08	DIR B	Read/Write	DIRECTION B. Used for controlling DBRB. This bit, in conjunction with the USER RDY signal, controls the direction of data transfer. When the DIR bit is cleared, the DRV11J RDY output signal is asserted and the DRV11J is the input device. When set and the USER RDY signal is asserted, the DRV11J is the output device. The negation of either DIR or USER RDY will cause the DRV11J outputs to remain in their high impedance state. Cleared by BINIT.
14:09			Unused. Read as zeros.
15	RDY B	Read Only	USER READY B. Used for controlling DBRB. When read, yields the state of the USER RDY signal. A zero equals negated (0) and a one equals asserted (1). It is used in conjunction with the DIR bit to enable DRV11J output operations. The user device asserts this signal when it desires the DRV11J to output data. Unaffected by BINIT.

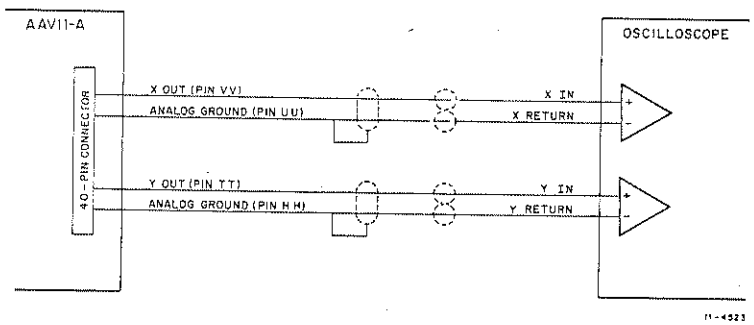


Figure 5 J1 Connector Pin Assignments

Table 5 AAV11-A Digital-to-Analog Conversions*

Input Code (octal)	Bipolar ±2.56 V (volts)	Bipolar ±5.12 V (volts)	Bipolar ±10.24 V (volts)	Unipolar 0 V to +5.12 V (volts)	Unipolar 0 V to +10.24 V (volts)
0000	−2.56	−5.12	−10.24	+0.0	+0.0
0001	−2.55875	−5.1175	−10.235	+0.00125	+0.025
3777**	−0.00125	−0.0025	−0.005	+2.55875	+5.1175
4000	0.0	0.0	0.0	+2.56	+5.12
4001	+0.00125	+0.0025	+0.005	+2.56125	+5.1225
7777	+2.55875	+5.1175	+10.235	+5.11875	+10.2375

* Offset binary for bipolar, straight binary for unipolar operating modes. Conversions may be made between 2's complement signed binary and offset binary numbers by subtracting 4000_8 from the 2's complement number (or adding 4000_8 to the offset binary number) and using only the low-order 12 bits of the result.

** Note that in all ranges, actual maximum positive voltage output is 1 LSB less than nominal maximum positive output.

ADV11-A

ADV11-A ANALOG TO DIGITAL CONVERTER

GENERAL

The ADV11-A is a 12-bit successive approximation analog-to-digital converter that samples analog data at specified rates and stores the digital equivalent value for processing. A multiplexer section can accommodate up to 16 single-ended or 8 quasi-differential inputs. The converter section uses a patented auto-zeroing design that measures the sample data with respect to its own circuitry offset and therefore cancels out its own offset error.

A/D conversions are initiated by program command, clock overflow, or external events. The program control is determined by the control and status register (CSR). The clock overflow command is supplied by the KWV11-A option. External event inputs can originate at the user's equipment or from the Schmitt trigger output on the KWV11-A clock. The digital data output is routed through a buffer register to the bus, from which it can be transferred into memory. This buffer optimizes the throughput rate of the converter.

Three reference signals are provided for self-testing on any channel input: two dc levels and one bipolar triangular waveform. This output can be used with DIGITAL diagnostic software to produce a data base for extremely thorough and precise analog linearity testing.

FEATURES

- 16-channel multiplexer
- Sample-and-hold functions
- Auto-zeroing technique
- Buffered data output
- Self-testing features

SPECIFICATIONS

Identification	A012
Type	Quad
Power	+5 Vdc ±5% at 2.0 A +12 Vdc ±3% at 450 mA
Bus Loads	
AC	3.25
DC	1

Inputs

Analog input protection	Fusible resistor guaranteed to open at ±85 V within 6.25 seconds. Guaranteed not to open from −25 V to +20 V at the input. Overload affects no components other than the fusible resistor on the overloaded channel; no other channels are affected.
Logic input protection	Fusible resistor guaranteed to open at ±25 V within 6.25 seconds. Guaranteed not to open from −4 V to +9 V at the input.
Analog input full scale range (FSR)	10.24 V bipolar (−5.12 V to +5.12 V)
Analog input dynamic resistance (Vin ≤ 5.12 V)	100 MΩ minimum
Analog input bias current (Vin ≤ 5.12 V)	50 nA, maximum
Logic input voltages	Low = 0.0 to +0.7 V High = +2 V to +5 V
Logic input currents	Low = −6.8 mA at 0 V High = +1.3 mA at +5 V
Logic input rise/fall time	400 ns maximum

Coding

A/D Converter Resolution	12 bits, binary weighted (2.5 mV nominal)
Format	Parallel offset binary, right justified
	Input Voltage
	+FS-1 LSB
	0
	−FS
	Output Code
	7777
	4000
	0

	(FS = 5.12 V; 1 LSB = 2.5 mV)
Vernier D/A Resolution	8 bits, binary weighted
Format	Offset binary encoded
	Input Code — **Approximate Offset Voltage**
	377 — +2.5 A/D LSB (+6.4 mV)
	200 — 0
	0 — −2.5 A/D LSB (−6.4 mV)

Performance

Gain error	Adjustable to zero
Offset error	Adjustable to zero
Differential linearity	No skipped states; no states wider than 2 LSB. 99% of state widths ±½ LSB
Integral linearity	±1 LSB, maximum non-linearity (referenced to end points)
Temperature coefficients	Gain = 6 PPM per °C Linearity = 2 PPM of full-scale range per °C Offset = 7.5 PPM of full-scale range per °C
Noise	Module = 0.4 LSB rms; 2 LSB peak System = 0.5 LSB rms; 2 LSB peak
Warm-up time	5 minutes, maximum

Timing

External start	Low level pulse, 50 ns minimum to 10 μs maximum; conversion starts on leading edge

Synchronization	0 to T
Conversion time	16 T (T = Clock period = 2 μs)
Transition interval (reacquisition interval between end of conversion or channel change and start of new conversion)	9 μs

Test Signals

The ADV11-A provides three output voltages for test purposes:

1. Positive dc level, +4.4 V (±15%)
2. Negative dc level, −4.4 V (±15%)
3. Triangular wave, 15 Hz nominal (±15%)

DESCRIPTION

General

The function of the ADV11-A module is to convert analog input data to a 12-bit digital word that is representative of the input. This is done by the channel selection, control logic, A/D converter, and bus interface functions as shown in Figure 1.

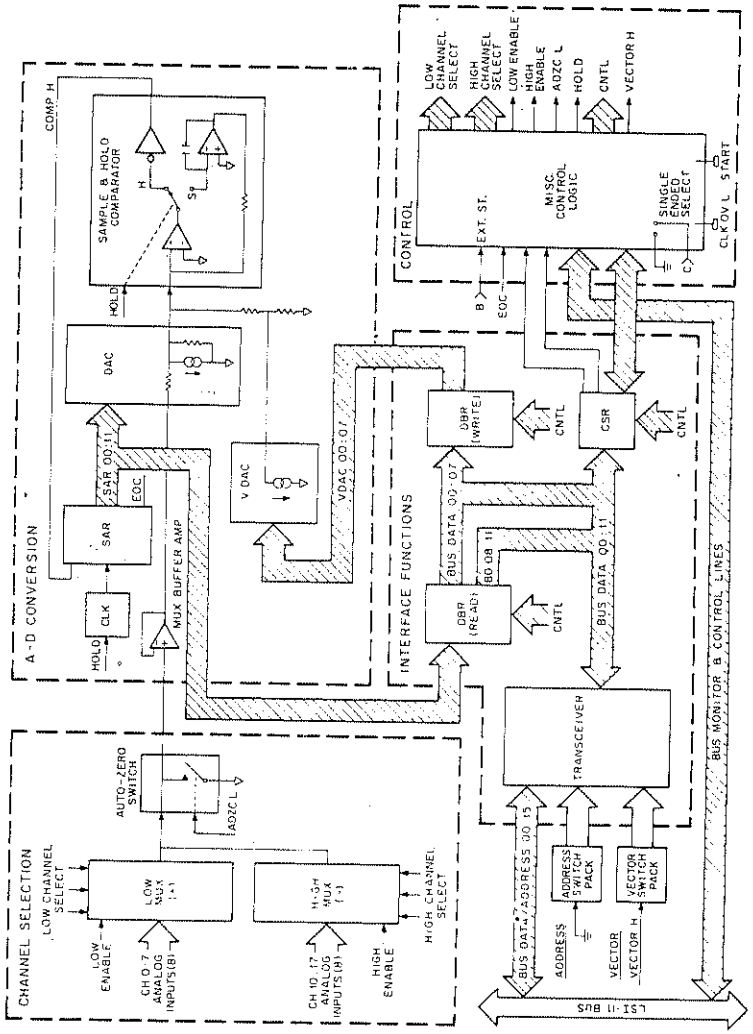


Figure 1 ADV11-A Functional Block Diagram

Registers

The control and status register (CSR) address can be selected in the range of 170000 to 177774 by using the S2 dip switch as shown in Figure 3. Switch S2 is factory-set at 170400, which is the recommended address as illustrated in Figure 3. The functions of the CSR bits are shown in Figure 4 and detailed in Table 2.

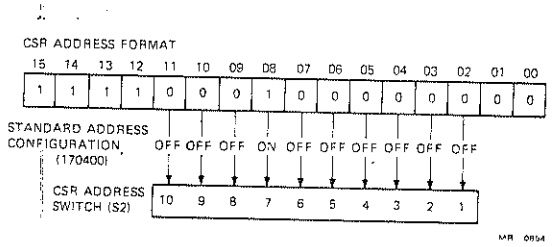


Figure 3 CSR Switch-Selectable Address

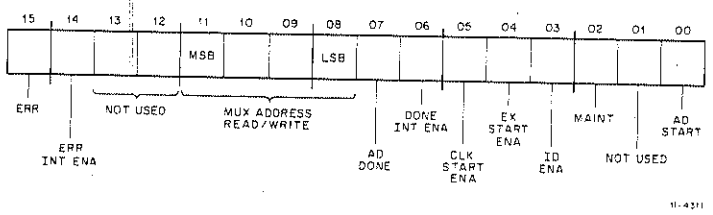


Figure 4 CSR Bit Format

Table 2 CSR Bit Functions

Bit	Description
15	A/D Error (Read/Write)—The A/D Error bit may be program set or cleared and is cleared by asserting BINIT L. It is set by any of the following conditions: 1. Attempting an external or clock start during the transition interval. 2. Attempting any start during a conversion in progress. 3. Failing to read the result of a previous conversion before the end of the current conversion.
14	Error Interrupt Enable (Read/Write)—When set, enables a program interrupt upon an error condition (A/D Error). Interrupt is generated whenever bits 14 and 15 are set, regardless of which was set first.
13-12	Not used.
11-8	Multiplexer Address (Read/Write)—Contains the number of the current analog input channel being addressed.
7	A/D Done (Read)—Set at the completion of a conversion when the data buffer is updated. Cleared when the data buffer is read by asserting BINIT L. If enabled, interrupts are requested simultaneously by both bits 7 and 15; bit 7 has the higher priority.
6	Done Interrupt Enable (Read/Write)—When set, enables a program interrupt at the completion of a conversion (A/D Done). Interrupt is generated when bit 7 and bit 6 are both set regardless of sequence.
5	Clock Start Enable (Read/Write)—When set, enables conversions to be initiated by an overflow from the clock option.
4	External Start Enable (Read/Write)—When set, enables conversions to be initiated by an external signal or through a Schmitt trigger from the clock option.

3:4

Table 2 CSR Bit Functions (Cont)

Bit	Description
3	ID Enable (Read/Write)—When set, causes bit 12 of the data buffer register to be loaded to 1 at the end of any conversion.
2	Maintenance (Read/Write)—When set, loads all bits of the converted data output equal to multiplexer address LSB (bit 8) at the completion of the next conversion. Cleared by asserting BINIT L. Used for all 0s and all 1s = test of A/D conversion logic.
1	Not used
0	A/D Start (Read/Write)—Initiates a conversion when set. Cleared at the completion of the conversion and by asserting BINIT L.

The data buffer register (DBR) address will be the next even address following the selected CSR address. This address has two separate DBR registers: one read-only and the other write-only. The functions of the register bits are shown in Figure 5 and described in Table 3.

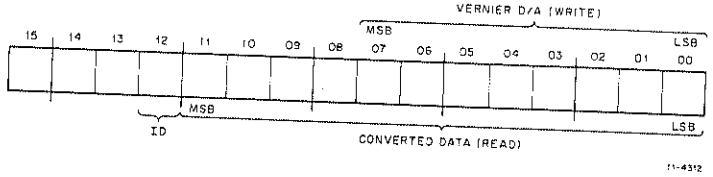


Figure 5 DBR Bit Format

Table 3 DBR Bit Functions

Bit	Function
Read-Only	
15-13	Not used. Should read as 0.
12	ID—When ID Enable (bit 3) of the CSR has been set, DBR bit 12 will be set to 1 at the end of the conversion.
11-0	Converted Data—These bits contain the results of the last A/D conversion.
Write-Only	
15-8	Not used.
7-0	Vernier D/A—These bits provide a programmed offset to the converted value (scaled 1 D/A LSB = 1/50 A/D LSB). The hardware initializes this value to 200_8 (mid-range). Values greater than 200_8 make this input voltage appear more positive.

Vector Interrupt

The A/D conversion complete interrupt vector is set by dip switch S1 (Figure 2). Any address in the range of 000_8 to 777_8 can be selected by the user. The switch is factory-configured for 400_8, the recommended vector, as shown in Figure 6. The error interrupt vector will be four words higher than the A/D conversion complete interrupt vector.

3-5

matrox
electronic systems ltd.

5800 ANDOVER AVE., T.M.R., QUE., H4T 1H4, CANADA
TEL.: 514—735-1182 TELEX: 05-825651

QRGB-256

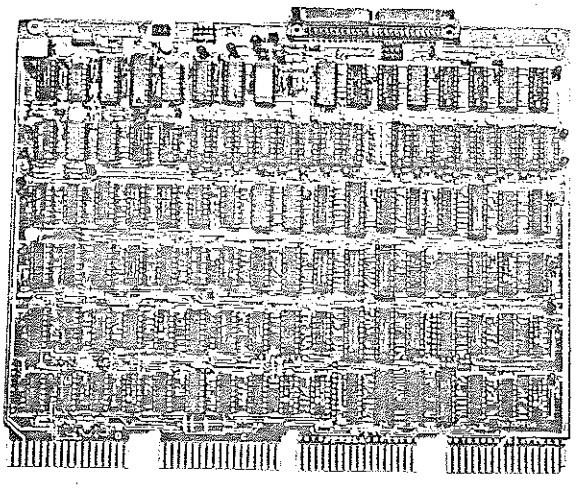
LSI-11 Plug-in Single Board Color Imaging System

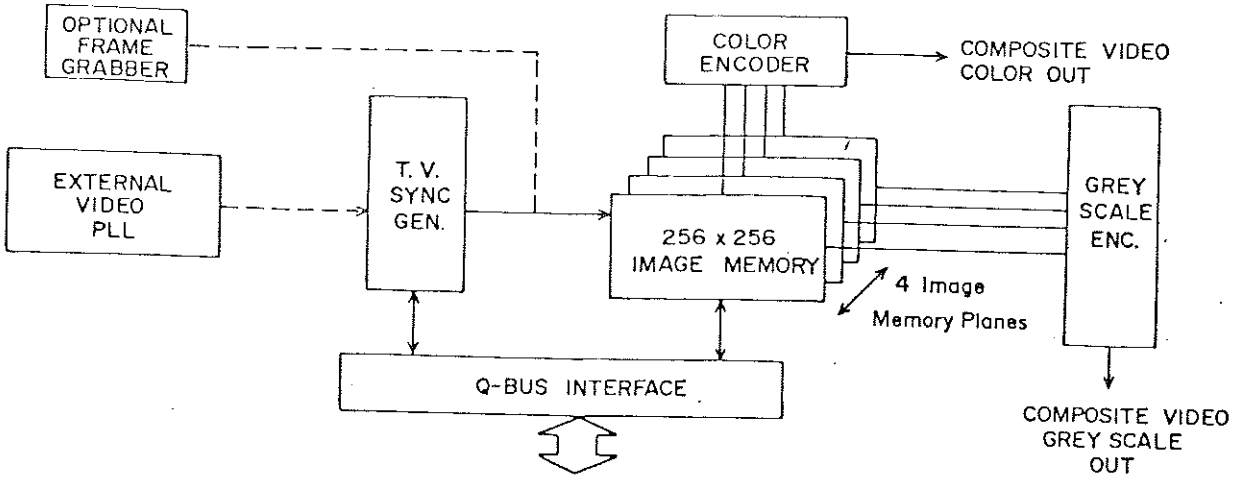
- -Separate image bit outputs for driving RGB monitors.
- -Built in phase lock loop.
- -Operates on NTSC or PAL color television standards.
- -Multiple cards can be combined to produce imaging systems with a higher number of bits/pixel.
- -Additional frame grabber cards allow capture of a single field of live video.
- -Direct plug-in graphics for LSI-11 based microcomputers.
- -Single card color/grey scale imaging system.
- -16 grey levels or colors.
- -Built in high speed 8 bit video D/A converter and 8 bit color encoder.
- -Composite color output drives standard color monitors directly.

The Matrox QRGB-256 plugs into the LSI-11 bus giving the LSI-11 user a complete color/grey scale imaging system, integrated on a single Quad height PC board.

The card features 256 by 256 dot resolution with 4 bit planes on a single PC board. The card includes built-in NTSC (American) or PAL (European) color and grey scale encoders which can provide up to 16 shades or colors. The encoders permit the QRGB-256 to directly drive standard low cost color, or black and white TV monitors on a single 75 Ohm cable.

A separate frame grabber card, the QFG-01, is available allowing the QRGB-256 to store TV pictures.





2.Ø FUNCTIONAL DESCRIPTION:

The QRGB-256 is a 256 x 256 dot resolution by 4 bit plane colour graphics board designed to be used by LSI-II based computers.

On-board colour and grey scale encoders provide up to 16 colours or grey levels in either American or European TV standards (strap selectable). The encoders permit the QRGB-256 to directly drive standard low cost colour, or black and white TV monitors on a single 75 ohm cable.

An on-board phase locked loop allows the QRGB-256 video outputs to be synchronized to an external video source, such as a TV camera. This allows the QRGB-256 to be used in broadcasting, or wherever its output will be mixed with other video signals.

Multiple QRGB-256 cards can be combined to give more bits per pixel. Specifically two QRGB-256 cards can be combined to give 8 bits per pixel. The two card system, with no additional hardware, will give a total of 256 colours or grey levels.

I/O mapped registers allow the user to: read from or write to any addressed display dot; turn video on or off; load any given value into entire video display memory; start a frame grab sequence; shift the display vertically; and check the status of the board.

A separate frame grabber card, the QFG-Ø1, is available. This card allows the QRGB-256 to directly store TV pictures.

3.Ø SOFTWARE SPECIFICATIONS:

3.1 INTRODUCTION:

This section describes the various registers and flags present in the QRGB-256. This information will enable the user to understand how to address and use the QRGB-256 to generate a display.

The QRGB-256 is strappable anywhere in the I/O address space of the LSI-II (see section 4.1), and uses eight consecutive byte locations. These locations will be referred to below by relative locations Ø-7. Figure 1 gives the address map of the QRGB-256. Boxes on the left of the diagram have been left empty. It is intended that these boxes be filled with the base address of the board as strapped less the least significant digit (e.g. if the base address = 16ØØØØ the boxes are filled with 16ØØØ). This will make fig. 1 a handy reference for the absolute address of each register.
Note: The board is shipped with base address 16ØØØØ.

3.2 X, Y REGISTERS:

The QRGB-256 uses a simple (X, Y) coordinate addressing scheme. X specifies the horizontal column and reads left to right, i.e. Ø is the left most dot, 255 (377 octal) the right most dot. Y specifies the line, and reads top to bottom i.e. Ø is the top line, however the top line can be changed by the scroll register, see 3.6.

Two directly addressable registers store the (X, Y) coordinates of a given dot (see fig 1a). The X and Y registers are 8 bits each allowing for a 256 x 256 dot resolution. The registers can be written to by byte (location 2 and 3) or word instructions (location 2). For word instructions both X and Y are loaded at the same time (Y is the high order byte) see fig. 1b.

LOCATION	DESCRIPTION	D15								DØ
Ø	Data						- -	- - D	D D D	
1	Control	C C	C C -	- R R						
2	X-Register						X X	X X X	X X X	
3	Y-Register	Y Y	Y Y Y	Y Y Y						
4	Scroll Register						S S	S S S	S S S	
5	Unused	- -	- - -	- - -						
6	Unused						- -	- - -	- - -	
7	Unused	- -	- - -	- - -						

Figure 1a Register locations under a byte write instruction

LOCATION	DESCRIPTION	D15								DØ
Ø	Control/Data	C	C C C	- - R	R - -		- - D	D D D		
2	Y and X Registers	Y	Y Y Y	Y Y Y	Y X X		X X X	X X X		
4	Scroll Register	-	- - -	- - -	- S S		S S S	S S S		
6	Unused	-	- - -	- - -	- - -		- - -	- - -		

Figure 1b Register locations under a word write instruction

LOCATION	DESCRIPTION	D15								DØ
Ø	Data						Ø Ø	Ø Ø D	D D D	
1	Flags	F F	Ø Ø Ø	Ø R R						
2	Unused						Ø Ø	Ø Ø Ø	Ø Ø Ø	
3	Unused	Ø Ø	Ø Ø Ø	Ø Ø Ø						
4	Unused						Ø Ø	Ø Ø Ø	Ø Ø Ø	
5	Unused	Ø Ø	Ø Ø Ø	Ø Ø Ø						
6	Unused						Ø Ø	Ø Ø Ø	Ø Ø Ø	
7	Unused	Ø Ø	Ø Ø Ø	Ø Ø Ø						

Figure 1c Register locations under a byte read instruction

LOCATION	DESCRIPTION	D15								DØ
Ø	Data/Flags	F	F Ø Ø	Ø Ø R	R Ø Ø		Ø Ø D	D D D		
2	Unused	Ø	Ø Ø Ø	Ø Ø Ø	Ø Ø Ø		Ø Ø Ø	Ø Ø Ø		
4	Unused	Ø	Ø Ø Ø	Ø Ø Ø	Ø Ø Ø		Ø Ø Ø	Ø Ø Ø		
6	Unused	Ø	Ø Ø Ø	Ø Ø Ø	Ø Ø Ø		Ø Ø Ø	Ø Ø Ø		

Figure 1d Register locations under a word read instruction

KEY

SYMBOL	MEANING	SYMBOL	MEANING
C	Control bit	S	Scroll Register bit
D	Data bit	R	Reserved
F	Flag bit	-	Don't care
X	X-Register bit	Ø	Always = Ø
Y	Y-Register bit		

3.3 DATA REGISTER:

After the dot coordinates have been loaded into the X and Y registers the dot colour or intensity level can be loaded by outputting data to location Ø (Byte), see fig. 1a,b. Data bits DØ-D3 determine the colour or grey level displayed, as in Fig 2. In a two card system connected as in 4.5 all 8 bits will define the colour or grey level.

The colour or grey level of the addressed dot can be read by reading from location Ø with a byte or word instruction. In a word instruction bits 15 & 14 are flags, see 3.5.

Data Bit:	D7 - D4	D3	D2	D1	DØ
Colour Affected	None	Green	Red	Blue	Mid-Green

Figure 2a Colours affected by a high on a specific data bit in a one card system

D3	DØ	Grey Scale Intensity		Colour
Ø	Ø Ø Ø	Black	Ø	Black
Ø	Ø Ø 1		1	Dark Green
Ø	Ø 1 Ø		2	Dark Blue
Ø	Ø 1 1		3	Light Blue
Ø	1 Ø Ø		4	Red
Ø	1 Ø 1		5	Brown
Ø	1 1 Ø		6	Magenta
Ø	1 1 1		7	Light Magenta
1	Ø Ø Ø		8	Green
1	Ø Ø 1		9	Light Green
1	Ø 1 Ø		1Ø	Dark Cyan
1	Ø 1 1		11	Light Cyan
1	1 Ø Ø		12	Brown-Green
1	1 Ø 1		13	Yellow-Green
1	1 1 Ø		14	White-Purple
1	1 1 1	White	15	White

Figure 2b Colours and grey scale levels available with a one card system

3.4 CONTROL REGISTER:

Writing a byte into location 1 loads the control register. Bits, 15-12 only are used and are defined below (fig. 3, see also fig 1a,b).

Data Bit:	D15	D14	D13	D12	D13 - D8
Function when = 1:	START ERASE/DIGITIZE	START FRAME GRAB	VIDEO OFF	CONTINUOUS FRAME GRAB	UNUSED
Function when = Ø:	DON'T ERASE/DIGITIZE	NO FRAME GRAB	VIDEO ON	NO CONT. FRAME GRAB	UNUSED

Figure 3 Control bit definitions

3.4 CONTROL REGISTER (Cont'd):

Bit 15 = ERASE/DIGITIZE; takes data from the data register and writes it into entire display memory.

Bit 14 = FRAME GRAB; loads one frame of video from QFG-Ø1 into display memory.

NOTE: The FRAME GRAB and ERASE/DIGITIZE instructions can be given asynchronously with vertical blanking. The time required to execute these instructions is 16-33 msec. No other instructions should be given to the QRGB-256 during this time. A busy flag is provided for this purpose (see 3.5).

Bit 13 = VIDEO ON/OFF: when this bit is Ø video is turned on, and when this bit is 1 video is turned off. When the board is initialized by the bus INITL signal video is turned on.

Bit 12 = CONTINUOUS FRAME GRAB: when this bit is 1 the QRGB-256 continuously executes a frame grab sequence. The X and Y registers may be written to and data may be read from the addressed location. No other commands should be issued to the QRGB-256 while the continuous frame grab is in progress. When this bit is Ø the busy flag should be checked before resuming normal QRGB-256 operation.

Writing a word into location Ø loads the control register and the data register, the control register is the high order byte. Normally bits 12, 13, 14 and 15 will be clear (no ERASE/DIGITIZE, CONTINUOUS FRAME GRAB, or FRAME GRAB and VIDEO ON).

3.5 FLAG REGISTER:

Two flags are available when reading a byte from location 1, or a word from location Ø, they are defined as follows (See fig. 4, see also fig 1c,d).

Data Bit:	D15	D14	D13 - D8
Function when = 1:	BUSY	VERTICAL BLANK	UNUSED
Function when = Ø:	NOT BUSY	NO VERTICAL BLANK	UNUSED

Figure 4 Flag bit functions

Bit 15 = BUSY: Indicates that the QRGB-256 is busy executing an ERASE/DIGITIZE or FRAME GRAB instruction, no other instructions should be issued to the QRGB-256 while this flag is set.

Bit 14 = VERTICAL BLANK: This flag is clear during the vertical blank period, and is provided to assist in animation synchronization.

3.6 SCROLL REGISTER:

The line to be displayed at the top of the screen is written into the scroll register. The register is 8 bits wide and can be written to by a byte or word instruction to location 4 (see fig 1a,b). In American standard, 24Ø of the lines are displayed at a time, in other words 16 of the lines remain hidden, scrolling however will bring these lines on the screen. For example loading 25 decimal into the scroll register will produce a display as in fig. 5.

3.6 SCROLL REGISTER (Cont'd):

Standard:	American	European
Display Scan lines	25 ↓ 256 Ø ↓ 9	25 ↓ 256 Ø ↓ 24

Figure 5 Scan Lines Displayed

Normally the scroll register should be initialized to Ø especially before a Frame Grab sequence. The INITL signal does not clear the scroll register.

4.Ø STRAP OPTIONS:

4.1 ADDRESS STRAPS:

The following diagram gives the straps for the address decoder, these must be configured to give the desired 8 location boundary in the I/O space. The desired address is converted to binary (3 least significant bits Ø), and for every 1 in the remaining bits the strap between the respective pins is removed, and for every Ø in the remaining bits the pins are strapped together.

Note: For BS7 to be asserted, and the board selected, address bits 13-15 must be high.

Example: 163Ø7Ø (octal) converts to 1 11Ø Ø11 ØØØ 111 ØØØ, therefore for A1Ø, A9, A5, A4 and A3 the strap is removed and for A12, A11, A8, A7 and A6 a strap is inserted.

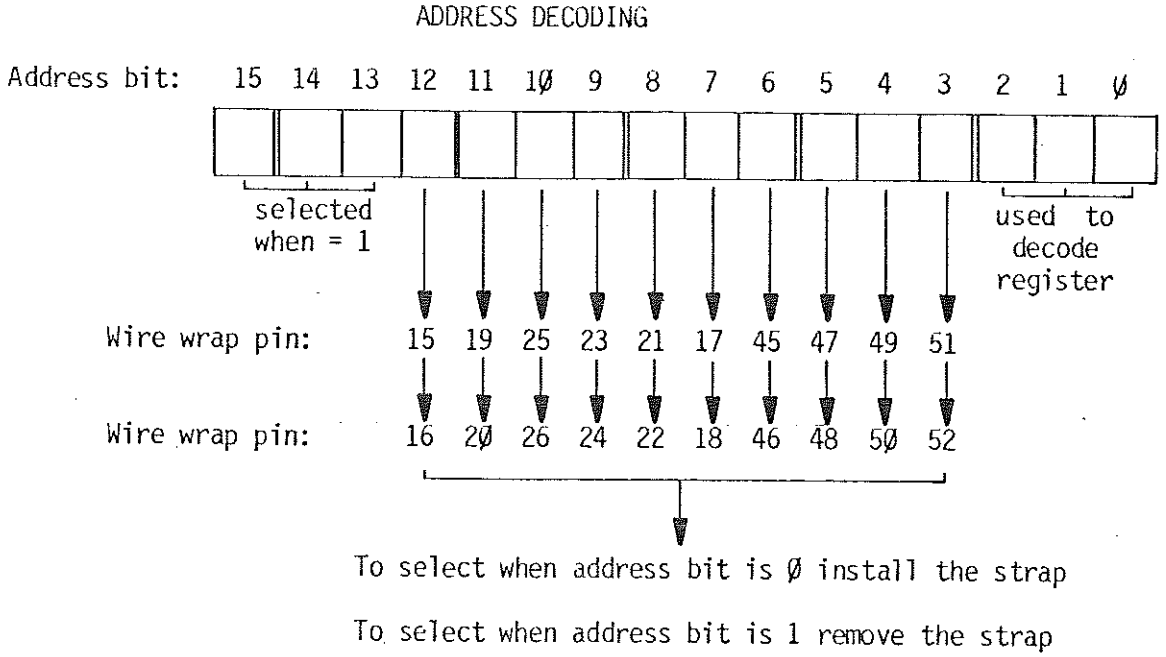
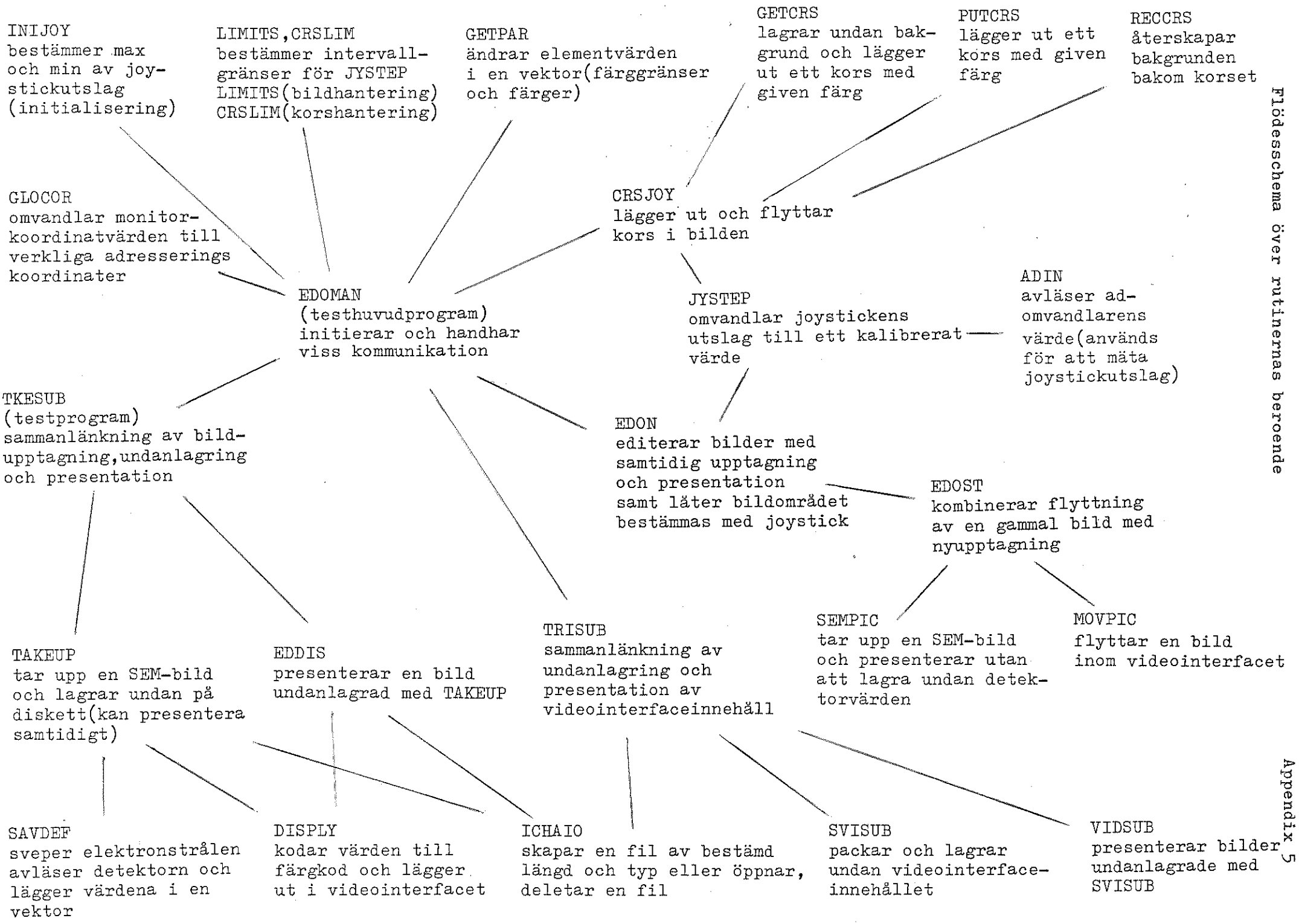


Figure 6 Address Decoder Strap Selection



Flödesschema över rutinernas beroende

Appendix 5

Flödesschema SAVDEF

- sätt parallellportens kanal A (horisontell avlänkning) och B (vertikal avlänkning) i skrivmod

- sätt ad-omvandlaren att följa kanal 0 (detektorkanalen)

- lagra den första avlänkningsadressens x-koordinat i minnescellen STARTX och i parallellport A

- lagra den första avlänkningsadressens y-koordinat i paralellport B

- sampla (enbart för att få tidsreferens så att elektronstrålen skall hinna stabilisera sig)

- lagra horisontellt avstånd i R1 (general purpose register 1)

- lagra vertikalt avstånd i minnescellen DISTY

- lagra antal punkter per horisontell linje i R2

- minska punkträknaren (R2) med ett (R2=R2-1)

- lagra undan antalet punkter per linje i minnescellen NUMADP

- lagra antalet linjer att avsöka i minnescellen LINC

- minska innehållet i LINC med ett (LINC=LINC-1) (första linjen har börjat avsökas)

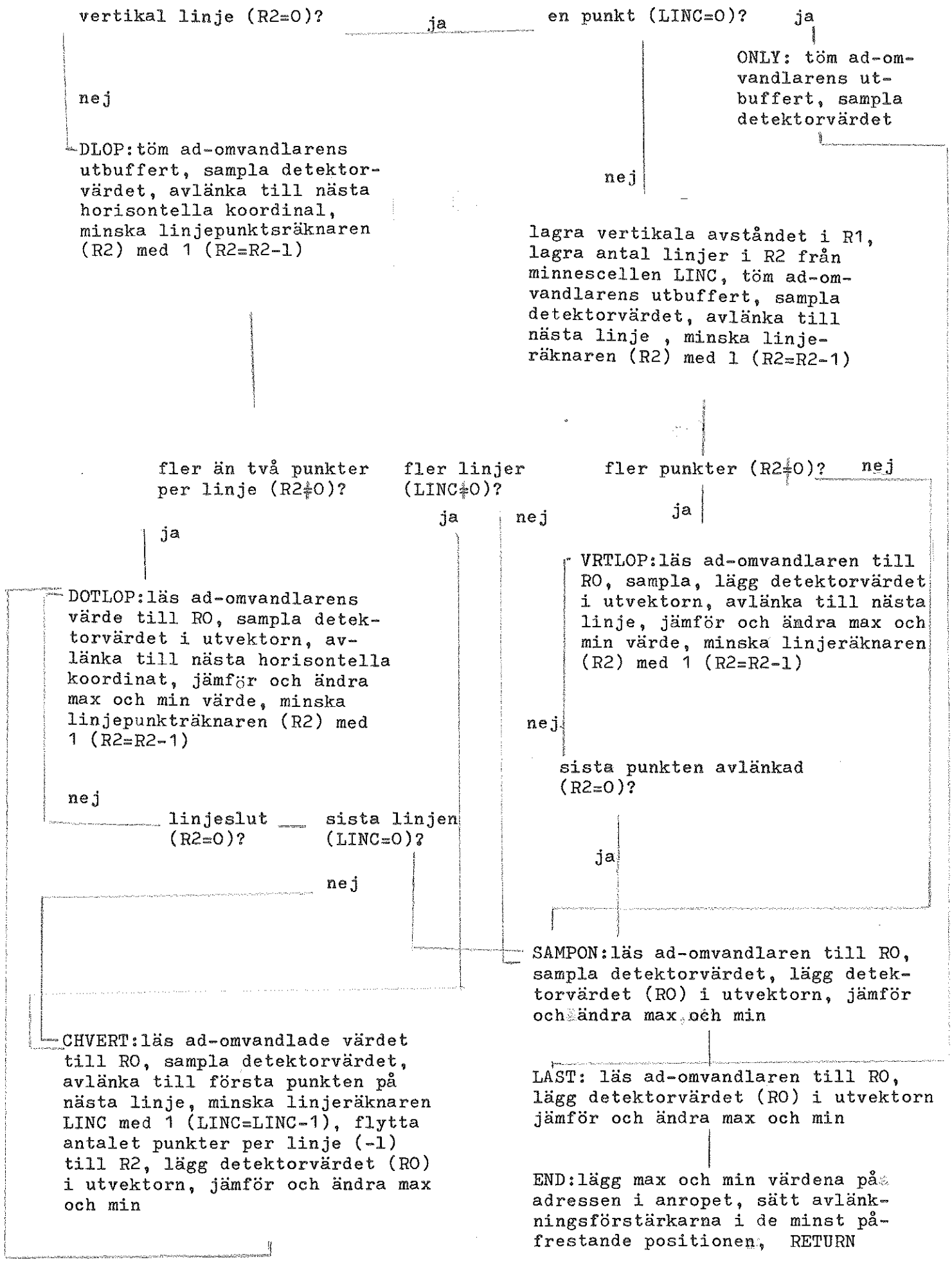
- lagra maxvärdesadressen i minnescellen MAXADR

- lagra hittillsvarande maxvärde i R3

- lagra minvärdesadressen i minnescellen MINADR

- lagra minvärdet i R4

- lagra utvektorns startadress i R5


vertikal linje (R2=0)?
ja
en punkt (LINC=0)?
ja
ONLY: töm ad-omvandlarens utbuffert, sampla detektorvärdet
nej
nej
DLOP:töm ad-omvandlarens utbuffert, sampla detektorvärdet, avlänka till nästa horisontella koordinal, minska linjepunktsräknaren (R2) med 1 (R2=R2-1)
lagra vertikala avståndet i R1, lagra antal linjer i R2 från minnescellen LINC, töm ad-omvandlarens utbuffert, sampla detektorvärdet, avlänka till nästa linje , minska linjeräknaren (R2) med 1 (R2=R2-1)
fler än två punkter per linje (R2≠0)?
fler linjer (LINC≠0)?
fler punkter (R2≠0)?
nej
ja
ja
nej
ja
VRTLOP:läs ad-omvandlaren till RO, sampla, lägg detektorvärdet i utvektorn, avlänka till nästa linje, jämför och ämdra max och min värde, minska linjeräknaren (R2) med 1 (R2=R2-1)
DOTLOP:läs ad-omvandlarens värde till RO, sampla detektorvärdet i utvektorn, avlänka till nästa horisontella koordinat, jämför och ändra max och min värde, minska linjepunkträknaren (R2) med 1 (R2=R2-1)
nej
nej
sista punkten avlänkad (R2=0)?
linjeslut (R2=0)?
sista linjen (LINC=0)?
ja
nej
SAMPON:läs ad-omvandlaren till RO, sampla detektorvärdet, lägg detektorvärdet (RO) i utvektorn, jämför och ändra max och min
CHVERT:läs ad-omvandlade värdet till RO, sampla detektorvärdet, avlänka till första punkten på nästa linje, minska linjeräknaren LINC med 1 (LINC=LINC-1), flytta antalet punkter per linje (-1) till R2, lägg detektorvärdet (RO) i utvektorn, jämför och ändra max och min
LAST: läs ad-omvandlaren till RO, lägg detektorvärdet (RO) i utvektorn jämför och ändra max och min
END:lägg max och min värdena på adressen i anropet, sätt avlänkningsförstärkarna i de minst påfrestande positionen, RETURN

Flödesschema för SEMPIC

-sätt DRV11-J portens kanal A och B i skrivmod
-sätt ADV11-A DA-omvandlaren att följa kanal 0 (detektorkanalen)
-avlänka till första horisontella koordinat och lagra undan koordinaten i minnescellen STARTX
-avlänka till första vertikala koordinat
-lagra det horisontella avståndet i R1(general purpose register 1)
-lagra det vertikala avståndet i minnescellen DISTY
-lagra antalet punkter i den horisontella avlänkningen i R2

antal punkter per horisontell linje (R2)?

\>2 =2 <2(=1)

SU2:minska antalet punkter per linje med 2(R2=R2-2) | VERTY1: flagga VERTY=-1 | flagga VERTY=0

flagga JUMPY=-1 | flagga JUMPY=1 | flagga JUMPY=1

SU3:lagra undan antalet punkter per linje före linjebyte(R2) i minnescellen NUMADP
-lagra antalet horisontella linjer i R3
-initialisera färggränstabellen (lägg ut i programkoden)
-sampla och ad-omvandla det första detektorvärdet
-flagga JUMPy (fler än två punkter per linje)?

-1(≤0)(fler) 1(≤0)(mindre eller lika med)

flagga VERTY (en eller två punkter per linje)?

-1(≠)(tvåpunktslinje) 0(=)vertikal linje)

NMDEC:minska antal linjer att läsa med ett (R3=R3-1) | minska antal linjer att läsa med ett (R3=R3-1)

fler linjer att svepa(R3>0)?

NORM:adressera nästa horisontella koordinat (öka med det horisontella avståndet (R1)) | nej (R3=0) | ja(R3>0)

adressera nästa vertikala koordinat (öka med det vertikala avståndet(minnescellen DISTY))

NMADR: initialisera färgtabellen (lägg ut i programkoden)

-lägg första monitorpunktens horisontella koordinat i den undre byten i buffertminnescellen STMON
-lägg första monitorpunktens vertikala koordinat i den övre byten i buffertminnescellen STMON
-första monitorpunktens koordinat (minnescellen STMON) lagras i R4
-det första ad-omvandlade värdet läses till R0

flagga JUMPY (fler än två punkter per linje)?

-1(=0,fler)

1(> 0, mindre eller lika med)

fler linjer att svepa (R3 ⩾ 0)?

ja (R3 ⩾ 0)

nej(R3 < 0)

NMSC:sampla detektorvärdet
läs det vertikala avståndet
från minnescellen DISTY till R5

flagga JUMPY (fler än två punkter per linje)?

-1(⩽0,fler)

1(0, mindre eller lika med)

flagga VERTY (en eller två punkter per linje)?

0(=)(vertikal linje)

-1(≠)(tvåpunktslinje)

VRTSCA:fler linjer att svepa(R3≠0)?

fler linjer att svepa(R3 0)?

ja(R3≠0)

nej

ja

nej

avlänka till nästa linje

avlänka till första punkt på nästa linje

BRBL9:sätt punkträknaren (R2) till ett(R2=1)

DOTLOP:läs det ad-omvandlade värdet till RO, sampla detektorvärdet, öka monitoradressen (R4) (nästa horisontella punkt)

START:avlänka till nästa horisontella punkt

BL9:lägg monitoradressen(R4) i videointerfacet

koda det ad-omvandlade detektorvärdet(RO) till färg och lägg ut på monitorn(binär jämförelse enligt schemat nedan)

L1	ingen färgändring på monitorn
L2	färg C1
	färg C2
L3	färg C3
L4	färg C4
L5	färg C5
L6	färg C6
L7	färg C7
L8	färg C8
L9	färg C9
L10	färg C10
L11	färg C11
L12	färg C12
L13	färg C13
L14	färg C14
L15	färg C15
L16	färg C16
L17	ingen färgändring på monitorn

vid jämförelse mellan värde och färggräns gäller att strikt mindre än medför hopp nedåt i schemat

räkna ner punkträknaren(R2)
R2=0?

ja / nej

B9:flagga JUMPY (slut på monitorlinje, avlänkningslinje eller en-/två-punkts linje)?

0(=)(slut på monitorlinje) / -1(<0)(slut på avlänkningslinje / 1(0)(en-eller två-punktslinjer)

B10:läs av ad-omvandlaren till RO
sampla detektorvärdet startpunkten för nästa monitorlinje i R4
läs antal punkter före linjebyte(minnescellen NUMADP) till R2
sätt flagga JUMPY=-1 (ändrat monitorlinje sist)
räkna ner linjeräknaren (R3) med ett (R3=R3-1)

har sista linjen svepts(R3=0)?

nej / ja

B12:töm ad-omvandlarens utbuffert

sätt linjepunktsräknaren (R2) till två (R2=2)(två punkter sedan monitorlinjeskift)
läs ad-omvandlaren till RO
sampla detektorvärdet
sätt flagga JUMPY=0 (ändrat avlänkningslinje sist)
räkna upp monitorpunktadressen (R4)

sveps sista linjen (R3=1)?

nej / ja

adressera nästa avlänkningslinje

LT2D:flagga VERTY(en-eller två-punktslinje)?

-1(< 0) (tvåpunkts) / 0(≥) (enpunkts)

DOT2:minska linjeräknaren (R3) med ett (R3=R3-1)
fler linjer (R3>0)?

nej / ja

minska linjeräknaren (R3) med ett(R3=R3-1)

fler punkter att söka (R3>0)?

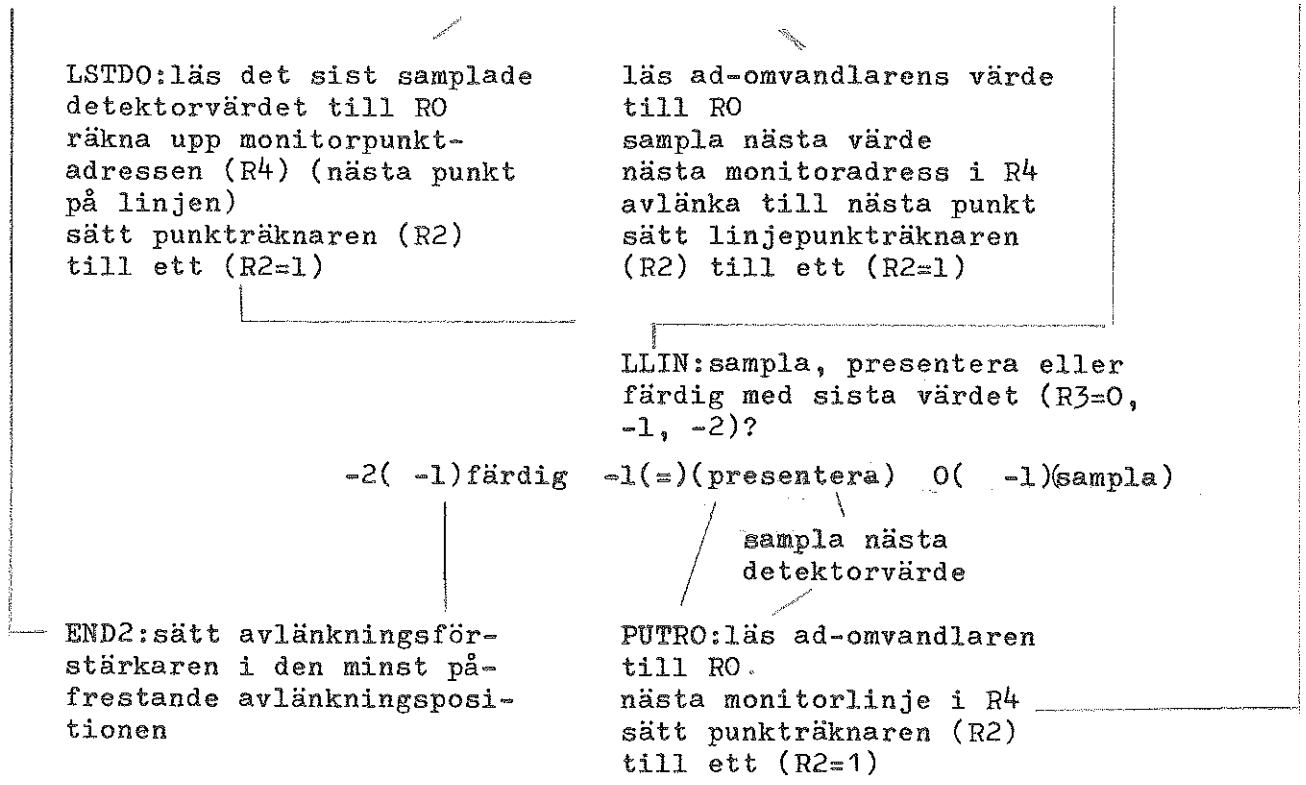
ja / nej

läs ad-omvandlarens värde till RO
sampla nästa värde
nästa monitoradress i R4
avlänka till nästa punkt
sätt linjepunkträknaren (R2) till ett (R2=1)

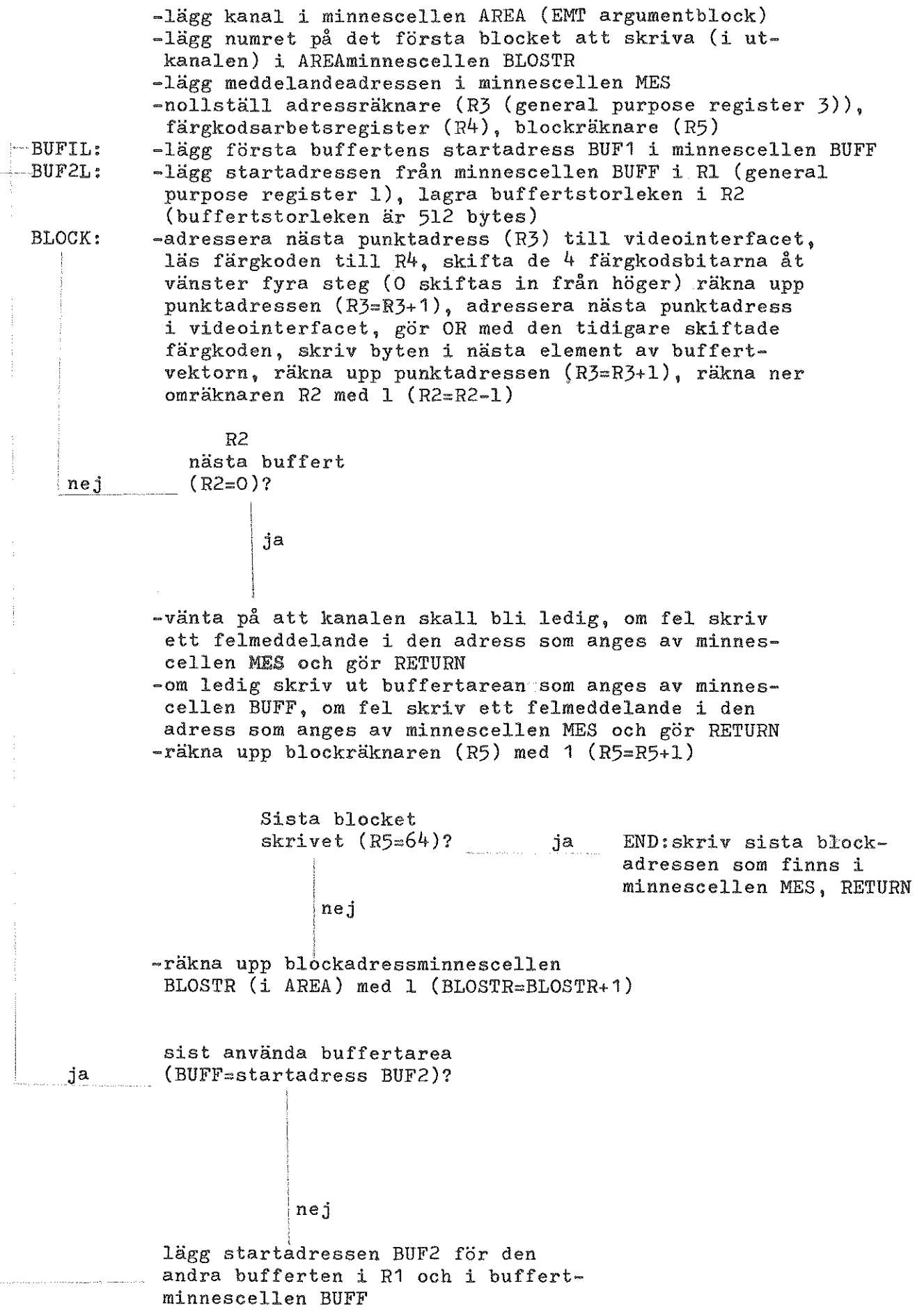
läs ad-omvandlarens värde till RO
sampla detektorvärdet
lägg nästa monitorlinje i R4
sätt punkträknaren (R2) till två
avlänka till nästa linje

LSTLIN:sampla presentera eller färdig med sista värdet(R3=0,-1,-2)?

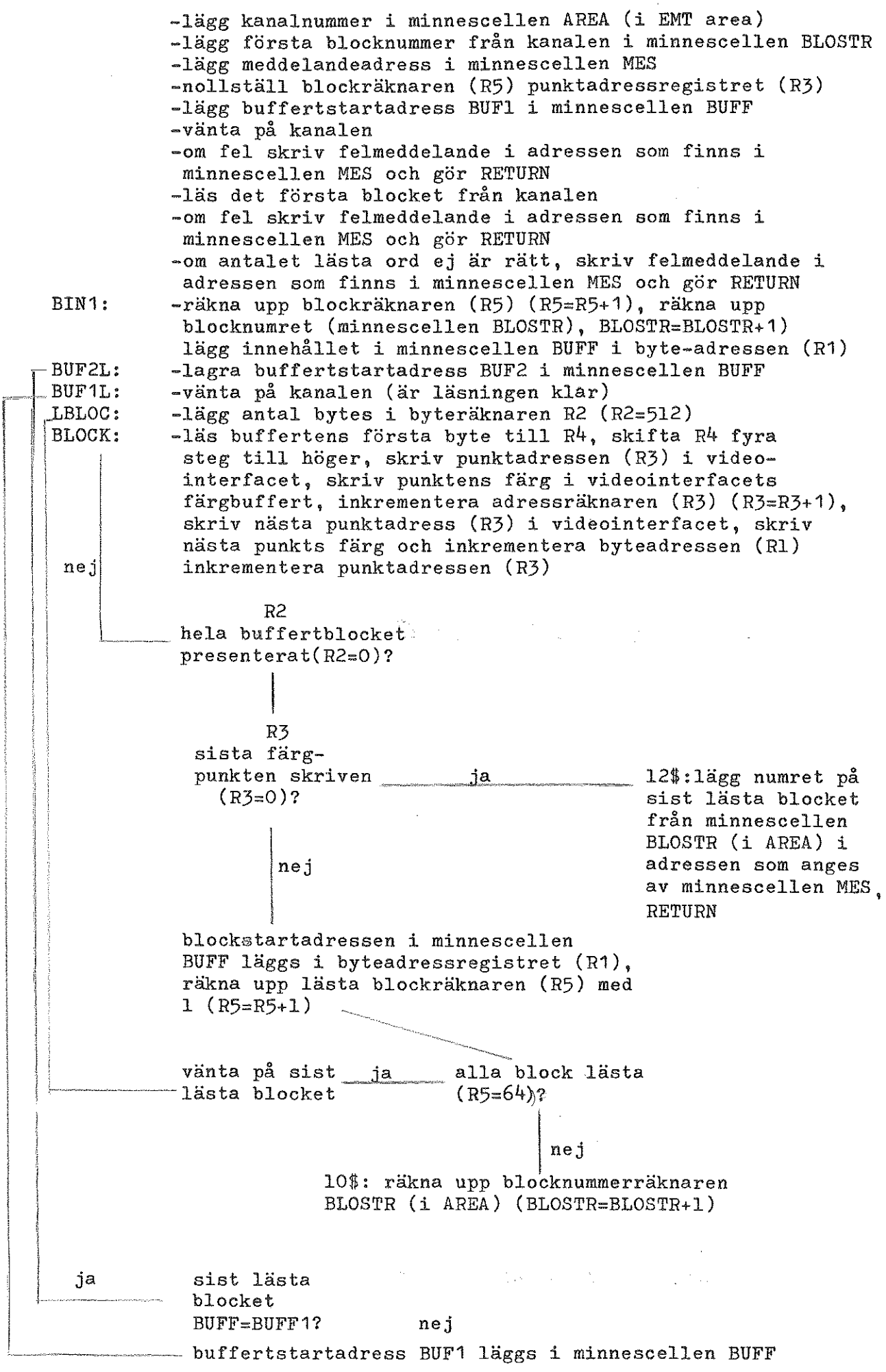
-2(<-1,färdig) / -1(=,presentera) / 0(>-1,sampla)


LSTDO:läs det sist samplade detektorvärdet till RO
räkna upp monitorpunkt-adressen (R4) (nästa punkt på linjen)
sätt punkträknaren (R2) till ett (R2=1)
läs ad-omvandlarens värde till RO
sampla nästa värde
nästa monitoradress i R4
avlänka till nästa punkt
sätt linjepunkträknaren (R2) till ett (R2=1)
LLIN:sampla, presentera eller färdig med sista värdet (R3=0, -1, -2)?
-2(-1)färdig
-1(=)(presentera)
0(-1)sampla)
sampla nästa detektorvärde
END2:sätt avlänkningsför-stärkaren i den minst på-frestande avlänkningsposi-tionen
PUTRO:läs ad-omvandlaren till RO.
nästa monitorlinje i R4
sätt punkträknaren (R2) till ett (R2=1)

Flödesschema SVISUB



Flödesschema för VIDSUB



Användarbeskrivning

Kommunikationen med operatören sköts
till största del genom huvudprogrammet FDOMAN.
Detta är enbart till för att prova subroutinerna tillsammans.

Här följer en kort förteckning över möjliga kommandon.
Siffran inom parantes anger den tangent som
utför momentet eller ömmar parametern.
Om parametern är ömmad kan den därefter ges nytt värde.
Siffran 0 innebär att uppritning av bilder on-line startar.

DISPLAYFORM(1)=11..44bestämmer den presenterade bildens storlek
Bilden kan byggas upp av 1 till 16 rutor,vilka kan
kombineras liksom matriselement.Den första siffran i formatet
anger antalet rutor i vertikalled den andra i horisontalled.
Möjliga format är 11,12,13,14,21,22,23,24,31,32,33,34,41,42,43,44
(11=64x64punkter...44=fullbild256x256).

DPPOSITION(2)=11..44bestämmer bildposition
Bildens placering på monitorn anges med matriselementbeteckning
(11 motsvarar överst till vänster och 44 nederst till höger).

STARTX(3)=,STARTY(4)=startpunkt(horisontellt,vertikalt) för
avläsningen(motsvarande bildens övre vänstra hörn).
Koordinaten anges i intervallet 0..-32768..-1(16-bitar).

DISTX(5)=,DISTY(6)=1..256 avståndet mellan avlästa koordinater
(horisontellt,vertikkalt).

STEPX(7)=,STEPY(8)=stegförflyttning(horisontellt,vertikkalt).
Bildområdet flyttas ett visst antal avläsningssteg från.
(motsvarar joystickutslag)

JOYMOD(9)=1..6 anger hur joysticken styr bildområde(när,hur).
1=ta upp bilden och återvänd till huvudprogrammet
2=ta upp samma bild om igen(loop)
3=ta upp en bild när joysticken är skild från centrum
4=loop och kombinera flyttning och nyuppläsning av
flyttning styrs joysticken
5=loop och alltid ny uppläsning av bild (ingen
flyttning av samma bild) joysticken styr
6=ny uppläsning om joystick skild från centrum
bildområde styrs av joysticken

PICWAIT(10)=tiden mellan bilduppläsningarna vid loop
anges i ticks (1 tick=20ms)
MAXTIM(11)=maximal väntetid mellan joystickanrop
DIFFTIM(12)=skillnad mellan max och min väntetid
(mellan joystickanrop)

STEPTIM(13)=tid mellan joystickanrop om joystickutslaget
är så stort att bilden flyttas i ett stort steg

TWINKLESPEED(14)= aktiv tid för korset vid blinkning (i ticks)

CROSS(15)=lässer ut ett kors på monitorn och låter detta flyttas med joysticken

CRSWING(16)=vinslänsden på detta kors

CONVERTVIDEOCOORDINATE TO DEFL(17)=omvandlar monitor koordinater till adresseringskoordinater (bilden får ej ändrats)

CHANGE COLCODF(18)=ger möjlighet att ändra färger på kodningen

CHANGE COLLIM(19)=kallar rutin för ändring av färgsränser (se beskrivning SEMPIC.DISPLY)

SAVE OR DISPLAY VIDEO(20)=kallar rutin för undanlagring eller återrresentation av videointerface innehåll (undanlagring sker på diskett)

TAKEUP OR EDIT SAVED PICTURE(21)=kallar rutin för undanlagring,framtagning av ad-värden(lagring på diskett). Värden kodas till bilder och presenteras.

(0)= startar bilduppbygnings on-line
Följande händer.
Uppbygningsparametrarna kollas
Bilden tas upp eventuellt med hjälp av joystick
om joymod=1 se görs omedelbart return efter att
bilden tagits
annars görs return till huvudprogrammet genom att trycka
ned godtycklig tangent

med korset kan man
ändra färg genom att trycka ner en hexadecimal tangent
få det börja blinka genom att trycka T och sluta med N
behålls det och återgå till huvudprogrammet genom att H trycks ned
ta bort korset med annan tangent än 1..F,T,N och H(se ovan)

om man skall ändra färger(18) eller gränser(19) så görs
det genom att ange parameternummer och nytt värde
vid undanlagring och visning av monitorbilder måste
en fil skapas eller öppnas.Denna rutin är själv-
instruerande.

Rutinen som lagrar undan ad-värden motsvarande en bild
är ej färdig.Den nuvarande varianten saknar kommunika-
tion till huvudprogrammet med parametrar.Dessa får vid
anrop ges genom underprogrammet.
Man kan välja om man vill
 ta upp ta fram,deleta en fil
 ändra i färggränstabellen eller återvända till operatörsprogrammet.
Färggränstabellen ändras linjärt mellan ett
av operatören givet max och min.

Om man skall ta ur en bild från sin
efter displayformat(DISPFORMAT) och displayposition(DISPOS).
Här avses bildstorlek(11,,44) och var på monitorn bilden skall
placeras(11,,,44).
Om man ej vill ha bilden visad sätts DISPOS=0.
Startpunkt och adresseringsavstånd anges med samma
uttryck som i huvudprogrammet(STARTPOSX,STARTPOSY i 0,,-32768,,-1
och DISTX,DISTY i 1,,256).

Vid presentation av en tidigare undanlagrad bild frågas efter
DISPFORMAT (11,,,44)vilket format den visade bilden skall ha
DISPOS (11,,44) var påmonitorn den framtagna bilden skall vi
sas.

TAKEUPFORMAT är det format som bilden togs upp med(11,,44).
STARTDISPOS anger vilken del av den upptagna bilden som skall
visas.Denna del kan bestämmas på en helt standardposition när.
Detta genom att man lägger till 55,50,05 eller 00 till
standardpositionskoordinaten(11 i övre vänstra hörnet).
Om STARTDISPOS sätts 1105 börjar
den presenterade bilden ett halvt standardformat horisontellt
in på den upptagna bilden.På detta vis kan intressanta delområden
presenteras i mindre format än upptagningsformat.

För att få en fil deletad skrivs ett device DY0: eller DY1:
plus filnamn in(ej tur) tillsammans med option /DOLD

Testprogram för videointerfacet

```
;*****  ORGBTL ADDRESS
;INORGB  .EXIT
ADR=170740
BEGIN:  CLRB    @#<ADR+1>
        CLRB    @#<ADR+4>
        CLR     R1
        CLR     R2
1$:     CLR     R3
        BITB    #200,@#<ADR+1>
        BMI     1$
3$:     MOVB    R1,@#<ADR+2>
        BITB    #200,@#<ADR+1>
        BMI     3$
4$:     MOVB    R2,@#<ADR+3>
        BITB    #200,@#<ADR+1>
        BMI     4$
        MOVB    R3,@#ADR
        INCB    R2
        BNE     1$
        COMB    R1
        BITB    #17,R1
        BNE     2$
        INCB    R3
        COMB    R1
        INCB    R1
        BITB    #377,R1
        BNE     1$
.EXIT
.END    BEGIN

.TITLE  INORGB
.GLOBL  INORGB
ADR=170740
INORGB: TST     (R5)+
        MOVB    @(R5)+  ,@#ADR
        MOVB    #UR10000000     ,@#<ADR+1>
        RTS     PC

.TITLE  PUTDOT
ADR=170740
PUTDOT: TST     (R5)+
        MOVB    @(R5)+  ,@#<ADR+2>
        MOVB    @(R5)+  ,@#<ADR+3>
        MOVB    @(R5)+  ,@#ADR
        RTS     PC
        .END
        .TITLE  GETDOT
        ADR=170740
GETDOT: TST     (R5)+
        MOVB    @#<ADR+2>,(R5)+
        MOVB    @#<ADR+3>,(R5)+
        MOVB    @#ADR   ,@(R5)+
        RTS     PC


        CALLED TO SCROLL

        S TO SCROLL 0,,255

        ;ORGBADRESS

        ADR+4>  ;SCROLL REGISTER
```

Testprogram för videointerfacet

```
          .TITLE  ORGBTL ADDRESS
          .GLOBL  .EXIT
          ADR=170740
BEGIN:    CLRB    @#<ADR+1>
          CLRB    @#<ADR+4>
          CLR     R1
          CLR     R2
          CLR     R3
1$:       BITB    #200,@#<ADR+1>
          BMI     1$
          MOVB    R1,@#<ADR+2>
3$:       BITB    #200,@#<ADR+1>
          BMI     3$
          MOVB    R2,@#<ADR+3>
4$:       BITB    #200,@#<ADR+1>
          BMI     4$
          MOVB    R3,@#ADR
          INCB    R2
          BNE     1$
          COMB    R1
          BITB    #17,R1
          BNE     2$
          INCB    R3
2$:       COMB    R1
          INCB    R1
          BITB    #377,R1
          BNE     1$
          .EXIT
          .END    BEGIN

          .TITLE  INQRGB
          .GLOBL  INQRGB
          ADR=170740
INQRGB::  TST     (R5)+
          MOVB    @(R5)+  ,@#ADR
          MOVB    #UB10000000     ,@#<ADR+1>
          RTS     PC
          .END
          .TITLE  PUTDOT
          ADR     =170740
PUTDOT::  TST     (R5)+
          MOVB    @(R5)+  ,@#<ADR+2>
          MOVB    @(R5)+  ,@#<ADR+3>
          MOVB    @(R5)+  ,@#ADR
          RTS     PC
          .END
          .TITLE  GETDOT
          ADR=170740
GETDOT::TST     (R5)+
          MOVB    @(R5)+  ,@#<ADR+2>
          MOVB    @(R5)+  ,@#<ADR+3>
          MOVB    @#ADR   ,@(R5)+
          RTS     PC
          .END

          .TITLE SCROL
;+THIS PROGRAM MAY BE CALLED TO SCROLL
;THE QRGB
;  CALL SCROL(ISCROL)
;  ISCROL=NUMBER OF LINES TO SCROLL 0,,255
;-
          ADR=170740        ;QRGBADRESS

SCROL::   TST     (R5)+
          MOV     @(R5)+,<ADR+4>   ;SCROLL REGISTER
          RTS     PC
          .END SCROL
```

```
          .TITLE SAVDEF.MAC
$+THIS PROGRAM MAY BE CALLED TO TAKE UP A PICTURE
$STORE IT IN A VECTOR GIVEN IN THE ARGUMENTS
$CALL SAVDEF(DEFLSTARTX,DEFLSTARTY,DISTX,DISTY,NUMADP,NUMLIN,
$       MAXV,MINV,VECT)
$       (ALL VALUES INTEGER)
$WHERE:DEFLSTARTX,DEFLSTARTY=STARTADRESS OF DEFLECTION(0,65536)
$        (INTEGER*4) AND SECOND WORD IS DISREGARDED
$       DISTX=DISTANCES BETWEEN ADRESSED POINTS IN XSCAN
$       DISTY=DISTANCES BETWEEN ADRESSED POINTS IN YSCAN
$       NUMADP=NUMBER OF ADRESSED POINT IN ONE LINE
$       NUMLIN=NUMBER OF LINES TO SCAN
$       MAXV=MAXIMUMVALUE IS STARTMAX ON ENTER AND ABSMAX ON RETURN
$       MINV=MINIMUMVALUE IS STARTMIN ON ENTER AND ABSMIN ON RETURN
$       VECT=VECTOR TO PUT AD VALUES IN
$-

          CSRA=164160       ;ADRESS TO CONTROLWORD DRV11-J CHAN A
          CSRB=164164       ;ADRESS TO CONTROLWORD DRV11-J CHAN B
          ADSR=170400       ;ADRESS TO CONTROLWORD ADV-11 A

SAVDEF:: TST     (R5)+              ;SKIP ARGUMENT COUNT
          MOV     #400,@#CSRA       ;PUT PORT A IN WRITEMODE
          MOV     #400,@#CSRB       ;PUT PORT B IN WRITEMODE
          CLR     @#ADSR            ;CLEAR CONTROL AD
          MOV     @(R5)+,STARTX     ;START(INT*4)LOW WORD VALUE INTO STARTX
          MOV     STARTX,@#<CSRA+2> ;LOAD START X VALUE INTO DRA
          MOV     @(R5)+,@#<CSRB+2> ;STARTY(INT*4)LOW WORD VALUE INTO DRB
          INC     @#ADSR            ;SAMPLE TO GET SETTLINGTIME (DEFL)
          MOV     @(R5)+,R1         ;DOTDIST X IN R1
          MOV     @(R5)+,DISTY      ;DOTDIST Y IN DISTY
          MOV     @(R5)+,R2         ;PUT NUMBER OF POINTS IN A LINE
          DEC     R2                ;FIRST DEFL DONE
          MOV     R2,NUMADP         ;NUMADP(WHEN NEW LINE DEFL DONE)
          MOV     @(R5)+,LINC       ;PUT NUMBER OF LINES INTO LINC
          DEC     LINC              ;DEF FIRST LINE STARTED
          MOV     (R5),MAXADR       ;STORE ADRESS OF MAXVALUE INMAXADR
          MOV     @(R5)+,R3         ;MAXVALUE TO COMPARE IN R3
          MOV     (R5),MINADR       ;STORE MINVALUE ADRESS IN MINADR
          MOV     @(R5)+,R4         ;MINVALUE TO COMPARE IN R4
          MOV     (R5),R5           ;START ADRESS OF OUTVECTOR
          TST     R2
          BNE     DLOP              ;VERTICAL DEFLECTION
          TST     LINC              ;TEST IF ONLY ONE LINE
          BEQ     ONLY              ;ONLY ONE VALUE
          MOV     DISTY,R1          ;DISTY IN R1
          MOV     LINC,R2           ;R2 AS LOOPCOUNTER
VERT:     TSTB    @#ADSR            ;CHECK DONE FLAG
          BPL     VERT              ;WAIT UNTIL SET
          MOV     @#<ADSR+2>,R0     ;PUT THE AD VALUE IN R0
          INC     @#ADSR            ;TAKE FIRST SAMPEL AND AD-CONVERT
          ADD     R1,@#<CSRB+2>;NEW DEFL POINT
          DEC     R2
          BEQ     SAMPON
```

```
VRTLOP:TSTB     @#ADSR          ;CHECK DONE FLAG
       BPL      VRTLOP          ;WAIT UNTIL SET
       MOV      @#<ADSR+2>,R0   ;PUT THE PREVIOUS AD VALUE IN R0
       INCB     @#ADSR          ;TAKE SAMPEL AND DA-CONVERT
       MOV      R0,(R5)+        ;PUT VALUE INTO BUFFER
       ADD      R1,@#<CSRB+2>   ;ADRESS NEXT XPOINT
       CMP      R0,R3           ;CMP TO PREVIOUS MAX(R3)
       BLO      MINVA           ;IF NOT GREATER BRANCH
       MOV      R0,R3
MINVA: CMP      R0,R4    ;CMP TO PREVIOUS MIN
       BGT      VLOPE           ;IF GREATER BRANCH
       MOV      R0,R4           ;R4=MIN
VLOPE: SOB      R2,VRTLOP
       BR       SAMPON

DLOP:  TSTB     @#ADSR          ;CHECK DONE FLAG
       BPL      DLOP            ;WAIT UNTIL SET
       MOV      @#<ADSR+2>,R0   ;PUT THE AD VALUE IN R0
       INC      @#ADSR          ;TAKE FIRST SAMPEL AND AD-CONVERT
       ADD      R1,@#<CSRA+2>;NEW DEFL POINT
       DEC      R2
       BNE      DOTLOP
       TST      LINC
       BEQ      SAMPON
       BR       CHVERT


DOTLOP: TSTB    @#ADSR          ;CHECK DONE FLAG
       BPL      DOTLOP          ;WAIT UNTIL SET
       MOV      @#<ADSR+2>,R0   ;PUT THE PREVIOUS AD VALUE IN R0
       INCB     @#ADSR          ;TAKE SAMPEL AND DA-CONVERT
       MOV      R0,(R5)+        ;PUT VALUE INTO BUFFER
       ADD      R1,@#<CSRA+2>   ;ADRESS NEXT XPOINT
       CMP      R0,R3           ;CMP TO PREVIOUS MAX(R3)
       BLO      MINVB           ;IF NOT GREATER BRANCH
       MOV      R0,R3
MINVB: CMP      R0,R4    ;CMP TO PREVIOUS MIN
       BGT      DLOPE           ;IF GREATER BRANCH
       MOV      R0,R4           ;R4=MIN
DLOPE: SOB      R2,DOTLOP
       TST      LINC
       BEQ      SAMPON

CHVERT: TSTB    @#ADSR          ;CHECK DONE FLAG
       BPL      CHVERT          ;WAIT UNTIL SET
       MOV      @#<ADSR+2>,R0   ;PUT THE PREVIOUS AD VALUE IN R0
       INCB     @#ADSR          ;TAKE SAMPEL AND DA-CONVERT
       MOV      STARTX,@#<CSRA+2>;FIRST XDEFLADR
       ADD      DISTY,@#<CSRB+2>;ADRESS NEXT LINE
       DEC      LINC
       MOV      NUMADP,R2
       MOV      R0,(R5)+        ;PUT VALUE INTO BUFFER
       CMP      R0,R3           ;CMP TO PREVIOUS MAX(R3)
       BLO      MINVC           ;IF NOT GREATER BRANCH
```

```
        MOV     R0,R3
MINVC:  CMP     R0,R4    ;CMP TO PREVIOUS MIN
        BGT     DOTLOP           ;IF GREATER BRANCH
        MOV     R0,R4            ;R4=MIN
        BR      DOTLOP

ONLY:   TSTB    @#ADSR           ;CHECK DONE FLAG
        BPL     ONLY             ;WAIT UNTIL SET
        MOV     @#<ADSR+2>,R0    ;PUT THE PREVIOUS AD VALUE IN R0
        INCB    @#ADSR           ;TAKE SAMPEL AND DA-CONVERT
        BR      LAST

SAMPON: TSTB    @#ADSR           ;CHECK DONE FLAG
        BPL     SAMPON           ;WAIT UNTIL SET
        MOV     @#<ADSR+2>,R0    ;PUT THE PREVIOUS AD VALUE IN R0
        INCB    @#ADSR           ;TAKE SAMPEL AND DA-CONVERT
        MOV     R0,(R5)+         ;PUT VALUE INTO BUFFER
        CMP     R0,R3            ;CMP TO PREVIOUS MAX(R3)
        BLO     MINVD            ;IF NOT GREATER BRANCH
        MOV     R0,R3
MINVD:  CMP     R0,R4    ;CMP TO PREVIOUS MIN
        BGT     LAST             ;IF GREATER BRANCH
        MOV     R0,R4            ;R4=MIN

LAST:   TSTB    @#ADSR           ;CHECK DONE FLAG
        BPL     LAST             ;WAIT UNTIL SET
        MOV     @#<ADSR+2>,R0    ;PUT THE PREVIOUS AD VALUE IN R0
        MOV     R0,(R5)+         ;PUT VALUE INTO BUFFER
        CMP     R0,R3            ;CMP TO PREVIOUS MAX(R3)
        BLO     MINVE            ;IF NOT GREATER BRANCH
        MOV     R0,R3
MINVE:  CMP     R0,R4    ;CMP TO PREVIOUS MIN
        BGT     END              ;IF GREATER BRANCH
        MOV     R0,R4            ;R4=MIN

END:    MOV     R3,@MAXADR       ;MAXVALUE OUT
        MOV     R4,@MINADR       ;MINVALUE OUT
        MOV     #100000,@#<CSRA+2>;LEAST STRESSING DEFL
        MOV     #100000,@#<CSRB+2>;LEAST STRESSING DEFL
        RTS     PC               ;END OF SUBROUTINE


STARTX: .WORD
NUMADF: .WORD
DISTY:  .WORD
LINC:   .WORD
MAXADR: .WORD
MINADR: .WORD
        .EVEN
        .END SAVDEF
```

```
        .TITLE DISPLY.MAC
;+THIS PROGRAMPACK MAY BE CALLED TO DISPLAY A PICTURE
;ON THE VIDEOMONITOR
;FIRST FILL THE COLORVECTOR(COLOR TO BE DISPLAYED IF
;VALUES ARE WITHIN LIMITS)WITH
;CALL DISCOL(COLCODETAB)
;       COLCODETAB=A 16WORD VECTOR(INTEGER) OF COLORCODE 0..15
;       THE COLORVECTOR WILL REMAIN SO UNTIL CHANGED WITH ANOTHER
;       CALL
;THEN FILL THE LIMITVECTOR WITH
;CALL DISLIM(LIMITS)
;       LIMITS=A 17WORD VECTOR(INTEGER) OF LIMITS TO BE USED TO DESIDE
;       WICH WORD IN THE COLORVECTOR TO PUT OUT ON THE VIDEOMONITOR
;TO START THE DISPLAY THEN
;CALL DISPLY(STARTMONX,STARTMONY,NUMADP,NUMLIN,VALUEVECTOR)
;       STARTMONX=STARTPOINT ON THE MONITOR X-COORDINATE BETWEEN
;       0:LEFT...255:RIGHT
;       STARTMONY=STARTPOINT ON THE MONITOR Y-COORDINATE BETWEEN
;       0:TOP...255:BOTTOM
;       NUMADP=NUMBER OF ADRESSED POINT IN ONE LINE
;       NUMLIN=NUMBER OF LINES TO SCAN
;       VALUEVECTOR=THE VALUES TO CODE AND DISPLAY
;       VALUES LESS THEN LIMITS(17) OR GREATER EQUAL
;       LIMITS(1) IN LIMVECTOR WILL NOT BE DISPLAYED
;       THE SPECIFIC DOTS WILL REMAIN AS THEY WHERE BEFORE
;       (ALL ARGUMENTS INTEGER)
;-
        COLADR=170740     ;ADRESS TO QRGB:S COLORBYTE
        ADR=170742        ;ADRESS TO QRGB:S DOTADRESSWORD
        A=123456          ;DUMMYVALUE
        C=165432          ;DUMMYVALUE

DISPLY::TST     (R5)+             ;SKIP ARGUMENT COUNT
        MOV     @(R5)+,STX
        MOVB    STX,DOTADR
        MOVB    @(R5)+,DOTADR+1
        MOV     DOTADR,R4
        MOV     @(R5)+,NADP       ;NUMBER OF ADRESSINGPOINTS
        MOV     NADP,R2
        MOV     @(R5)+,R3         ;NUMBER OF LINES INTO R1
        MOV     (R5)+,R1          ;ARGUMENTVECTORSTART INTO R1
        BR      LOP
DOTLOP: INC     R4
LOP:    MOV     (R1)+,R0          ;NEXT VECTORVALUE INTO R0
        MOV     R4,@#ADR          ;PUT OUT ADRESS TO THE QRGB
L9:     CMP     R0,#A             ;COMP IF GREATER THEN CENTERVALUE
        BGE     L5
L13:    CMP     R0,#A             ;COMP IF GREATER THEN LOWER QUARTER
        BGE     L11               ;VALUE
L15:    CMP     R0,#A             ;COMP IF GREATER THEN LOWER LOWER
        BGE     L14               ;EIGHTH VALUE
L16:    CMP     R0,#A             ;CMP IF LESS THEN LOWER LOWER LOWER
        BLO     L17               ;16:TH VALUE BRANCH IF LESS
C15:    MOVB    #C,@#COLADR       ;PUT OUT COLORCODE BETWEEN LIN 15,16
        SOB     R2,DOTLOP         ;IF NOT END OF LINE DO THE NEXT DOT
```

```
        BR      B9
C7:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 7..8
        SOB     R2,DOTLOP       ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C11:    MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 11..12
        SOB     R2,DOTLOP       ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C13:    MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 13..14
        SOB     R2,DOTLOP       ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C3:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 3..4
        SOB     R2,DOTLOP       ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C5:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 5..6
        SOB     R2,DOTLOP       ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C9:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 9..10
        SOB     R2,DOTLOP       ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
L17:    CMP     R0,#A           ;CMP IF GREATER THEN LOWER LIMIT
        BLO     B1              ;IF LESS NO COLOR OUTPUT
C16:    MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 16,17
B1:     SOB     R2,DOTLOP       ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
L14:    CMP     R0,#A           ;CMP IF GREATER THEN LOWER LOWER
        BGE     C13             ;UPPER 16:TH LIMITVALUE
C14:    MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 14..15
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BNE     DOTLOP
        BR      B9              ;EOFLINE
L6:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER LOWER
        BGE     C5              ;UPPER 16:TH LIMITVALUE
C6:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 6..7
        DEC     R2
        BNE     DOTLOP          ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
L10:    CMP     R0,#A           ;CMP IF GREATER THEN LOWER UPPER
        BGE     C9              ;UPPER 16:TH LIMITVALUE
C10:    MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 10..11
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BNE     DOTLOP
        BR      B9
L2:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER UPPER
        BGE     L1              ;UPPER 16:TH LIMIT VALUE
C2:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 2..3
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BNE     DOTLOP
        BR      B9
L1:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER UPPER
        BGE     B8              ;UPPER 16:TH VALUE LIMIT NO OUTPUT
C1:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 1..2
B8:     DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BNE     DOTLOP
        BR      B9
L11:    CMP     R0,#A           ;CMP IF GREATER THEN LOWER UPPER
        BGE     L10             ;EIGHTHLIMIT
```

```
L12:    CMP     R0,#A           ;CMP IF GREATER THEN LOWER UPPER
        BGE     C11             ;LOWER 16:TH LIMIT VALUE
C12:    MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 12..13
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BNE     DOTLOP
        BR      B9
L3:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER UPPER
        BGE     L2              ;EIGHTH LIMITVALUE
L4:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER UPPER
        BGE     C3              ;LOWER 16:TH LIMIT VALUE
C4:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 4..5
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BEQ     B9
        JMP     DOTLOP
L5:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER QUARTER
        BGE     L3              ;LIMIT
L7:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER LOWER
        BGE     L6              ;EIGHTH LIMITVALUE
L8:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER LOWER
        BGE     C7              ;LOWER 16:TH LIMIT
C8:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 8,9
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BEQ     B9
        JMP     DOTLOP
B9:     DEC     R3              ;DECREMENT LINECOUNTER
        BLE     LLINE           ;IF LESS OR EQUAL 0 EOF  DISPLAY
        INCB    <DOTADR+1>      ;NEXT MONITORLINE
        MOVB    STX,DOTADR
        MOV     DOTADR,R4
        MOV     NADP,R2         ;NUMBER OF DOTS IN ALINE INTO R2
        JMP     LOP             ;IF NOT THE LAST LINE JMP START
LLINE:  RTS     PC
DISCOL::TST     (R5)+           ;SKIP ARGUMENT COUNT
        MOV     (R5)+,R4        ;START ADRESS OF COLORTABLE IN R4
        MOV     (R4)+,<C1+2>    ;
        MOV     (R4)+,<C2+2>    ;COLOR TABLE
        MOV     (R4)+,<C3+2>    ;
        MOV     (R4)+,<C4+2>    ;THE COLORS CORRESPOND TO
        MOV     (R4)+,<C5+2>    ;THE LIMITS IN THE LIMITVECTOR
        MOV     (R4)+,<C6+2>    ;WITH FIRST COLORVECTORELEMENT
        MOV     (R4)+,<C7+2>    ;USED TOGETHER WITH VALUES
        MOV     (R4)+,<C8+2>    ;BETWEEN FIRST AND SECOND
        MOV     (R4)+,<C9+2>    ;LIMITVECTORELEMENT
        MOV     (R4)+,<C10+2>   ;AND SO FORTH
        MOV     (R4)+,<C11+2>   ;
        MOV     (R4)+,<C12+2>   ;
        MOV     (R4)+,<C13+2>   ;
        MOV     (R4)+,<C14+2>   ;
        MOV     (R4)+,<C15+2>   ;
        MOV     (R4),<C16+2>    ;
        RTS     PC              ;RETURN FROM DISCOL
DISLIM::TST     (R5)+           ;SKIP ARGUMENT COUNT
        MOV     (R5)+,R4        ;START ADRESS OF LIMITARRAY INTO R4
        MOV     (R4)+,<L1+2>    ;THIS IS A TRICK TO MAKE A FASTER
        MOV     (R4)+,<L2+2>    ;COMPARISION BY USING DIRECT MODE
        MOV     (R4)+,<L3+2>    ;AND ASSIGNING THE VALUES HERE BEFORE
```

```
        MOV     (R4)+,<L.4+2>     ;THE BEGINING OF THE PROGRAM
        MOV     (R4)+,<L.5+2>
        MOV     (R4)+,<L.6+2>     ;THE LIMITS ARE TO BE ASSIGNED IN
        MOV     (R4)+,<L.7+2>     ;DISCENDING ORDER WITH THE HIGHEST
        MOV     (R4)+,<L.8+2>     ;LIMIT IN VECTORPOSISION 1
        MOV     (R4)+,<L.9+2>
        MOV     (R4)+,<L.10+2>
        MOV     (R4)+,<L.11+2>
        MOV     (R4)+,<L.12+2>
        MOV     (R4)+,<L.13+2>
        MOV     (R4)+,<L.14+2>
        MOV     (R4)+,<L.15+2>
        MOV     (R4)+,<L.16+2>
        MOV     (R4),<L.17+2>
        RTS     PC                ;RETURN FROM DISLIM

STX:  .WORD
DOTADR:  .WORD
NADF:    .WORD
         .EVEN
         .END DISPLY
```

```
          .TITLE SEMPIC.MAC                                    PAGE     1


          .TITLE SEMPIC.MAC
;+THIS PROGRAM MAY BE CALLED TO TAKE UP A PICTURE
;AND DISPLAY IT ON THE VIDEOMONITOR
;CALL SEMPIC(DEFLSTARTX,DEFLSTARTY,DISTX,DISTY,NUMADP,NUMLIN,
;       LIMITS,COLCODETAB,STARTMONX,STARTMONY)
;       (ALL VALUES INTEGER)
;WHERE:DEFLSTARTX,DEFLSTARTY=STARTADRESS OF DEFLECTION
;       DISTX,Y=DISTANCES BETWEEN ADRESSED POINTS IN A SCAN(X,Y)
;       NUMADP=NUMBER OF ADRESSED POINT IN ONE LINE
;       NUMLIN=NUMBER OF LINES TO SCAN
;       LIMITS=A 17WORD VECTOR(INTEGER) OF LIMITS TO BE USED TO DESIDE
;       WICH WORD IN THE COLORVECTOR TO PUT OUT ON THE VIDEOMONITOR
;       LIMITS OVER 4095 AND UNDER 0 RESULT IN DISPLAY OF ALL
;       ADRESSED POINTS
;       COLCODETAB=A 16WORD VECTOR(INTEGER) OF COLORCODE 0..15
;       STARTMONX=STARTPOINT ON THE MONITOR X-COORDINATE BETWEEN
;       0:LEFT...255:RIGHT
;       STARTMONY=STARTPOINT ON THE MONITOR Y-COORDINATE BETWEEN
;       0:TOP...255:BOTTOM
;       THE PROGRAM ADRESSES DEFLECTION IN SAME WAY
;-
          COLADR=170740     ;ADRESS TO RRGB:S COLORBYTE
          ADR=170742        ;ADRESS TO RRGB:S DOTADRESSWORD
          CSRA=164160       ;ADRESS TO CONTROLWORD DRV11-J CHAN A
          CSRB=164164       ;ADRESS TO CONTROLWORD DRV11-J CHAN B
          ADSR=170400       ;ADRESS TO CONTROLWORD ADV-11 A

          A=123456          ;DUMMYVALUE
          C=165432          ;DUMMYVALUE


SEMPIC::  TST       (R5)+             ;SKIP ARGUMENT COUNT
          MOV       #400,@#CSRA       ;PUT PORT A IN WRITEMODE
          MOV       #400,@#CSRB       ;PUT PORT B IN WRITEMODE
          CLR       @#ADSR            ;MAKE AD-CONV FOLLOW CHAN 0
          MOV       @(R5),@#<CSRA+2>  ;LOAD START X VALUE INTO DRA
          MOV       @(R5)+,STARTX     ;PUT START VALUE INTO STARTX
          MOV       @(R5)+,@#<CSRB+2> ;LOAD START Y VALUE INTO DRB
          MOV       @(R5)+,R1         ;PUT POINT DISTANCE(X) INTO R1
          MOV       @(R5)+,DISTY      ;SAVE DEFLDIST Y IN DISTY
          MOV       @(R5)+,R2         ;PUT NUMBER OF POINTS IN A LINE
          CMP       R2,#2             ;IF NR DOTS LE 2 SPECIAL  SCAN
          BGT       SU2               ;IF GT NORMAL SCAN
          BEQ       VERTY1            ;IF EQ TWO DOTS SCAN
          CLR       VERTY             ;R2=1, VERTY=0 (VERTICAL SCAN)
          MOV       #1,JUMPY          ;SPECIAL SCAN
          BR        SU3               ;BRANCH TO MAKE JUMPY=1
VERTY1:   MOV       #-1,VERTY         ;R2=2 (2 DOTS LINE)
          MOV       #1,JUMPY          ;SPECIAL SCAN
          BR        SU3

SU2:      SUB       #2,R2             ;DEFLEC.CHANGE TWO POINTS BEFORE EOL
          MOV       #-1,JUMPY         ;MAKE DEFL CHANGE FIRST NORMAL SCAN
SU3:      MOV       R2,NUMADP         ;INTO R2 AND NUMADP
          MOV       @(R5)+,R3         ;PUT NUMBER OF LINES INTO R3
```

```
        MOV     (R5)+,R4            ;START ADRESS OF LIMITARRAY INTO R4
        MOV     (R4)+,<L1+2>        ;THIS IS A TRICK TO MAKE A FASTER
        MOV     (R4)+,<L2+2>        ;COMPARISION BY USING DIRECT MODE
        MOV     (R4)+,<L3+2>        ;AND ASSIGNING THE VALUES HERE IN
        MOV     (R4)+,<L4+2>        ;THE BEGINING OF THE PROGRAM
        MOV     (R4)+,<L5+2>
        MOV     (R4)+,<L6+2>        ;THE LIMITS ARE TO BE ASSIGNED IN
        MOV     (R4)+,<L7+2>        ;DISCENDING ORDER WITH THE HIGHEST
        MOV     (R4)+,<L8+2>        ;LIMIT IN VECTORPOSISION 1
        MOV     (R4)+,<L9+2>
        MOV     (R4)+,<L10+2>
        MOV     (R4)+,<L11+2>
        MOV     (R4)+,<L12+2>
        MOV     (R4)+,<L13+2>
        MOV     (R4)+,<L14+2>
        MOV     (R4)+,<L15+2>
        MOV     (R4)+,<L16+2>
        MOV     (R4),<L17+2>
        INCB    @#ADSR              ;TAKE FIRST SAMPEL AND AD-CONVERT
        MOV     (R5)+,R4            ;START ADRESS OF COLORTABLE IN R4
        TST     JUMPY               ;IF JUMPY=-1 (MORE THEN 2 DOTS IN ALINE)
        BLE     NORM                ;NORMAL SCAN
        TST     VERTY               ;TEST IF VERTY
        BNE     NMDEC               ;IF VERTY NOT EQ 0(2 DOT LINE) ADRESS NEXT
        DEC     R3                  ;ONE MORE LINE TO ADRESS IF GT 0
        BLE     NMADR               ;NO FURTHER ADRESS
        ADD     DISTY,@#<CSRB+2>    ;ADRESS NEXT LINE
        BR      NMADR
NMDEC:  DEC     R3                  ;FIRST LINE SCANNED
NORM:   ADD     R1,@#<CSRA+2>       ;ADRESS NEXT XPOINT IN SCAN
NMADR:  MOV     (R4)+,<C1+2>        ;SAME TRICK AS ABOVE WITH THE
        MOV     (R4)+,<C2+2>        ;COLOR TABLE
        MOV     (R4)+,<C3+2>        ;
        MOV     (R4)+,<C4+2>        ;THE COLORS CORRESPOND TO
        MOV     (R4)+,<C5+2>        ;THE LIMITS IN THE LIMITVECTOR
        MOV     (R4)+,<C6+2>        ;WITH : IRST COLORVECTORELEMENT
        MOV     (R4)+,<C7+2>        ;USED TOGETHER WITH VALUES
        MOV     (R4)+,<C8+2>        ;BETWEEN FIRST AND SECOND
        MOV     (R4)+,<C9+2>        ;LIMITVECTORELEMENT
        MOV     (R4)+,<C10+2>       ;AND SO FORTH
        MOV     (R4)+,<C11+2>       ;
        MOV     (R4)+,<C12+2>       ;
        MOV     (R4)+,<C13+2>       ;
        MOV     (R4)+,<C14+2>       ;
        MOV     (R4)+,<C15+2>       ;
        MOV     (R4),<C16+2>        ;
        MOVB    @(R5)+,STMON        ;STARTADRESS X MONITOR IN LOW BYTE
        MOVB    @(R5)+,<STMON+1>    ;STARTADRESS Y MONITOR IN HIGH BYTE
        MOV     STMON,R4            ;FIRST MONITORADRESS IN R4
        MOV     @#<ADSR+2>,R0       ;PUT THE AD VALUE IN R0
        TST     JUMPY               ;TEST IF NORMAL SCAN
        BLE     NMSC
        TST     R3
        BLT     BL9

NMSC:   INCB    @#ADSR              ;MAKE NEW SAMPEL
```

```
        MOV     DISTY,R5          ;PUT DEFLDISTANCE Y INTO R5
        TST     JUMPY
        BLE     START
        TST     VERTY
        BEQ     VRTSCA
        TST     R3
        BEQ     BRBL9
        MOV     STARTX,@#<CSRA+2>;
        ADD     R5,@#<CSRB+2>     ;
        BR      BL9
BRBL9:  MOV     #1,R2
        BR      BL9
VRTSCA: TST     R3
        BEQ     BL9
        ADD     R5,@#<CSRB+2>;
        BR      BL9
DOTLOP: MOV     @#<ADSR+2>,R0     ;PUT THE PREVIOUS AD VALUE IN R0
        INCB    @#ADSR            ;TAKE SAMPEL AND DA-CONVERT
        INC     R4                ;INCREMENT X-COORDINATE TO DISPLAY
START:  ADD     R1,@#<CSRA+2>     ;ADRESS NEXT XPOINT
BL9:    MOV     R4,@#ADR          ;PUT OUT ADRESS TO THE QRGB
L9:     CMP     R0,#A             ;COMP IF GREATER THEN CENTERVALUE
        BGE     L5
L13:    CMP     R0,#A             ;COMP IF GREATER THEN LOWER QUARTER
        BGE     L11               ;VALUE
L15:    CMP     R0,#A             ;COMP IF GREATER THEN LOWER LOWER
        BGE     L14               ;EIGHTH VALUE
L16:    CMP     R0,#A             ;CMP IF LESS THEN LOWER LOWER LOWER
        BLO     L17               ;16:TH VALUE BRANCH IF LESS
C15:    MOVB    #C,@#COLADR       ;PUT OUT COLORCODE BETWEEN LIM 15.16
        SOB     R2,DOTLOP         ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C7:     MOVB    #C,@#COLADR       ;PUT OUT COLORCODE BETWEEN LIM 7..8
        SOB     R2,DOTLOP         ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C11:    MOVB    #C,@#COLADR       ;PUT OUT COLORCODE BETWEEN LIM 11..12
        SOB     R2,DOTLOP         ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C13:    MOVB    #C,@#COLADR       ;PUT OUT COLORCODE BETWEEN LIM 13..14
        SOB     R2,DOTLOP         ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C3:     MOVB    #C,@#COLADR       ;PUT OUT COLORCODE BETWEEN LIM 3..4
        SOB     R2,DOTLOP         ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C5:     MOVB    #C,@#COLADR       ;PUT OUT COLORCODE BETWEEN LIM 5..6
        SOB     R2,DOTLOP         ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
C9:     MOVB    #C,@#COLADR       ;PUT OUT COLORCODE BETWEEN LIM 9..10
        SOB     R2,DOTLOP         ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
L17:    CMP     R0,#A             ;CMP IF GREATER THEN LOWER LIMIT
        BLO     B1                ;IF LESS NO COLOR OUTPUT
C16:    MOVB    #C,@#COLADR       ;PUT OUT COLORCODE BETWEEN LIM 16.17
B1:     SOB     R2,DOTLOP         ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
L14:    CMP     R0,#A             ;CMP IF GREATER THEN LOWER LOWER
```

```
        BGE     C13             ;UPPER 16:TH LIMITVALUE
C14:    MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 14.,15
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BNE     DOTLOP
        BR      B9              ;EOFLINE
L6:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER LOWER
        BGE     C5              ;UPPER 16:TH LIMITVALUE
C6:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 6.,7
        DEC     R2
        BNE     DOTLOP          ;IF NOT END OF LINE DO THE NEXT DOT
        BR      B9
L10:    CMP     R0,#A           ;CMP IF GREATER THEN LOWER UPPER
        BGE     C9              ;UPPER 16:TH LIMITVALUE
C10:    MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 10.,11
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BNE     DOTLOP
        BR      B9
L2:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER UPPER
        BGE     L1              ;UPPER 16:TH LIMIT VALUE
C2:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 2.,3
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BNE     DOTLOP
        BR      B9
L1:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER UPPER
        BGE     B8              ;UPPER 16:TH VALUE LIMIT NO OUTPUT
C1:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 1.,2
B8:     DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BNE     DOTLOP
        BR      B9
L11:    CMP     R0,#A           ;CMP IF GREATER THEN LOWER UPPER
        BGE     L10             ;EIGHTHLIMIT
L12:    CMP     R0,#A           ;CMP IF GREATER THEN LOWER UPPER
        BGE     C11             ;LOWER 16:TH LIMIT VALUE
C12:    MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 12.,13
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BNE     DOTLOP
        BR      B9
L3:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER UPPER
        BGE     L2              ;EIGHTH LIMITVALUE
L4:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER UPPER
        BGE     C3              ;LOWER 16:TH LIMIT VALUE
C4:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 4.,5
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BEQ     B9
        JMP     DOTLOP
L5:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER QUARTER
        BGE     L3              ;LIMIT
L7:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER LOWER
        BGE     L6              ;EIGHTH LIMITVALUE
L8:     CMP     R0,#A           ;CMP IF GREATER THEN UPPER LOWER
        BGE     C7              ;LOWER 16:TH LIMIT
C8:     MOVB    #C,@#COLADR     ;PUT OUT COLORCODE BETWEEN LIM 8.9
        DEC     R2              ;IF NOT END OF LINE DO THE NEXT DOT
        BEQ     B9
        JMP     DOTLOP
B9:     TST     JUMPY           ;TEST IF END OF LINE FOR DEFL OR MONITOR
```

```
        BEQ     B10             ;BRANCH IF END OF LINE FOR MONITOR
        BGT     LT2D
        ADD     #2,R2           ;TWO DOTS BEFORE CHANGE MONITORLINE
        MOV     @#<ADSR+2>,R0   ;PUT FORMER ADCONVERSION INTO R0
        INCB    @#ADSR          ;SAMPLE AND ADCONVERT
        INC     JUMPY           ;NEXT LOOP JMP TO END OF MONITORLINE
        INC     R4              ;NEXT XADRESS ON MONITOR
        CMP     #1,R3           ;TEST IF LAST LINE
        BEQ     B11             ;IF SO NO MORE DEFLECTION
        MOV     STARTX,@#<CSRA+2>;START A NEW DEFLECTION LINE
        ADD     R5,@#<CSRB+2>   ;ADRESS THE NEW Y COORDINATE
B11:    JMP     BL9             ;START A NEW COLOR CONVERSION
B10:    MOV     @#<ADSR+2>,R0   ;FORMER ADCONVERSION INTO R0
        INCB    @#ADSR          ;SAMPLE AND ADCONVERT
        INCB    <STMON+1>       ;NEXT MONITORLINE
        MOV     STMON,R4        ;FIRST X-POSITION
        MOV     NUMADP,R2       ;NUMBER OF DOTS IN ALINE INTO R2
        DEC     JUMPY           ;AT THE END OF LINE CHANGE DEFL FIRST
        DEC     R3
        BEQ     B12
        JMP     START    ;IF NOT THE LAST LINE JMP START
B12:    TSTB    @#ADSR          ;TEST IF LAST VALUE CONVERTED
        BPL     B12
        MOV     @#<ADSR+2>,R0   ;READ LAST VALUE TO EMPTY ADBUFF
        JMP     END2            ;JUMP TO RETURN FROM SUBROUTINE


LT2D:   TST     VERTY           ;TEST IF END OF LINE FOR DEFL OR MONITOR
        BLT     DOT2
        DEC     R3
        BLE     LLIN
        MOV     @#<ADSR+2>,R0   ;PUT FORMER ADCONVERSION INTO R0
        INCB    @#ADSR          ;SAMPLE AND ADCONVERT
        INCB    <STMON+1>       ;NEXT XADRESS ON MONITOR
        MOV     STMON,R4
        ADD     R5,@#<CSRB+2>
        INC     R2              ;VERTICAL LINE
        JMP     BL9             ;START A NEW COLOR CONVERSION
LLIN:   CMP     #-1,R3
        BGT     END1
        BEQ     PUTR0
        INCB    @#ADSR          ;SAMPLE BEFORE READ (SEE INTERFACE-HANDBOOK)
PUTR0:  MOV     @#<ADSR+2>,R0
        INCB    <STMON+1>
        MOV     STMON,R4
        INC     R2
        JMP     BL9
END1:   JMP     END2            ;JUMP TO RETURN FROM SUBROUTINE

DOT2:   DEC     R3
        BLE     LSTLIN
        MOV     @#<ADSR+2>,R0   ;FORMER ADCONVERSION INTO R0
        INCB    @#ADSR          ;SAMPLE AND ADCONVERT
        INCB    <STMON+1>       ;NEXT MONITORLINE
        MOV     STMON,R4        ;FIRST X-POSITION
        MOV     #2,R2           ;NUMBER OF DOTS IN ALINE INTO R2
```

```
        MOV     STARTX,@#<CSRA+2>;START A NEW DEFLECTION LINE
        ADD     R5,@#<CSRB+2>    ;ADRESS THE NEW Y COORDINATE
        JMP     BL9              ;IF NOT THE LAST LINE JMP START
LSTLIN: CMP     #-1,R3           ;LASTLINE
        BGT     END2
        BEQ     LSTDO
        MOV     @#<ADSR+2>,R0
        INCB    <STMON+1>
        MOV     STMON,R4
        INCB    @#ADSR
        INC     R2
        JMP     BL9

LSTDO:  MOV     @#<ADSR+2>,R0
        INC     R4
        INC     R2
        JMP     BL9
END2:   MOV     #77777,@#<CSRA+2>;DEFL-AMP INLEAST STRESSED POSITION
        MOV     #77777,@#<CSRB+2>;DEFL-AMP IN.EAST STRESSED POSITION
        RTS     PC

STARTX: .WORD
NUMADP: .WORD
STMON:  .WORD
JUMPY:  .WORD
DISTY:  .WORD
VERTY:  .WORD
        .EVEN
        .END SEMPIC
```

```
    .TITLE SVISUB.MAC
;+THIS SUBROUTINE TAKES THE PICTURE DISPLAYED ON THE
;VIDEOMONITOR PACKS 4 DOTVALUES OF FOUR BITS IN ONE WORD
;AND STORES IT IN A FILE CREATED BY THE USER
;CALL SVISUB(ICHAN,BLOSTR,MES)
;  ICHAN=CHANNEL OF FILE
;  BLOSTR=NR ON FIRST BLOCK TO WRITE
;  MES=ERRORMESSAGE LT 0 IF ERROR
;    =NUMBER ON LAST BLOCK WRITTEN ON NORMAL RETURN
;    =0 IF TRIED TO WRITE PAST END OF FILE
;    =-1 IF HARDWAREERROR (WRITE)
;    =-2 CHANNEL NOT OPENED(WRITE)
;    =-3 -------""-------(WAIT)
;    =-4 HARDWAREERROR(WAIT)
;    =-5 WORDS READ NOT EQUAL EXPECTED
;-
        .MCALL .WAIT,.WRITE

        NUMBL=UD64              ;NUMBER OF BLOCKS
        BFZ=UD256               ;NUMBER OF WORDS IN BUFFERT
        BUFFZ=UD512             ;BUFFERTSIZE=512BYTES
        ADR=170740              ;BASEADRESS TO ORGR
        ERRBYT=52               ;ERRORMESSAGELOCATION

SVISUB::TST   (R5)+             ;SKIP ARGUMENT COUNTER
        MOVB @(R5)+,AREA        ;OUT CHAN IN AREA
        MOV @(R5)+,BLOSTR       ;NR ON FIRST BLOCK IN BLOSTRT
        MOV (R5)+,MES           ;MESSAGEADRESS IN MES


TSTART: CLR R3                  ;CLEAR ADRESSCOUNTER
        CLR R4                  ;CLEAR WORKREGISTER
        CLR R5                  ;CLEAR BLOCKCOUNTER
BUF1L:: MOV #BUF1,BUFF          ;PUT BUFFERTSTARTADRESS IN BUFF
BUF2L:: MOV BUFF,R1             ;PUT FIRST BUFFADRESS INTO R1
        MOV #BUFFZ,R2           ;INIT NUMBER OF BYTES IN R2
BLOCK:: MOV R3,@#<ADR+2>        ;PUT OUT THE POINTADRESS
        MOVB @#ADR,R4           ;READ VALUEBYTE OOOOMMMM 4 SIGNIFICANT BITS
        ASH #4,R4               ;SHIFT 4 TO THE LEFT
        INC R3                  ;INCREMENT POINTADRESS
        MOV R3,@#<ADR+2>        ;PUT OUT THE POINTADRESS
        BISB @#ADR,R4           ;"OR" THE VALUE TO THE PRIFVIOUS SHIFTED
        MOVB R4,(R1)+           ;WRITE INTO BUFFER AND INCREMENT BUFFADR
        INC R3                  ;INCREMENT POINTADRESS
        SOB R2,BLOCK            ;DECREMENT NUMBER OF BYTES AND BRANCH IF 0
        .WAIT AREA              ;WAIT FOR OUTPUT
        BCC 11$                 ;BRANCH IF OK
        TSTB  @#ERRBYT          ;CHECK ERRBYT
        BEQ  10$                ;BRANCH IF CHANNEL NOT OPEN
        MOV #-4,@MES            ;HARDWAREERROR(WAIT)
        RTS PC
10$::   MOV #-3,@MES            ;CHANNEL NOT OPEN
        RTS PC
11$::   .WRITE #AREA            ;WRITE NEXT BLOCK
        BCC 13$                 ;BRANCH IF OK
        CMPB #1,@#ERRBYT        ;COMPARE ERRBYT
```

```
        BLE  12$
        CLR  @MES              ;TRIED TO WRITE PAST END OF FILE
        RTS  PC
12$:    BEQ  14$
        MOV  $-2,@MES          ;CHANNEL NOT OPENED
        RTS  PC
14$:    MOV  $-1,@MES          ;HARDWARE ERROR
        RTS  PC
13$:    CMP  WCNT,R0           ;TEST IF CORRECT NUMBER OF WORDS WRITTEN
        BNE  END
        INC  R5                ;INCREMENT BLOCKCOUNTR
        CMP  R5,#NUMBL         ;COMPARE WRITTEN BLOCKS
        BEQ  END1              ;SUCCESSFULL END
        INC  BLOSTR            ;NEXT BLOCKNR TO READ
        CMP  BUFF,#BUF2        ;SEE IF WRITTEN IN BUFF2
        BEQ  BUF1L             ;IF BRANCH TO BUFFLABEL 1
        MOV  #BUF2,R1          ;PUT STARTADRESS OF SECOND BUFF INTOR1
        MOV  #BUF2,BUFF        ;PUT STARTADRESS OF SECOND BUFF INTO BUFF
        JMP  BUF2L             ;JUMP NEXT LOOP
END:    MOV  $-5,@MES          ;WORDS TO READ NOT EQUAL TO BE READ
        RTS  PC
END1:   MOV  BLOSTR,@MES       ;NUMBER ON LAST BLOCK IN MES
        RTS  PC                ;RETURN
AREA:   .BYTE                  ;CHANNEL NR PUT HERE
        .BYTE   11             ;EMT ARGUMENTBLOCK(5 WORD)
BLOSTR: .WORD                  ;BLOCK NR TO WRITE HERE
BUFF:   .WORD   0              ;MEMORYBUFFERADRESS
WCNT:   .WORD  D256            ;WORDS TO BE WRITTEN
        .WORD  1
        .EVEN
MES:    .WORD
BUF1:   .BLKW  D256            ;FIRST OUTPUTBUFFAREA
BUF2:   .BLKW  D256            ;SECOND OUTBUFFBUFFAREA
        .END   SVISUB
```

```
        .TITLE VIDSUB.MAC
        ;+THIS SUBROUTINE TAKES THE CONTENTS SPECIFIED
        ; BY THE CHANNELNUMBER UNPACKS IT
        ;AND DISPLAYS IT ON THE VIDEOMONITOR
        ;CALL VIDSUB(CHAN,BLOSTR,ERMES)
        ; CHAN=CHANNEL WITH DATA TO DISPLAY
        ; BLOSTR=NR ON FIRST BLOCK TO READ
        ; ERMES=ERRORMESSAGE
        ;  =IF NOERROR NUMBER ON LAST BLOCK READ
        ;  =
        ;-
        .MCALL .READ,.WAIT

        ERRBYT=52               ;ERRORMESSAGELOCATION
        BUFFZ=^D512             ;NUMBER OF BYTES TO READ IN A BLOCK
        NUMBL=^D64              ;NUMBER OF BLOCKS TO READ
        ADR=170740              ;ADRESS OF QRQR INTERFACE
        BFZ=^D256               ;WORDS IN ONE BLOCK
VIDSUB::TST     (R5)+           ;SKIP ARGUMENT COUNT
        MOVB @(R5)+,AREA        ;CHANNEL NR IN EMTBLOCK
        MOV @(R5)+,BLOSTR       ;FIRST BLOCK TO READ
        MOV (R5),MES            ;ERRORMESSAGE ADRESS IN MES

        CLR     R5              ;CLEAR BLOCKCOUNTER
        CLR     R3              ;START DISPLAY ADRESS 0
        MOV     #BUF1,BUFF      ;PUT STARTADRESS OF FIRSTBUFFER INTO BUFF
        .WAIT AREA              ;CHECK CHANNEL
        BCC     L7              ;BRANCH IF NO ERROR
WAITER: TSTB    @#ERRBYT        ;TEST ERROR
        BEQ     5$              ;BRANCH TO PRINT CHANNEL NOT OPEN
        MOV #-4,@MES            ;HARDWAREERROR!
        RTS PC
5$:     MOV #-3,@MES            ;CHANNEL NOT OPEN
        RTS PC
L7:     .READ #AREA             ;READ FIRST BLOCK
        BCC  OK                 ;BRANCH IF NO ERROR
RDERR:  CMPB #1,@#ERRBYT        ;CHECK ERRORMESSAGE
        BLE     2$
        CLR @MES                ;TRIED TO READ PAST END OF FILE
        RTS PC
2$:     BLO 3$
        MOV #-1,@MES            ;HARDWAREERROR
        RTS PC
3$:     MOV #-2,@MES            ;CHANNEL NOT OPENED
        RTS PC
OK:     CMP #BFZ,R0
        BEQ     BIN1
BIN2:   MOV #-5,@MES            ;UNCORRECT NUMBER OF WORDS READ
        RTS PC
BIN1:   INC     R5              ;INCREMENT BLOCKCOUNTER
        INC BLOSTR
        MOV     BUFF,R1         ;PUT BUFFSTARTADRESS IN R1
BUF2L:  MOV     #BUF2,BUFF      ;PUT STARTADRESS OF SECONDBUFF INTO BUFF
BUF1L:  .WAIT AREA              ;WAIT TO BE READ
        BCS  WAITER             ;BRANCH IF ERROR
4$:     .READ #AREA             ;READ NEXT BLOCK
```

```
        BCS     RDERR           ;BRANCH IF ERROR
BINE:   CMP #BFZ,R0
        BNE BIN2                ;IF UNCORRECT BLOCK READ RETURN
LBLOC:  MOV     #BUFFZ,R2       ;NUMBER OF BYTES IN R2
BLOCK:  MOVB    (R1),R4         ;READ BYTE
        ASH     #-4,R4          ;SHIFT 4 STEPS RIGHT
        MOV     R3,@#<ADR+2>    ;PUT OUT ADRESS
        MOVB    R4,@#ADR        ;PUT OUT DOTVALUE ----DDDD
        INC     R3              ;INCREMENT ADRESS
        MOV     R3,@#<ADR+2>    ;PUT OUT ADRESS
        MOVB    (R1)+,@#ADR     ;PUT OUT DOTVALUE AND INCREMENT
                                ;BYTEPOINTER
        INC     R3              ;INC ADRESS
        SOB     R2,BLOCK        ;DECREMENT BYTECOUNTER AND BRANCH IF
                                ;NOT ZERO
        TST     R3              ;TEST IF ALL DISPLAYED
        BEQ     12$             ;IF ZERO ALL DOTS DISPLAYED
        MOV     BUFF,R1         ;PUT THE READ BUFFSTARTADRESS TO
                                ;STARTADRESS OF NEXT BLOCK TO PUTOUT
        INC     R5              ;ONE MORE BLOCK READ
        CMP     R5,#NUMBL       ;TEST IF ALL BLOCKS READ
        BNE     10$
        .WAIT AREA              ;WAIT TO BE READ
        BCS  WAITER             ;BRANCH IF ERROR
        BR      LBLOC           ;IF SO DISPLAY THE LAST BLOCK
10$:    INC     BLOSTR          ;INC BLOCK TO READ
        CMP     BUFF,#BUF1      ;WAS THE LAST BLOCK READ INTO BUF1
        BEQ     BUF2L           ;BRANCH TO READ INTO BUF2
        MOV     #BUF1,BUFF      ;PUT BUF1 TO STARTADRESS OF NEXT BLOCK
        JMP     BUF1L           ;BRANCH TO READ INTO BUFF1
12$:    MOV BLOSTR,@MES         ;LAST BLOCK READ IN MESSAGE
        RTS     PC
AREA:   .BYTE           ;CHANNEL BYTE HERE
        .BYTE 10        ;EMT ARE/ BLOCK
BLOSTR: .WORD           ;BLOCK NR
BUFF:   .WORD           ;BUFFERT AREA ADRESS
WCNT:   .WORD 0D256     ;WORD IN BLOCK
        .WORD 1
        .EVEN
MES:    .WORD
BUF1:   .BLKW 0D256
BUF2:   .BLKW 0D256
        .END  VIDSUB
```

```
          .TITLE  MOVPIC
          .TITLE  MOVPIC
;+THIS MACRO MAY BE CALLED TO MOVE WHAT IS DISPLAYED
;ON THE MONITOR
;THE PROGRAM DOES NOT CHECK ANYTHING
;  CALL MOVPIC(CORNX,CORNY,XOFFSET,YOFFSET,NDOT,NLIN)
;CORNX,CORNY=THE CORNERCOORDINATES WHERE THE PICTURE
;  IS SHIFTED OUT
;  IF IF ANY OFFSET =0 CHOOSE HIGHSET(IN ADRESS)POSSIBLE CORNER
;XOFFSET=THE NUMBER OF STEPS TO MOVE PICTURE IN X COORDINATE
;  -TO MOVE LEFT +TO MOVE RIGHT (-255,255)
;YOFFSET=THE NUMBER OF STEPS TO MOVE PICTURE IN Y COORDINATE
;  -TO MOVE UP +TO MOVE DOWN (-255,255)
;NDOT=NUMBER OF DOTS IN ONE LINE BEFORE THE MOVE
;NLIN=NUMBER OF LINES TO SCAN BEFORE THE MOVE
;THE SUBROUTINE CALCULATES THE NUMBER OF DOTS AND LINES
;  TO MOVE ON ITS OWN

CRTADR = 170742             ;QRGB ADRESSADRESS
CRTBUF = 170740             ;QRGB COLORBYTEADRESS

        .GLOBL      MOVPIC

        .MACRO BLOCK  LABEL,XDIR,YDIRB
LABEL:  MOV     R0,@#CRTADR         ;R0 INTO ADR
        MOVB    @#CRTBUF,R4         ;COLOR IN R4
        MOV     R1,@#CRTADR         ;R1 INTO ADR
        MOVB    R4,@#CRTBUF
        XDIR    R0                  ;ADRESS NEXT DOT
        XDIR    R1                  ;IN X COORDINATE
        SOB     R2,LABEL            ;UNTIL EOL
        YDIRB   <STO+1>             ;NEXT Y COORDINATE
        YDIRB   <SFROM+1>
        MOV     SFROM,R0
        MOV     STO,R1
        MOV     ND,R2
        SOB     R3,LABEL            ;UNTIL LAST LINE
        RTS     PC
        .ENDM

MOVPIC: TST     (R5)+               ;SKIP ARGUMENTCOUNT
        MOV     @(R5)+,R0           ;CORNERCOORDINATE X
        MOVB    R0,STO              ;LOW BYTE STO(STARTADRESS TO)
        MOV     @(R5)+,R1           ;CORNERCOORDINATE Y
        MOVB    R1,<STO+1>          ;UPPER BYTE
        MOV     @(R5)+,R3           ;OFFSET X
        MOV     R3,XOFF
        SUB     R3,R0               ;CALCULATE FROMADRESS X
        MOVB    R0,SFROM            ;LOW BYTE
        MOV     @(R5)+,R4           ;OFFSET Y
        MOV     R4,YOFF
        SUB     R4,R1               ;CALCULATE FROMADRESS Y
        MOVB    R1,<SFROM+1>        ;UPPER BYTE
        MOV     @(R5)+,R2           ;FORMAT IN DOTS(BEFORE MOV)
        TST     R3
        BGE     XOFFE               ;IF POSITIVE OR ER BRANCH XOFFE
        NEG     R3                  ;CHANGE SIGN
```

```
XOFFE:  SUB     R3,R2           ;CALCULATENUMBER OF DOTS TO SCAN
        MOV     R2,ND           ;NUMBER OF DOTS IN ND
        MOV     @(R5),R3        ;NLIN IN R3
        TST     R4              ;TST YOFF
        BGE     YOFFE           ;IF POS OR ER BRANCH YOFFE
        NEG     R4              ;CHAMGE SIGN
YOFFE:  SUB     R4,R3           ;CALCULATE NUMBER OF LINES
        MOV     SFROM,R0
        MOV     STO,R1

        TST      YOFF           ;TEST YOFF
        BEQ      YZERO          ;IF 0 BRANCH YZERO
        BMI      YNEG           ;IF NEGATIVE BRANCH
        TST      XOFF           ;TEST XOFF
        BPL      XPOYPO         ;IF POS OR ER BRANCH XPOYPO
        BR       XNEYPO         ;BRANCH
YZERO:  TST      XOFF           ;TEST XOFF
        BGT      XPOYPO         ;IF XOFF>0 BRANCH
        BMI      XNEYPO         ;IF XOFF<0 BRANCH
        RTS      PC             ;IF XOFF=0 FAILURE RETURN

YNEG:   TST      XOFF           ;TEST XOFF
        BPL      XPOYNE         ;IF POS OR ER BRANCH

        BLOCK XNEYNE,INCR,INCR
        BLOCK XPOYNE,DECR,INCR
        BLOCK XPOYPO,DECR,DECR
        BLOCK XNEYPO,INCR,DECR

XOFF:   .WORD 0
YOFF:   .WORD 0

SFROM:  .WORD 0
STO:    .WORD 0

ND:     .WORD 0
       .END MOVPIC
```

```
          .TITLE GETCRS.MAC
;+TO SAVE THE DOTS AND PUT OUT CROSS
; CALL GETCRS(MONX,MONY,IWING,ICOL,ISAVEC)
; MONX=CENTERPOS X(0..255)
; MONY=CENTERPOS Y(0..255)
; IWING=WINGLENGTH OF CROSS(1..255)
; ICOL=COLOR ON CROSS PUT OUT
; ISAVEC=VECTOR TO SAVE THE DOTVALUES IN
;  MUST HAVE DIMENSION OF 4*IWING
;  NOT CHECKED
;ALL INTEGER
;-
          COLADR=170740     ;COLBYTEADR
          ADR=170742        ;GRGBADR

GETCRS::TST       (R5)+             ;SKIP ARG COUNT
          MOVB    @(R5)+,STMONX     ;CENTER X
          MOVB    @(R5)+,STMONY     ;-''- Y
          MOVB    @(R5)+,IWING      ;WINGLENGHT
          MOV     @(R5)+,ICOL       ;START OF SAVE VECTOR IN R4
          MOV     (R5),R4           ;STARTADRESS OF VECTOR IN R4
          MOVB    STMONX,MON        ;LET MON BE TOTAL ADRESS
          MOVB    STMONY,<MON+1>
          MOV     IWING,R3          ;LOPCOUNT=NUMBER OF WINGELEMENT
LOPUP:    DECB    <MON+1>           ;GO UP ON SCREEN
          CMPB    #377,<MON+1>      ;IF PASS 0 NO MORE TAKEUP
          BEQ     ENUP
          MOV     MON,ADR
          MOVB    COLADR,(R4)+      ;SCREENVALUE IN VECTOR
          MOV     MON,ADR
          MOVB    ICOL,COLADR       ;COLORCODE ON SCREEN
          SOB     R3,LOPUP          ;UNTIL WINGEND
ENUP:     MOV     IWING,R3
          MOVB    STMONY,<MON+1>
LOPDO:    INCB    <MON+1>           ;GO DOWN
          BEQ     ENDO
          MOV     MON,ADR
          MOVB    COLADR,(R4)+      ;ANALOG WITH ABOVE
          MOV     MON,ADR
          MOVB    ICOL,COLADR
          SOB     R3,LOPDO
ENDO:     MOV     IWING,R3
          MOVB    STMONX,MON
          MOVB    STMONY,<MON+1>
LOPRI:    INCB    MON               ;GO RIGHT
          BEQ     ENPRI
          MOV     MON,ADR
          MOVB    COLADR,(R4)+
          MOV     MON,ADR
          MOVB    ICOL,COLADR
          SOB     R3,LOPRI
ENPRI:    MOV     IWING,R3
          MOVB    STMONX,MON
LOPLE:    DECB    MON               ;GO LEFT
          CMPB    #377,MON
          BEQ     END
          MOV     MON,ADR
          MOVB    COLADR,(R4)+
          MOV     MON,ADR
          MOVB    ICOL,COLADR
          SOB     R3,LOPLE
END:      RTS     PC                ;RETURN FROM SUB
STMONX:   .WORD 0
STMONY:   .WORD 0
IWING:    .WORD
ICOL:     .WORD
MON:      .WORD
          .END GETCRS
```

```
;THIS (PROGRAMPACK) MAY BEE CALLED TO CHANGE
;CROSS-COLOR ON THE SCREEN
;

;+TO CHANGE COLOR
; CALL PUTCRS(STMONX,STMONY,IWING,ICOL)
; STMONX=MONITORPOS X(0,,,255)
; ----"---        Y
; IWING=WINGLENGHT
; ICOL=COLORCODE FOR CROSS (0,,,15)
;-
        .TITLE PUTCRS.MAC
        COLADR=170740    ;COLBYTEADR
        ADR=170742       ;QRGBADR

PUTCRS::TST     (R5)+    ;SKIP ARG COUNT
        MOVB    @(R5)+,STMONX    ;XPOS
        MOVB    @(R5)+,STMONY    ;YPOS
        MOVB    @(R5)+,IWING     ;WINGLENGTH ON CROSS
        MOV     @(R5)+,ICOL      ;COLOR
        MOVB    STMONX,MON       ;CONVERT TO QRGB ADRESS
        MOVB    STMONY,<MON+1>
        MOV     IWING,R3

LOPUP:  DECB    <MON+1>          ;WING UP
        CMPB    #377,<MON+1>
        BEQ     ENUP
        MOV     MON,ADR          ;ADRESS FOR QRGB TO PUT DOT
        MOVB    ICOL,COLADR      ;COLORCODE IN COLBYTEADRESS
        SOB     R3,LOPUP
ENUP:   MOV     IWING,R3
        MOVB    STMONY,<MON+1>
LOPDO:  INCB    <MON+1>          ;WING DOWN
        BEQ     ENDO             ;ANALOG WITH ABOVE
        MOV     MON,ADR
        MOVB    ICOL,COLADR
        SOB     R3,LOPDO
ENDO:   MOV     IWING,R3
        MOVB    STMONX,MON
        MOVB    STMONY,<MON+1>
LOPRI:  INCB    MON              ;WING RIGHT
        BEQ     ENPRI
        MOV     MON,ADR          ;ANALOG WITH ABOVE
        MOVB    ICOL,COLADR
        SOB     R3,LOPRI
ENPRI:  MOV     IWING,R3
        MOVB    STMONX,MON

LOPLE:  DECB    MON              ;WING LEFT
        CMPB    #377,MON
        BEQ     END              ;ANALOG WITH ABOVE
        MOV     MON,ADR
        MOVB    ICOL,COLADR
        SOB     R3,LOPLE
END:    RTS     PC
```

```
STMONX: .WORD 0
STMONY: .WORD 0
IWING:  .WORD
ICOL:   .WORD
MON:    .WORD
        .END PUTCRS
```

```
      SUBROUTINE INIJOY(MZN6,MZN7)
C     THIS SUBROUTINE INITIATES THE JOYSTICKLEVELS
C     MAXVALUE,ZEROVALUE,MINVALUE FOR AD CHANNELS 6,7
      DIMENSION MZN6(3),MZN7(3)
100   WRITE(7,120)
120   FORMAT(1X,'ZEROPOINT!')
      CALL ISLEEP(0,0,3,0)
      CALL ADIN(6,MZN6(2))
      CALL ADIN (7,MZN7(2))
      WRITE(7,140) MZN6(2),MZN7(2)
140   FORMAT(1X,'ZER06= ',I6,'ZER07= ',I6)
      WRITE(7,160)
160   FORMAT(1X,'MAX!')
      CALL ISLEEP(0,0,3,0)
      CALL ADIN(6,MZN6(3))
      CALL ADIN(7,MZN7(3))
      WRITE(7,180)MZN6(3),MZN7(3)
180   FORMAT(1X,'MAX6= ',I6,'MAX7= ',I6)
      WRITE(7,200)
200   FORMAT(1X,'MINIMUM!')
      CALL ISLEEP(0,0,3,0)
      CALL ADIN(6,MZN6(1))
      CALL ADIN(7,MZN7(1))
      WRITE(7,220)MZN6(1),MZN7(1)
220   FORMAT(1X,'MIN6= ',I6,'MIN7= ',I6)
230   WRITE(7,240)
240   FORMAT(1X,'SATISFIED?')
      READ(5,260)A
260   FORMAT(A)
      IF(A.EQ.'Y')RETURN
      IF(A.NE.'N')GOTO 230
      GOTO 100
      RETURN
      END
      SUBROUTINE LIMITS(MZN,L,CDIST)
C     THIS SUBROUTINE PRODUCES THE DIFFERENT LIMITS AND
C     CONSTANTS TO BE USED TOGETHER WITH THE JOYSTICK
C     L(1)=MAX L(6)=MIN
C     CDIST=1/THE DISTANCE BETWEEN LIMITS (1)GREATEST
      DIMENSION MZN(3),L(6),CDIST(2)
      A=(MZN(3)-MZN(1))/20.0
      LA=INT(A)
      WRITE(7,130)(MZN(N),N=1,3)
130   FORMAT(3X,'1=',I6,3X,'2=',I4,3X,'3=',I6)
      WRITE(7,140)A,LA
140   FORMAT(3X,F10.4,5X,I6)
      L(1)=MZN(3)
      L(2)=MZN(3)-4*LA
      L(3)=MZN(3)-9*LA
      L(4)=MZN(1)+9*LA
      L(5)=MZN(1)+4*LA
      L(6)=MZN(1)
      CDIST(1)=1./FLOAT(4*LA)
      CDIST(2)=1./FLOAT(5*LA)
      RETURN
      END
```

```
      SUBROUTINE JYSTEP(ISLX,ISLY,ITIME,MAXTIM,MNX,ISTP,
     + LIM6,LIM7,CDIST6,CDIST7,ITX,ITY)
C       THIS SUBROUTINE IS USED TO CALCULATE THE STEPS TO MOVE
C       AND TAKE UP NEW AREAS
C       ISLX=STEP IN X
C       ISLY=STEP IN Y
C       ITIME=TIME TO PROCEED BEFORE NEXT CALL IS POSSIBLE
C       MAXTIM=MAXIMUM WAITTIME
C       MNX=DIFFERENCE BETWEEN MAX AND MIN TIME
C       ISTP=ITIME IF STEP NOT 1
C       LIM6,7=LIMITS TO BE USED TOGETHER WITH AD TO DISIDE STEPLENGTH
C       LIM(1)=MAX,LIM(6)=MIN
C       CDIST=(1)DIST LIM(6)-LIM(5),(2) LIM(5)-LIM(4)
C       ITX,Y=MAXIMUM STEPLENGTH TO USE
        DIMENSION LIM6(6),LIM7(6),CDIST6(2),CDIST7(2)

      CALL ADIN(6,IVAL6)
C       X COORDINATE
      IF(IVAL6.GT.LIM6(1))IVAL6=LIM6(1)
      IF(IVAL6.LT.LIM6(2))GOTO 100
      ISLX=FLOAT(ITX)*FLOAT(IVAL6-LIM6(2))*CDIST6(1)+5
      ISLX=MIN0(ITX,ISLX)
      ITIM6=ISTP
C       TIME IS STEPTIME
      GOTO 300
100   IF(IVAL6.LT.LIM6(3))GOTO 150
      ISLX=1
      ITIM6=MAXTIM-FLOAT(MNX*(IVAL6-LIM6(3)))*CDIST6(2)
      GOTO 300
150   IF(IVAL6.LT.LIM6(4))GOTO 200
      ISLX=0
      ITIM6=MAXTIM
      GOTO 300
200   IF(IVAL6.LT.LIM6(5))GOTO 250
      ISLX=-1
      ITIM6=MAXTIM+FLOAT(MNX*(IVAL6-LIM6(4)))*CDIST6(2)
      GOTO 300
250   IF(IVAL6.LT.LIM6(6))IVAL6=LIM6(6)
      ISLX=FLOAT(ITX)*FLOAT(IVAL6-LIM6(5))*CDIST6(1)-5
      ISLX=MAX0(-ITX,ISLX)
      ITIM6=ISTP

300   CALL ADIN(7,IVAL7)
C       Y COORDINATE
      IF(IVAL7.GT.LIM7(1))IVAL=LIM7(1)
      IF(IVAL7.LT.LIM7(2))GOTO 400
      ISLY=FLOAT(ITY)*FLOAT(IVAL7-LIM7(2))*CDIST7(1)+5
      ISLY=MIN0(ISLY,ITY)
      ITIME=MIN0(ITIM6,ISTP)
      RETURN
400   IF(IVAL7.LT.LIM7(3))GOTO 450
      ISLY=1
      ITIM7=MAXTIM-FLOAT(MNX*(IVAL7-LIM7(3)))*CDIST7(2)
      ITIME=MIN0(ITIM7,ITIM6)
      RETURN
450   IF(IVAL7.LT.LIM7(4))GOTO 500
```

```
      ISLY=0
      ITIME=MINO(ITIM6,MAXTIM)
      RETURN
500   IF(IVAL7.LT.LIM7(5))GOTO 550
      ISLY=-1
      ITIM7=MAXTIM+FLOAT(MNX*(IVAL7-LIM7(4)))*CDIST7(2)
      ITIME=MINO(ITIM6,ITIM7)
      RETURN
550   ISLY=FLOAT(ITY)*FLOAT(IVAL7-LIM7(5))*CDIST7(1)-5
      ISLY=MAXO(-ITY,ISLY)
      ITIME=MINO(ITIM6,ISTP)
      RETURN
      END
500
      ISLY=0
      ITIME=MINO(ITIM6,MAXTIM)
       WRITE(7,*)ISLX,ISLY,ITIME
      RETURN
500   IF(IVAL7.LT.LIM7(5))GOTO 550
      ISLY=-1
      ITIM7=MAXTIM+FLOAT(MNX*(IVAL7-LIM7(4)))*CDIST7(2)
      ITIME=MINO(ITIM6,ITIM7)
       WRITE(7,*)ISLX,ISLY,ITIME
      RETURN
550   ISLY=FLOAT(ITY)*FLOAT(IVAL7-LIM7(5))*CDIST7(1)-1
      ISLY=MAXO(-ITY,ISLY)
      ITIME=MINO(ITIM6,ISTP)
       WRITE(7,*)ISLX,ISLY,ITIME
      RETURN
      END
```

```
      FUNCTION ICHAIO(ITYPE,ISIZE)
C     ICHAIO=CHANNELNUMBER
C     ITYPE=DEFAULT FILETYPE IN ASCII LOGICAL*1 DIM(3)
C     ISIZE=NUMBER OF BLOCKS
C     IF ISIZE=0 LOOKUP
C     IF ISIZE.LT.0 AND DOLD OPTION DELETES FILE
C     IF ISIZE.GT.0 CREATE A NEW FILE IF IT DONT EXIST
C     IF ISIZE.GT.0 AND DOLD OPTION DELETE AND CREATE A NEW FILE
      LOGICAL*1 INP(81),PROMT(11),HLPBUF(5),DY0B(5),DY1B(5)
      LOGICAL*1 OPT(5),NAME(12),ITYPE(3),LOG1,LOG2,NAME2(12)
      REAL*8 FSPEC
      DATA PROMT/'F','I','L','E','N','A','M','E','?',':',"200/
      DATA HLPBUF/' ',' ',' ',' ',0/
      DATA DY0B/'D','Y',0,':',0/,DY1B/'D','Y','1',':',0/
      DATA OPT/' ',' ',' ',' ',0/,NAME(1),NAME(2)/'D','Y'/
30    IDEL=0
      ICHAIO=-1
      DO 40 K=4,9
40    NAME(K)=' '
      DO 43 K=1,80
43    INP(K)=0
45    WRITE(7,500)
500   FORMAT(1X,'DEV,FILE(ACHAR)(/DOLD)(TO DELETE OLD)(BLANK',
     + '=RETURN)(TO EXIT)',/)
      CALL GTLIN(INP,PROMT)
      DO 50 I=1,4
50    HLPBUF(I)=INP(I)
      NAME(3)='2'
      IF(ISCOMP(HLPBUF,DY0B).EQ.0)NAME(3)='0'
      IF(ISCOMP(HLPBUF,DY1B).EQ.0)NAME(3)='1'
      IF(ISCOMP(INP,' ').EQ.0)RETURN
C     IF BLANK LINE RETURN
      IF(NAME(3).NE.'2')GOTO 55
      WRITE(7,*)'ILLEGAL DEVICE!(LEGAL DY1:,DY0:)'
      GOTO 30

55    IF(INP(5).LT.'A'.OR.INP(5).GT.'Z')GOTO 180
C     IF FIRST CHAR.NOT ALPH, WRITE ILLEGAL CHARACTER

      DO 100 I=5,10
      IF(INP(I).EQ.0)GOTO 150
      IF(INP(I).EQ.'/')GOTO 200
      LOG1=(INP(I).GE."101.AND.INP(I).LE."132)
C     TRUE IF ALPH. CHARACTER
      LOG2=(INP(I).GE."60.AND.INP(I).LE."71)
C     TRUE IF NUM. CHARACTER
      IF(.NOT.(LOG1.OR.LOG2))GOTO 180
C     IF CHAR.NOT NUMALPH. WRITE ILLEGAL CHARACTER
100   NAME(I-1)=INP(I)

      IK=11
      IF(INP(11).EQ.'/')GOTO 230
      IF(INP(11).NE.0)GOTO 180
      GOTO 300
150   IK=I
      DO 160 JB=IK+1,80
```

```
160     IF(INP(JB).NE.0)GOTO 180
        IF(IK.GT.5)GOTO 300
180     WRITE(7,*)'ILLEGAL CHARACTER IN FILENAME!'
        WRITE(7,*)'(DEV:1ALPHA+0..5ALPHANUMERIC CHAR+(OPTION))'
        GOTO 30

200     IK=I
        IF(IK.LT.6)GOTO 180
C       IF CHAR 5=/ ILLEGAL NAME
230     DO 250 J=1,4
250     OPT(J)=INP(IK+J)
        IF(ISCOMP(OPT,'DOLD ').NE.0)GOTO 260
        DO 255 JB=IK+5,80
255     IF(INP(JB).NE.0)GOTO 260
        IDEL=1
        GOTO 290
260     WRITE(7,*)'ONLY OPTION PERMITTED(/DOLD=(DELETE OLD))'
        GOTO 30

290     IDEL=1
300     DO 310 J=1,3
310     NAME(J+9)=ITYPE(J)


        JI=IRAD50(12,NAME,FSPEC)
        ICHAN=IGETC()
        ICHAIO=ICHAN
        IF(ICHAN.GE.0)GOTO 350
        WRITE(7,*)'NO MORE CHANNELS!'
        RETURN
350     IL=LOOKUP(ICHAN,FSPEC)
        IF(IL.LT.1.OR.IDEL.EQ.1.OR.ISIZE.NE.0)GOTO 352
        ISIZE=IL
        RETURN
C       FILE OPENED
352     ICL=PURGE(ICHAN)
        IF(IL.GE.0.AND.IDEL.EQ.1.AND.ISIZE.NE.0)GOTO 460
C       JUMP TO DELETE FILE
        IF(IL.EQ.0)GOTO 460
C       IF NOBLOCK DELETE
        IF(IL.EQ.-2.AND.ISIZE.GT.0)GOTO 370
C       IF NO FILE JUMP AND ENTER NEW FILE
        IF(IL.LT.0)GO TO 355
        WRITE(7,*)'FILE ALREADY EXIST!'
        IFC=IFREEC(ICHAN)
        GOTO 30
355     GOTO(360,365,380),IABS(IL)
        IFC=IFREEC(ICHAN)
        GOTO 30
360     WRITE(7,*)'CHANNEL ALREADY OPEN!'
        IFC=IFREEC(ICHAN)
        GOTO 30
365     WRITE(7,*)'FILE NOT FOUND!'
        IFC=IFREEC(ICHAN)
        GOTO 30
```

```
380     WRITE(7,*)'DEVICE IN USE!'
        IFC=IFREEC(ICHAN)
        GOTO 30

370     IF(IDEL.NE.1)GOTO 405
        WRITE(7,*)'FILE NOT FOUND!,LOOK'
        IFC=IFREEC(ICHAN)
        GOTO 30
405     IE=IENTER(ICHAN,FSPEC,ISIZE)
        IF(IE.NE.ISIZE)GOTO 407
        RETURN
407     GOTO(410,420,430,440),IABS(IE)
        GOTO 420
410     WRITE(7,*)'CHANNELERROR(IN USE)'
        IFC=IFREEC(ICHAN)
        GOTO 30
C       IMPOSSIBLE!
420     WRITE(7,*)'OUT OF FILE SPACE!'
        IFC=IFREEC(ICHAN)
        ICHAIO=-1
        RETURN
430     WRITE(7,*)'DEV IN USE!'
        IFC=IFREEC(ICHAN)
        GOTO 30
440     WRITE(7,*)'FILE ALREADY EXIST!(PROTECTED)'
        IFC=IFREEC(ICHAN)
        GOTO 30
460     ID=IDELET(ICHAN,FSPEC)
        IF(ID.EQ.0.AND.ISIZE.GT.0)GOTO 405
        IF(ID.NE.0)GOTO 490
        IFC=IFREEC(ICHAN)
        ICHAIO=-1
        RETURN
490     GOTO(510,520,530,540),ID
        IFC=IFREEC(ICHAN)
        GOTO 30
510     WRITE(7,*)'CHAN IN USE!'
        IFC=IFREEC(ICHAN)
        GOTO 30
520     WRITE(7,*)'DEV IN USE!'
        IFC=IFREEC(ICHAN)
        GOTO 30
530     WRITE(7,*)'FILE NOT FOUND!'
        IFC=IFREEC(ICHAN)
        GOTO 30
540     WRITE(7,*)'FILE PROTECTED!'
        IFC=IFREEC(ICHAN)
        GOTO 30
        END
```

```
;THIS PROGRAM MAY BEE CALLED TO RECOVER                    PAGE    1


;THIS PROGRAM MAY BEE CALLED TO RECOVER
; A CROSS ON THE SCREEN


;  TO RECOVER A CROSS PUT OUT WITH GETCRS()
;  CALL RECCRS(STMONX,STMONY,IWING,ISAVEC)
;  STMONX=CENTER X OF CROSS
;  STMONY=CENTER Y OF CROSS
;  IWING=WINGLENGTH OF CROSS
;  ISAVEC=VECTOR WERE THE ORIGINAL VALUES STORED
;  ALL INTEGER
          .TITLE RECCRS.MAC

          COLADR=170740     ;COLBYTEADR
          ADR=170742        ;QRGBADR


RECCRS::  TST       (R5)+             ;SKIP ARG COUNT
          MOVB      @(R5)+,STMONX
          MOVB      @(R5)+,STMONY
          MOVB      @(R5)+,IWING
          MOV       (R5),R4           ;STARTADRESS OF VECTOR IN R4
          MOVB      STMONX,MON
          MOVB      STMONY,<MON+1>
          MOV       IWING,R3

LOPUP:    DECB      <MON+1>           ;UP
          CMPB      #377,<MON+1>
          BEQ       ENUP
          MOV       MON,ADR           ;QRGBADRESS TO PUT VECTOR
          MOVB      (R4)+,COLADR      ;COLORCODE
          SOB       R3,LOPUP          ;UNTIL END OF WING
ENUP:     MOV       IWING,R3
          MOVB      STMONY,<MON+1>
LOPDO:    INCB      <MON+1>           ;DOWN
          BEQ       ENDO
          MOV       MON,ADR           ;ANALOG WITH ABOVE
          MOVB      (R4)+,COLADR
          SOB       R3,LOPDO
ENDO:     MOV       IWING,R3
          MOVB      STMONX,MON
          MOVB      STMONY,<MON+1>

LOPRI:    INCB      MON               ;RIGHT
          BEQ       ENPRI
          MOV       MON,ADR           ;ANALOG WITH ABOVE
          MOVB      (R4)+,COLADR
          SOB       R3,LOPRI
ENPRI:    MOV       IWING,R3
          MOVB      STMONX,MON

LOPLE:    DECB      MON               ;LEFT
          CMPB      #377,MON
          BEQ       END               ;ANALOG WITH ABOVE
          MOV       MON,ADR
          MOVB      (R4)+,COLADR
```

```
        SOB     R3,LOOPLE
END:    RTS     PC
STMONX: .WORD 0
STMONY: .WORD 0
IWING:  .WORD
MON:    .WORD
        .END RECORS
```

```
          .TITLE ADIN
;-THIS MAY BE CALLED BY A FORTRAN PROGRM
; CALL ADIN(CHANNEL,RESULT)
;READS THE ADV11-A
;+

ADSR=170400
ADBR=170402

ADIN::  TST     (R5)+    ;SKIP ARGCOUNT
        MOV     @(R5)+,R0        ;CHAN NUMBER
        CLR     @#ADSR   ;CLR STATUSREGISTER
        MOVB    R0,@#ADSR+1   ;CHANNEL
        INC     @#ADSR   ;START CONVERSION
     1$:TSTB    @#ADSR   ;WAIT UNTIL COMPLETE
        BPL     1$
        MOV     @#ADBR,@(R5)      ;RESULT ADRESSED BY R5
        RTS     PC
        .END ADIN
```

```
      SUBROUTINE GETPAR(IVEC,IELVEC)
      DIMENSION IVEC(IELVEC)
50    DO 100 I=1,IELVEC
100   WRITE(7,*)'VECTEL',I,'=',IVEC(I)
      WRITE(7,*)'RETURN(0) OR CHANGE(PARNR)'
      READ(5,*)IR
      IF(IR.EQ.0)RETURN
      IF(IR.LT.1.OR.IR.GT.IELVEC)GOTO 50
      WRITE(7,*)'PAR',IR,'NEWVAL'
      READ(5,*)IVEC(IR)
      GOTO 50
      END
```

```
      SUBROUTINE GLOCOR(IDP,ISX,ISY,IDX,IDY,ISCRX,ISCRY)
      INTEGER*4 ISX,ISY,I4DISX,I4DISY,I4ADX,I4ADY,ISADX,ISADY
      IDPY=IDP/10
      IDPX=IDP-10*IDPY
      IDISX=ISCRX-(IDPX-1)*64
      IDISY=ISCRY-(IDPY-1)*64
      IS=JICVT(IDX,I4DX)
      IS=JICVT(IDY,I4DY)
      IS=JICVT(IDISX,I4DISX)
      IS=JICVT(IDISY,I4DISY)
      IS=JMUL(I4DX,I4DISX,I4ADX)
      IS=JMUL(I4DY,I4DISY,I4ADY)
      IS=JADD(ISX,I4ADX,ISADX)
      IS=JADD(ISY,I4ADY,ISADY)
      WRITE(7,*)'DEFLCCORD X=',ISADX,'DEFLCCORD Y=',ISADY
C100   FORMAT(5X,'DEFLCOORDINATE X=',I12,5X,'DEFLCOORDINATE Y=',I12)
      RETURN
      END
```

```
      SUBROUTINE TAKEUP(IDF,IDP,ISX,ISY,IDX,IDY,LL,ICC,IEM
     + ,ICODE)
C       IDF=DISPLAYFORMAT(11,,44)
C       IDP=DISPLAYPOSITION(11,,44)
C         =0 IF NO DISPLAY
C       ISX=STARTADRESS TO SCAN FROM MAY BE DIFFERENT WHEN RETURN
C       ISY= ------"----------------
C       IDX,IDY=DISTANCE BETWEEN ADRESSED POINTS
C       LL=LIMITTABLE FOR DISPLY(SEE SEMPIC)
C       ICC=COLORCODE TO USE TOGETHER WITH LL
C        IEM=ERRORMESSAGE

        INTEGER*4 ISX,ISY,ILO,IUP,I4DX,I4DY,I4F0HX,I4F0HY
        INTEGER*4 IM4X,IM4Y,ISCX,ISCY,I432DX,I4D64Y,ISHY,ISHX
        INTEGER*2 LL(17),ICC(16)
        INTEGER*2 IHLO(2),IHUP(2)
        INTEGER*2 IOVEC(2048)
        LOGICAL*1 ITYPE(3)
        DATA ITYPE/'S','A','D'/
        EQUIVALENCE (IUP,IHUP),(ILO,IHLO)
        DATA IHLO/0,0/,IHUP/-1,0/
        IEM=0
        MAX=0
        MIN=4099


50      IDFY=IDF/10
        IF(IDFY.GE.1.AND.IDFY.LE.4)GOTO 150
100     IEM=1
C       FORMATFAILURE
        WRITE(7,*)' FORMATFAILURE'
        RETURN
150     IDFX=IDF-10*IDFY
        IF'IDFX.LT.1.OR.IDFX.GE.5)GOTO 100
        IF(IDP.EQ.0)GOTO 360
        IDPY=IDP/10
        IF(IDPY.GE.1.AND.IDPY.LE.4)GOTO 250
200     IEM=2
C       POSITIONFAILURE
        WRITE(7,*)'POSITIONFAILURE'
        RETURN
250     IDPX=IDP-10*IDPY
        IF(IDPX.LT.1.OR.IDPX.GE.5)GOTO 200

        IF((IDPX+IDFX).LE.5)GOTO 350
300      IEM=3
        WRITE(7,*)'POS AND FORMAT MISSFITS'
C       POS AND FORMAT MISSFITS
        RETURN
350     IF((IDPY+IDFY).GT.5)GOTO 300

360         IF(IDX.GE.1.AND.IDX.LE.256)GOTO 450
400      IEM=4
         WRITE(7,*)'ILLEGAL DEFLECTION'
C        ILLEGAL DEFLECTION
         RETURN
```

```
450     IF(IDY.LT.1.OR.IDY.GT.256)GOTO 400

C       TEST IF ADRESSED POINTS ARE WITHIN ADRESSROOM
C        WRITE(7,*)'BEFX',ISX,ILO
         IS=JCMP(ISX,ILO)
C        TEST IF STARTPOS WITHIN RANGE
         IF(IS.LT.0)GOTO 500
C        WRITE(7,*)'XLOWOK'
         IS=JCMP(ISY,ILO)
         IF(IS.LT.0)GOTO 500
C        WRITE(7,*)'LOWOK'
        IFOHX=64*IDFX-1
        IS=JICVT(IDX,I4DX)
        IS=JICVT(IFOHX,I4FOHX)
        IS=JMUL(I4FOHX,I4DX,IM4X)
        IS=JADD(IM4X,ISX,ISCX)
        IS=JCMP(IUP,ISCX)
        IF(IS.GE.0)GOTO 550
500     IEM=5
        WRITE(7,*)'OUT OF ADRESSROOM'
C       OUT OF ADRESSROOM
        RETURN
550     IFOHY=64*IDFY-1
        IS=JICVT(IFOHY,I4FOHY)
        IS=JICVT(IDY,I4DY)
        IS=JMUL(I4FOHY,I4DY,IM4Y)
        IS=JADD(IM4Y,ISY,ISCY)
        IS=JCMP(IUP,ISCY)
        IF(IS.LT.0)GOTO 500


         IS=JMOV(ISX,ISHX)
         IS=JMOV(ISY,ISHY)
        ISMX=(IDPX-1)*64
        ISMY=(IDPY-1)*64
        ISMHY=ISMY
        IFOTX=2*IDFX

        I32DX=IDX*32
        IS=JICVT(I32DX,I432DX)
        I64DY=IDY*64
        IS=JICVT(I64DY,I4D64Y)

        NBLC=IDFX*IDFY*16+1
        ICHAN=ICHAIO(ITYPE,NBLC)
CC      WRITE(7,*)'CHAN',ICHAN
        IF(ICHAN.LT.0)RETURN
        CALL DISCOL(ICOL)
        CALL DISLIM(LL)
        IBLNR=1
        L32=32
        L64=64
C       FIRST BLOCK(0) CONTAINS INFORMATION ABOUT TAKEUP

        DO 2000 K=1,IFOTX
        DO 1500 J=1,IDFY
```

```
      CALL SAVDEF(ISHX,ISHY,IDX,IDY,L32,L64,MAX,MIN,IOVEC)
      IF(IDP.EQ.0)GOTO 1000
      CALL DISPLY(ISMX,ISMHY,L32,L64,IOVEC)
CC    WRITE(7,*)'ISMX=',ISMX,'ISMHY=',ISMHY
      ISMHY=ISMHY+64
1000  IW=IWRITW(2048,IOVEC,IBLNR,ICHAN)
      IF(IW.LT.0)GOTO 2200
      WRITE(7,*)'MAXVALUE',MAX,'MINVALUE',MIN
CC    WRITE(7,*)'WORDS WRITTEN',IW
      IS=JADD(ISHY,I4D64Y,ISHY)
1500  IBLNR=IBLNR+8
      IS=JMOV(ISY,ISHY)
      IS=JADD(ISHX,I432DX,ISHX)
      ISMX=ISMX+32
2000  ISMHY=ISMY

2010  IC=CLOSEC(ICHAN)
      IF(IC.NE.0)GOTO 2100
2050  IFC=IFREEC(ICHAN)
      IF(IFC.EQ.0)RETURN
C     PROTECTIONERROR
CC    WRITE(7,*)'IFREEC=',IFC
      IEM=11
      RETURN
2100  WRITE(7,*)'PROTECTED FILE ALREADY EXIST '
      WRITE(7,*)'WARNING!TWO FILES WITH SAME NAME'
      GOTO 2050

2200  GOTO(2201,2202,2203)IABS(IW)
2201  IEM=8
C     TRIED TO WRITE PAST END OF FILE
      WRITE(7,*)'TRIED TO WRITE PAST END OF FILE'
      GOTO 2010
2202  IEM=9
      WRITE(7,*)'HARDWAREERROR!'
C     HARDWARE ERROR
      GOTO 2010
2203  IEM=10
      WRITE(7,*)'CHANNEL NOT OPENED'
C     CHANNEL NOT OPEN
      GOTO 2010
      END
```

```
      SUBROUTINE EDDIS(IDF,IDP,IEM,ITFO,INSTD,LL,ICC)
C       IDF=DISPLAYFORMAT(11,,44)
C       IDP=DISPLAYPOSITION(11,,44)
C       IEM=ERRORMESSAGE
C       ITFO=TAKEUP FORMAT OF INPUTFILE
C       INSTD=STARTDISPLAYPOSITION OF INPUTFILE
C        (11,,44,1100,,,,1155,,,4400)55 IF TO USE HALF DISPLAY
C        FORMAT WICH MEANS THAT IT IS POSSIBLE TO DISPLAY LESS
C        THEN TAKEUPFORMAT
C       LL=LIMITTABLE FOR DISPLY(SEE SEMPIC)
C       ICC=COLORCODE TO USE TOGETHER WITH LL
        INTEGER*2 LL(17),ICC(16)
        INTEGER*2 IOVEC(2048)
        LOGICAL*1 ITYPE(3)
        DATA ITYPE/'S','A','D'/
        IEM=0

      IDFY=IDF/10
      IF(IDFY.GE.1.AND.IDFY.LE.4)GOTO 150
100   IEM=1
C     FORMATFAILURE
      WRITE(7,*)' FORMATFAILURE'
      RETURN
150   IDFX=IDF-10*IDFY
      IF(IDFX.LT.1.OR.IDFX.GE.5)GOTO 100


      IDPY=IDP/10
      IF(IDPY.GE.1.AND.IDPY.LE.4)GOTO 250
200   IEM=2
C     POSITIONFAILURE
      WRITE(7,*)'POSITIONFAILURE'
      RETURN
250   IDPX=IDP-10*IDPY
      IF(IDPX.LT.1.OR.IDPX.GE.5)GOTO 200

      IF((IDPX+IDFX).LE.5)GOTO 350
300    IEM=3
      WRITE(7,*)'POS AND FORMAT MISSFITS'
C     POS AND FORMAT MISSFITS
      RETURN
350   IF((IDPY+IDFY).GT.5)GOTO 300

      ITFOY=ITFO/10
      IF(ITFOY.GE.1.AND.ITFOY.LE.4)GOTO 450
400   IEM=14
C     FORMATFAILURE
      WRITE(7,*)' TAKEUP FORMATFAILURE'
      RETURN
450   ITFOX=ITFO-10*ITFOY
      IF(ITFOX.LT.1.OR.ITFOX.GE.5)GOTO 400


C       TEST START DISPLAY POSITION
        ISTD=INSTD
```

```
      ISTDHX=0
      ISTDHY=0
      IF(ISTD.LE.44)GOTO 750
      ISTD=INSTD/100

      ISTDH=INSTD-ISTD*100
      ISTDHY=ISTDH/10
      IF(ISTDHY.NE.0.AND.ISTDHY.NE.5)GOTO 760
      ISTDHX=ISTDH-ISTDHY*10
      IF(ISTDHX.NE.0.AND.ISTDHX.NE.5)GOTO 760

750   ISTDY=ISTD/10
      IF(ISTDY.GE.1.AND.ISTDY.LE.4)GOTO 780
760   IEM=15
      WRITE(7,*)'ILLEGAL DISPLAY POS TO INPUTFILE '
C     ILLEGAL DISPLAY START
780   ISTDX=ISTD-10*ISTDY
      IF(ISTDX.LT.1.OR.ISTDX.GT.4)GOTO 760

C     TEST IF DISPLAYFORMAT FITS TAKEUPFORMAT
      IF((ITFOX-ISTDHX/5).GE.IDFX)GOTO 820
800   IEM=16
      WRITE(7,*)'DISPLAY POS OR FORMAT OF INPUTFILE MISSFITS'
C     FORMAT MISSFITS
      RETURN
820   IF((ITFOY-ISTDHY/5).LT.IDFY)GOTO 800


      ISMX=(IDPX-1)*64
      ISMY=(IDPY-1)*64
      ISMHY=ISMY
      IFOTX=2*IDFX
      IGHT=8
C     EIGHT BLOCKS FOR UNPACKED DATA
C     NBLC=IDFX*IDFY*16+1
C     TOTAL NUMBER OF BLOCKS
      NL=0
C     LOOKUP
      ICHAN=ICHAIO(ITYPE,NL)
CC    WRITE(7,*)'CHAN',ICHAN
      IF(I.LT.0)RETURN
      CALL DISCOL(ICC)
      CALL DISLIM(LL)
      INF=1
C     FIRST BLOCK(0) CONTAINS INFORMATION ABOUT TAKEUP
CC    WRITE(7,*)'ISTDX',ISTDX,'ISTDY',ISTDY
CC    WRITE(7,*)'ISTDHX',ISTDHX,'ISTDHY',ISTDHY
C     CALCULATE STARTBLOCK AND BLOCKS BETWEEN THEM TO DISPLAY
      IBLNR=ITFOY*((ISTDX-1)*IGHT*2+IGHT*ISTDHX/5)+
     + (ISTDY-1)*IGHT+IGHT/2*ISTDHY/5+INF
      NBBW=(ITFOY-IDFY)*IGHT
C     BLOCKS TO READ BEFORE CHANGE OF HORISONTAL COORDINATE
      IF((ITFOX*ITFOY*2*IGHT+INF).GT.NL)IEM=13
C     BLOCKS BETWEEN IN HORIZONTAL CHANGE
C     WRITE(7,*)'WARNING LESS BLOCKS THEN WANTED'
C     WARNING LESS BLOCKS THEN WANTED
```

```
      DO 2000 K=1,IFOTX
      DO 1500 J=1,IDFY
CC    WRITE(7,*)'STBLNR',IBLNR
      IR=IREADW(2048,IOVEC,IBLNR,ICHAN)
      IF(IR.LT.0)GOTO 2200
CC    WRITE(7,*)'BLOCKS READ',IR
      CALL DISPLY(ISMX,ISMHY,32,64,IOVEC)
      ISMHY=ISMHY+64
1500  IBLNR=IBLNR+IGHT
      ISMX=ISMX+32
      ISMHY=ISMY
2000  IBLNR=IBLNR+NRRW

2010  IC=CLOSEC(ICHAN)
      IF(IC.NE.0)GOTO 2100
2050  IFC=IFREEC(ICHAN)
      IF(IFC.EQ.0)RETURN
C     PROTECTIONERROR
CC    WRITE(7,*)'IFREEC=',IFC
      IEM=11
      RETURN
2100  WRITE(7,*)'PROTECTED FILE ALREADY EXIST '
      WRITE(7,*)'WARNING!TWO FILES WITH SAME NAME'
      GOTO 2050

2200  GOTO(2201,2202,2203)IABS(IR)
2201  IEM=12
C     TRIED TO READ PAST END OF FILE
      WRITE(7,*)'TRIED TO READ PAST END OF FILE'
      GOTO 2010
2202  IEM=9
      WRITE(7,*)'HARDWAREERROR!'
C     HARDWARE ERROR
      GOTO 2010
2203  IEM=10
      WRITE(7,*)'CHANNEL NOT OPENED'
C     CHANNEL NOT OPEN
      GOTO 2010
      END
```

```
      SUBROUTINE EDON(IDF,IDP,ISX,ISY,IDX,IDY,ISLX,ISLY,LL,ICC,
     + JOYMOD,IWPIC,IEM,TXT,LIM6,LIM7,CDIST6,CDIST7,MAXTIM,MNX,ISTP)
C     IDF=DISPLAYFORMAT(11.,44)
C     IDP=DISPLAYPOSITION(11.,44)
C     ISX=STARTADRESS TO SCAN FROM MAY BE DIFFERENT WHEN RETURN
C     ISY= -----"--------------
C     IDX,IDY=DISTANCE BETWEEN ADRESSED POINTS
C     ISLX,ISLY=STEPS TO MOVE ALREADY SHOWN PICTURE
C     IF BOOTH 0 A NEW PICTURE IS TAKEN UP
C     LL=LIMITTABLE FOR SEMPIC(SEE SEMPIC)
C     ICC=COLORCODE TO USE TOGETHER WITH LL
C      JOYMOD=1 IF ONE TAKEUP AND RETURN
C      =2 IF LOOP AND UNCHANGED POSITION(NO JOY)
C      =3 IF WITH UNCHANGED POSITION NEW SCAN WHEN JOY NOT 0
C      =4 IF TO LOOP AND MOVE OLD PIC
C      =5 IF TO LOOP AND ALWAYS TAKE UP A NEW SCAN(NO MOVE)
C      =6 IF TO WAIT UNTIL JOY NOT 0 AND NEW SCAN ACCORDING TO JOY
;     IWPIC=TIME TO PROCEED BEFORE NEW TAKEUP OF PICTURE
C      IN TICKS
C      IF 0 MAXIMUM TAKEUPSPEED
C      IEM=ERRORMESSAGE
C      TXT=TEXT TOGETHER WITH EM

      INTEGER*4 ISX,ISY,ILO,IUP,I4DX,I4DY,I4FOHX,I4FOHY
      INTEGER*4 IM4X,IM4Y,ISCX,ISCY,ISCLTX,ISCLTY
      INTEGER*4 ISTLR,ISTLL,ISTLU,ISTLD,I4STPX,I4STPY,ISTPX
      INTEGER*4 ISTPY
      DIMENSION CDIST6(2),CDIST7(2)

      INTEGER*2 LL(17),ICC(16),LIM6(6),LIM7(6)
      INTEGER*2 IHLO(2),IHUP(2)
      EQUIVALENCE (IUP,IHUP),(ILO,IHLO)
      DATA IHLO/0,0/,IHUP/-1,0/
      IEM=0


      IF(JOYMOD.GE.1.AND.JOYMOD.LE.6)GOTO 50
      IEM=6
      WRITE(7,*)'ILLEGAL JOYMODE'
C     ILLEGAL JOYMOD
      RETURN

50    IDFY=IDF/10
      IF(IDFY.GE.1.AND.IDFY.LE.4)GOTO 150
100   IEM=1
C     FORMATFAILURE
      WRITE(7,*)' FORMATFAILURE'
      RETURN
150   IDFX=IDF-10*IDFY
      IF(IDFX.LT.1.OR.IDFX.GE.5)GOTO 100

      IDPY=IDP/10
      IF(IDPY.GE.1.AND.IDPY.LE.4)GOTO 250
200   IEM=2
C     POSITIONFAILURE
      WRITE(7,*)'POSITIONFAILURE'
```

```
      RETURN
250   IDPX=IDP-10*IDPY
      IF(IDPX.LT.1.OR.IDPX.GE.5)GOTO 200

      IF((IDPX+IDFX).LE.5)GOTO 350
300    IEM=3
      WRITE(7,*)'POS AND FORMAT MISSFITS'
C     POS AND FORMAT MISSFITS
      RETURN
350   IF((IDPY+IDFY).GT.5)GOTO 300

       IF(IDX.GE.1.AND.IDX.LE.256)GOTO 450
400    IEM=4
       WRITE(7,*)'ILLEGAL DEFLECTION'
C      ILLEGAL DEFLECTION
       RETURN
450    IF(IDY.LT.1.OR.IDY.GT.256)GOTO 400

C      TEST IF ADRESSED POINTS ARE WITHIN ADRESSROOM
C
        IS=JCMP(ISX,ILO)
C       TEST IF STARTPOS WITHIN RANGE
        IF(IS.LT.0)GOTO 500
C
        IS=JCMP(ISY,ILO)
        IF(IS.LT.0)GOTO 500
C
       IFOHX=64*IDFX-1
       IS=JICVT(IDX,I4DX)
       IS=JICVT(IFOHX,I4FOHX)
       IS=JMUL(I4FOHX,I4DX,IM4X)
       IS=JADD(IM4X,ISX,ISCX)
       IS=JCMP(IUP,ISCX)
       IF(IS.GE.0)GOTO 550
500    IEM=5
       WRITE(7,*)'OUT OF ADRESSROOM'
C      OUT OF ADRESSROOM
       RETURN
550    IFOHY=64*IDFY-1
       IS=JICVT(IFOHY,I4FOHY)
       IS=JICVT(IDY,I4DY)
       IS=JMUL(I4FOHY,I4DY,IM4Y)
       IS=JADD(IM4Y,ISY,ISCY)
       IS=JCMP(IUP,ISCY)
       IF(IS.LT.0)GOTO 500

      CALL IPOKE("44,"10100.OR.IPEEK("44))
C     SET BIT 6 OF JSW TO MAKE ITTINR =NEG IF NOT ANY INPUT
C     SET BIT 12 TO MAKE IMMEDIATE READ

       IS=JSUB(IUP,ISCX,ISCLTX)
       IS=JDIV(ISCLTX,I4DX,ISTLR)
C      ISTLR=NUMBER OF POSSIBLE STEPS RIGHT
       IS=JDIV(ISX,I4DX,ISTLL)
C      ISTLL=NUMBER OF POSSIBLE STEPS LEFT
```

```
      IS=JSUB(IUP,ISCY,ISCLTY)
      IS=JDIV(ISCLTY,I4DY,ISTLD)
C     ISTLD=NUMBER OF POSSIBLE STEPS DOWN
      IS=JDIV(ISY,I4DY,ISTLU)
C     ISTLU=POSSIBLE STEP UP
      ITX=IDFX*64
      ITY=IDFY*64
      IMX=(IDPX-1)*64
      IMY=(IDPY-1)*64

570   IF(ISLX.EQ.0.AND.ISLY.EQ.0)GOTO 750
      IS=JICVT(ISLX,I4STPX)
      IS=JICVT(ISLY,I4STPY)
      IF(ISLX)600,610,620
600   IS=JADD(I4STPX,ISTLL,ISTPX)
      IF(IS.GE.0)GOTO 610
      IS=JSUB(I4STPX,ISTPX,I4STPX)
      IS=IJCVT(I4STPX,ISLX)
C      WRITE(7,*)ISLX
      GOTO 610

620   IS=JSUB(ISTLR,I4STPX,ISTPX)
      IF(IS.GE.0)GOTO 610
      IS=JADD(ISTPX,I4STPX,I4STPX)
      IS=IJCVT(I4STPX,ISLX)
C      WRITE(7,*)ISLX

610   IF(ISLY)700,710,720

700   IS=JADD(I4STPY,ISTLU,ISTPY)
      IF(IS.GE.0)GOTO 710
      IS=JSUB(I4STPY,ISTPY,I4STPY)
      IS=IJCVT(I4STPY,ISLY)
C      WRITE(7,*)ISLY
      GOTO 710

720   IS=JSUB(ISTLD,I4STPY,ISTPY)
      IF(IS.GE.0)GOTO 710
      IS=JADD(ISTPY,I4STPY,I4STPY)
      IS=IJCVT(I4STPY,ISLY)
C      WRITE(7,*)ISLY

710   IS=JADD(I4STPX,ISTLL,ISTLL)
      IS=JSUB(ISTLR,I4STPX,ISTLR)
C    CALCULATE NEW POSSIBLE STEPS
      IS=JADD(I4STPY,ISTLU,ISTLU)
      IS=JSUB(ISTLD,I4STPY,ISTLD)
      IF(JOYMOD.EQ.4)GOTO 750
C     NO MOVE
      IS=JMUL(I4DX,I4STPX,IM4X)
      IS=JADD(IM4X,ISX,ISX)
      IS=JMUL(I4DY,I4STPY,IM4Y)
      IS=JADD(IM4Y,ISY,ISY)
      ISLX=0
      ISLY=0
```

```
750     CALL EDOST(ISX,ISY,IDX,IDY,ISLX,ISLY,ITX,ITY,IMX,IMY,LL,ICC)
        IRET=ITTINR()
C        WRITE(7,*)ISX,ISY
C       TEST IF TO RETURN(IF IRET<0)
        IF(IRET.GE.0)GOTO 800
        IF(IWPIC.LE.0)GOTO 780
        CALL ISLEEP(0,0,0,IWPIC)
780     IF(JOYMOD.EQ.2)GOTO 750
        IF(JOYMOD.EQ.1)GOTO 800
785     IF(ITIME.LE.0)GOTO 790
        CALL ISLEEP(0,0,0,ITIME)
790     CALL JYSTEP(ISLX,ISLY,ITIME,MAXTIM,MNX,ISTP,LIM6,LIM7
     +  ,CDIST6,CDIST7,ITX,ITY)
        IF(JOYMOD.EQ.4.OR.JOYMOD.EQ.5)GOTO 570
        IRET=ITTINR()
        IF(ISLX.NE.0.OR.ISLY.NE.0)GOTO 795
C       TEST IF TO RETURN(IF IRET<0)
        IF(IRET.GE.0)GOTO 800
        GOTO 785
795     IF(JOYMOD.EQ.6)GOTO 570
C      JOYMOD=3
        ISLX=0
        ISLY=0
        GOTO 750
800     CALL IPOKE("44,"20100.AND.IPEEK("44))
C        ECHO CHARACTER AGAIN
C        SEE SSM 2-8(JSW)
        RETURN
        END
```

```
      SUBROUTINE FDOST(ISX,ISY,IDX,IDY,ISLEDX,ISLEDY,ITX,ITY,IMX,
     +IMY,LL,ICC)
C       ISX=STARTDEFPOSX INT*4
C       ISY=STARTDEFPOSY INT*4
C       ON RETURN ISX,ISY CONTAIN THE NEW STARTPOINTS
C       IDX=DEFLDISTX
C       IDY=DEFLDISTY
C       ISLEDX=STEPLENGTH X TO MOVE AND TAKEUP PICTURE
C       ISLEDY=-------*--- Y --------*-------------
C       ITX=TAKEUPFORMAT IN DOTS (X)
C       ITY=------*---------- LINES (Y)
C       IMX=MONITORDISPLAYPOSITION X(IN ADR COORDINATE)
C       IMY=-----------*---------  Y --------*---------
C       LL=LIMITTABLE
C       ICC=COLORTABLE
C       THE PROGRAM DOES NOT CHECK ANY PARAMETERS
C       ALL PARAMETERS INTEGER

        DIMENSION  LL(17),ICC(16)
        INTEGER*4 ISX,ISY,ISHX,ISHY,I4TX,I4TY,I4DY,I4DX
        INTEGER*4 I4SLX,I4SLY,I4SDX,I4SDY
C
C
        ISLX=-ISLEDX
C       MOVE PICTURE IN OPPOSITE DIRECTION TO NEW AREA
        ISLY=-ISLEDY

        IF(ISLX.EQ.0.AND.ISLY.EQ.0)GOTO 220

        IS=JICVT(ITX,I4TX)
        IS=JICVT(ITY,I4TY)
        IS=JICVT(IDX,I4DX)
        IS=JICVT(IDY,I4DY)
        IS=JICVT(ISLX,I4SLX)
        IS=JICVT(ISLY,I4SLY)
        IS= JMUL(I4TX,I4DX,I4TDX)
        IS=JMUL(I4TY,I4DY,I4TDY)
        IS=JMUL(I4SLX,I4DX,I4SDX)
        IS=JMUL(I4SLY,I4DY,I4SDY)
        IF(IABS(ISLX).NE.ITX.AND.IABS(ISLY).NE.ITY)GOTO 50
        IS=JSUB(ISX,I4SDX,ISX)
        IS=JSUB(ISY,I4SDY,ISY)
        GOTO 220
50      IF(ISLX)100,200,300
100     IF(ISLY)110,120,130
C       ISLX<0
200     IF(ISLY)210,220,230
C       ISLX=0
300     IF(ISLY)310,320,330
C       ISLX>0


110     CALL MOVPIC(IMX,IMY,ISLX,ISLY,ITX,ITY)
        IS=JADD(ISX,I4TDX,ISHX)
C       ISLX<0,ISLY<0!
        IS=JSUB(ISY,I4SDY,ISHY)
```

```
      NP=-ISLX
      NL=ITY
      IMHX=IMX+ITX+ISLX
      CALL SEMPIC(ISHX,ISHY,IDX,IDY,NP,NL,LL,ICC,IMHX,IMY)
      IS=JSUB(ISX,I4SDX,ISHX)
      IS=JADD(ISY,I4TDY,ISHY)
      NP=ITX+ISLX
      NL=-ISLY
      IMHY=IMY+ISLY+ITY
      CALL SEMPIC(ISHX,ISHY,IDX,IDY,NP,NL,LL,ICC,IMX,IMHY)
C     END OF MOVE IN 2ND QUADRANT
      IS=JSUB(ISX,I4SDX,ISX)
      IS=JSUB(ISY,I4SDY,ISY)
C     ISX=ISX-ISLX*IDX
C     ISY=ISY-ISLY*IDY
      RETURN

130   CALL MOVPIC(IMX,IMY+ITY-1,ISLX,ISLY,ITX,ITY)
C     ISLX<0 ISLY>0
      IS=JADD(ISX,I4TDX,ISHX)
C     ISHX=ISX+ITX*IDX
      IS=JSUB(ISY,I4SDY,ISHY)
C     ISHY=ISY-ISLY*IDY
      NP=-ISLX
      NL=ITY
      IMHX=IMX+ITX+ISLX
      CALL SEMPIC(ISHX,ISHY,IDX,IDY,NP,NL,LL,ICC,IMHX,IMY)
      IS=JSUB(ISX,I4SDX,ISHX)
C     ISHX=ISX-ISLX*IDX
C     ISHY=ISY-ISLY*IDY
      NP=ITX+ISLX
      NL=ISLY
      CALL SEMPIC(ISHX,ISHY,IDX,IDY,NP,NL,LL,ICC,IMX,IMY)
C     END OF MOVE 3RD QUADRANT
      IS=JMOV(ISHX,ISX)
      IS=JMOV(ISHY,ISY)
C     ISX=ISHX
C     ISY=ISHY
      RETURN

310   CALL MOVPIC(IMX+ITX-1,IMY,ISLX,ISLY,ITX,ITY)
C     ISLX>0,ISLY<0
      IS=JADD(ISY,I4TDY,ISHY)
C     ISHY=ISY+ITY*IDY
      NP=ITX-ISLX
      NL=-ISLY
      IMHY=IMY+ISLY+ITY
      CALL SEMPIC(ISX,ISHY,IDX,IDY,NP,NL,LL,ICC,IMX+ISLX,IMHY)
      IS=JSUB(ISX,I4SDX,ISHX)
C     ISHX=ISX-ISLX*IDX
      IS=JSUB(ISY,I4SDY,ISHY)
C     ISHY=ISY-ISLY*IDY
      NP=ISLX
      NL=ITY
      CALL SEMPIC(ISHX,ISHY,IDX,IDY,NP,NL,LL,ICC,IMX,IMY)
C     END OF MOVE IN FIRST QUADRANT
```

```
      IS=JMOV(ISHX,ISX)
      IS=JMOV(ISHY,ISY)
C     ISX=ISHX
C     ISY=ISHY
      RETURN

330   CALL MOVPIC(IMX+ITX-1,IMY+ITY-1,ISLX,ISLY,ITX,ITY)
C     ISLX>0,ISLY>0
      IS=JSUB(ISY,I4SDY,ISHY)
C     ISHY=ISY-ISLY*IDY
      NP=ITX-ISLX
      NL=ISLY
      CALL SEMPIC(ISX,ISHY,IDX,IDY,NP,NL,LL,ICC,IMX+ISLX,IMY)
      IS= JSUB(ISX,I4SDX,ISHX)
C     ISHX=ISX-ISLX*IDX
      IS=JSUB(ISY,I4SDY,ISHY)
C     ISHY=ISY-ISLY*IDY
      NP=ISLX
      NL=ITY
      CALL SEMPIC(ISHX,ISHY,IDX,IDY,NP,NL,LL,ICC,IMX,IMY)
C     END OF MOVE 4TH QUADRANT
      IS=JMOV(ISHX,ISX)
      IS=JMOV(ISHY,ISY)
C     ISX=ISHX
C     ISY=ISHY
      RETURN

120   CALL MOVPIC(IMX,IMY+ITY-1,ISLX,ISLY,ITX,ITY)
C     ISLX<0 ISLY=0
      IS=JADD(ISX,I4TDX,ISHX)
C     ISHX=ISX+IDX*ITY
C     ISLY=0
      NP=-ISLX
      NL=ITY
      IMHX=IMX+ITX+ISLX
      CALL SEMPIC(ISHX,ISY,IDX,IDY,NP,NL,LL,ICC,IMHX,IMY)
C     END ONLY MOVE X
      IS=JSUB(ISX,I4SDX,ISHX)
      IS=JMOV(ISHX,ISX)
C     ISX=ISX-ISLX*IDX
      RETURN

320   CALL MOVPIC(IMX+ITX-1,IMY+ITY-1,ISLX,ISLY,ITX,ITY)
C     ISLX>0 ISLY=0
      IS=JSUB(ISX,I4SDX,ISHX)
C     ISX=ISX-ISLX*IDX
      NP=ISLX
      NL=ITY
      CALL SEMPIC(ISHX,ISY,IDX,IDY,NP,NL,LL,ICC,IMX,IMY)
      IS=JMOV(ISHX,ISX)
C     ONLY X MOVE
      RETURN


210   CALL MOVPIC(IMX+ITX-1,IMY,ISLX,ISLY,ITX,ITY)
C     ISLX=0 ISLY<0
```

```
      IS=JADD(ISY,I4TDY,ISHY)
C     ISHY=ISY+ITY*IDY
C     ISLX=0
      NP=ITX
      NL=-ISLY
      IMHY=IMY+ITY+ISLY
      CALL SEMPIC(ISX,ISHY,IDX,IDY,NP,NL,LL,ICC,IMX,IMHY)
C     END ONLY MOVE Y
      IS=JSUB(ISY,I4SDY,ISHY)
      IS=JMOV(ISHY,ISY)
C     ISY=ISY-IDY*ISLY
      RETURN

230   CALL MOVPIC(IMX+ITX-1,IMY+ITY-1,ISLX,ISLY,ITX,ITY)
C     ISLX=0 ISLY>0
      IS=JSUB(ISY,I4SDY,ISHY)
C     ISY=ISY-ISLY*IDY
      NP=ITX
      NL=ISLY
      CALL SEMPIC(ISX,ISHY,IDX,IDY,NP,NL,LL,ICC,IMX,IMY)
      IS=JMOV(ISHY,ISY)
C     MOVE ONLY Y
      RETURN

220   CALL SEMPIC(ISX,ISY,IDX,IDY,ITX,ITY,LL,ICC,IMX,IMY)
C     END OF NEW TAKEUP
      RETURN

      END
```

```
      SUBROUTINE CRSJOY(IDF,IDP,ICRSX,ICRSY,IWING,ICOL,IVEC,IDV,
     + IWPIC,LIM6,LIM7,CDIST6,CDIST7,ITWINK,IEM)
C       IDF=DISPLAYFORMAT(11.,44)
C       IDP=DISPLAYPOSITION(11.,44)
C       ICRSX=STARTADRESS FOR CROSS X (0.,.255)
C       ICRSY= ------"--------------- Y
C       IWING=WINGLENGTH OF CROSS
C       ICOL=COLOR OF CROSS
C       IVEC=VECTOR TO STORE THE QRGBVALUES IN
C       IDV=DIM OF IVEC
C       IWPIC=TIME TO PROCEED BEFORE NEW TAKEUP OF PICTURE
C        IN TICKS
C       LIM6,7=LIMITS TO BE USED WITH JYSTEP
C       CDIST6,7=KONSTANTS TO USE IN JYSTEP
C       ITWINK=LOGICAL =.TRUE.IF TO TWINKLE GROSS
C       IEM=FAILUREMESSAGE =1(FORMAT),2(POS),3(MISSFIT)

        DIMENSION CDIST6(2),CDIST7(2),IVEC(IDV)

        INTEGER*2 LIM6(6),LIM7(6)
        LOGICAL ITWINK
        MAXTIM=30
C       TIME TO WAIT IF JOYSTEP=0
        MNX=30
C       MAXTIM-MNX=LEAST TIMEDIST TO WAIT
        ISTP=1
C       TIME TO WAIT IF STEPING
        ISLX=0
        ISLY=0
        IWPIC=MAX0(0,IWPIC)
C       WRITE(7,*)(CDIST6(K),K=1,2),(CDIST7(K ,K=1,2)

50      IDFY=IDF/10
        IF(IDFY.GE.1.AND.IDFY.LE.4)GOTO 150
100     IEM=1
C       FORMATFAILURE
        RETURN
150     IDFX=IDF-10*IDFY
        IF(IDFX.LT.1.OR.IDFX.GE.5)GOTO 100

        IDPY=IDP/10
        IF(IDPY.GE.1.AND.IDPY.LE.4)GOTO 250
200     IEM=2
C       POSITIONFAILURE
        RETURN
250     IDPX=IDP-10*IDPY
        IF(IDPX.LT.1.OR.IDPX.GE.5)GOTO 200

        IF((IDPX+IDFX).LE.5)GOTO 350
300      IEM=3
C       POS AND FORMAT MISSFITS
        RETURN
```

```
350     IF((IDPY+IDFY).GT.5)GOTO 300

        IFOHX=64*IDFX-1
        IFOHY=64*IDFY-1

        CALL IPOKE("44,"10100.OR.IPEEK("44))
C       SET BIT 6 OF JSW TO MAKE ITTINR =NEG IF NOT ANY INPUT
C       SET BIT 12 TO MAKE IMMEDIATE READ .

        MINX=(IDPX-1)*64
        MINY=(IDPY-1)*64
        MAXX=MINX+IDFX*64-1
        MAXY=MINY+IDFY*64-1
        ITX=IDFX*64
        ITY=IDFY*64
        MDX=6
        MDY=6
        GOTO 420

400     CALL PUTCRS(ICRSX,ICRSY,IWING,ICOL)
        GOTO 785
410     CALL RECCRS(ICRSX,ICRSY,IWING,IVEC)

420      ICLX=ICRSX+ISLX
         ICLY=ICRSY+ISLY
         IF(ISLX)430,440,450
430      ICRSX=MAX0(MINX,ICLX)
440      IF(ISLY)530,540,550
450      ICRSX=MIN0(MAXX,ICLX)
         IF(ISLY)530,540,550

530      ICRSY=MAX0(MINY,ICLY)
         GOTO 540
550      ICRSY=MIN0(MAXY,ICLY)
540      CALL GETCRS(ICRSX,ICRSY,IWING,ICOL,IVEC)
         IZLOP=0



650      IRET=ITTINR()
C        TEST IF TO RETURN OR CHANGE PAR (IF IRET>0)
         IF(IRET.GE.0)GOTO 800

785      IF(ITIME.LE.0)GOTO 790
         CALL ISLEEP(0,0,0,ITIME)
790      CALL JYSTEP(ISLX,ISLY,ITIME,MAXTIM,MNX,ISTP,LIM6,LIM7
     +  ,CDIST6,CDIST7,MDX,MDY)
         IF(ISLX.NE.0.OR.ISLY.NE.0)GOTO 410
C        ISLX,ISLY=0
         IF(ITWINK)GOTO 750
         IF(IZLOP.GT.4)GOTO 650
         IZLOP=IZLOP+1
         IF(IZLOP.NE.4)GOTO 650
         WRITE(7,*)'XCOOR=',ICRSX,'YCOOR=',ICRSY
         GOTO 650
750      CALL RECCRS(ICRSX,ICRSY,IWING,IVEC)
```

```
      IF(IWPIC.EQ.0)GOTO 755
      CALL ISLEEP(0,0,0,IWPIC)
755   ITM=MAX0(0,IWPIC-ITIME)
      CALL GETCRS(ICRSX,ICRSY,IWING,ICOL,IVEC)
      IF(IWPIC.EQ.0)GOTO 760
      CALL ISLEEP(0,0,0,ITM)
760   IF(IZLOP.GT.4)GOTO 650
      IZLOP=IZLOP+1
      IF(IZLOP.NE.4)GOTO 650
      WRITE(7,*)'XCOOR=',ICRSX,'YCOOR=',ICRSY
      GOTO 650

800   IF(IRET.NE.48)GOTO 810
C     IRET='0'
      ICOL=0
      GOTO 400
810   IF(IRET.NE.49)GOTO 820
C     IRET='1'
      ICOL=1
      GOTO 400
820   IF(IRET.NE.50)GOTO 830
C     IRET=2
      ICOL=2
      GOTO 400
830   IF(IRET.NE.51)GOTO 840
C     IRET='3'
      ICOL=3
      GOTO 400
840   IF(IRET.NE.52)GOTO 850
C     IRET='4'
      ICOL=4
      GOTO 400
850   IF(IRET.NE.53)GOTO 860
C     IRET='5'
      ICOL=5
      GOTO 400
860   IF(IRET.NE.54)GOTO 870
C     IRET='6'
      ICOL=6
      GOTO 400
870   IF(IRET.NE.55)GOTO 880
C     IRET='7'
      ICOL=7
      GOTO 400
880   IF(IRET.NE.56)GOTO 890
C     IRET='8'
      ICOL=8
      GOTO 400
890   IF(IRET.NE.57)GOTO 900
C     IRET='9'
      ICOL=9
      GOTO 400
900   IF(IRET.NE.65)GOTO 910
C     IRET='A'
      ICOL=10
      GOTO 400
```

```
910     IF(IRET.NE.66)GOTO 920
C       IRET='B'
        ICOL=11
        GOTO 400
920     IF(IRET.NE.67)GOTO 930
C       IRET='C'
        ICOL=12
        GOTO 400
930     IF(IRET.NE.68)GOTO 940
C       IRET='D'
        ICOL=13
        GOTO 400
940     IF(IRET.NE.69)GOTO 950
C       IRET='E'
        ICOL=14
        GOTO 400
950     IF(IRET.NE.70)GOTO 960
C       IRET='F'
        ICOL=15
        GOTO 400

960     IF(IRET.NE.84)GOTO 970
C       IRET='T'
        ITWINK=.TRUE.
        GOTO 785
970     IF(IRET.NE.78)GOTO 980
C       IRET='N'
        ITWINK=.FALSE.
        GOTO 785

980     IF(IRET.EQ.72)GOTO 990
C       IRET='H' IF NOT TO RECOVER CROSS ON RETURN
        CALL RECCRS(ICRSX,ICRSY,IWING,IVEC)
990     CALL IPOKE("44,"20100.AND.IPEEK("44)
C        ECHO CHARACTER AGAIN
C        SEE SSM 2-8(JSW)
        RETURN
        END
         SUBROUTINE CRSLIM(MZN,L,CDIST)
C        THIS SUBROUTINE PRODUCES THE DIFFERENT LIMITS AND
C        CONSTANTS TO BE USED TOGETHER WITH THE JOYSTICK
C        TOGETHER WITH THE CROSS
C        L(1)=MAX L(6)=MIN
C        CDIST=1/THE DISTANCE BETWEEN LIMITS (1)GREATEST
         DIMENSION MZN(3),L(6),CDIST(2)
         A=(MZN(3)-MZN(1))/20.0
         LA=INT(A)
         WRITE(7,130)(MZN(N),N=1,3)
130      FORMAT(3X,'1=',I6,3X,'2=',I6,3X,'3=',I6)
         WRITE(7,140)A,LA
140      FORMAT(3X,F10.4,5X,I6)
         L(1)=MZN(3)
         L(2)=MZN(3)-4*LA
         L(3)=MZN(3)-9*LA
         L(4)=MZN(1)+9*LA
         L(5)=MZN(1)+4*LA
```

```
L(6)=MZN(1)
CDIST(1)=1./FLOAT(4*LA)
CDIST(2)=1./FLOAT(5*LA)
RETURN
END
```

```
      SUBROUTINE TKESUB(LIM,ICC)
      LOGICAL*1 ITYPE(3)
      INTEGER*2 ICC(16),ISX2(2),ISY2(2),LIM(17)
      INTEGER*4 ISX,ISY
      EQUIVALENCE (ISX,ISX2),(ISY,ISY2)
C     DATA ICC/0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15/
      DATA ITYPE/'S','A','D'/,ISX2/0,0/,ISY2/0,0/
C     DATA MINV/4097/,MAXV/0/
3     WRITE(7,*)'MAXLIM,MINLIM'
      READ(5,*)MAXLIM,MINLIM
      ISTEP=(MAXLIM-MINLIM)/16
      DO 5 IL=1,17
5     LIM(18-IL)=(IL-1)*ISTEP+MINLIM
      I=IQSET(4)
      IF(I.EQ.0)GOTO 10
      WRITE(7,*)'LACK OF MEMORYSPACE'
      CALL LOCK
C     LOCK USR IN MEMORY

10    WRITE(7,*)'TAKEUP(1),EDDISPLAY(2),DEFINETF(3),CHANGELIM(4)
     +,RETURNSUB(5)'
      READ(5,*)IN
      IF(IN.LT.1.OR.IN.GT.5)GOTO 10
      GOTO(20,30,40,3,50),IN
20    WRITE(7,*)'IDSPFORMAT(11,,,44),DISPOS(1),,,44)'
      READ (5,*)IDF,IDP
      WRITE(7,*)'STARTPOSX,STARTPOSY,DISTX,DISTY'
      READ(5,*)ISX,ISY,IDX,IDY
      CALL TAKEUP(IDF,IDP,ISX,ISY,IDX,IDY,LIM,ICC,IEM,ICODE)
      IF(IEM.EQ.0)GOTO 10
      WRITE(7,*)'TAKUPERROR',IEM
      GOTO 10
C     WRITE(7,*)'MAXVALUE',MAXV,'MINVALUE',MINV
30    WRITE(7,*)'DISPFORMAT,DISPPOS,TAKEUPFORMAT,STARTDISPOS'
      READ(5,*)IDFE,IDPE,ITFO,INSTD
      CALL EDDIS(IDFE,IDPE,IEM,ITFO,INSTD,LIM,ICC)
      IF(IEM.EQ.0)GOTO 10
      WRITE(7,*)'EDDERROR',IEM
      GOTO 10
40    I=ICHAIO(ITYPE,-1)
C     DELETE A FILE
      GOTO 10
50    RETURN
      END



      SUBROUTINE TRISUB()
C     PROGRAM TRIFI
      LOGICAL*1 ITYPE(3)
C     INTEGER*2 IBUF(256)
      DATA ITYPE/'F','V','D'/
      I=IQSET(4)
      IF(I.EQ.0)GOTO 10
      WRITE(7,*)'LACK OF SPACE'
      CALL LOCK
C     LOCK USR IN MEMORY
```

```
10      WRITE(7,*)'SAVE(1),DISPLAY(2),DELFIF(3),RETURN(4)'
        READ(5,*)IN
        IF(IN.LT.1.OR.IN.GT.4)GOTO 10
        GOTO(20,30,40,50),IN
20      I=ICHAIO(ITYPE,A4)
        IF(I.LT.0)GOTO 10
        CALL SVISUB(I,0,IEM)
        IC=CLOSEC(I)
        CALL IFREEC(I)
        IF(IFM.GT.0)GOTO 10
C       WRITE(7,*)'SAVIDERROR!',IABS(IEM)
        GOTO 10
30      L=0
        I=ICHAIO(ITYPE,L)
        IF(I.LT.0)GOTO 10
        CALL VIDSUB(I,0,IEM)
        IC=CLOSEC(I)
        CALL IFREEC(I)
        IF(IFM.GT.0)GOTO 10
C       WRITE(7,*)'VIDISDERROR!',IABS(IEM)
        GOTO 10
40      I=ICHAIO(ITYPE,-1)
        DELETE A FILE
        GOTO 10
50      RETURN
        END


        PROGRAM FROMAN
        DIMENSION ICC(16),LL(17),MZN6(3),MZN7(3),LIM6(6),LIM7(6)
        DIMENSION CDIST6(2),CDIST7(2),IS2X(2),IS2Y(2),IVEC(512)
        DIMENSION CRIST6(2),CRIST7(2),LIMC6(6),LIMC7(6)
        INTEGER*4 ISX,ISY
        LOGICAL ITWINK
        EQUIVALENCE (IS2X,ISX),(IS2Y,ISY)
        DATA IDF,IDP/44,11/,IS2X/0,0/,IS2Y/0,0/
        DATA IDX,IDY/50,50/,ISLX,ISLY/0,0/,JOYMOD/4/,ICOL/0/
        DATA IWPIC/0/,MAXTIM/50/,MNX/30/,ISTP/30/,IWING/7/,IWCR/30/
        DATA ICC/0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15/,IDV/512/

        ITWINK=.TRUE.
        DO 10 K=1,17
        A=(K-1)/16.
10      LL(18-K)=A*2000.
        CALL IQSET(2)
        CALL INIJOY(MZN6,MZN7)
        CALL LIMITS(MZN6,LIM6,CDIST6)
        CALL LIMITS(MZN7,LIM7,CDIST7)
        CALL CRSLIM(MZN6,LIMC6,CRIST6)
        CALL CRSLIM(MZN7,LIMC7,CRIST7)


20      WRITE(7,*)'DISPLAYFORM(1)=',IDF,'DPPOSITION(2)=',IDP
        WRITE(7,*)'STARTX(3)=',ISX,'STARTY(4)=',ISY
        WRITE(7,*)'DISTX(5)=',IDX,'DISTY(6)=',IDY
```

```
      WRITE(7,*)'STEPX(7)=',ISX,'STEPY(8)=',ISY
      WRITE(7,*)'JOYMOD(9)=',JOYMOD,'PICWAIT(10)=',IWPIC
      WRITE(7,*)'MAXTIM(11)=',MAXTIM,'DIFFTIM(12)=',MNX
      WRITE(7,*)'STEPTIM(13)=',ISTP,'CRSWING(14)=',IWING
      WRITE(7,*)'TWINKLESPEED(14)=',IWCR,'CROSS(15)'
      WRITE(7,*)'CONVERTVIDEOCOORDINATE TO DEFI(17)'
      WRITE(7,*)'CHANGE COLCODE(18)','CHANGE COLLIM(19)'
      WRITE(7,*)'SAVE OR DISPLAY VIDEO (20)'
      WRITE(7,*)'TAKEUP OR ENDID SAVED PICTURE(21)'
      WRITE(7,*)'CHANGE?(PARNR(0=NOMORECHANGE),NEWVAL)'
      READ(5,*)IPAR
      IF(IPAR.EQ.0)GOTO 300
      IF(IPAR.LT.0.OR.IPAR.GT.21)GOTO 20
      GOTO(140,150,160,170,180,190,200,210,220,230,240,250,
     + 260,270,280,290,295,296,297,298,299),IPAR
140   READ(5,*)IDF
      GOTO 20
150   READ(5,*)IDP
      GOTO 20
160   READ(5,*)ISX
      GOTO 20
170   READ(5,*)ISY
      GOTO 20
180   READ(5,*)IDX
      GOTO 20
190   READ(5,*)IDY
      GOTO 20
200   READ(5,*)ISLX
      GOTO 20
210   READ(5,*)ISLY
      GOTO 20
220   READ(5,*)JOYMOD
      GOTO 20
230   READ(5,*)IWPIC
      GOTO 20
240   READ(5,*)MAXTIM
      GOTO 20
250   READ(5,*)MNX
      GOTO 20
260   READ(5,*)ISTP
      GOTO 20
270   READ(5,*)IWCR
      GOTO 20
280   CALL CRSJOY(IDF,IDP,ICRSX,ICRSY,IWING,ICOL,IVEC,IDV,IWCR,
     + LIMC6,LIMC7,CRIST6,CRIST7,ITWINK,IEM)
      GOTO 20
290   READ(5,*)IWING
      GOTO 20
295   CALL GLOCOR(IDP,ISX,ISY,IDX,IDY,ICRSX,ICRSY)
      GOTO 20
296   CALL GETPAR(ICC,16)
      GOTO 20
297   CALL GETPAR(LL,17)
      GOTO 20
298   CALL TRISUB()
      GOTO 20
299   CALL TKESUB(LL,ICC)
      GOTO 20
300   IEM=0

      CALL EDON(IDF,IDP,ISX,ISY,IDX,IDY,ISLX,ISLY,LL,ICC,JOYMOD
     +,IWPIC,IEM,TXT,LIM6,LIM7,CDIST6,CDIST7,MAXTIM,MNX,ISTP)
      IF(IEM.EQ.0)GOTO 20
      WRITE(7,*)'ERROR=',IEM
      GOTO 20
      END
```