CODEN:LUTFD2/(TFRT-5276)/1-053/(1982)

HÖGUPPLÖSANDE FÄRGGRAFIK FÖR OPERATÖRSKOMMUNIKATION

TORBJÖRN PERSSON

DEPARTMENT OF AUTOMATIC CONTROL
LUND INSTITUTE OF TECHNOLOGY

JUNE 1982

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

Document name
MASTER THESIS

Date of issue
June 1982

Document number
CODEN:LUTFD2/(TFRT-5276)/1-053/(1982)

Author(s)
Torbjörn Persson

Supervisor
Karl Johan Åström

Sponsoring organization

Title and subtitle
Högupplösande färggrafik för operatörskommunikation
(High-resolution colour graphics in control systems)

Abstract
This paper describes the design and construction of a microprocessor-based video display control unit intended for use in industrial control systems. The unit features graphics of up to 1024 X 1024 pixels resolution and eight colours using the NEC µPD 7220 Graphics Display Controller. The design idea is based on a discussion of basic principles in colour graphics outlined in the first part of the paper. This is followed by a detailed description of the units' construction and performance including some software.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
53

Recipient's notes

Security classification

0. Förord

Denna rapport utgör dokumentationen för ett examensarbete vars syfte varit att konstruera en videokontrollenhet för högupplösande färggrafik. Examensarbetet har utförts dels vid Institutionen för Reglerteknik på Lunds Tekniska Högskola, dels vid Telsand Elektronics AB i Arlöv.

Färggrafik är ett relativt nytt område inom man-maskinkommunikation. Av denna anledning finns det ännu ganska få allmänna regler om hur den ska utformas. Ett antal artiklar i fackpressen och några böcker har dock skrivits och den första delen av denna rapport försöker sammanfatta grundläggande fakta om färggrafik. Ur dessa får vi till sist fram en lämplig konstruktionsidé.

Konstruktionen finns redovisad i den andra delen av rapporten. Denna beskriver hur de olika detaljerna från första delen kan realiseras. Här finns också en testrapport och några förslag till förbättringar och synpunkter på konstruktionen.

Den valda hårdvarukonstruktionen innehåller kretsar som senare visade sig vara svåra att få tag på. Bland annat videoprocessorn som kom ut på marknaden först ett år efter fabrikantens första tillkännagivelse. Eftersom konstruktionsarbetet därför dragit ut på tiden har inskränkningar fått göras i den planerade mjukvaruutvecklingen för videokontrollenheten.

Jag vill här också tacka för all hjälp och idéinspiration som jag fått av personalen på Reglerteknik och Telsand, och då speciellt min handledare prof. Karl Johan Åström, Telsands Axel Westrenius och Lars Knutsson.

Lund maj 1982

Torbjörn Persson

0.1 Innehållsförteckning

CODEN:LUTFD2/(TFRT-5276)/1-053/(1982)

HÖGUPPLÖSANDE FÄRGGRAFIK FÖR OPERATÖRSKOMMUNIKATION

TORBJÖRN PERSSON

DEPARTMENT OF AUTOMATIC CONTROL
LUND INSTITUTE OF TECHNOLOGY

JUNE 1982

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

Document name
MASTER THESIS

Date of issue
June 1982

Document number
CODEN:LUTFD2/(TFRT-5276)/1-053/(1982)

Author(s)
Torbjörn Persson

Supervisor
Karl Johan Åström

Sponsoring organization

Title and subtitle
Högupplösande färggrafik för operatörskommunikation
(High-resolution colour graphics in control systems)

Abstract
This paper describes the design and construction of a microprocessor-based video display control unit intended for use in industrial control systems. The unit features graphics of up to 1024 X 1024 pixels resolution and eight colours using the NEC µPD 7220 Graphics Display Controller. The design idea is based on a discussion of basic principles in colour graphics outlined in the first part of the paper. This is followed by a detailed description of the units' construction and performance including some software.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
53

Recipient's notes

Security classification

Distribution: The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

0. Förord

Denna rapport utgör dokumentationen för ett examensarbete vars syfte varit att konstruera en videokontrollenhet för högupplösande färggrafik. Examensarbetet har utförts dels vid Institutionen för Reglerteknik på Lunds Tekniska Högskola, dels vid Telsand Elektronics AB i Arlöv.

Färggrafik är ett relativt nytt område inom man-maskinkommunikation. Av denna anledning finns det ännu ganska få allmänna regler om hur den ska utformas. Ett antal artiklar i fackpressen och några böcker har dock skrivits och den första delen av denna rapport försöker sammanfatta grundläggande fakta om färggrafik. Ur dessa får vi till sist fram en lämplig konstruktionsidé.

Konstruktionen finns redovisad i den andra delen av rapporten. Denna beskriver hur de olika detaljerna från första delen kan realiseras. Här finns också en testrapport och några förslag till förbättringar och synpunkter på konstruktionen.

Den valda hårdvarukonstruktionen innehåller kretsar som senare visade sig vara svåra att få tag på. Bland annat videoprocessorn som kom ut på marknaden först ett år efter fabrikantens första tillkännagivelse. Eftersom konstruktionsarbetet därför dragit ut på tiden har inskränkningar fått göras i den planerade mjukvaruutvecklingen för videokontrollenheten.

Jag vill här också tacka för all hjälp och idéinspiration som jag fått av personalen på Reglerteknik och Telsand, och då speciellt min handledare prof. Karl Johan Åström, Telsands Axel Westrenius och Lars Knutsson.

Lund maj 1982

Torbjörn Persson

0.1 Innehållsförteckning

1. Inledning

"En färgbild säger mer än tusen svart-vita"

Även om detta modifierade talesätt är något överdrivet belyser det vilka fördelar färgbilder har. En bild i sig har klara fördelar framför en verbal beskrivning. Med färg kan man förbättra förståelsen av bilden ytterligare.

Ända sen datorns födelse har ett stort problem varit att få ut någon vettig information av alla data som matats ut efter en exekvering. I många år har enda möjligeten varit att försöka tyda sidor av tal i olika tabeller som skrivits ut på papper av en printer. Detta har varit oöverskådligt och skulle mycket bättre kunna presenteras i någon form av diagram. I en processindustri skulle det vara helt otänkbart att försöka styra läget hos ventiler och nivåer i tankar med en pappers-utskrift som enda kommunikationsmöjlighet. Allt detta har naturligtvis skapat ett behov för andra metoder att visa datamängder. Detta har bland annat lett fram till möjligheten att använda färggrafik som detta examensarbete kommer att försöka utveckla ytterligare.

Vad kan man då uppnå med färggrafik på bildskärm som man inte kan få med andra metoder ? Ett svar är att en bild kan ge översiktlighet åt en datamängd. Det faktum att bilden finns på en bildskärm gör att man snabbt och enkelt kan ändra delar av bilden eller få fram en helt ny bild. Med färgen får man en tredje parameter att variera sin bild med.

När det gäller att snabbt hitta en viss detalj i en bild är färgen den överlägset mest betydelsefulla urvalsmekanismen. I process-industriella tillämpningar har man visat

(litt.ref 2) att färgen har en mycket stor betydelse för söktiden d.v.s. den tid det tar att hitta en viss detalj i en bild. Detta förstås under förutsättning att man innan vet vilken färg detaljen har (Bra argument för vettigt färgval i bilderna). Inom forskning och utvecklingsarbete gör färganvändningen att man lättare uppfattar komplicerade strukturer. I speltillämpningar har man fått en ny dimension att röra sig i, och dessutom är färgen i detta fall ett viktigt försäljningsargument. Gemensamt för alla tillämpningar för färgbilder är att man kan få en bättre struktur på bilden och totalt en mera "lättläst" bild.

Anledningen till att man inte använt färgbilder förrän på senare år (~1980) är inte att det saknats ett behov för det. Snarare är det så att det tidigare varit svårt och framför allt mycket dyrt att få fram en någorlunda hygglig grafisk färgbild på en bildskärm. Detta beror bland annat på att det behövs minst tre gånger så mycket minneskapacitet som motsvarande svart-vita bild. (Med svart-vita i detta sammanhang menar jag bilder som endast kan bestå av två färger, förgrunds- och bakgrundsfärg. Man kan alltså även tänka sig till exempel gult - brunt som vissa terminaler använder sig av.)

Eftersom minnesutrymmet utgör en stor del av kostnaden för ett färggrafiskt system kan man förstå att de nu snabbt sjunkande priserna på minneselement påverkat möjligheterna att konstruera avancerade sådana bildskärmssystem. Bara de två senaste åren har priset på dynamiska minnen sjunkit från ett par kronor till under 70 öre per Kbit. Samtidigt har lagringskapaciteten i en kapsel ökat från 16 Kbits till 64 Kbits.

Även på bildskärmssidan kommer det troligen att ske en prisrevolution. I nuläget kostar en normal TV-bildskärm (alltså en monitor med TV-bildrör och motsvarande kringelektronik) kring 2 000 SEK (not 1) medan en bildskärm med högre kvalitet lämplig för datoranvändning kostar från 20 000 SEK och uppåt. Detta förhållande kan emellertid komma att förändras inom de närmaste åren eftersom hemdatormarknaden har fått en explosionsartad

* not 1: SEK = svenska kronor

utveckling. Denna marknad behöver bildskärmar, och då företrädesvis färgbildskärmar. Det blir alltså mer lönsamt att göra relativt högkvalitativa skärmar och man kan med stora serier pressa ned kostnaderna för tillverkningen. En lämplig skärm skulle därför kunna närma sig vanliga TV-skärmar prismässigt sett.

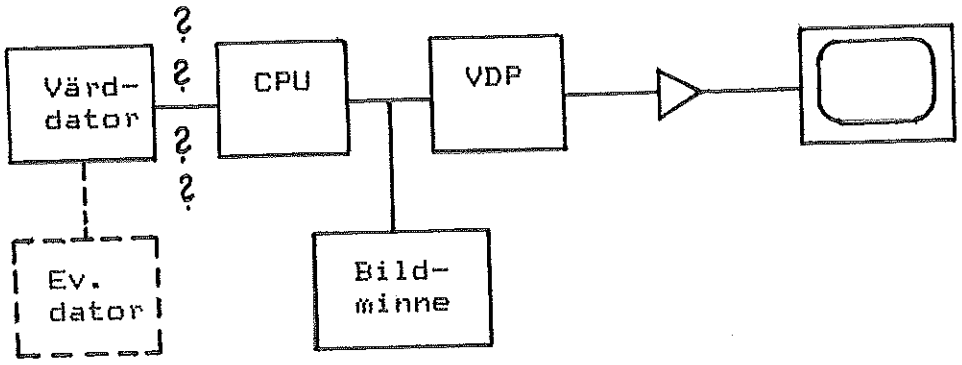
En annan faktor som påverkat färgbildsanvändningen är att det tidigare inte funnits någon speciell hårdvara för genereringen av bilden. Man har varit tvungen att bygga upp färgbildsgeneratorn av ett stor antal integrerade kretsar. En konsekvens av detta är att konstruktionen har tagit stor plats. Numera finns de flesta behövliga funktioner (synkgenerering, bildminnesadressering m.m.) samlade i en eller ett fåtal IC:s.

I detta examensarbete ingår en konstruktion av ett färgbildskärmsinterface. Alltså den del av hårdvara och, i viss mån, mjukvara mellan en dator och en bildskärm som behövs för att generera färgbilder på skärmen. Interfacet är i första hand avsett att användas i system för process-styrning och -övervakning. Det bör dock konstrueras så pass generellt att det med inga eller små modifikationer kan användas även i andra tillämpningar. Eventuella nödvändiga förändringar bör i så fall ligga i mjukvara för att förenkla ändringen.

För att veta efter vilka principer man ska konstruera interfacet ingår i denna redogörelse en diskussion av de olika delar som ingår i konstruktionen, till exempel hur man överför en bild från dator till interface på lämpligaste sätt. Framför allt måste man reda ut var gränsen mellan dator och interface ska gå (markerad med frågetecken i figur 1.1). Detta gäller till största del mjukvaran.

Vidare följer en översikt på tillgängliga videoprocessorer. Dessa är alltså den tidigare omtalade hopsamlade hårdvaran för bildgenereringen.

För att underlätta förståelsen för kommande uttryck och termer kan vi redan här föreställa oss



Figur 1.1

en tänkbar uppdelning av det totala systemet. (Se figur 1.1).

Om vi går "baklänges" så finns först bildskärmen. Denna kommer att behandlas i kommande avsnitt. Bildskärmens intensitetsförändringar styrs via en anpassningskoppling av en video-displayprocessor (VDP), eller något kortare, videoprocessor. Nu tillgängliga videoprocessorer är tämligen ointelligenta, och för att uppdatera i bilden behövs någonting som hanterar bildminnet. Detta kan vara en vanlig microprocessor. Angående bildminnet kan detta vara inkopplat på olika sätt, men mer om detta i kommande kapitel. Micro-processorn får i sin tur instruktioner från en överordnad dator om vad som ska uppdateras i bilden. Eventuellt är sedan denna dator hopkopplad med andra datorer, men detta kommer att bero på den specifika tillämpningen.

Detta var alltså bara en tänkbar lösning och den slutliga lösningen kan naturligtvis bli annor-lunda.

2. Förutsättningar

2.1 Det totala systemet

Det totala systemet omfattar i detta fall allting från någon typ av mätvärdesrapportering till bildskärmen. Vad som sedan exakt finns här emellan kommer att variera med olika tillämpningar. I ett minimalt system behövs det åtminstone någonting som kan ta emot och konvertera mätvärdena till en lämplig representation för bilden, och någonting som sedan kan presentera de bearbetade värdena på bildskärmen. I dessa system behöver man kanske inte mer än fem till tio olika bilder som är relativt schematiskt uppbyggda. I större system kan man behöva flera bildskärmar och många (i storleksordningen hundratals) olika bilder för presentationen. Till detta krävs då en effektiv bildhantering och också ett lämpligt sätt att lagra bilderna på.

Ur operatörens synvinkel ska systemet inte vara mer än ett "förlängt öga". Hon ska inte behöva ta hänsyn till om det sitter några få komponenter eller en stor komplex dator mellan mätställe och motsvarande representation på bildskärmen. En annan sak som berör operatören är att systemet ska reagera på samma sätt i alla situationer. Detta gäller speciellt vid larmsituationer. Några saker som ger exempel på detta är att man inte ska behöva vänta på en bild man vill ha fram (åtminstone inte längre än någon bråkdel av en sekund), och om man har givit fel kommando ska det gå snabbt att rätta till misstaget.

I systemet ingår också den del där man designar de bilder som används. Denna del behöver inte alltid sitta tillsammans med de delar som används för processövervakningen. I små tillämpningar behöver man oftast inte ändra på bilderna när de väl en gång fåtts att fungera. Då finns det heller inget behov av de verktyg som behövs för bild-editering. I de större systemen kan det vara

vettigt att ha direkt tillgång till bild-editeringsfunktioner.

En utförligare presentation av de olika delarna följer i kommande kapitel.

2.2 Nivåuppdelning

Det tidigare diskuterade totala systemet kan delas upp i olika nivåer beroende på delarnas komplexitet. (Se tabell 2.1).

De delar som finns upptagna i figuren ger inte en komplett bild av den totala strukturen, men de försöker visa vilken typ av funktioner som kan finnas på respektive nivå.

Nivå 1 är oförändrad i alla olika tillämpningar. Nivå 2 anpassas däremot till den specifika tillämpningen och kan i stora system tänkas uppdelad i ytterligare undernivåer. I de allra minsta systemen behövs nivå 2 inte alls. I sådana fall får nivå 1 nödvändiga kommandon direkt från nivå 3.

Nivå 3 består av två delar, dels "utveckling" dels "process". Denna uppdelning behöver inte finnas i verkligheten, men i små system kan man utesluta utvecklingsdelen från det levererade systemet.

När det gäller programvaran på de olika nivåerna ska man försöka att använda högnivåspråk i så stor utsträckning som möjligt. Högnivåspråken ger bättre struktur och mindre felmöjligheter än motsvarande assemblyprogram. Eftersom mjukvaran på nivå 2 och 3 (processidan) i de flesta fall måste anpassas till varje specifik tillämpning, behöver man ett effektivt språk på dessa nivåer. Detta gör att utvecklingstiden och därmed kostnaden kan hållas nere. På nivå 1 är emellertid uppgifterna relativt enkla och de ska inte behövas

	Utvecklings-miljö	Process-miljö
Nivå 3:	* BILD-EDITOR * BILDKOMPILATOR	* MÄTVÄRDES-BEARBETNING * (MÄTVÄRDES-RAPPORTERING)
Nivå 2:	* BILDLAGRING * KOMMANDOGIVNING TILL NIVÅ 1 * BILDÖVERFÖRING TILL NIVÅ 1 * SPECIALTECKEN SOM SKA ÖVERFÖRAS TILL NIVÅ 1 * TANGENTBORD FÖR BILDKONTROLL * KONTROLL AV FLERA BILDSKÄRMAR (MOTSV. ~ NIVÅ 1)	
Nivå 1:	* STANDARD TECKEN UPPSÄTTNING * TA EMOT SPECIALTECKEN FRÅN NIVÅ 2 * UTFÖR ENKLA KOMMANDON FRÅN NIVÅ 2 * TAR EMOT NYA HELA BILDER * HANTERAR EN BILDSKÄRM	

Tabell 2.1

ändras med olika tillämpningar när de väl fungerar. Dessutom kommer utrymmet för programvaran att vara begränsat och man kräver också snabbhet hos rutinerna. Detta gör att assembly är ett lämpligt språk här.

Kommunikationen mellan nivåerna ska vara så kort och koncis som möjligt. I tabell 2.2 finns de huvudsakliga kommunikationsvägarna uppställda. Kommunikationen mellan nivå 1 och 2 blir i normala fall helt enkelriktad. Nivå 2 har ofta flera olika nivå 1 att hantera och ge order till. Detta kräver att nivå 1 inte stoppar upp nivå 2 i dess arbete. Dessa olika synpunkter måste betänkas vid valet av

kommunikationsmetod.

2.3 Krav på hårdvaran

För detta examensarbete finns det vissa förutsättningar i valet av hårdvara. Som tidigare nämnts utföres konstruktionen hos Telsand Elektronics AB i Arlöv. Detta företag har ett komplett och mycket användbart utvecklingssystem för Texas Instruments Inc.:s produkter. Detta gör

---

NIVÅ 3 ---) NIVÅ 2

* "U" EDITERADE BILDER SOM SKA LAGRAS
* "U" NYA SPECIALTECKEN
* "P" MÄTVÄRDEN SOM SKA ÄNDRAS
* "P" ALARMSITUATIONER

NIVÅ 2 ---) NIVÅ 3

* "U" BILDER SOM SKA ÄNDRAS
* FELMEDDELANDE

NIVÅ 2 ---) NIVÅ 1

* NYA HELA BILDER
* UPPDATERING AV ELEMENT I BILDEN (NYA MÄTVÄRDEN)
* SPECIALTECKEN

NIVÅ 1 ---) NIVÅ 2

* FELMEDDELANDE

"U" = UTVECKLING , "P" = PROCESS

Tabell 2.2

---

att det krävs synnerligen starka själ att välja något annat fabrikat på de mjukvaruberoende delarna i konstruktionen. Detta gäller i första hand mikroprocessorer och kommunikationsdelar. För övriga komponenter (t.ex. videoprocessorer) finns det inga begränsningar då det gäller fabrikat.

3. Grundläggande principer

3.1 Upplösning

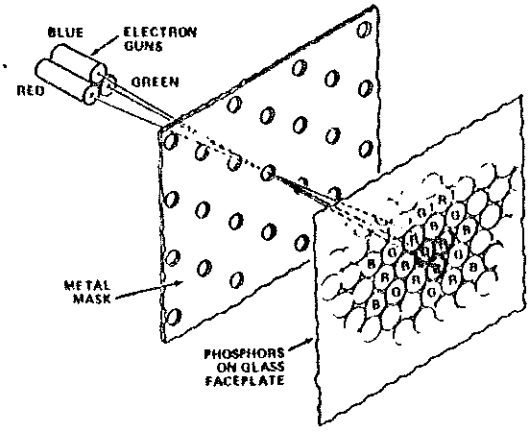
Med upplösning menas här hur många enskilda punkter som bilden kan byggas upp av (på engelska: pixels som är en sammandragning av picture elements). Upplösningen anges vanligtvis som antalet punkter per linje X antalet linjer, t.ex. 768 X 576. Förhållandet mellan dessa värden brukar vara 4:3 (vanligast) eller 1:1. Dessa punkter ska inte blandas samman med de trippel-färgpunkter som en bildskärmsyta är uppbyggd av. En sammankoppling finns dock. Antalet pixels kan inte vara större än antalet trippelfärgpunkter.

Det finns olika benämningar på upplösningsgraden. En upplösning kring 250 punkter per linje benämns "lågupplösande", kring 500 punkter kallas "mellanupplösande" och upplösningar från 1000 punkter och uppåt benämns "högupplösande". I annonser för bildskärmssystem brukar man dock kalla allting över 200 punkter för högupplösande.

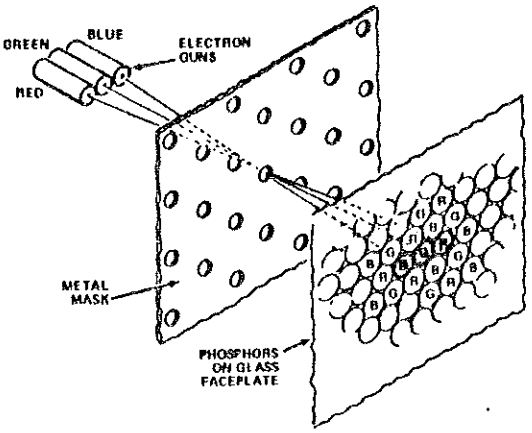
Upplösningen begränsas av i huvudsak två faktorer, nämligen, som tidigare antytts, antalet färgpunkter på bildskärmen och videoprocessorns kapacitet. Dessa båda diskuteras i detta avsnitt.

3.1.1 Monitorer

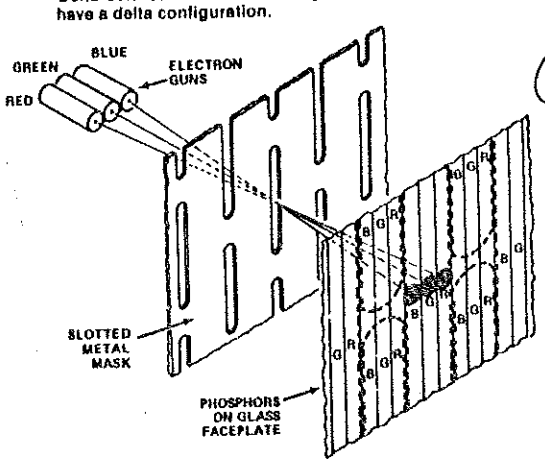
Vi ska nu se hur stor upplösning en bildskärm kan ge. Först tittar vi på hur en bildskärm och dess yta är uppbyggd. I figur 3.1 finns några olika typer av uppbyggnad avbildade. I huvudsak är det antalet trippelfärgpunkter som bestämmer hur stor upplösning man kan få. Man kan tänka sig att de övriga delarna hos monitorn, t.ex. avlänkning,



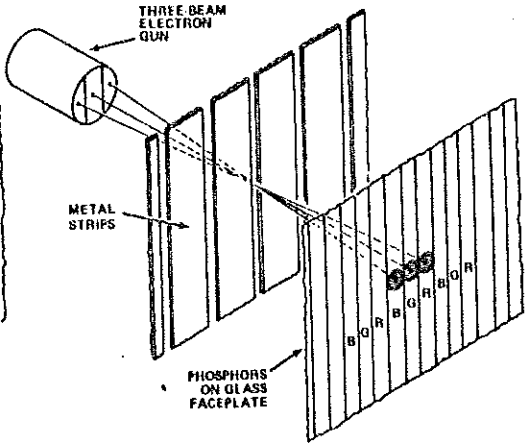
Delta-delta color CRT. Electron guns and aperture mask both have a delta configuration.



PIL-delta color CRT. In-line electron guns are combined with delta-type aperture mask.



In-line electron guns combined with slotted aperture mask and vertical phosphor stripes.



Single-lens, three-beam electron gun with a metal-strip mask and vertical phosphor stripes.

Figur 3.1

skulle kunna begränsa, men i allmänhet är dessa anpassade till skärmens möjligheter.

En normal TV-bildskärms punkter har en diameter av ca 0.6 - 0.8 mm, medan en högupplösande skärm har ca 0.2 - 0.3 mm punkter. För en monitor i mellanstorlek (18 till 20 tum) ger detta för TV-skärmen ca 500 - 600 punkter per linje respektive 1200 - 1500 punkter för den högupplösande skärmen.

3.1.2 Videoprocessorer

De idag tillgängliga videoprocessorerna och dess funktioner kommer att beskrivas till fullo i kapitlet "Teknikövesikt". Här ska vi bara se på deras upplösningsförmåga. Som vi ska se senare i det angivna kapitlet kan man dela upp videoprocessorerna i två huvudgrupper, "enkla" och "avancerade".

De "enkla" kännetecknas av en maximal upplösning av ca 256 punkter per linje. Detta medför att de kan användas tillsammans med vanliga TV-skärmar. De används därför ofta i avancerade TV-spel och hemdatorer. Den andra gruppen är avsedd för "proffesionella" tillämpningar och har upplösningar från 500 ända upp till 4000 punkter per linje. På vissa fabrikat kan man själv ange vilken upplösning man vill ha, men det finns ändå en övre gräns för dessa inom det nämnda området.

3.2 Informationstäthet

Först ska vi konstatera att det finns två olika sorters information i en bild. Den ena typen är de statiska delarna d.v.s. bakgrund, rubriker och andra tecken som definierar själva process-situationen. Den andra typen, den dynamiska, är mätvärden, diagram och dylikt som

ändras beroende på processens uppförande. Det är naturligtvis viktigt att båda dessa typer kan uppfattas snabbt och utan missförstånd, men det är speciellt viktigt för den dynamiska delen. Eftersom den statiska delen av bilden är oförändrad hela tiden kommer operatören att, efter ett tag, lära sig hur bilden är uppbyggd och var de viktiga fälten finns. Som nämnts tidigare spelar här också färgkodningen av detaljer en stor roll.

Det som i första hand bestämmer informationstätheten är alltså hur många dynamiska fält man kan ha i en bild, även om man inte kan bortse från de statiska delarna.

Det har gjorts ett antal utredningar om informationstäthet och man har kommit fram till i stort sett liknande resultat. Vi ska här se på en sammanfattning av de faktorer man behöver ta hänsyn till. I övrigt hänvisas till de nämnda utredningarna vilka finns upptagna i referenslitteraturförteckningen.

Designen av bilden kan delas upp i två problem, dels hur mycket information man kan ha i bilden, dels var i bilden denna ska placeras. Först ser vi på problemet "hur mycket". Man har gjort undersökningar av hur personer uppfattar en bild och då kommit fram till att man bör ha maximalt 25 % av den totala bildytan täckt med information. Av denna del bör inte mer än 75 % vara av den dynamiska typen. Detta ger att max 18 % av bildytan kan innehålla föränderlig information.

Angående "var" man ska placera sin information ska vi först betänka att vårt öga ser klart bara i en smal kon med ca 5° toppvinkel. Med en bilddiagonal på ca 20 tum och ett betraktningsavstånd på ca 70 cm betyder detta att man ser klart inom en cirkel med ca 6 cm:s diameter. Inom ett sådant synfält bör man inte ha mer än en enda detalj, t.ex. ett mätvärde. (Om man har två mätvärden som jämförs mot varandra är det lämplig att ha båda dessa inom synfältscirkeln).

En annan konsekvens av det begränsade synfältet är att man bör undvika detaljer som är större än denna 6 cm:s cirkel. Enligt det tidigare siffer-exemplet blir ett alfanumeriskt tecken ca 5 mm

brett (om man använder 80 tecken per rad). Då skulle ett ord kunna ha 12 tecken för att passa inom cirkeln. Normalt rekomenderas dock inte mer än sex tecken i ett ord. För överskrifter och rubriker kan man förstås ha längre ord.

Ord har en inbyggd redundans som gör dem lättare att snabbt förstå. Detta gäller emellertid inte för numeriska värden. Dessa kan tolkas rätt bara om varje enskilt tecken tolkas rätt. Därför bör man inte ha tal med mer än fyra siffror (om det krävs att man ska tolka talet snabbt). Ett sätt att ersätta eller komplettera ett numeriskt mätvärde är att använda stapeldiagram eller visare. Detta ökar avsevärt chansen att tolka värdet på rätt sätt.

3.3 Färgkomposition

Liksom i övriga delar av detta kapitel kan jag här bara ge grundläggande rekommendationer om vad som är lämpligt. Man får förlita sig på bild-designerns kunskap och sunda förnuft. En viktig punkt måste man emellertid tänka på. I process-industrin finns det ofta redan inarbetade färgkodningar av olika saker. Lika ofta är denna kodning olika från industri till industri. På ett ställe kan grön och röd färg beteckna öppen respektive stängd ventil, på ett annat ställe kan det vara tvärt om. Därför är det viktigt att man verkligen tar reda på hur färgkodningen är vid den industri man ska installera ett system. Detta för att undvika felmanövrar i krissituationer då gamla inlärda symboler och färger lätt kan förvilla begreppen.

Frånsett redan använd färgkodning, hur kan man då välja färger i bilden ? Ja, det är ganska klart att man inte ska sträva efter att göra ett kubistiskt mästerverk. Ju mer konsekvent man kan vara i färgvalet desto lättare för operatören. Samma färger för samma saker i olika bilder. Jag har tagit med ett exempel (tabell 3.2) på färgval som kan vara vettigt i många tillämpningar. Exemplet är hämtat ur en artikel i

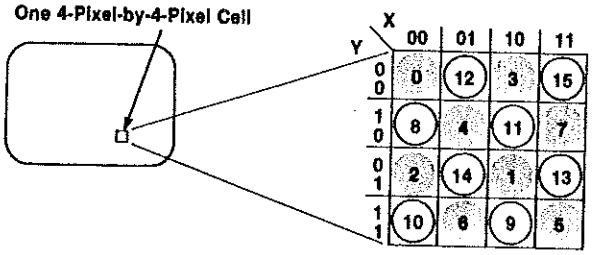
Färg	Användning
svart	Bakgrund
blå	Överskrifter, rubriker (statisk inf)
cyan	Alfanumerisk data (dynamisk inf)
grön	Från, stängd, normal
röd	Till, öppen, varning
vit	Lägen mellan grön och röd t.ex. halvöppen
gul	Uppmärksamhet krävs
magenta	Fara, omedelbar åtgärd krävs

Ur Instrumentation Technology, okt -76.

Tabell 3.2

Instrumentation Technology, okt -76, "CRT Displays for Power Plants". Som tidigare sagts får man emellertid kontrollera att några färger inte används i ombytta funktioner.

Antalet färger är givet av hur många bitar per pixel man har i bildminnet. I de tillämpningar som denna konstruktion är avsedd för är det fullt tillräckligt med åtta olika färger, vilket skulle kräva tre bitar. Med hög upplösning kan man få fler färgnyanser i fyllda areor än de givna åtta grundfärgerna. Detta åstadkommer man genom att blanda två av de åtta färgerna i bestämda mönster (se figur 3.3). På detta sätt kan man få ända upp till 4000 olika färger. Observera att detta bara gäller areor.



Area fill colors are obtained by varying the pattern in 4 × 4 arrays of pixels.

Figur 3.3

3.4 Operatörskommunikation

I systemet måste ingå en eller flera enheter som operatören kan styra bildinformationen med. Det kan gälla att välja en ny bild, ändra inställbara parametrar i processen och dylikt. Här kan man tänka sig ett antal olika metoder. Ett normalt alfanumeriskt tangentbord är kanske den vanligaste enheten i detta sammanhang. Men det kan också finnas behov av ett eller flera tangentbord som är specialgjorda för den aktuella tillämpningen. Andra enheter kan vara ljuspenna, digitizer (not 1) eller röstkommunikation. Ett smidigt sätt att peka ut en viss detalj i bilden är att använda sig av en markör. Denna kan förflyttas på olika sätt t. ex. med piltangenter, "boll", "mus", "joy-stick" eller ljuspenna.

* not 1: Inorgan utformat som en platta där man kan "rita" med en speciell penna. Används ung. som vanligt "papper och penna".

I ett mycket litet system som används endast för processövervakning (innehåller upp till ca tio bilder) behöver enda kommunikationsmedlet vara ett litet tangentbord t.ex. ett numeriskt. I detta fall behöver man ju bara kunna välja någon av de tillgängliga bilderna. I något större system med fler bilder och där man även vill ha möjlighet till styrning av processen kan ett alfanumeriskt tangentbord vara lämpligt. Med detta kan man då dels välja bilder, dels ändra parametrar i den aktuella bilden.

I många tillämpningar kan ett specialgjort tangentbord vara nödvändigt. På detta kan man ha sådana funktioner som snabbt måste åtgärdas i larmsituationer. Detta kan vara anrop efter bilder som visar larmstället, direkt manöverering av viktiga delar i processen o.dyl.. Man kan även tänka sig särskilda knappar för översiktsbilder och andra ofta använda bilder. Utformningen av detta tangentbord får bedömas för varje enskild användning.

Bollen och musen (eng. mouse) är i princip lika. Båda bygger på att man rör sin hand i relation till önskad markörförflyttning. Bollenheten är stillastående och man snurrar på bollen som styr markörens läge. Musenheten är en låda med boll eller vinkelrätt monterade hjul på undersidan. Man flyttar hela enheten på ett plant underlag. Ofta har man också några tryckknappar på lådan som kan användas till olika funktioner t.ex. för markering av valda delar i bilden.

Lite mer udda kommunikationsmedel är ljuspenna och digitizer. Dessa båda används ungefär på samma sätt med ett viktigt undantag, nämligen vinkeln på underlaget. Den normala ställningen för en bildskärmsyta är i det närmaste vertikal. Därför kan man inte stödja handen mot något när man använder en ljuspenna, d.v.s. man blir snabbt ordentligt trött i både hand och arm. Ljuspennan kan därför bara användas vid sporadiska tillfällen, och är därför mindre lämplig som huvudsaklig kommunikationsmedel. Om däremot bildskärmsytan kan läggas horisontellt får man en ställning som i likhet med digitizern är mycket mindre tröttande och därmed bättre lämpad för ändamålet. Nackdelen med liggande bildrör är deras djup som nu är vänt nedåt och konkurrerar med operatörens

knän om utrymmet, såvida man inte placerar skärmen vid sidan om operatören.

I samband med talet om liggande bildskärmar kan jag ta upp ett tänkbart användningsområde för färggrafik. På senare tid har det börjat dyka upp s. k. touch sensitive screens, d. v. s. bildskärmar med ett beröringskänsligt skikt på ytan. Då skulle man kunna designa ett tangentbord mjukvarumässigt och rita upp det på skärmen. Man får på detta sätt ett nära nog oändligt antal variationer på tangentplaceringarna och dess utformning och funktion. Man kan ha många olika sådana tangentbord i samma system där sedan operatören kan få fram den version han behöver för just det tillfället. Ett extra eller nydesignat tangentbord kräver ingen ny hårdvara och man kan därför hålla kostnaderna nere.

Röstkommunikation är i nuläget fortfarande i experimentstadiet och bör undvikas, speciellt i kritiska positioner i processstyrningen. Röstkommunikation går i två riktningar. Den ena riktningen, som kallas "voice recognition" d.v.s. operatören talar till datorn, kan emellertid vara tänkbar i mindre viktiga funktioner och då operatören har sina händer upptagna med annat. Det andra hållet, "voice synthesizing", datorn talar till operatören, kan man använda som "varningsröst" vid felkommandon och larm.

4. Teknikövesikt, videoprocessorer

På dagens marknad finns det i stort sett två typer av videoprocessorer. Som nämnts i tidigare kapitel (3.1.2) kan vi kalla dessa olika typer för "enkla" respektive "avancerade". Denna beteckning hänför sig i första hand till deras tänkta användningsområden. Tekniskt sett kan den "enkla" typen vara väl så avancerad som den "avancerade" typen. Det som skiljer är istället vilka funktioner som finns i de olika videoprocessorerna.

Vi ska börja med att se på de enkla VDPerna (Video Display Processor). Som också nämndes i det tidigare kapitlet är dessas upplösning begränsad till ca 256 bildpunkter (= pixels) per linje och ungefär lika många linjer per bildruta. Detta medför att den maximala frekvensen på videosignalen blir ca 2.5 MHz (en linje = 64 µs - sync och blankintervall = 52 µs som 256 pixels ska visas på => 5 MHz. I varje punkt behöver man bara kunna visa en halvperiod => 2.5 MHz). Med denna relativt låga frekvens kan man ha all videosignalshantering direkt på chipet, vilket också är utmärkande för den "enkla" typen.

De har ofta begränsningar i färgval, så att man får välja mellan antingen många färger (i flesta fall åtta) och låg upplösning eller maximal upplösning med kanske bara två färger. Ofta kan man inte direkt adressera varje enskild pixel, vilket gör det svårt att rita kurvor.

I övrigt kan sägas att den "enkla" typen är utrymmessnål. Oftast behöver man bara videoprocessorn och bildminne. På "ena sidan" hänger man på en mikroprocessor för styrningen och på "andra sidan" kan man ta ut videosignalen direkt från en pinne på kretsen. Beträffande bildminnet kan man med vissa VDPer direkt koppla till dynamiska minneskapslar med multiplexade adressingångar och separata in- och utdatapinnar. Detta medför att man behöver max tio kapslar till

den grundläggande hårdvaran.

Om vi går över till den "avancerade" typen kan vi konstatera att man här arbetar med betydligt högre videofrekvenser. Ända upp till 80 MHz kan bli aktuella. Av denna anledning har de "avancerade" VDPerna ingen som helst logik för videosignalens generering på chipet. Detta får byggas utanför VDPn med snabb TTL eller liknande. Istället gör flertalet av de "avancerade" VDPerna det möjligt att mjukvarumässigt bestämma video-parametrar som upplösning, synk- och blank-intervaller. Detta gör dem mycket flexibla och lätta att anpassa till olika tillämpningar.

En annan skillnad mellan de två typerna är att de "enkla" sköter all färghantering internt. För de "avancerade" VDPernas del spelar det ingen roll om man vill ha bara svart-vitt eller ett stort antal färger, eftersom all videosignalshantering finns utanför kretsen.

Nackdelen med de "avancerade" VDPerna är dess utrymmesbehov. Förutom videoprocessorn och bildminnet (som för övrigt kan vara upp till tio gånger större än för de "enkla" VDPerna) måste man ha skiftregister, buffrar, klockgenerering m. m.. För att ge en uppfattning av storleken kan man generellt säga att det krävs minst 50 ICs för den grundläggande konstruktionen.

I nästa avsnitt ska vi se lite närmare på de olika fabrikat som finns på marknaden våren 1982.

4.1 "Enkla" typen

De tillgängliga fabrikaten av "enkla" videoprocessorer finns sammanställda i tabell 4.1. Vissa av dessa ska man kanske inte kalla tillgängliga eftersom man i bästa fall bara kan få tag på provexemplar, men de finns i alla fall beskrivna i datablad och handledningar. Denna typ av VDPer ger alltså en videosignal ut från kretsen och finns därför i olika versioner för NTSC (amerikanska TV-systemet) eller PAL

Fabrikat	Typ	Upplösning
A M I	S68047	256 X 192
Motorola	MC6847	256 X 192
R C A	CDI1861/62	64 X 192
Signetics	2637	~150 X 200
Texas Instr.	TMS 9918	256 X 192
MOS Techn.	VIC 6561	192 X 200

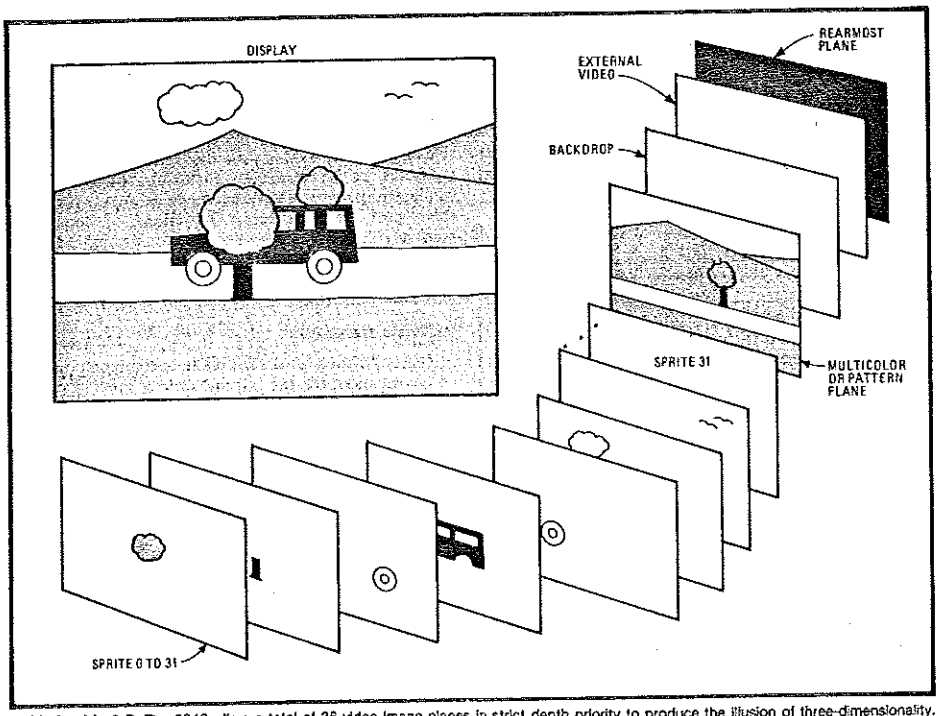
Tabell 4.1

("europeiska" systemet). Vissa kretsar har utgångar för luminans och krominans som man blandar externt till önskat videosignalsformat.

RCAs version består av två kapslar som är utförda i CMOS-teknik. Det krävs dessutom en mikroprocessor för adressgenereringen till bildminnet. Motorola och AMIs versioner är till innehållet lika men de är av någon anledning inte pinn-kompatibla. I dessa finns tolv olika moder för alfanumerisk och grafisk presentation. I alfanumerisk mod kan man välja på fyra eller åtta färger, men i den högupplösande grafikmoden (256 X 192 pixels) finns det tyvärr bara två färger tillgängliga.

MOS Technologys krets har till viss del variabelt bildformat (max 192 X 200 pixels). Även denna har begränsningar i antalet färger vid högre upplösningar. Kretsen har även en ingång för ljuspenna samt två analog-digital omvandlare. Dessutom innehåller den en ljudgenerator med tre tonkanaler och en bruskanal. Den används bland annat i Commodores hemdator VIC-20.

Texas Instruments krets används också i hemdatorer, i detta fall Texas egna 99/4. I grunden har den ungefär samma uppbyggnad som de tidigare, men man kan använda alla femton färgerna även vid högsta upplösning (256 X 192). Dessutom har kretsen en funktion som gör den till den mest avancerade i denna grupp. Som kan ses i figur 4.2 byggs bilden upp av 36 plan staplade ovanpå



Flatland in 3-D. The 9918 offers a total of 36 video image planes in strict depth priority to produce the illusion of three-dimensionality. Thirty-two planes containing objects, called sprites, combine with transparent areas that let the background show through.

Figur 4.2

varandra. I de 32 översta planen kan man ha tecken, kallade sprites, som är 8 X 8 eller 16 X 16 pixels stora. I varje plan kan det finnas en sprite och denna kan placeras var som helst på skärmen med en pixels noggrannhet genom att man ändrar dess x- och ykoordinat. Med denna funktion kan man skapa en "tredimensionell" bild där ovanliggande sprites kan "köra över" underliggande sprites utan att de sistnämnda förstörs.

Som också framgår av figuren är det möjligt att koppla in en extern videosignal som visas i motsvarande plan. Detta kan vara en annan videoprocessor eller en kamera e.dyl..

4.2 "Avancerade" typen

De olika fabrikaten av denna typ finns uppställda i tabell 4.3. Liksom för den "enkla" typen är det lite tveksamt med tillgängligheten. Vissa av dessa kretsar är kompatibla med andra fabrikat. De är då antingen helt och hållet identiska (ofta second source fabrikat) eller så är de bara pinkompatibla och har vissa olika funktioner inuti sig. Gemensamt för alla är att de genererar adresser till bildminnet och synkroniseringspulser.

Den mest finessrika VDPn kommer från NEC. Den kan efter ett fåtal kommandon själv rita linjer, rektanglar, bågar, cirklar och fylla areor. Man kan skriva valfria tecken om 8 X 8 pixels var som helst i bilden i åtta olika riktningar och med varierbar förstoringsgrad (mellan 1 och 16 ggr). Dessutom kan man zooma bilden från 1 till 16 gångers förstoring. Scrolling och panorering kan ske i två oberoende fält på skärmen. Upplösningen som är varierbar är max 1024 X 1024 pixels med fyra bildplan vilket kan ge 16 färger (om man har en VDP till varje färgplan är upplösningen max

Fabrikat	Typ	Upplösning	Ljuspenna
Intel	8275	640 X 512	Ja
Motorola	MC6845	ext.	Ja
A M I	S68045	ext.	Nej
National S.	DP8350	~900 X 512	Nej
Standard M.	CRT5027	ext.	Nej
Texas Ins.	TMS9927	ext.	Nej
Synertek	SY6545	ext.	Ja
N E C	µPD7220	2048 X 2048	Ja
Hitachi	HD46505	ext.	Ja
Thomson	EF9365	512 X 512	Ja

ext. : Upplösningen beror på externa kopplingar.

Tabell 4.3

2048 X 2048).

Thomsons krets kan också rita, men bara linjer och tecken. Den har fast upplösning på 512 X 512, 512 X 256 eller 256 X 256 pixels. Övriga kretsar har inga speciella finesser utöver sin grundfunktion. Upplösningen, som är varierbar, är mellan 256 och 1024 punkter. Vissa kretsar kräver yttre räknare och/eller speciella tecken-generatorer för att få hög upplösning.

4.3 Val av videoprocessor

Ett av kraven på konstruktionen är att den hårdvarumässigt ska vara så flexibel som möjligt. Vidare ska den kunna ta nytta av de framsteg som görs på bildskärmssidan, där man kan förvänta sig billiga högupplösande bildskärmar. Detta betyder bl. a. att man ska kunna använda samma mjukvara även om man behöver bygga om hårdvaran. Redan dessa krav antyder att den "enkla" typen av videoprocessorer med max 256 pixels upplösning är mindre intressant. Även ekonomiskt sett finns det inget som motiverar användandet av den "enkla" typen. Prisskillnaden mellan "enkla" och "avancerade" VDPer är i det närmaste lika med noll, och den merkostnad som den "avancerade" typens större bildminne utgör (1000-3000 SEK) är en relativt liten del av ett komplett systems totala kostnad. Mjukvaran kommer att bli ungefär lika omfattande för båda typerna så denna påverkar inte kostnadsbilden.

Vi kan därför koncentrera oss på den "avancerade" typen av VDPer och där är NEC µPD 7220 den mest intressanta. Den har alla funktioner som de andra tillsammans har. Dessutom har den funktioner för linje och cirkeldragning och areafyllning. Dessa kan underlätta programmering och systemarbete och därmed sänka kostnaden för systemet.

Som framgår av tekniköversikten är NECkretsens upplösning fullt tillräcklig för framtida tänkbara uppgraderingar (upp till 2048 pixels / linje)

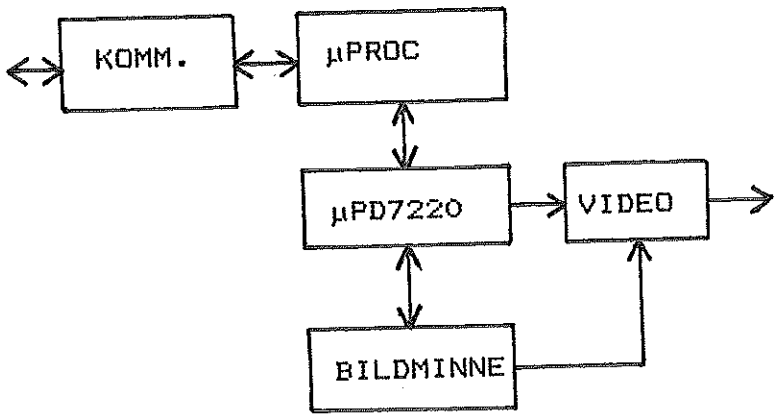
och samtliga videoparametrar kan ändras mjukvarumässigt.

Vi väljer alltså att använda NEC µPD 7220 som videoprocessor i konstruktionen. Datablad för denna krets finns i appendix 1.

5. Konstruktionen

När nu videoprocessorn är vald gäller det att bygga upp all kringutrustning till denna. Vi börjar med att i stora drag se efter vilka enheter som krävs. (Se figur 5.1).

För att styra VDPn behöver vi något som ger kommandon och parametrar till VDPn. Detta något skulle kunna vara den dator som konstruktionen ska användas tillsammans med, men eftersom VDPns kommandostruktur ligger på en relativt låg nivå skulle denna lösning belasta datorn onödigt mycket. Ett bättre alternativ är att ha en mikroprocessor med tillhörande minne som kan ta emot högnivåkommandon och sedan omvandla dessa till VDPns kommandonivå. Detta medför att värddatorn bara behöver sända ett fåtal instruktioner till videokonstruktionen och därmed hålls belastningen nere. Av samma anledning krävs det också en effektiv kommunikation mellan



Figur 5.1

värddator och mikroprocessor.

För att lagra bilden krävs det ett bildminne. Med den valda videoprocessorn är det i stort sett också bestämt hur detta ska se ut. I detta fall blir det ett minne med 16-bitars ordlängd och maximalt 256 Kord.

Vidare vill vi ha ut informationen från bildminnet till bildskärmen och detta handhas av ett antal logikkretsar som vi sammantaget kan kalla videodelen.

Dessa olika delar kommer att behandlas i de följande avsnitten.

5.1 Beskrivning av huvuddelar

5.1.1 Mikroprocessor med minne

Som tidigare nämnts bör vi välja mjukvaruberoende komponenter från Texas Instruments sortiment. I detta fall är det ingen begränsning eftersom den valda mikroprocessorn är en av de kraftfullaste på marknaden. Denna processor kallas TMS 9995 och är en 16-bitars konstruktion med multiplexad 8-bitars extern databuss. Övriga intressanta finesser är ett internt 256 bytes RAM, prefetch av instruktioner, automatiskt waitstate för långsamma minnen, signed multiply- och divide-instruktioner och inbyggd klockgenerator. Cykeltiden för en access av yttre minne är 330 ns.

Den är mycket utrymmessnål eftersom ett fungerande system endast kräver processor, en ROMkapsel med program och tre diskreta komponenter (bl. a. kristall). En mera ingående beskrivning finns i appendix 1 som är en artikel hämtad från Electronic Design, 22 nov 1980.

Till mikroprocessorn behöver vi plats för programvaran, teckenuppsättningar m. m., det vill säga minne. Eftersom det är svårt att i detta läge

uppskatta hur mycket minne som behövs, kan vi i prototypen bara göra ett preliminärt val. Det som begränsar storleken är utrymmet på det kort som prototypen byggs på. Vi bör få plats med tre kapslar i 24-28 pinnars storleken, varav vi väljer att ha två EPROM av 8K X 8 bits-modellen och en RAM-kapsel på 2K X 8. RAM-kapseln är i första hand avsedd för lagring av tecken som man enkelt vill ändra på. Som arbetsminne för programmen räcker det interna RAMet till i de flesta fall.

5.1.2 Kommunikation processor-värddator

Som tidigare nämnts måste kommunikationen ordnas så att värddatorn belastas så lite som möjligt. Detta åstadkommer man programmässigt genom att använda kommandon på "hög nivå" och hårdvarumässigt genom att värddatorn aldrig eller sällan behöver vänta på accessrätt till kommunikationsenheten. Samtidigt ska enheten uppta minimalt utrymme på kortet.

En enhet som uppfyller dessa krav är en ny krets från Texas som heter TMS 99650 MPIF (=Multi Processor InterFace). Tillsammans med en buffertkrets (20-pins) bildar denna krets (40-pins) en komplett kommunikationsenhet.

TMS 99650 består utifrån sett av två identiska portar med åttabitars databuss, trebitars adressbuss och några kontrollsignaler. Internt finns ett 256 bytes RAM som uppför sig som ett dubbelportat RAM. Detta adresseras av pekare (en för varje port) som kan ändras via databussen. RAMet kan konfigureras som en FIFO-buffert med automatisk utelåsning av ena porten när FIFOn blir tom eller full. Dessutom finns för varje port ett meddelande-, ett status- och ett kontrollregister. En fullständigare beskrivning av kretsen finns i appendix 1.

5.1.3 Bildminnet

För att kunna visa färger på bildskärmen behöver man ett visst antal bitar per pixel beroende på hur många färger man vill ha. Det enklaste sättet är att ha en bit för varje grundfärg på bildskärmen (röd, grön, blå). Detta ger åtta färger. Förutom dessa tre bitar kan det vara lämpligt med ytterligare en bit per pixel. Denna kan användas t. ex. för att invertera de andra bitarna och på så sätt markera vissa delar i bilden eller så kan biten markera blinkande fält. Funktionen byggs upp med hårdvara men kan lämpligen varieras mjukvarumässigt.

Detta ger totalt fyra bitar per pixel och alltså fyra skilda bildplan. De tillgängliga 256 Korden delas upp i fyra 64 Kordsblock som kommer att se likadana ut. När bildinformationen ska överföras till videodelen läser man ut ett ord = 16 bitar per plan till ett skiftregister som seriellt skickar iväg bitarna. Detta medför att man måste kunna läsa ut 64 bitar samtidigt. Med detta i åtanke kan vi välja lämpliga minneskapslar till bildminnet.

Med 64K X 1 dynamiska RAMkapslar får vi precis 64 stycken för hela minnesutrymmet. Dynamiska minnen ska refreshas inom en viss tid (vanligen 2 ms) och detta kan göras på flera sätt. Antingen kan man organisera adresserna så att minnet refreshas samtidigt med att datan läses ut till bildskärmen eller så kan man utnyttja den inbyggda refreshfunktionen i µPD 7220. VDPn utför detta under horisontalsynkintervallerna och man förlorar på så sätt en del av tiden för att uppdatera i bildminnet. Därför bör man om möjligt välja den första metoden.

I prototypen kommer varje minnesplan att byggas upp på ett enkelt europakort (160 X 100 mm). För multiplexingen av adresserna till de dynamiska minneskapslarna används en krets från Texas Instruments, TMS 4500 Dynamic RAM Controller, som även innehåller en refreshfunktion (se appendix 1). Denna funktion används emellertid inte för videosystemet eftersom den ibland interfererar med utläsningen av data till bildskärmen. Detta hade

då gett upphov till störningar i bilden. Minneskortet konstrueras för att kunna användas även tillsammans med det kortsystem (TLD 85) som finns hos Telsand och i detta fall sköts refreshen av TMS 4500-kretsen.

5.1.4 Videodelen

Videodelens uppgift är att överföra bild-informationen från bildminnet till en vettig signal för bildskärmen. I detta ingår också logik styrd av det tidigare diskuterade fjärde planet.

Första delen är ett skiftregister som utför en parallell till seriell omvandling av det utlästa 16-bitarsordet. Denna del finns i praktiken på minneskortet eftersom man då avsevärt reducerar ledningsdragningen. Härifrån går bitarna, en från varje plan, till logik för invertering och grindning styrd av fjärde planets bit. Grindningen styrs även av blankingsignalen från VDPn. För att kompensera skillnader i fördröjning av bildbitarna finns sedan en låskrets som synkroniseras med utskiftningen. Sist sitter drivstegen för RGB- och synk-utgångarna.

5.1.5 Övrig logik

För att kunna utnyttja VDPns möjlighet till zoomning måste man externt minska skiftregistrens klockfrekvens så att den motsvarar zoomfaktorn. Detta görs i konstruktionen genom att zoomfaktorn laddas in i en låskrets av mikroprocessorn och detta värde styr sedan en räknare som dividerar ned klockfrekvensen.

Skiftregistren ska laddas när sista biten skiftas ut och det nya dataordet finns klart att läsas in. För att få denna laddpuls rätt i tiden krävs en fördröjning av den signal som startar

utläsningen av nya dataordet.

När VDPn gör en RMW-cykel (read-modify-write) aktiverar den DBIN-signalen för att läsa in ordet. Däremot finns det ingen signal från VDPn som styr skrivningen till bildminnet, så en sådan måste genereras externt.

Alla kopplingar kommer att beskrivas utförligare i nästa kapitel.

5.2 Kopplingsschemor

Hela konstruktionen finns dokumenterad på kopplingsschemor i appendix 2. Detta kapitel ska försöka förklara och motivera de olika kretslösningarna. Dessutom tas det upp några alternativa lösningar på vissa detaljer. Konstruktionen är fysiskt uppdelad i två delar. På ett enkelt europakort finns en fjärdedel av bildminnet. Man använder alltså fyra sådana kort för att få ett fullt utbyggt bildminne. Detta kortet benämner vi minneskort. Alla övriga kretsar finns på ett eget europakort, som vi kan kalla videokort. Denna uppdelning finns också på ritningarna.

5.2.1 Videokort

Vi börjar med att titta på videokortet och då först på mikroprocessordelen. Här finns själva mikroprocessorn TMS 9995, två EPROMar typ 2564 (8K X 8), ett RAM 6116 eller 4016 (2K X 8) och en adressavkodare (krets 17). Dessutom ingår en komponentbärare för kristall och resetfunktionens resistor och kondensator. Adressbussens tretton lägsta bitar är utdragna till EPROMen och de elva lägsta till RAMet. Dessutom behöver MPIF-kretsen tre adressbitar och VDPn en bit. De återstående översta tre adressbitarna styr en avkodare som i

sin tur väljer ut en av de kretsar som är anslutna till mikroprocessorns databuss. Adresserna till dessa finns angivna i tabell 5.2.

Mikroprocessorns interna klockkrets behöver en kristall på max 12 MHz. Resetingången är av typ Schmittrigger och kan därför anslutas direkt till ett lämpligt RC-nät. Vid testningen visade det sig emellertid att funktionen var tämligen osäker varför en buffert från krets 6 (745244) lades in mellan RC-nätet och resetingången. Reset fungerade sedan som den ska. Resetsignalen används även till MPIF-kretsen.

En sak som måste iakttas är Texas Instruments "felaktiga" numrering av adress- och databitar på vissa kretsar. På TMS 9995 och MPIF-kretsen är bit 0 (noll) den mest signifikanta biten (MSB), medan den på Texas minneskretsar (och f. ö. på alla andra fabrikats kretsar också) är den minst signifikanta (LSB).

Om vi går vidare till MPIF-kretsen ser vi att denna mycket enkelt ansluts till mikroprocessorn utan någon extra logik. Port A är på mikroprocessorsidan och följdaktligen är port B avsedd för värddatorn. Här finns buffrar på alla utgående signaler d. v. s. databuss och readysignal. Riktningen på databussen styrs av värddatorns output enable-signal. När man använder MPIF-kretsens data/increment funktion måste man tänka på att inkrementeringen styrs av chip

---

Krets	Adress	Längd
EPROM 0	0000 - 1FFF	8K bytes
EPROM 1	2000 - 3FFF	8K bytes
RAM	4000 - 47FF	2K bytes
MPIF	6000 - 6007	8 bytes
Zoomlatch	8000	4 bits (LSB)
VDP	A000 - A001	2 bytes

Tabell 5.2

---

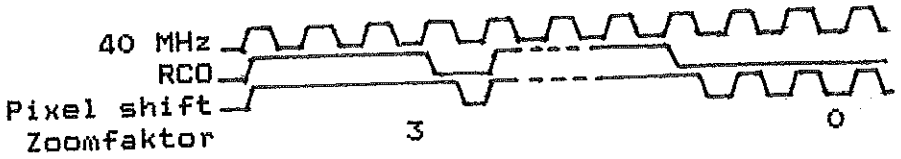
enable-signalen. Om man har flera CE-pulser för en instruktion kommer adresspekaren att räkna upp motsvarande antal steg. Observera även den "felaktiga" numreringen på MPIFs adress- och databussbitar.

Vi övergår nu till klockdrivningen för videodelen. För att skifta ut pixelbitarna till bildskärmen behöver vi en klocka med pixelfrekvensen 40 MHz (krets 24). Med denna frekvens får vi full upplösning, 1024 X 1024 pixels, med en bildfrekvens på ca 27 Hz (interlaced). Med upplösning på 1024 X 768 (4:3 som är förhållandet mellan sida och höjd för en normal bildskärm) ökar bildfrekvensen till ca 35 Hz vilket ger en flimmerfriare bild än den förra. Klockgeneratorn är en TTL-krets 74S124 och det frekvensbestämmande elementet är en kristall på 40 MHz.

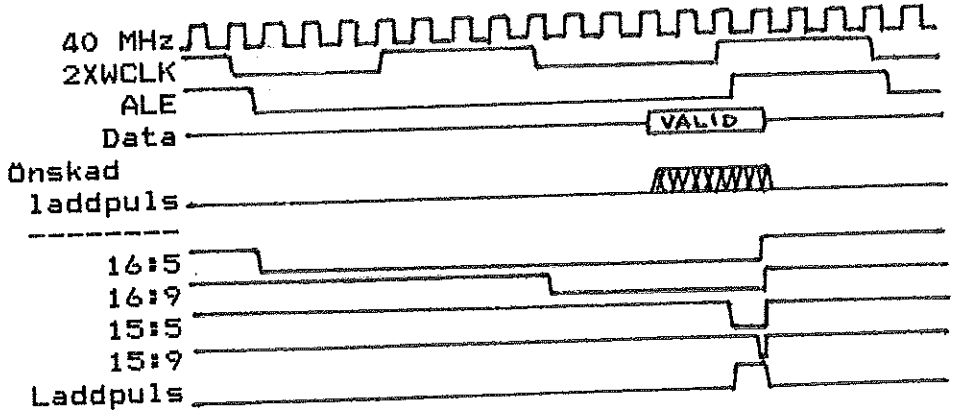
40 MHz dividerat med 8 ger 5 MHz som används till VDPns 2XWCLK-ingång. Detta är den högsta rekommenderade frekvensen och det medför att VDPn kan rita figurerna med maximal hastighet (= 800 ns per bit). Divisionen sker i kretsen 20 och utsignalen går via en buffert till ett antal kretsar. VDPn kräver att hög nivå på klockan är över 3.9 V, vilket kräver ett pull-up motstånd (24:1-18) till +5 V. Vidare finns en resistor (24:2-17) för att minska undershoten.

Om man inte vill använda sig av zoomningsmöjligheten kan man skicka ut 40 MHz-signalen direkt till skiftregistrens klockingångar. Om däremot zoomningen ska fungera måste pixelfrekvensen divideras ned med den aktuella zoomfaktorn. Detta sker genom att zoomfaktorn laddas in i krets 19, som egentligen är ett skiftregister men som här bara utnyttjas som låskrets. Zoomfaktorn laddas sedan in i en räknare som räknar ner till noll i takt med 40 MHz-klockan. Då går RCO (Ripple Carry Out) låg och genererar en pixelskiftpuls. Samtidigt laddas zoomfaktorn in i räknaren på nytt. Grindarna 13 och 7 behövs när zoomfaktorn är noll. I detta fall är RCO låg hela tiden och skulle utan grindarna inte ge några pulser. Se pulsschemat i figur 5.3.

När ett sextonbitars ord ska läsas ut från bildminnet till skiftregistren ska detta ske enligt den övre delen av figur 5.4. När ALE går

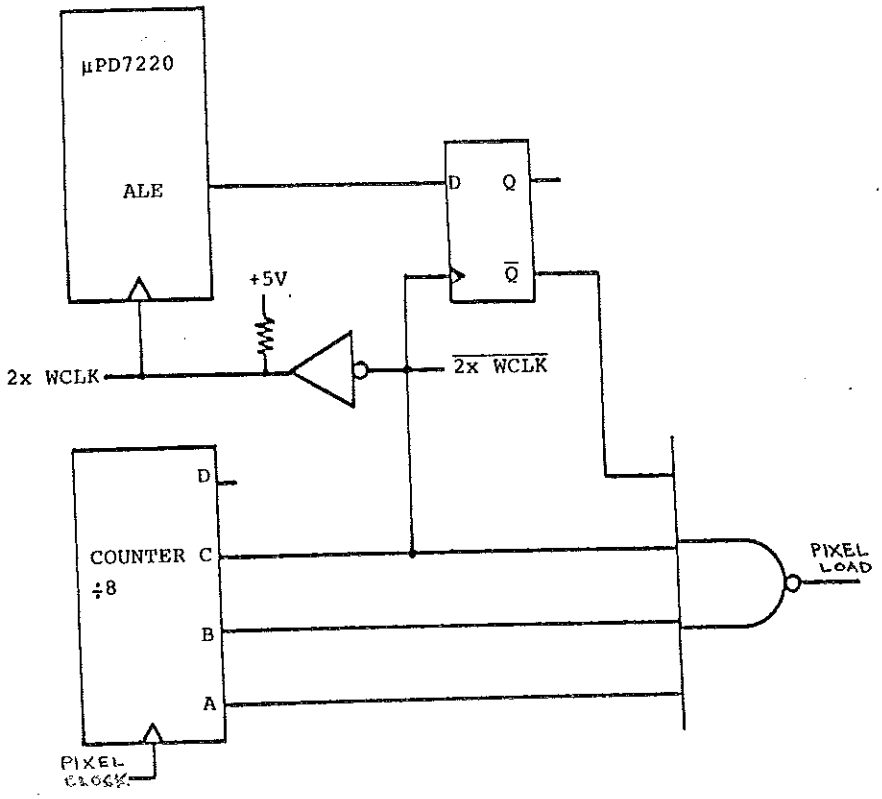


Figur 5.3



Figur 5.4

låg startar en minnesaccesscykel. Efter minnets accesstid finns dataordet klart på ingångarna till skiftregistren. Det finns kvar här tills ALE går hög plus ytterligare en tid på grund av fördröjningar i buffrar och adress-strobs-generering. Laddpulsen bör komma så sent som möjligt eftersom man då kan utnyttja långsammare minnen som är billigare. Den här valda lösningen utgörs av kretsarna 15 och 16. Dessa utgör totalt fyra D-vippor som via klockning av ALE och 2XWCLK ger en laddpuls enligt nedre halvan i figur 5.4.

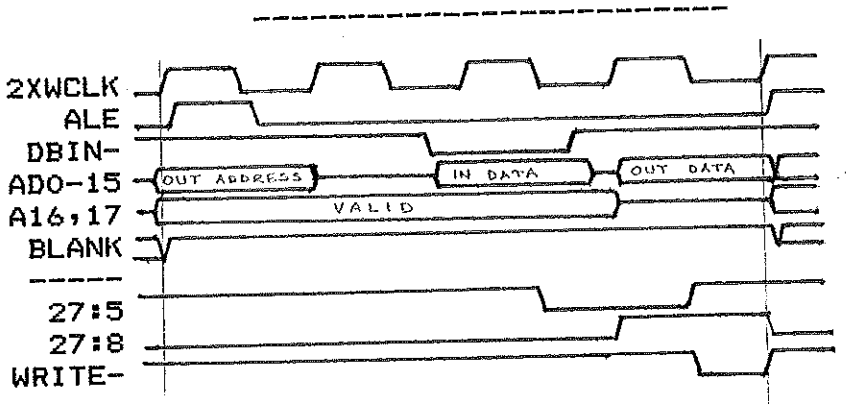


Figur 5.5

Laddpulsen kan också genereras med kopplingen i figur 5.5. Nackdelen är att pulsen kommer en pixelklockperiod tidigare än i den valda kopplingen, och eventuellt krävs därför snabbare minneskretsar.

När man vill ändra något i bilden sker detta via VDPn. Denna utför en read-modify-writecykel (RMW) enligt övre delen i figur 5.6. Anledningen till RMW-förfarandet är att man alltid arbetar med 16-bitars ord. De bitar som inte ska ändras måste ändå läsas in i VDPn och sedan skrivas tillbaks oförändrade. Hur modifieringen går till i VDPn framgår av appendix 1. Här ska vi se vad som händer utanför VDPn.

För att visa att VDPn vill göra en RMW-cykel låter den DBIN gå låg. Denna signal kan direkt styra en output enable-funktion på minneskorten



Figur 5.6

för inläsningen. Efter ytterligare en VDP-klockperiod läggs det modifierade dataordet ut på bussen och i detta skede måste man generera en write-puls externt. Eftersom enda gången man vill ha en write-puls är när DBIN precis tidigare varit aktiv använder vi DBIN för att styra två D-vippor (krets 27). Dessa klockas av 2XWCLK och hela tidsförloppet framgår av nedre delen av figur 5.6.

De två översta adressbitarna från VDPn (A16 och A17) används för att välja ut ett av de fyra minneskorten. Tyvärr är dessa bitar inte korrekta den sista klockperioden i RMW-cyklerna, varför man måste låsa dem i krets 12b. (Anledningen till b:et i numreringen är att kretsen lagts till i efterhand. I den första specifikationen på VDPn angavs nämligen felaktigt att A16 och A17 var korrekta hela cykeln.) Krets 12a är en avkodare som utför valet av minneskort.

Kretsarna 2 och 3 driver adress- och databussen till minneskorten. De kan stängas med en yttre enable-signal om man skulle vilja ha direkt åtkomst till bildminnet från någon annan enhet.

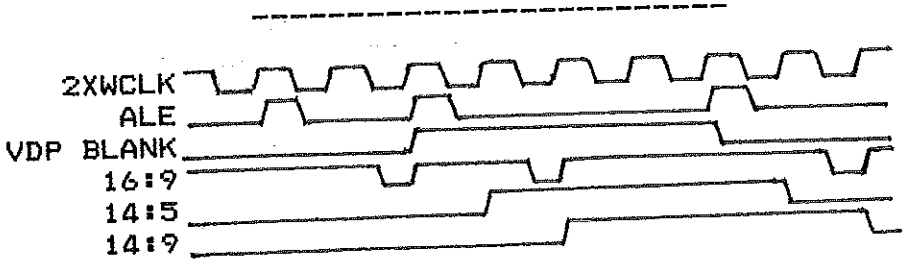
Det som återstår på videokortet är hanteringen av pixelbitarna. Dessa kommer, en från varje plan, i takt med pixelskiftklockan. De tre RGB-bitarna går först till en EXOR-grind. Denna utför en invertering av bitarna om biten från det fjärde planet (i schemat betecknad med X) är satt, och

man valt att utnyttja denna funktion genom att strappa 24:10-11. Vidare i behandlingsvägen sitter en NAND-krets som också den kan styras av X-signalen (via strap 24:9-11). Den styrs dessutom alltid av blankingsignalen från VDPn. Denna är aktiv under de mjukvarumässigt specificerade blank- och synkintervallerna.

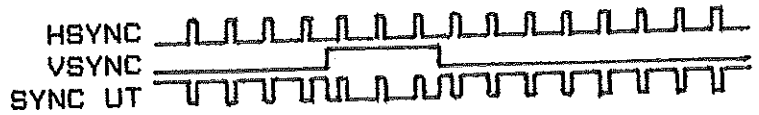
Utskiftningen av pixelbitarna ligger "ett steg efter" VDPns utläsning av dataordet och därför måste blankpulsen fördröjas med motsvarande tid. Detta görs med krets 14 vars tidsdiagram finns i figur 5.7.

Från NAND-grindarna går RGB-informationen till en låskrets (30) som klockas av pixelskiftpulserna. Denna krets förhindrar att eventuell olika fördröjning i tidigare steg blir märkbar på skärmen.

Som sista steg sitter för varje utgång en line driver av typ 74S140. Tillsammans med en resistor på 300 ohm ger dessa en utsignal på 0 till 2 volt vilket är standard för videosignalsöverföring med RGB-uppdelning. Synksignalen avslutas på samma sätt men utan resistor eftersom man här vill ha en nivå mellan 0 och 4 volt. Före slutsteget har de separata vertikala och horisontala synkutgångarna från VDPn mixats ihop i en EXOR-grind. Denna gör att man får inverterade horisontalsynkpulser inuti vertikalsynkpulsen. På så sätt får man en stabilare synkronisering av horisontalavlänkningen i monitorn. Se pulsschema i figur 5.8.



Figur 5.7



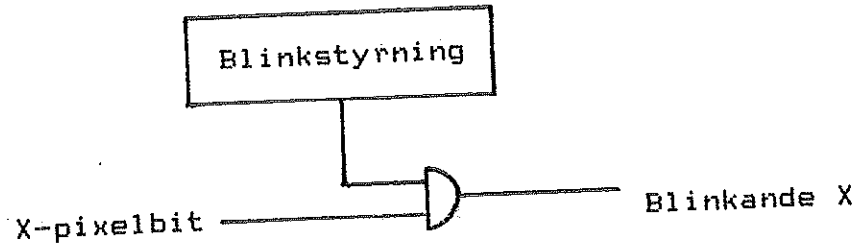
Figur 5.8

På schemat finns även en ingång för ljuspenna. Kravet för att VDPn ska registrera ljuspennans position är att samma punkt på bildskärmen i två efterföljande bildscan ska ge upphov till pulser på ingången. Denna funktion har ej testats.

Om vi återgår till informationen från det fjärde planet ska vi se på en annan möjlighet att utnyttja det på. Man kan låta detta plan markera blinkande fält i bilden. Hårdvarumässigt kan detta lösas på flera sätt, men grunden är en koppling enligt figur 5.9.

De ställen i bilden som ska blinka markeras med ettor i fjärde minnesplanet. Blinkstyrningen pulsar mellan ett och noll och utsignalen kan sedan styra en EXOR- eller NAND-grind precis som i den tidigare lösningen. Blinkstyrningen kan göras på olika sätt. En mycket flexibel lösning är att använda en latch som sätts och nollställs av mikroprocessorn. Detta gör att man kan välja blinkintervaller helt godtyckligt genom att ändra i mjukvaran. Nackdelen kan vara att man får ytterligare en sak att hålla reda på i sina program.

Man kan även ordna blinket helt med hårdvara. Då sätter man in en räknare som klockas av vertikalsynkutgången. Denna signal är normalt på ca 50 Hz och med en fyra eller fembits räknare får man då ca 3 respektive 1.5 Hz blinkfrekvens. Dessutom blir blinkningen synkroniserad med bildfrekvensen. Man kan också tänka sig att styra räknaren från mikroprocessorn på samma sätt som för zoomningsfunktionen och på så sätt få variabel blinkfrekvens.

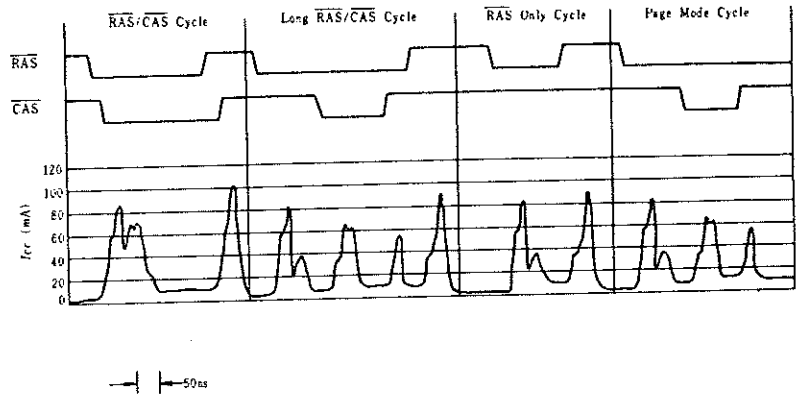


Figur 5.9

5.2.2 Minneskortet

Minneskortet är enklare att beskriva än videokortet, men i gengäld ska vi se på två något olika versioner. Först beskrivs den version som diskuterats tidigare, alltså ett kort med en fjärdedel av det totala minnet och där varje kort styr en grundfärg plus x-planet. Sedan ska vi se vilka förändringar som behövs för att köra systemet med bara ett minneskort som ändå rymmer fyra plan. (Naturligtvis blir det totala minnet också fyra gånger mindre och därmed blir upplösningen på skärmen hälften så stor. (Max 512 X 512 pixel)). Anledningen till den senare konfigurationen är att man får ett billigare och kompaktare system för de tillfällen då man kan nöja sig med lägre upplösning.

Vi börjar med att titta på den första varianten. Den mest dominerande delen är de sexton dynamiska minneskapslarna. Alla dessa har den multiplexade adressbussen på åtta bitar, RAS-, CAS-, och Write-signalerna gemensamma. De återstående pinnarna, data in och data out, är kopplade så att varje kapsel utgör en bit i sextonbitarsordet. En sak att notera är att varje minneskapsel har en kondensator kopplad över spänningsmatningen. Dessa kondensatorer finns där beroende på de dynamiska minnenas stora variationer i drivströmmen (Se figur 5.10).



Figur 5.10

De dynamiska minnenas datautgångar blir aktiva när CAS går låg och förblir så tills CAS åter går hög. Detta medför att under en RMW-cykels sista del (write) är både minnenas och VDPns datautgångar aktiva. Eftersom två aktiva utgångar inte kan kopplas samman sätter vi in en buffert (kretsarna 24 och 25) mellan dessa utgångar. Bufferten styrs av VDPns DBIN-signal så att dess utgångar bara är aktiva under VDPns read-faser. Ett annat sätt man kan lösa problemet på är att utnyttja de dynamiska minnenas "early write"-mod. Denna fungerar på så sätt att om write-ingången aktiveras före CAS blir datautgångarna aldrig aktiva under den minnescykeln. Vid RMW-operationer är denna metod svår att använda eftersom CAS måste gå låg redan när utläsningen görs. CAS skulle sedan behöva gå hög igen innan skrivdelen kunde göras enligt "early write"-sättet.

Dessutom måste man ändå ha en buffert om man har mer än ett minneskort kopplat till samma buss eftersom alla korten aktiveras vid pixeldata-utläsningen.

Mellan minneskapslarna och bufferten tar man ut pixeldatan till skiftregistren. Dessa består av två seriekopplade 745299 TTL-kretsar, som är ett åttabitars skiftregister med parallella in- och utgångar och möjlighet att skifta i båda

riktningarna. I konstruktionen utnyttjas bara de parallella ingångarna och en seriell utgång samt skiftning åt ett håll. De har en garanterad övre skiftfrekvens på 50 MHz och har alltså en marginal till de 40 MHz som pixelskiftklockan är på. Skiftklocka och laddpulser framgår av tidsdiagrammet i avsnittet om videokortet.

Om vi nu övergår från datahantering till adresshantering ser vi först att dessa två delar kan kopplas samman med strapparna 26 och 27. Detta beroende på att kortet ska kunna användas till dels system med multiplexad adress- och data-buss, dels system med separata sådana. VDPn har adresser och data på samma buss så i detta fall använder vi strapparna och kopplar VDPn till en av bussarna på kortkontakten.

Adressen från VDPn går till TMS 4500-kretsen och är här kopplad så att den minst signifikanta halvan utgör radadress till minnena. Denna koppling gör att radadressen räknas upp ett steg i taget när datan utläses till skiftregistren. Eftersom radadressen också fungerar som refreshadress vid en minnescykel kommer därför minnet att vara helt refreshat efter 128 minnesaccesser. Med upplösningen 1024 pixels per linje har man 64 accesser per linje och det behövs alltså två linjer för fullständig refresh. Två linjer tar max 100 µs som ligger väl innanför max tillåten refreshtid som är 2 ms. Även vid vertikalblankingen, då ingen utläsning sker, klarar man tidskraven eftersom denna tar max 1.3 ms. Om man tänker använda lägre upplösning eller zoomning (då minnet inte scannas sekvensiellt) måste man kontrollera att detta refreshsätt fungerar, annars får man utnyttja den inbyggda refreshmöjligheten i VDPn.

Från TMS 4500 går adressen, halva i taget, ut till de sexton minnena där de latchas in med RAS- och CAS-pulserna som genereras i 4500. Denna krets kan direkt driva upp till 34 adressingångar och man behöver därför ingen speciell drivkrets häremellan.

Selekteringen av minneskorten måste kunna ske på två sätt. När man ändrar i bilden ska man kunna välja ut ett av de fyra korten och när pixeldatan ska läsas ut ska alla fyra kort aktiveras. Detta

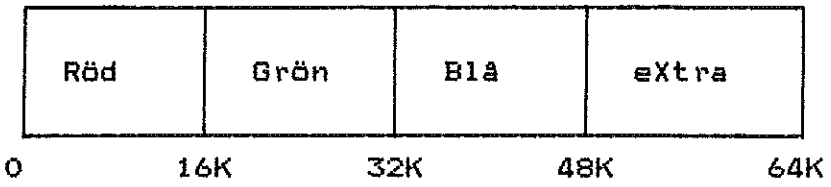
kan man lösa genom att låta ALE-signalen från VDPn starta en minnescykel på alla korten. Om den aktuella cykeln är en pixeldatautläsning laddas sedan skiftregistren på vanligt sätt. Om det är en RMW-cykel väljs ett av korten med card select-signalerna från videokortet. Som vi ser på kopplingschemat öppnar denna signal grindarna för DBIN och Write (krets 29). (Write-signalen går sedan till minnena via en buffert (krets 28) eftersom denna signal driver alla sexton minneskapslarna.) De övriga planen utför bara en pixeldatautläsning och denna har ingen betydelse eftersom blanksignalen från VDPn är aktiv under alla RMW-cykler. Detta förhindrar att oönskad data når skärmen.

När kortet används som ett vanligt minneskort i ett kortsystem har man normalt bara en signal att styra kortet med. Därför använder man strap 29:8-9 som kopplar samman ALE- och CS-ingångarna. En aktivering av en av dessa ingångar kommer då att både starta en minnescykel hos 4500-kretsen och öppna grindarna för DBIN och Write.

Den andra versionen av minneskortet finns uppritad på ett kopplingschema i appendix 2. Detta är inte fullständigt utan bara de förändrade delarna är redovisade. Resten är samma som i förra versionen.

Grundprincipen när vi nu ska använda ett kort är fortfarande att man har fyra pixelbitsplan, kan läsa ut 64 bitar på ett sätt så att en pixels alla bitar kommer fram samtidigt till videokortet och att minnesplanen är organiserade så att man kan utnyttja figurritningen i VDPn. Vi organiserar minnet enligt figur 5.11.

Nu kan vi utnyttja figurritningen precis som tidigare, eftersom VDPn för detta förutsätter ett sekvensiellt ordnat minne enligt figur 5.11. Vid pixeldatautläsning lägger VDPn också ut adresserna sekvensiellt, och på något sätt måste vi "lura" adresserna att hoppa mellan planen. Vi vill alltså kunna läsa ut orden i ordningen: Rött plans första ord, Grönts första, Blåtts första, eXtras första, Rötts andra, Grönts andra o. s. v.. Om vi tar bort de två lägsta adressbitarna ser vi att resten räknar upp efter denna ordning. För att välja plan använder vi de två lägsta bitarna (se tabell



Figur 5.11

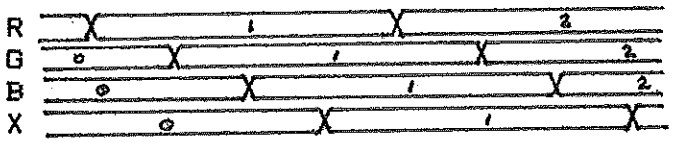
5.12).

I praktiken är detta lika med att skifta hela adressen två steg nedåt och flytta A1 och A0 till A15 resp. A14. Detta gör vi i konstruktionen med fyra fourbits data selectors (74LS157). Valet mellan de två adressmoderna styrs av blanksignalen från VDPn. Denna är alltid låg vid pixelutläsning och hög annars. Nu kan vi alltså läsa ut ett sextonbitarsord från varje plan. Dessa ord måste laddas in i skiftregistren, ett (16-bits) för varje plan. För att välja vilket register som ska laddas styr vi en decoder med den vanliga laddpulsen och de adressbitar som väljer ut planen (A0 och A1).

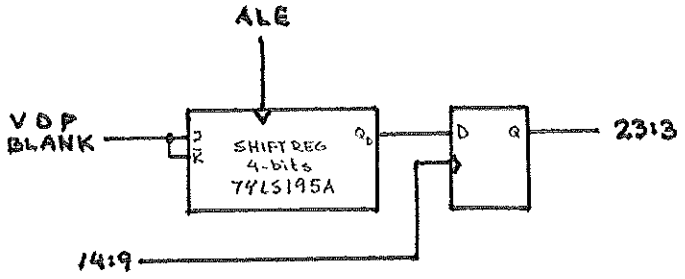
För att få ut en pixels alla bitar samtidigt fastän de laddas in vid olika tidpunkter, måste vi fördröja det först inladdade ordet tolv pixelklockpulser, det andra ordet åtta pulser och det tredje fyra. Detta görs med skiftregistren 30, 31, 32 och 33. Se figur 5.13. Dessutom måste fördröjningen av blankpulsen förlängas med tolv klockpulser. Detta görs på videokortet enligt figur 5.14. För att läsa ut 64 bitar och skifta ut dessa i grupper om fyra måste även pixelklockan divideras med fyra. Detta görs enkelt genom att sätta zoomlatchen till tre.

A15														A0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Röd första
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Grön första
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	Blå första
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	eXtra första
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	Röd andra
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	Grön andra
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	. . . . .

Tabell 5.12



Figur 5.13



figur 5.14

6. Vad blev det ?

I detta kapitel ska vi se om det verkligen blev det vi ville ha av konstruktionen. Först testningen för att se om konstruktionen fungerar. Därefter ska vi se om det finns några detaljer som kunde konstruerats på ett bättre sätt. Till sist en sammanfattning av det hela.

6.1 Testning och mjukvara

Detta kapitel handlar om både testningen och mjukvaran av den anledningen att den mjukvara som skrivits har använts vid testningen. Mjukvaran består av dels program i assembly och BASIC som använts för att testa konstruktionens funktion och VDPns kommandouppsättning, dels av ett demonstrationsprogram skrivet i assembly som visar VDPns snabbhet och möjligheter att rita figurer.

Vid den första delen av testningen användes ett enkelt assemblyprogram för mikroprocessorn på videokortet. Detta finns återgivet i appendix 3. Programmet initierar VDPn och hämtar sedan eventuella vidare kommandon från MPIF-kretsen. Dessa kommandon är av VDP-struktur och de skickas i stort sett direkt till VDPn. Den enda avkodning av kommandon gäller ändring av zoomfaktor där zoomlatchen måste sättas till motsvarande värde, samt uppritning av tecken som finns lagrade i EPROM 1. Som värddator har en Commodore VIC-20 bordsdator använts. Ett interaktivt BASIC-program har härifrån skickat kommandona till MPIF-kretsen. Detta program finns ej listat i denna skrift, men principen är att koden för det valda kommandot POKEats i en minnesposition där MPIF finns (POKE A,X betyder lägg värdet X i minnesposition A).

Vid denna testning upptäcktes bland annat, som tidigare nämnts, att resetfunktionen fungerade tvivelaktigt. Vidare fanns det vissa problem med överföringen mellan VIC-20 och MPIF-kretsen, orsakade av en för lång kabel häremellan. I övrigt fungerade kretsar och kommandon som de skulle.

Vid den första delen av testningen gjordes all kommandohantering i BASIC och var därför tämligen långsam. För att få en uppfattning om snabbheten i uppdatering vid bildbyten och figurritning kördes därför ett rent assemblyprogram (appendix 3) på videokortet. Detta program består av ett antal subrutiner för linjedragning, areafyllning, teckenskrivning, tidsfördröjning m. m.. Dessa subrutiner kan kallas in från ett huvudprogram som hämtar kommandosekvensen från en tabell DEMTAB. Kommandona i denna tabell är av typ LINE,320,65,45,258,RED (dra en röd linje från (320,65) till (45,258)). Genom att ändra i DEMTAB kan man enkelt göra om eller lägga till bilder. Tillgängliga kommandon och deras parametrar finns också i appendix 3.

Demonstrationsprogrammet kördes på ett system med ett minneskort och upplösning mellan 300 och 500 pixels per linje och motsvarande antal linjer.

Denna senare del av testningen visade att en areafyllning av ett 320 X 360 pixels område tar omkring en halv sekund. På denna tid har VDPn alltså ändrat på samtliga bitar i den del av bildminnet som visas på skärmen. Om vi nu kommer ihåg det tidigare resonemanget om att man inte bör ha mer än 25 % av bildytan täckt med information skulle detta medföra att en bild kunde ritas upp på ca en tiondels sekund. Detta förutsätter att mikroprocessorn hinner med att ge VDPn instruktioner i motsvarande takt.

6.2 Vidareutveckling

Vi ska här se på några saker som kan utvecklas mer än den givna konstruktionen. För det första är uppbyggnaden på enkla europakort (160 X 100 mm)

diskutabel. Dessa kort placeras i en rack där även andra kort kommer att sitta. Alla förbindningar mellan korten (t.ex. adress- och databuss) sker i bakplanet av racken. Dessa förbindelser bör vara standardiserade för alla rackar och alla kort, så att man hårdvarumässigt inte behöver låsa ett in/ut-kort till en bestämd plats. Videokortet och minneskorten behöver en egen adress- och databuss vilket inte överensstämmer med nämnda filosofi. Därför bör man ha hela videosystemet på ett kort med standardkoppling till rackens bakplan.

Nu är det omöjligt att packa in alla kretsar på ett enkelt europakort, men på två dubbla europakort (233 X 160 mm) får allt plats. På det ena kortet har man då kretsarna motsvarande videokortet plus ett minneskort kopplat enligt den senare beskrivna metoden för bildminnet d.v.s. fyra bildplan på ett kort. Detta kort ska kunna köras ensamt med lägre upplösing. Det andra dubbla europakortet har de övriga tre minnesplanen och kan mekaniskt kopplas samman med det första kortet till en sandwich-konstruktion. Elektriskt kopplas korten samman med t.ex. flatkabel så att man inte behöver bryta upp kopplingarna i rackens bakplan.

En annan förbättring i konstruktionen gäller behandlingen av pixelbiten från fjärde planet i videodelen. För att välja funktion hos denna bit används nu strappar som måste flyttas mekaniskt. Ett bättre sätt är att ha en latch, styrd av mikroprocessorn, som väljer funktionen. Detta kan realiseras med en latch och några NAND-grindar. På detta sätt kan man mjukvarumässigt bestämma om man vill ha blinkande fält, inverterade fält e.dyl..

6.3 Sammanfattning

Tro aldrig på en försäljares leveranstider. Detta kan vara sensmoralen av examensarbetet efter alla de fördröjningar som varit. Nåväl, konstruktionen blev till sist klar och visade sig fungera som den var avsedd att göra. Inga speciella svåra och tidsödande problem dök upp i själva hårdvarudelen, medan det på programsidan

varit något besvärligare. Detta beror till största del på att det inte funnits någon emulator för den nya mikroprocessorn TMS 9995. Programmen har fått lagras i EPROM vilket betyder att det varit svårare att hitta och tagit längre tid att ändra fel i dem.

Prestandamässigt hävdar sig denna konstruktion väl i förhållande till tänkbara konkurrenter. Upplösningen är varierbar från noll till 1024 pixels i både vertikal och horisontell led och VDPn klarar av att rita upp en bild med högsta upplösning och 25 % täckning på mindre än en sekund. Tack vare mikroprocessorn på videokortet kan man hålla kommunikationen till enheten på en hög nivå vilket ger en låg belastning på värddatorn. Dessutom kan man enkelt ändra videoenhetens "kunnande" om så skulle vara önskvärt.

Innan denna videoenhet är klar att säljas på öppna marknaden krävs en avsevärd mjukvaruutveckling, dels för mikroprocessorn på videokortet, dels på nivåerna ovanför. Det är dock min förhoppning att detta med hjälp av den gjorda konstruktionen uppbyggnad och en lämplig nivåuppdelning inte ska bli alltför betungande.

APPENDIX

1. Beskrivningar på använda kretsar

NEC µPD 7220	: Ur Electronics 7 apr. 1981
	: Utdrag ur datablad
TMS 9995	: Ur Electronic Design 22 nov. 1980
TMS 99650	: Ur Electronic Design 12 nov. 1981
TMS 4500	: Utdrag ur datablad

2. Kopplingsscheman

Videokort höger del
Videokort vänster del
Minneskort 64K X 16 bits
Minneskort modifierat (endast förändringar)
Komponentlista videokort
Komponentlista minneskort

3. Program

Parameteröverföring för första testdelen
Demonstrationsprogram i assembly

4. Litteraturförteckning

APPENDIX 1

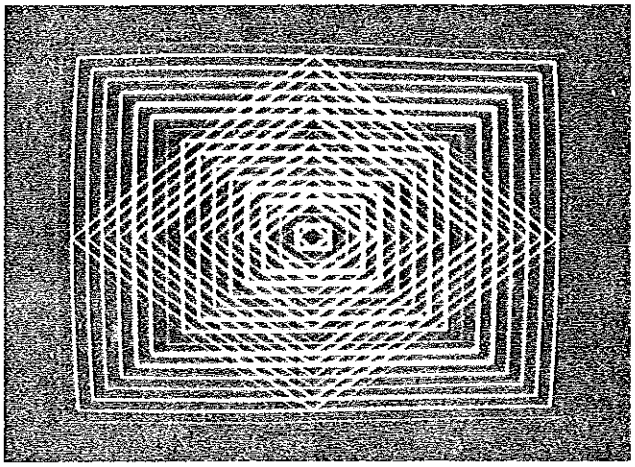
Beskrivningar på använda kretsar

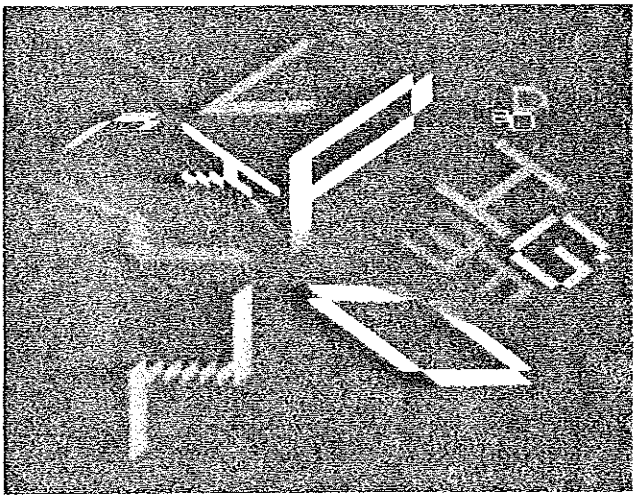
NEC µPD 7220	: Ur Electronics 7 apr. 1981
	: Utdrag ur datablad
TMS 9995	: Ur Electronic Design 22 nov. 1980
TMS 99650	: Ur Electronic Design 12 nov. 1981
TMS 4500	: Utdrag ur datablad

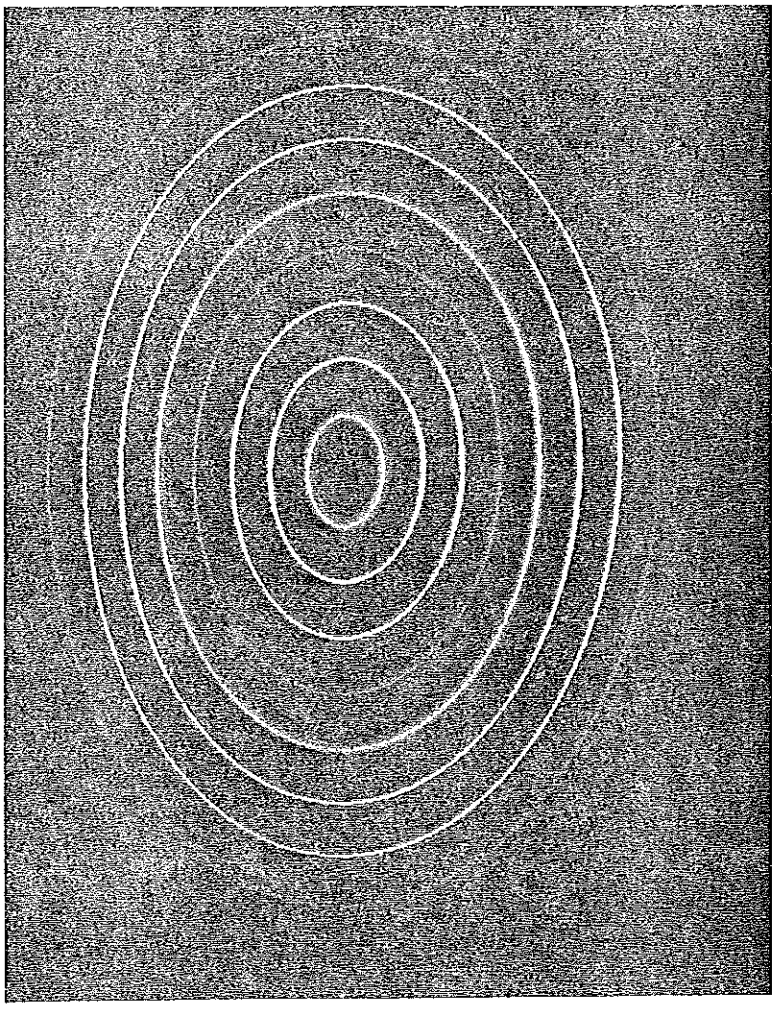
Display controller simplifies design of sophisticated graphics terminals

Dedicated chip makes high-density color displays with alphanumerics and figure-drawing capabilities an economical proposition

by Jeffrey L. Wise and Henryk Szejnwald
NEC Microcomputers Inc., Wellesley, Mass.





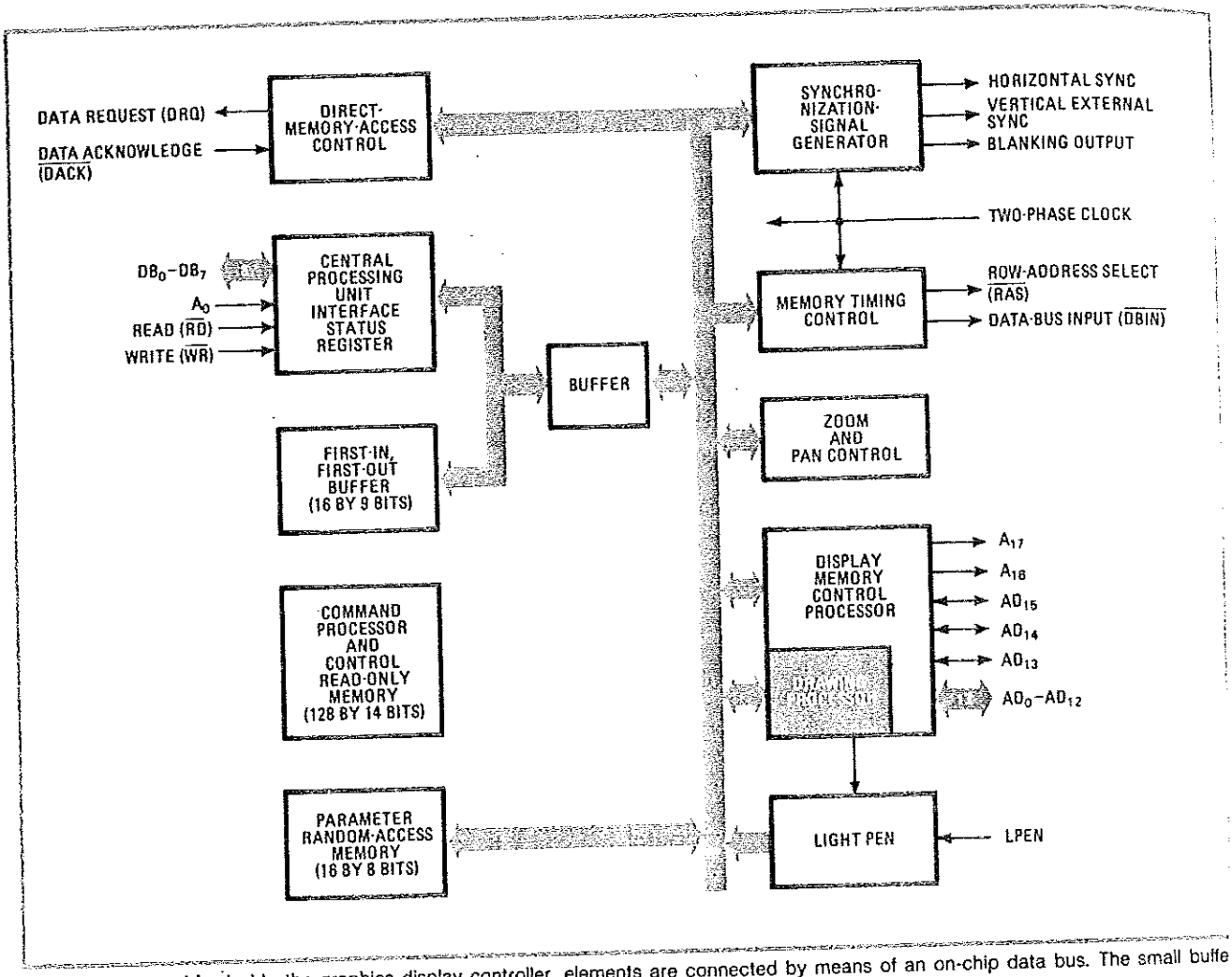


☐ The ongoing computerization of society brings with it the need for ever more sophisticated cathode-ray-tube displays. But lagging display controller technology has kept the price of high-resolution color graphics terminals high. Now, market demand has reached the threshold at which large-scale integration of the circuitry in these devices becomes economical and so will bring the terminals within reach of even small-system users.

The μPD7220 graphics display controller, or GDC, promises to lower the cost of color graphics significantly. This dedicated LSI chip can handle a display memory as large as 256-K 16-bit words—the equivalent of over 4 million picture elements (pixels). It can draw lines, arcs, circles, and rectangles at a rate of less than 800 nanoseconds per pixel.

Previous controller solutions have had some serious drawbacks. Medium-scale integrated controllers are fast

1. Living color. The μPD7220 graphics display controller can create complex color images on a 2,048-by-2,048 dot matrix and can address 256-K 16-bit words. In its graphics mode, they can be red, green, and blue overlay planes. The GDC can also draw figures like circles without processor intervention.



2. System on a chip. Inside the graphics display controller, elements are connected by means of an on-chip data bus. The small buffe permits independent transfers to take place through the first-in, first-out memory. The drawing processor constructs geometric objects.

and often adequate, but they consume considerable power and occupy significant board space. Controllers based on bipolar bit-slice processors have also been effective but require lengthy design work and are difficult to upgrade. The MOS microprocessor display controller offers lower package count and lower cost but is too slow to be considered a high-performance solution.

The GDC, however, fabricated with a 3-micrometer n-channel MOS process using over 13,000 transistors, simultaneously meets performance and cost requirements of sophisticated yet high-volume graphics terminals. It isolates the display memory from the microprocessor, freeing the processor to handle higher-level graphics calculations and communications with the terminal user and the host processor. It can work with almost any central processing unit to form a basis for a powerful computer graphics system with character capability at very low cost.

Flexible formats

How effective a graphics system is depends largely on the flexibility of the system's display controller. The GDC can control thousands of alphanumeric characters or graphics figures comprising millions of dots. These bit-mapped figures can be drawn so quickly that complex images can be created in one 16.7-millisecond video frame period. Up to four character-display or two graphics-display areas may be horizontally split and scrolled independently, with little local processor overhead.

The display memory is often larger than the displa area, so as to make possible double-buffered displa frames, multiple-frame movies and panning and the us of lower-cost video circuitry and monitors. Zoom magni fication factors of 1 to 16 may be selected unde program control, and a light-pen detection circuit i included. The GDC facilitates the strobe generation fo dynamic random-access memories, which it refreshe even during high zoom magnification.

For graphics, the GDC's display memory can be orgar ized as 2,048 pixels by 2,048 lines, or as 1,024 pixels b 1,024 lines with four bit planes per location, or in ju about any other combination. Data can be moved fro the display memory to the screen in 16-bit words in minimum cycle of under 400 ns. Optionally, two 16-b words can be moved simultaneously for a video pixel ra of 80 megahertz.

Allowing for RS-343 blanking, this rate yields a 6 hertz noninterlaced display of 1,024 pixels by 792 lin for a 4:3 display aspect ratio. A 2:1 interlaced display 1,024 by 1,024 dots with red, green, blue, and overl

CODEN:LUTFD2/(TFRT-5276)/1-053/(1982)

HÖGUPPLÖSANDE FÄRGGRAFIK FÖR OPERATÖRSKOMMUNIKATION

TORBJÖRN PERSSON

DEPARTMENT OF AUTOMATIC CONTROL
LUND INSTITUTE OF TECHNOLOGY

JUNE 1982

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

Document name
MASTER THESIS

Date of issue
June 1982

Document number
CODEN:LUTFD2/(TFRT-5276)/1-053/(1982)

Author(s)
Torbjörn Persson

Supervisor
Karl Johan Åström

Sponsoring organization

Title and subtitle
Högupplösande färggrafik för operatörskommunikation
(High-resolution colour graphics in control systems)

Abstract
This paper describes the design and construction of a microprocessor-based video display control unit intended for use in industrial control systems. The unit features graphics of up to 1024 X 1024 pixels resolution and eight colours using the NEC μPD 7220 Graphics Display Controller. The design idea is based on a discussion of basic principles in colour graphics outlined in the first part of the paper. This is followed by a detailed description of the units' construction and performance including some software.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
53

Recipient's notes

Security classification

Distribution: The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

0. Förord

Denna rapport utgör dokumentationen för ett examensarbete vars syfte varit att konstruera en videokontrollenhet för högupplösande färggrafik. Examensarbetet har utförts dels vid Institutionen för Reglerteknik på Lunds Tekniska Högskola, dels vid Telsand Elektronics AB i Arlöv.

Färggrafik är ett relativt nytt område inom man-maskinkommunikation. Av denna anledning finns det ännu ganska få allmänna regler om hur den ska utformas. Ett antal artiklar i fackpressen och några böcker har dock skrivits och den första delen av denna rapport försöker sammanfatta grundläggande fakta om färggrafik. Ur dessa får vi till sist fram en lämplig konstruktionsidé.

Konstruktionen finns redovisad i den andra delen av rapporten. Denna beskriver hur de olika detaljerna från första delen kan realiseras. Här finns också en testrapport och några förslag till förbättringar och synpunkter på konstruktionen.

Den valda hårdvarukonstruktionen innehåller kretsar som senare visade sig vara svåra att få tag på. Bland annat videoprocessorn som kom ut på marknaden först ett år efter fabrikantens första tillkännagivelse. Eftersom konstruktionsarbetet därför dragit ut på tiden har inskränkningar fått göras i den planerade mjukvaruutvecklingen för videokontrollenheten.

Jag vill här också tacka för all hjälp och idéinspiration som jag fått av personalen på Reglerteknik och Telsand, och då speciellt min handledare prof. Karl Johan Åström, Telsands Axel Westrenius och Lars Knutsson.

Lund maj 1982

Torbjörn Persson

0.1 Innehållsförteckning

1. Inledning

"En färgbild säger mer än tusen svart-vita"

Även om detta modifierade talesätt är något överdrivet belyser det vilka fördelar färgbilder har. En bild i sig har klara fördelar framför en verbal beskrivning. Med färg kan man förbättra förståelsen av bilden ytterligare.

Ända sen datorns födelse har ett stort problem varit att få ut någon vettig information av alla data som matats ut efter en exekvering. I många år har enda möjligeten varit att försöka tyda sidor av tal i olika tabeller som skrivits ut på papper av en printer. Detta har varit oöverskådligt och skulle mycket bättre kunna presenteras i någon form av diagram. I en processindustri skulle det vara helt otänkbart att försöka styra läget hos ventiler och nivåer i tankar med en pappers-utskrift som enda kommunikationsmöjlighet. Allt detta har naturligtvis skapat ett behov för andra metoder att visa datamängder. Detta har bland annat lett fram till möjligheten att använda färggrafik som detta examensarbete kommer att försöka utveckla ytterligare.

Vad kan man då uppnå med färggrafik på bildskärm som man inte kan få med andra metoder ? Ett svar är att en bild kan ge översiktlighet åt en datamängd. Det faktum att bilden finns på en bildskärm gör att man snabbt och enkelt kan ändra delar av bilden eller få fram en helt ny bild. Med färgen får man en tredje parameter att variera sin bild med.

När det gäller att snabbt hitta en viss detalj i en bild är färgen den överlägset mest betydelsefulla urvalsmekanismen. I process-industriella tillämpningar har man visat

(litt.ref 2) att färgen har en mycket stor betydelse för söktiden d.v.s. den tid det tar att hitta en viss detalj i en bild. Detta förstås under förutsättning att man innan vet vilken färg detaljen har (Bra argument för vettigt färgval i bilderna). Inom forskning och utvecklingsarbete gör färganvändningen att man lättare uppfattar komplicerade strukturer. I speltillämpningar har man fått en ny dimension att röra sig i, och dessutom är färgen i detta fall ett viktigt försäljningsargument. Gemensamt för alla tillämpningar för färgbilder är att man kan få en bättre struktur på bilden och totalt en mera "lättläst" bild.

Anledningen till att man inte använt färgbilder förrän på senare år (~1980) är inte att det saknats ett behov för det. Snarare är det så att det tidigare varit svårt och framför allt mycket dyrt att få fram en någorlunda hygglig grafisk färgbild på en bildskärm. Detta beror bland annat på att det behövs minst tre gånger så mycket minneskapacitet som motsvarande svart-vita bild. (Med svart-vita i detta sammanhang menar jag bilder som endast kan bestå av två färger, förgrunds- och bakgrundsfärg. Man kan alltså även tänka sig till exempel gult - brunt som vissa terminaler använder sig av.)

Eftersom minnesutrymmet utgör en stor del av kostnaden för ett färggrafiskt system kan man förstå att de nu snabbt sjunkande priserna på minneselement påverkat möjligheterna att konstruera avancerade sådana bildskärmssystem. Bara de två senaste åren har priset på dynamiska minnen sjunkit från ett par kronor till under 70 öre per Kbit. Samtidigt har lagringskapaciteten i en kapsel ökat från 16 Kbits till 64 Kbits.

Även på bildskärmssidan kommer det troligen att ske en prisrevolution. I nuläget kostar en normal TV-bildskärm (alltså en monitor med TV-bildrör och motsvarande kringelektronik) kring 2 000 SEK (not 1) medan en bildskärm med högre kvalitet lämplig för datoranvändning kostar från 20 000 SEK och uppåt. Detta förhållande kan emellertid komma att förändras inom de närmaste åren eftersom hemdatormarknaden har fått en explosionsartad

* not 1: SEK = svenska kronor

utveckling. Denna marknad behöver bildskärmar, och då företrädesvis färgbildskärmar. Det blir alltså mer lönsamt att göra relativt högkvalitativa skärmar och man kan med stora serier pressa ned kostnaderna för tillverkningen. En lämplig skärm skulle därför kunna närma sig vanliga TV-skärmar prismässigt sett.

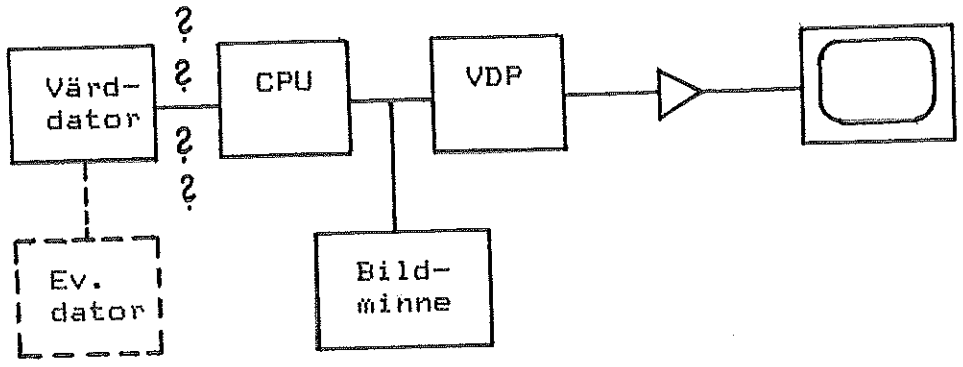
En annan faktor som påverkat färgbilds-användningen är att det tidigare inte funnits någon speciell hårdvara för genereringen av bilden. Man har varit tvungen att bygga upp färgbildsgeneratorn av ett stor antal integrerade kretsar. En konsekvens av detta är att konstruktionen har tagit stor plats. Numera finns de flesta behövliga funktioner (synkgenerering, bildminnesadressering m.m.) samlade i en eller ett fåtal IC:s.

I detta examensarbete ingår en konstruktion av ett färgbildskärmsinterface. Alltså den del av hårdvara och, i viss mån, mjukvara mellan en dator och en bildskärm som behövs för att generera färgbilder på skärmen. Interfacet är i första hand avsett att användas i system för process-styrning och -övervakning. Det bör dock konstrueras så pass generellt att det med inga eller små modifikationer kan användas även i andra tillämpningar. Eventuella nödvändiga förändringar bör i så fall ligga i mjukvara för att förenkla ändringen.

För att veta efter vilka principer man ska konstruera interfacet ingår i denna redogörelse en diskussion av de olika delar som ingår i konstruktionen, till exempel hur man överför en bild från dator till interface på lämpligaste sätt. Framför allt måste man reda ut var gränsen mellan dator och interface ska gå (markerad med frågetecken i figur 1.1). Detta gäller till största del mjukvaran.

Vidare följer en översikt på tillgängliga videoprocessorer. Dessa är alltså den tidigare omtalade hopsamlade hårdvaran för bild-genereringen.

För att underlätta förståelsen för kommande uttryck och termer kan vi redan här föreställa oss



Figur 1.1

en tänkbar uppdelning av det totala systemet. (Se figur 1.1).

Om vi går "baklänges" så finns först bildskärmen. Denna kommer att behandlas i kommande avsnitt. Bildskärmens intensitetsförändringar styrs via en anpassningskoppling av en video-displayprocessor (VDP), eller något kortare, videoprocessor. Nu tillgängliga videoprocessorer är tämligen ointelligenta, och för att uppdatera i bilden behövs någonting som hanterar bildminnet. Detta kan vara en vanlig microprocessor. Angående bildminnet kan detta vara inkopplat på olika sätt, men mer om detta i kommande kapitel. Micro-processorn får i sin tur instruktioner från en överordnad dator om vad som ska uppdateras i bilden. Eventuellt är sedan denna dator hopkopplad med andra datorer, men detta kommer att bero på den specifika tillämpningen.

Detta var alltså bara en tänkbar lösning och den slutliga lösningen kan naturligtvis bli annor-lunda.

2. Förutsättningar

2.1 Det totala systemet

Det totala systemet omfattar i detta fall allting från någon typ av mätvärdesrapportering till bildskärmen. Vad som sedan exakt finns här emellan kommer att variera med olika tillämpningar. I ett minimalt system behövs det åtminstone någonting som kan ta emot och konvertera mätvärdena till en lämplig representation för bilden, och någonting som sedan kan presentera de bearbetade värdena på bildskärmen. I dessa system behöver man kanske inte mer än fem till tio olika bilder som är relativt schematiskt uppbyggda. I större system kan man behöva flera bildskärmar och många (i storleksordningen hundratals) olika bilder för presentationen. Till detta krävs då en effektiv bildhantering och också ett lämpligt sätt att lagra bilderna på.

Ur operatörens synvinkel ska systemet inte vara mer än ett "förlängt öga". Hon ska inte behöva ta hänsyn till om det sitter några få komponenter eller en stor komplex dator mellan mätställe och motsvarande representation på bildskärmen. En annan sak som berör operatören är att systemet ska reagera på samma sätt i alla situationer. Detta gäller speciellt vid larmsituationer. Några saker som ger exempel på detta är att man inte ska behöva vänta på en bild man vill ha fram (åtminstone inte längre än någon bråkdel av en sekund), och om man har givit fel kommando ska det gå snabbt att rätta till misstaget.

I systemet ingår också den del där man designar de bilder som används. Denna del behöver inte alltid sitta tillsammans med de delar som används för processövervakningen. I små tillämpningar behöver man oftast inte ändra på bilderna när de väl en gång fåtts att fungera. Då finns det heller inget behov av de verktyg som behövs för bild-editering. I de större systemen kan det vara

vettigt att ha direkt tillgång till bild-editeringsfunktioner.

En utförligare presentation av de olika delarna följer i kommande kapitel.

2.2 Nivåuppdelning

Det tidigare diskuterade totala systemet kan delas upp i olika nivåer beroende på delarnas komplexitet. (Se tabell 2.1).

De delar som finns upptagna i figuren ger inte en komplett bild av den totala strukturen, men de försöker visa vilken typ av funktioner som kan finnas på respektive nivå.

Nivå 1 är oförändrad i alla olika tillämpningar. Nivå 2 anpassas däremot till den specifika tillämpningen och kan i stora system tänkas uppdelad i ytterligare undernivåer. I de allra minsta systemen behövs nivå 2 inte alls. I sådana fall får nivå 1 nödvändiga kommandon direkt från nivå 3.

Nivå 3 består av två delar, dels "utveckling" dels "process". Denna uppdelning behöver inte finnas i verkligheten, men i små system kan man utesluta utvecklingsdelen från det levererade systemet.

När det gäller programvaran på de olika nivåerna ska man försöka att använda högnivåspråk i så stor utsträckning som möjligt. Högnivåspråken ger bättre struktur och mindre felmöjligheter än motsvarande assemblyprogram. Eftersom mjukvaran på nivå 2 och 3 (processidan) i de flesta fall måste anpassas till varje specifik tillämpning, behöver man ett effektivt språk på dessa nivåer. Detta gör att utvecklingstiden och därmed kostnaden kan hållas nere. På nivå 1 är emellertid uppgifterna relativt enkla och de ska inte behövas

	Utvecklings-miljö	Process-miljö
Nivå 3:	* BILD-EDITOR * BILDKOMPILATOR	* MÄTVÄRDES-BEARBETNING * (MÄTVÄRDES-RAPPORTERING)
Nivå 2:	* BILDLAGRING * KOMMANDOGIVNING TILL NIVÅ 1 * BILDÖVERFÖRING TILL NIVÅ 1 * SPECIALTECKEN SOM SKA ÖVERFÖRAS TILL NIVÅ 1 * TANGENTBORD FÖR BILDKONTROLL * KONTROLL AV FLERA BILDSKÄRMAR (MOTSV. ~ NIVÅ 1)	
Nivå 1:	* STANDARD TECKEN UPPSÄTTNING * TA EMOT SPECIALTECKEN FRÅN NIVÅ 2 * UTFÖR ENKLA KOMMANDON FRÅN NIVÅ 2 * TAR EMOT NYA HELA BILDER * HANTERAR EN BILDSKÄRM	

Tabell 2.1

ändras med olika tillämpningar när de väl fungerar. Dessutom kommer utrymmet för programvaran att vara begränsat och man kräver också snabbhet hos rutinerna. Detta gör att assembly är ett lämpligt språk här.

Kommunikationen mellan nivåerna ska vara så kort och koncis som möjligt. I tabell 2.2 finns de huvudsakliga kommunikationsvägarna uppställda. Kommunikationen mellan nivå 1 och 2 blir i normala fall helt enkelriktad. Nivå 2 har ofta flera olika nivå 1 att hantera och ge order till. Detta kräver att nivå 1 inte stoppar upp nivå 2 i dess arbete. Dessa olika synpunkter måste betänkas vid valet av

kommunikationsmetod.

2.3 Krav på hårdvaran

För detta examensarbete finns det vissa förutsättningar i valet av hårdvara. Som tidigare nämnts utföres konstruktionen hos Telsand Elektronics AB i Arlöv. Detta företag har ett komplett och mycket användbart utvecklingssystem för Texas Instruments Inc.:s produkter. Detta gör

---

NIVA 3 ---> NIVA 2

* "U" EDITERADE BILDER SOM SKA LAGRAS
* "U" NYA SPECIALTECKEN
* "P" MÄTVÄRDEN SOM SKA ÄNDRAS
* "P" ALARMSITUATIONER

NIVA 2 ---> NIVA 3

* "U" BILDER SOM SKA ÄNDRAS
* FELMEDDELANDE

NIVA 2 ---> NIVA 1

* NYA HELA BILDER
* UPPDATERING AV ELEMENT I BILDEN (NYA MÄTVÄRDEN)
* SPECIALTECKEN

NIVA 1 ---> NIVA 2

* FELMEDDELANDE

"U" = UTVECKLING , "P" = PROCESS

Tabell 2.2

---

att det krävs synnerligen starka själ att välja något annat fabrikat på de mjukvaruberoende delarna i konstruktionen. Detta gäller i första hand mikroprocessorer och kommunikationsdelar. För övriga komponenter (t.ex. videoprocessorer) finns det inga begränsningar då det gäller fabrikat.

3. Grundläggande principer

3.1 Upplösning

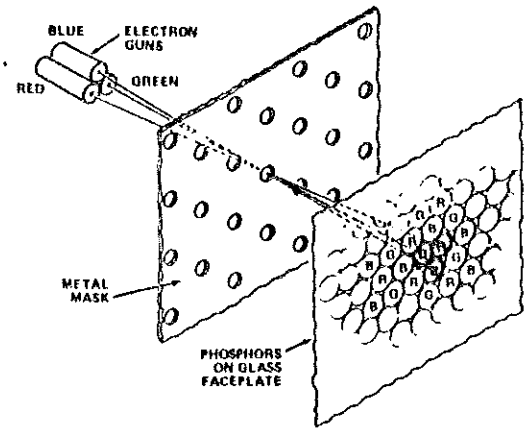
Med upplösning menas här hur många enskilda punkter som bilden kan byggas upp av (på engelska: pixels som är en sammandragning av picture elements). Upplösningen anges vanligtvis som antalet punkter per linje X antalet linjer, t.ex. 768 X 576. Förhållandet mellan dessa värden brukar vara 4:3 (vanligast) eller 1:1. Dessa punkter ska inte blandas samman med de trippel-färgpunkter som en bildskärmsyta är uppbyggd av. En sammankoppling finns dock. Antalet pixels kan inte vara större än antalet trippelfärgpunkter.

Det finns olika benämningar på upplösningsgraden. En upplösning kring 250 punkter per linje benämns "lågupplösande", kring 500 punkter kallas "mellanupplösande" och upplösningar från 1000 punkter och uppåt benämns "högupplösande". I annonser för bildskärmssystem brukar man dock kalla allting över 200 punkter för högupplösande.

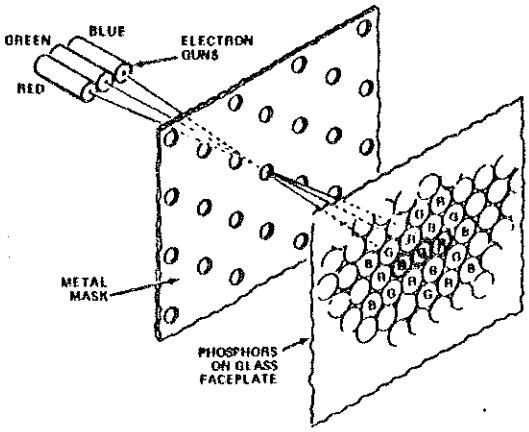
Upplösningen begränsas av i huvudsak två faktorer, nämligen, som tidigare antytts, antalet färgpunkter på bildskärmen och videoprocessorns kapacitet. Dessa båda diskuteras i detta avsnitt.

3.1.1 Monitorer

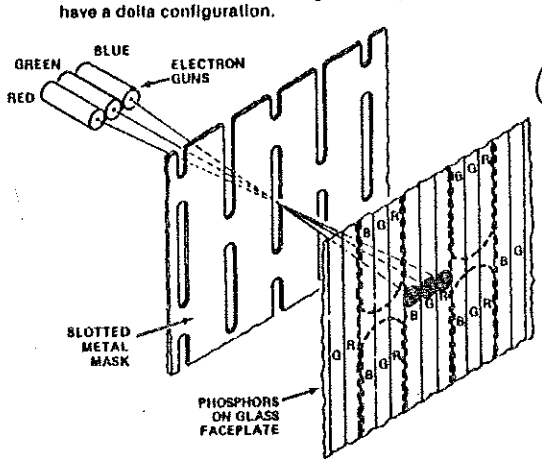
Vi ska nu se hur stor upplösning en bildskärm kan ge. Först tittar vi på hur en bildskärm och dess yta är uppbyggd. I figur 3.1 finns några olika typer av uppbyggnad avbildade. I huvudsak är det antalet trippelfärgpunkter som bestämmer hur stor upplösning man kan få. Man kan tänka sig att de övriga delarna hos monitorn, t.ex. avlänkning,



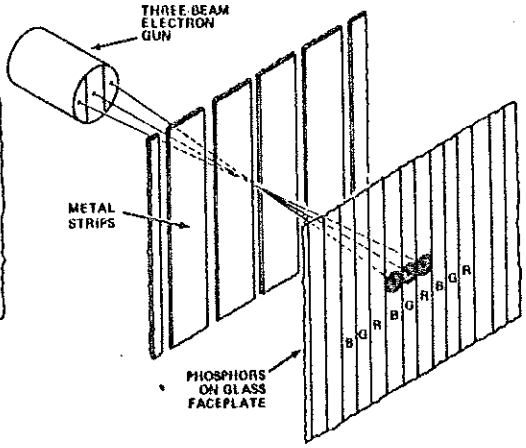
Delta-delta color CRT. Electron guns and aperture mask both have a delta configuration.



PIL-delta color CRT. In-line electron guns are combined with delta-type aperture mask.



In-line electron guns combined with slotted aperture mask and vertical phosphor stripes.



Single-lens, three-beam electron gun with a metal-strip mask and vertical phosphor stripes.

Figur 3.1

skulle kunna begränsa, men i allmänhet är dessa anpassade till skärmens möjligheter.

En normal TV-bildskärms punkter har en diameter av ca 0.6 - 0.8 mm, medan en högupplösande skärm har ca 0.2 - 0.3 mm punkter. För en monitor i mellanstorlek (18 till 20 tum) ger detta för TV-skärmen ca 500 - 600 punkter per linje respektive 1200 - 1500 punkter för den högupplösande skärmen.

3.1.2 Videoprocessorer

De idag tillgängliga videoprocessorerna och dess funktioner kommer att beskrivas till fullo i kapitlet "Teknikövesikt". Här ska vi bara se på deras upplösningsförmåga. Som vi ska se senare i det angivna kapitlet kan man dela upp videoprocessorerna i två huvudgrupper, "enkla" och "avancerade".

De "enkla" kännetecknas av en maximal upplösning av ca 256 punkter per linje. Detta medför att de kan användas tillsammans med vanliga TV-skärmar. De används därför ofta i avancerade TV-spel och hemdatorer. Den andra gruppen är avsedd för "proffesionella" tillämpningar och har upplösningar från 500 ända upp till 4000 punkter per linje. På vissa fabrikat kan man själv ange vilken upplösning man vill ha, men det finns ändå en övre gräns för dessa inom det nämnda området.

3.2 Informationstäthet

Först ska vi konstatera att det finns två olika sorters information i en bild. Den ena typen är de statiska delarna d.v.s. bakgrund, rubriker och andra tecken som definierar själva process-situationen. Den andra typen, den dynamiska, är mätvärden, diagram och dylikt som

ändras beroende på processens uppförande. Det är naturligtvis viktigt att båda dessa typer kan uppfattas snabbt och utan missförstånd, men det är speciellt viktigt för den dynamiska delen. Eftersom den statiska delen av bilden är oförändrad hela tiden kommer operatören att, efter ett tag, lära sig hur bilden är uppbyggd och var de viktiga fälten finns. Som nämnts tidigare spelar här också färgkodningen av detaljer en stor roll.

Det som i första hand bestämmer informationstätheten är alltså hur många dynamiska fält man kan ha i en bild, även om man inte kan bortse från de statiska delarna.

Det har gjorts ett antal utredningar om informationstäthet och man har kommit fram till i stort sett liknande resultat. Vi ska här se på en sammanfattning av de faktorer man behöver ta hänsyn till. I övrigt hänvisas till de nämnda utredningarna vilka finns upptagna i referenslitteraturförteckningen.

Designen av bilden kan delas upp i två problem, dels hur mycket information man kan ha i bilden, dels var i bilden denna ska placeras. Först ser vi på problemet "hur mycket". Man har gjort undersökningar av hur personer uppfattar en bild och då kommit fram till att man bör ha maximalt 25 % av den totala bildytan täckt med information. Av denna del bör inte mer än 75 % vara av den dynamiska typen. Detta ger att max 18 % av bildytan kan innehålla föränderlig information.

Angående "var" man ska placera sin information ska vi först betänka att vårt öga ser klart bara i en smal kon med ca 5° toppvinkel. Med en bilddiagonal på ca 20 tum och ett betraktningsavstånd på ca 70 cm betyder detta att man ser klart inom en cirkel med ca 6 cm:s diameter. Inom ett sådant synfält bör man inte ha mer än en enda detalj, t.ex. ett mätvärde. (Om man har två mätvärden som jämförs mot varandra är det lämplig att ha båda dessa inom synfältscirkeln).

En annan konsekvens av det begränsade synfältet är att man bör undvika detaljer som är större än denna 6 cm:s cirkel. Enligt det tidigare siffer-exemplet blir ett alfanumeriskt tecken ca 5 mm

brett (om man använder 80 tecken per rad). Då skulle ett ord kunna ha 12 tecken för att passa inom cirkeln. Normalt rekomenderas dock inte mer än sex tecken i ett ord. För överskrifter och rubriker kan man förstås ha längre ord.

Ord har en inbyggd redundans som gör dem lättare att snabbt förstå. Detta gäller emellertid inte för numeriska värden. Dessa kan tolkas rätt bara om varje enskilt tecken tolkas rätt. Därför bör man inte ha tal med mer än fyra siffror (om det krävs att man ska tolka talet snabbt). Ett sätt att ersätta eller komplettera ett numeriskt mätvärde är att använda stapeldiagram eller visare. Detta ökar avsevärt chansen att tolka värdet på rätt sätt.

3.3 Färgkomposition

Liksom i övriga delar av detta kapitel kan jag här bara ge grundläggande rekommendationer om vad som är lämpligt. Man får förlita sig på bild-designerns kunskap och sunda förnuft. En viktig punkt måste man emellertid tänka på. I process-industrin finns det ofta redan inarbetade färgkodningar av olika saker. Lika ofta är denna kodning olika från industri till industri. På ett ställe kan grön och röd färg beteckna öppen respektive stängd ventil, på ett annat ställe kan det vara tvärt om. Därför är det viktigt att man verkligen tar reda på hur färgkodningen är vid den industri man ska installera ett system. Detta för att undvika felmanövrar i krissituationer då gamla inlärda symboler och färger lätt kan förvilla begreppen.

Frånsett redan använd färgkodning, hur kan man då välja färger i bilden ? Ja, det är ganska klart att man inte ska sträva efter att göra ett kubistiskt mästerverk. Ju mer konsekvent man kan vara i färgvalet desto lättare för operatören. Samma färger för samma saker i olika bilder. Jag har tagit med ett exempel (tabell 3.2) på färgval som kan vara vettigt i många tillämpningar. Exemplet är hämtat ur en artikel i

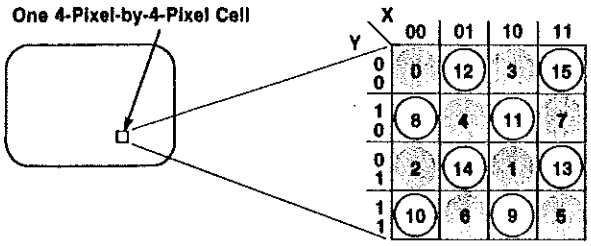
Färg	Användning
svart	Bakgrund
blå	Överskrifter, rubriker (statisk inf)
cyan	Alfanumerisk data (dynamisk inf)
grön	Från, stängd, normal
röd	Till, öppen, varning
vit	Lägen mellan grön och röd t.ex. halvöppen
gul	Uppmärksamhet krävs
magenta	Fara, omedelbar åtgärd krävs

Ur Instrumentation Technology, okt -76.

Tabell 3.2

Instrumentation Technology, okt -76, "CRT Displays for Power Plants". Som tidigare sagts får man emellertid kontrollera att några färger inte används i ombytta funktioner.

Antalet färger är givet av hur många bitar per pixel man har i bildminnet. I de tillämpningar som denna konstruktion är avsedd för är det fullt tillräckligt med åtta olika färger, vilket skulle kräva tre bitar. Med hög upplösning kan man få fler färgnyanser i fyllda areor än de givna åtta grundfärgerna. Detta åstadkommer man genom att blanda två av de åtta färgerna i bestämda mönster (se figur 3.3). På detta sätt kan man få ända upp till 4000 olika färger. Observera att detta bara gäller areor.



Area fill colors are obtained by varying the pattern in 4 × 4 arrays of pixels.

Figur 3.3

3.4 Operatörskommunikation

I systemet måste ingå en eller flera enheter som operatören kan styra bildinformationen med. Det kan gälla att välja en ny bild, ändra inställbara parametrar i processen och dylikt. Här kan man tänka sig ett antal olika metoder. Ett normalt alfanumeriskt tangentbord är kanske den vanligaste enheten i detta sammanhang. Men det kan också finnas behov av ett eller flera tangentbord som är specialgjorda för den aktuella tillämpningen. Andra enheter kan vara ljuspenna, digitizer (not i) eller röstkommunikation. Ett smidigt sätt att peka ut en viss detalj i bilden är att använda sig av en markör. Denna kan förflyttas på olika sätt t. ex. med piltangenter, "boll", "mus", "joy-stick" eller ljuspenna.

* not 1: Inorgan utformat som en platta där man kan "rita" med en speciell penna. Används ung. som vanligt "papper och penna".

I ett mycket litet system som används endast för processövervakning (innehåller upp till ca tio bilder) behöver enda kommunikationsmedlet vara ett litet tangentbord t.ex. ett numeriskt. I detta fall behöver man ju bara kunna välja någon av de tillgängliga bilderna. I något större system med fler bilder och där man även vill ha möjlighet till styrning av processen kan ett alfanumeriskt tangentbord vara lämpligt. Med detta kan man då dels välja bilder, dels ändra parametrar i den aktuella bilden.

I många tillämpningar kan ett specialgjort tangentbord vara nödvändigt. På detta kan man ha sådana funktioner som snabbt måste åtgärdas i larmsituationer. Detta kan vara anrop efter bilder som visar larmstället, direkt manövrering av viktiga delar i processen o.dyl.. Man kan även tänka sig särskilda knappar för översiktsbilder och andra ofta använda bilder. Utformningen av detta tangentbord får bedömas för varje enskild användning.

Bollen och musen (eng. mouse) är i princip lika. Båda bygger på att man rör sin hand i relation till önskad markörförflyttning. Bollenheten är stillastående och man snurrar på bollen som styr markörens läge. Musenheten är en låda med boll eller vinkelrätt monterade hjul på undersidan. Man flyttar hela enheten på ett plant underlag. Ofta har man också några tryckknappar på lådan som kan användas till olika funktioner t.ex. för markering av valda delar i bilden.

Lite mer udda kommunikationsmedel är ljuspenna och digitizer. Dessa båda används ungefär på samma sätt med ett viktigt undantag, nämligen vinkeln på underlaget. Den normala ställningen för en bildskärmsyta är i det närmaste vertikal. Därför kan man inte stödja handen mot något när man använder en ljuspenna, d.v.s. man blir snabbt ordentligt trött i både hand och arm. Ljuspennan kan därför bara användas vid sporadiska tillfällen, och är därför mindre lämplig som huvudsaklig kommunikationsmedel. Om däremot bildskärmsytan kan läggas horisontellt får man en ställning som i likhet med digitizern är mycket mindre tröttande och därmed bättre lämpad för ändamålet. Nackdelen med liggande bildrör är deras djup som nu är vänt nedåt och konkurrerar med operatörens

knän om utrymmet, såvida man inte placerar skärmen vid sidan om operatören.

I samband med talet om liggande bildskärmar kan jag ta upp ett tänkbart användningsområde för färggrafik. På senare tid har det börjat dyka upp s. k. touch sensitive screens, d. v. s. bildskärmar med ett beröringskänsligt skikt på ytan. Då skulle man kunna designa ett tangentbord mjukvarumässigt och rita upp det på skärmen. Man får på detta sätt ett nära nog oändligt antal variationer på tangentplaceringarna och dess utformning och funktion. Man kan ha många olika sådana tangentbord i samma system där sedan operatören kan få fram den version han behöver för just det tillfället. Ett extra eller nydesignat tangentbord kräver ingen ny hårdvara och man kan därför hålla kostnaderna nere.

Röstkommunikation är i nuläget fortfarande i experimentstadiet och bör undvikas, speciellt i kritiska positioner i processstyrningen. Röstkommunikation går i två riktningar. Den ena riktningen, som kallas "voice recognition" d.v.s. operatören talar till datorn, kan emellertid vara tänkbar i mindre viktiga funktioner och då operatören har sina händer upptagna med annat. Det andra hållet, "voice synthesizing", datorn talar till operatören, kan man använda som "varningsröst" vid felkommandon och larm.

4. Teknikövesikt, videoprocessorer

På dagens marknad finns det i stort sett två typer av videoprocessorer. Som nämnts i tidigare kapitel (3.1.2) kan vi kalla dessa olika typer för "enkla" respektive "avancerade". Denna beteckning hänför sig i första hand till deras tänkta användningsområden. Tekniskt sett kan den "enkla" typen vara väl så avancerad som den "avancerade" typen. Det som skiljer är istället vilka funktioner som finns i de olika videoprocessorerna.

Vi ska börja med att se på de enkla VDPerna (Video Display Processor). Som också nämndes i det tidigare kapitlet är dessas upplösning begränsad till ca 256 bildpunkter (= pixels) per linje och ungefär lika många linjer per bildruta. Detta medför att den maximala frekvensen på videosignalen blir ca 2.5 MHz (en linje = 64 µs - sync och blankintervall = 52 µs som 256 pixels ska visas på => 5 MHz. I varje punkt behöver man bara kunna visa en halvperiod => 2.5 MHz). Med denna relativt låga frekvens kan man ha all videosignalshantering direkt på chipet, vilket också är utmärkande för den "enkla" typen.

De har ofta begränsningar i färgval, så att man får välja mellan antingen många färger (i flesta fall åtta) och låg upplösning eller maximal upplösning med kanske bara två färger. Ofta kan man inte direkt adressera varje enskild pixel, vilket gör det svårt att rita kurvor.

I övrigt kan sägas att den "enkla" typen är utrymmessnål. Oftast behöver man bara videoprocessorn och bildminne. På "ena sidan" hänger man på en mikroprocessor för styrningen och på "andra sidan" kan man ta ut videosignalen direkt från en pinne på kretsen. Beträffande bildminnet kan man med vissa VDPer direkt koppla till dynamiska minneskapslar med multiplexade adressingångar och separata in- och utdatapinnar. Detta medför att man behöver max tio kapslar till

den grundläggande hårdvaran.

Om vi går över till den "avancerade" typen kan vi konstatera att man här arbetar med betydligt högre videofrekvenser. Ända upp till 80 MHz kan bli aktuella. Av denna anledning har de "avancerade" VDPerna ingen som helst logik för videosignalens generering på chipet. Detta får byggas utanför VDPn med snabb TTL eller liknande. Istället gör flertalet av de "avancerade" VDPerna det möjligt att mjukvarumässigt bestämma video-parametrar som upplösning, synk- och blank-intervaller. Detta gör dem mycket flexibla och lätta att anpassa till olika tillämpningar.

En annan skillnad mellan de två typerna är att de "enkla" sköter all färghantering internt. För de "avancerade" VDPernas del spelar det ingen roll om man vill ha bara svart-vitt eller ett stort antal färger, eftersom all videosignalshantering finns utanför kretsen.

Nackdelen med de "avancerade" VDPerna är dess utrymmesbehov. Förutom videoprocessorn och bildminnet (som för övrigt kan vara upp till tio gånger större än för de "enkla" VDPerna) måste man ha skiftregister, buffrar, klockgenerering m. m.. För att ge en uppfattning av storleken kan man generellt säga att det krävs minst 50 ICs för den grundläggande konstruktionen.

I nästa avsnitt ska vi se lite närmare på de olika fabrikat som finns på marknaden våren 1982.

4.1 "Enkla" typen

De tillgängliga fabrikaten av "enkla" videoprocessorer finns sammanställda i tabell 4.1. Vissa av dessa ska man kanske inte kalla tillgängliga eftersom man i bästa fall bara kan få tag på provexemplar, men de finns i alla fall beskrivna i datablad och handledningar. Denna typ av VDPer ger alltså en videosignal ut från kretsen och finns därför i olika versioner för NTSC (amerikanska TV-systemet) eller PAL

Fabrikat	Typ	Upplösning
A M I	S68047	256 X 192
Motorola	MC6847	256 X 192
R C A	CDI1861/62	64 X 192
Signetics	2637	~150 X 200
Texas Instr.	TMS 9918	256 X 192
MOS Techn.	VIC 6561	192 X 200

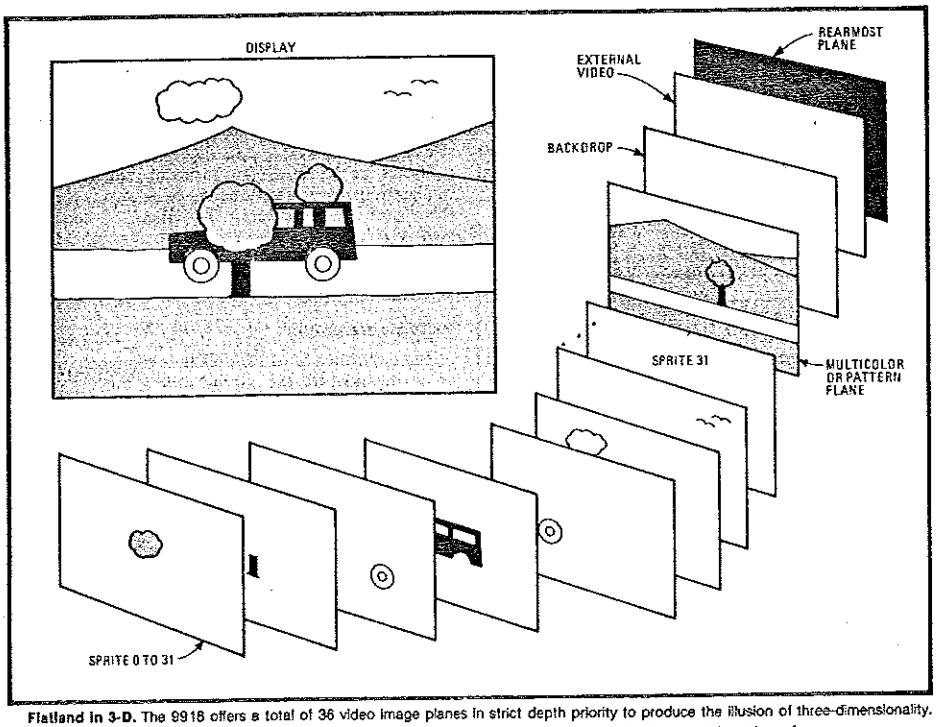
Tabell 4.1

("europeiska" systemet). Vissa kretsar har utgångar för luminans och krominans som man blandar externt till önskat videosignalsformat.

RCAs version består av två kapslar som är utförda i CMOS-teknik. Det krävs dessutom en mikroprocessor för adressgenereringen till bildminnet. Motorola och AMIs versioner är till innehållet lika men de är av någon anledning inte pinn-kompatibla. I dessa finns tolv olika moder för alfanumerisk och grafisk presentation. I alfanumerisk mod kan man välja på fyra eller åtta färger, men i den högupplösande grafikmoden (256 X 192 pixels) finns det tyvärr bara två färger tillgängliga.

MOS Technologys krets har till viss del variabelt bildformat (max 192 X 200 pixels). Även denna har begränsningar i antalet färger vid högre upplösningar. Kretsen har även en ingång för ljuspenna samt två analog-digital omvandlare. Dessutom innehåller den en ljudgenerator med tre tonkanaler och en bruskanal. Den används bland annat i Commodores hemdator VIC-20.

Texas Instruments krets används också i hemdatorer, i detta fall Texas egna 99/4. I grunden har den ungefär samma uppbyggnad som de tidigare, men man kan använda alla femton färgerna även vid högsta upplösning (256 X 192). Dessutom har kretsen en funktion som gör den till den mest avancerade i denna grupp. Som kan ses i figur 4.2 byggs bilden upp av 36 plan staplade ovanpå



Flatland in 3-D. The 9918 offers a total of 36 video image planes in strict depth priority to produce the illusion of three-dimensionality. Thirty-two planes containing objects, called sprites, combine with transparent areas that let the background show through.

Figur 4.2

varandra. I de 32 översta planen kan man ha tecken, kallade sprites, som är 8 X 8 eller 16 X 16 pixels stora. I varje plan kan det finnas en sprite och denna kan placeras var som helst på skärmen med en pixels noggrannhet genom att man ändrar dess x- och ykoordinat. Med denna funktion kan man skapa en "tredimensionell" bild där ovanliggande sprites kan "köra över" underliggande sprites utan att de sistnämnda förstörs.

Som också framgår av figuren är det möjligt att koppla in en extern videosignal som visas i motsvarande plan. Detta kan vara en annan videoprocessor eller en kamera e.dyl..

4.2 "Avancerade" typen

De olika fabrikaten av denna typ finns uppställda i tabell 4.3. Liksom för den "enkla" typen är det lite tveksamt med tillgängligheten. Vissa av dessa kretsar är kompatibla med andra fabrikat. De är då antingen helt och hållet identiska (ofta second source fabrikat) eller så är de bara pinkompatibla och har vissa olika funktioner inuti sig. Gemensamt för alla är att de genererar adresser till bildminnet och synkroniseringspulser.

Den mest finessrika VDPn kommer från NEC. Den kan efter ett fåtal kommandon själv rita linjer, rektanglar, bågar, cirklar och fylla areor. Man kan skriva valfria tecken om 8 X 8 pixels var som helst i bilden i åtta olika riktningar och med varierbar förstoringsgrad (mellan 1 och 16 ggr). Dessutom kan man zooma bilden från 1 till 16 gångers förstoring. Scrolling och panorering kan ske i två oberoende fält på skärmen. Upplösningen som är varierbar är max 1024 X 1024 pixels med fyra bildplan vilket kan ge 16 färger (om man har en VDP till varje färgplan är upplösningen max

Fabrikat	Typ	Upplösning	Ljuspenna
Intel	8275	640 X 512	Ja
Motorola	MC6845	ext.	Ja
A M I	S68045	ext.	Nej
National S.	DP8350	~900 X 512	Nej
Standard M.	CRT5027	ext.	Nej
Texas Ins.	TMS9927	ext.	Nej
Synertek	SY6545	ext.	Ja
N E C	µPD7220	2048 X 2048	Ja
Hitachi	HD46505	ext.	Ja
Thomson	EF9365	512 X 512	Ja

ext. : Upplösningen beror på externa kopplingar.

Tabell 4.3

2048 X 2048).

Thomsons krets kan också rita, men bara linjer och tecken. Den har fast upplösning på 512 X 512, 512 X 256 eller 256 X 256 pixels. Övriga kretsar har inga speciella finesser utöver sin grundfunktion. Upplösningen, som är varierbar, är mellan 256 och 1024 punkter. Vissa kretsar kräver yttre räknare och/eller speciella tecken-generatorer för att få hög upplösning.

4.3 Val av videoprocessor

Ett av kraven på konstruktionen är att den hårdvarumässigt ska vara så flexibel som möjligt. Vidare ska den kunna ta nytta av de framsteg som görs på bildskärmssidan, där man kan förvänta sig billiga högupplösande bildskärmar. Detta betyder bl. a. att man ska kunna använda samma mjukvara även om man behöver bygga om hårdvaran. Redan dessa krav antyder att den "enkla" typen av videoprocessorer med max 256 pixels upplösning är mindre intressant. Även ekonomiskt sett finns det inget som motiverar användandet av den "enkla" typen. Prisskillnaden mellan "enkla" och "avancerade" VDPer är i det närmaste lika med noll, och den merkostnad som den "avancerade" typens större bildminne utgör (1000-3000 SEK) är en relativt liten del av ett komplett systems totala kostnad. Mjukvaran kommer att bli ungefär lika omfattande för båda typerna så denna påverkar inte kostnadsbilden.

Vi kan därför koncentrera oss på den "avancerade" typen av VDPer och där är NEC µPD 7220 den mest intressanta. Den har alla funktioner som de andra tillsammans har. Dessutom har den funktioner för linje och cirkeldragning och areafyllning. Dessa kan underlätta programmering och systemarbete och därmed sänka kostnaden för systemet.

Som framgår av tekniköversikten är NECkretsens upplösning fullt tillräcklig för framtida tänkbara uppgraderingar (upp till 2048 pixels / linje)

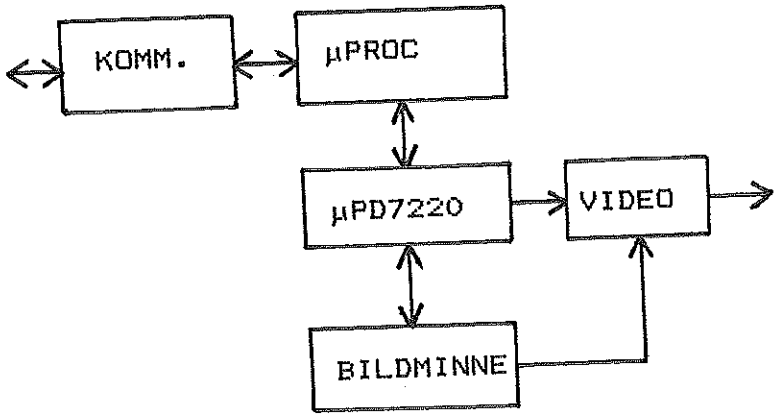
och samtliga videoparametrar kan ändras mjukvarumässigt.

Vi väljer alltså att använda NEC µPD 7220 som videoprocessor i konstruktionen. Datablad för denna krets finns i appendix 1.

5. Konstruktionen

När nu videoprocessorn är vald gäller det att bygga upp all kringutrustning till denna. Vi börjar med att i stora drag se efter vilka enheter som krävs. (Se figur 5.1).

För att styra VDPn behöver vi något som ger kommandon och parametrar till VDPn. Detta något skulle kunna vara den dator som konstruktionen ska användas tillsammans med, men eftersom VDPns kommandostruktur ligger på en relativt låg nivå skulle denna lösning belasta datorn onödigt mycket. Ett bättre alternativ är att ha en mikroprocessor med tillhörande minne som kan ta emot högnivåkommandon och sedan omvandla dessa till VDPns kommandonivå. Detta medför att värddatorn bara behöver sända ett fåtal instruktioner till videokonstruktionen och därmed hålls belastningen nere. Av samma anledning krävs det också en effektiv kommunikation mellan



Figur 5.1

värddator och mikroprocessor.

För att lagra bilden krävs det ett bildminne. Med den valda videoprocessorn är det i stort sett också bestämt hur detta ska se ut. I detta fall blir det ett minne med 16-bitars ordlängd och maximalt 256 Kord.

Vidare vill vi ha ut informationen från bildminnet till bildskärmen och detta handhas av ett antal logikkretsar som vi sammantaget kan kalla videodelen.

Dessa olika delar kommer att behandlas i de följande avsnitten.

5.1 Beskrivning av huvuddelar

5.1.1 Mikroprocessor med minne

Som tidigare nämnts bör vi välja mjukvaruberoende komponenter från Texas Instruments sortiment. I detta fall är det ingen begränsning eftersom den valda mikroprocessorn är en av de kraftfullaste på marknaden. Denna processor kallas TMS 9995 och är en 16-bitars konstruktion med multiplexad 8-bitars extern databuss. Övriga intressanta finesser är ett internt 256 bytes RAM, prefetch av instruktioner, automatiskt waitstate för långsamma minnen, signed multiply- och divide-instruktioner och inbyggd klockgenerator. Cykeltiden för en access av yttre minne är 330 ns.

Den är mycket utrymmessnål eftersom ett fungerande system endast kräver processor, en ROMkapsel med program och tre diskreta komponenter (bl. a. kristall). En mera ingående beskrivning finns i appendix 1 som är en artikel hämtad från Electronic Design, 22 nov 1980.

Till mikroprocessorn behöver vi plats för programvaran, teckenuppsättningar m. m., det vill säga minne. Eftersom det är svårt att i detta läge

uppskatta hur mycket minne som behövs, kan vi i prototypen bara göra ett preliminärt val. Det som begränsar storleken är utrymmet på det kort som prototypen byggs på. Vi bör få plats med tre kapslar i 24-28 pinnars storleken, varav vi väljer att ha två EPROM av 8K X 8 bits-modellen och en RAM-kapsel på 2K X 8. RAM-kapseln är i första hand avsedd för lagring av tecken som man enkelt vill ändra på. Som arbetsminne för programmen räcker det interna RAMet till i de flesta fall.

5.1.2 Kommunikation processor-värddator

Som tidigare nämnts måste kommunikationen ordnas så att värddatorn belastas så lite som möjligt. Detta åstadkommer man programmässigt genom att använda kommandon på "hög nivå" och hårdvarumässigt genom att värddatorn aldrig eller sällan behöver vänta på accessrätt till kommunikationsenheten. Samtidigt ska enheten uppta minimalt utrymme på kortet.

En enhet som uppfyller dessa krav är en ny krets från Texas som heter TMS 99650 MPIF (=Multi Processor InterFace). Tillsammans med en buffertkrets (20-pins) bildar denna krets (40-pins) en komplett kommunikationsenhet.

TMS 99650 består utifrån sett av två identiska portar med åttabitars databuss, trebitars adressbuss och några kontrollsignaler. Internt finns ett 256 bytes RAM som uppför sig som ett dubbelportat RAM. Detta adresseras av pekare (en för varje port) som kan ändras via databussen. RAMet kan konfigureras som en FIFO-buffert med automatisk utelåsning av ena porten när FIFOn blir tom eller full. Dessutom finns för varje port ett meddelande-, ett status- och ett kontrollregister. En fullständigare beskrivning av kretsen finns i appendix 1.

5.1.3 Bildminnet

För att kunna visa färger på bildskärmen behöver man ett visst antal bitar per pixel beroende på hur många färger man vill ha. Det enklaste sättet är att ha en bit för varje grundfärg på bildskärmen (röd, grön, blå). Detta ger åtta färger. Förutom dessa tre bitar kan det vara lämpligt med ytterligare en bit per pixel. Denna kan användas t. ex. för att invertera de andra bitarna och på så sätt markera vissa delar i bilden eller så kan biten markera blinkande fält. Funktionen byggs upp med hårdvara men kan lämpligen varieras mjukvarumässigt.

Detta ger totalt fyra bitar per pixel och alltså fyra skilda bildplan. De tillgängliga 256 Korden delas upp i fyra 64 Kordsblock som kommer att se likadana ut. När bildinformationen ska överföras till videodelen läser man ut ett ord = 16 bitar per plan till ett skiftregister som seriellt skickar iväg bitarna. Detta medför att man måste kunna läsa ut 64 bitar samtidigt. Med detta i åtanke kan vi välja lämpliga minneskapslar till bildminnet.

Med 64K X 1 dynamiska RAMkapslar får vi precis 64 stycken för hela minnesutrymmet. Dynamiska minnen ska refreshas inom en viss tid (vanligen 2 ms) och detta kan göras på flera sätt. Antingen kan man organisera adresserna så att minnet refreshas samtidigt med att datan läses ut till bildskärmen eller så kan man utnyttja den inbyggda refreshfunktionen i µPD 7220. VDPn utför detta under horisontalsynkintervallerna och man förlorar på så sätt en del av tiden för att uppdatera i bildminnet. Därför bör man om möjligt välja den första metoden.

I prototypen kommer varje minnesplan att byggas upp på ett enkelt europakort (160 X 100 mm). För multiplexingen av adresserna till de dynamiska minneskapslarna används en krets från Texas Instruments, TMS 4500 Dynamic RAM Controller, som även innehåller en refreshfunktion (se appendix 1). Denna funktion används emellertid inte för videosystemet eftersom den ibland interfererar med utläsningen av data till bildskärmen. Detta hade

då gett upphov till störningar i bilden. Minneskortet konstrueras för att kunna användas även tillsammans med det kortsystem (TLD 85) som finns hos Telsand och i detta fall sköts refreshen av TMS 4500-kretsen.

5.1.4 Videodelen

Videodelens uppgift är att överföra bild-informationen från bildminnet till en vettig signal för bildskärmen. I detta ingår också logik styrd av det tidigare diskuterade fjärde planet.

Första delen är ett skiftregister som utför en parallell till seriell omvandling av det utlästa 16-bitarsordet. Denna del finns i praktiken på minneskortet eftersom man då avsevärt reducerar ledningsdragningen. Härifrån går bitarna, en från varje plan, till logik för invertering och grindning styrd av fjärde planets bit. Grindningen styrs även av blankingsignalen från VDPn. För att kompensera skillnader i fördröjning av bildbitarna finns sedan en låskrets som synkroniseras med utskiftningen. Sist sitter drivstegen för RGB- och synk-utgångarna.

5.1.5 Övrig logik

För att kunna utnyttja VDPns möjlighet till zoomning måste man externt minska skiftregistrens klockfrekvens så att den motsvarar zoomfaktorn. Detta görs i konstruktionen genom att zoomfaktorn laddas in i en låskrets av mikroprocessorn och detta värde styr sedan en räknare som dividerar ned klockfrekvensen.

Skiftregistren ska laddas när sista biten skiftas ut och det nya dataordet finns klart att läsas in. För att få denna laddpuls rätt i tiden krävs en fördröjning av den signal som startar

utläsningen av nya dataordet.

När VDPn gör en RMW-cykel (read-modify-write) aktiverar den DBIN-signalen för att läsa in ordet. Däremot finns det ingen signal från VDPn som styr skrivningen till bildminnet, så en sådan måste genereras externt.

Alla kopplingar kommer att beskrivas utförligare i nästa kapitel.

5.2 Kopplingsschemor

Hela konstruktionen finns dokumenterad på kopplingsschemor i appendix 2. Detta kapitel ska försöka förklara och motivera de olika kretslösningarna. Dessutom tas det upp några alternativa lösningar på vissa detaljer. Konstruktionen är fysiskt uppdelad i två delar. På ett enkelt europakort finns en fjärdedel av bildminnet. Man använder alltså fyra sådana kort för att få ett fullt utbyggt bildminne. Detta kortet benämner vi minneskort. Alla övriga kretsar finns på ett eget europakort, som vi kan kalla videokort. Denna uppdelning finns också på ritningarna.

5.2.1 Videokort

Vi börjar med att titta på videokortet och då först på mikroprocessordelen. Här finns själva mikroprocessorn TMS 9995, två EPROMar typ 2564 (8K X 8), ett RAM 6116 eller 4016 (2K X 8) och en adressavkodare (krets 17). Dessutom ingår en komponentbärare för kristall och resetfunktionens resistor och kondensator. Adressbussens tretton lägsta bitar är utdragna till EPROMen och de elva lägsta till RAMet. Dessutom behöver MPIF-kretsen tre adressbitar och VDPn en bit. De återstående översta tre adressbitarna styr en avkodare som i

sin tur väljer ut en av de kretsar som är anslutna till mikroprocessorns databuss. Adresserna till dessa finns angivna i tabell 5.2.

Mikroprocessorns interna klockkrets behöver en kristall på max 12 MHz. Resetingången är av typ Schmittrigger och kan därför anslutas direkt till ett lämpligt RC-nät. Vid testningen visade det sig emellertid att funktionen var tämligen osäker varför en buffert från krets 6 (74S244) lades in mellan RC-nätet och resetingången. Reset fungerade sedan som den ska. Resetsignalen används även till MPIF-kretsen.

En sak som måste iakttas är Texas Instruments "felaktiga" numrering av adress- och databitar på vissa kretsar. På TMS 9995 och MPIF-kretsen är bit 0 (noll) den mest signifikanta biten (MSB), medan den på Texas minneskretsar (och f. ö. på alla andra fabrikats kretsar också) är den minst signifikanta (LSB).

Om vi går vidare till MPIF-kretsen ser vi att denna mycket enkelt ansluts till mikroprocessorn utan någon extra logik. Port A är på mikroprocessorsidan och följdaktligen är port B avsedd för värddatorn. Här finns buffrar på alla utgående signaler d. v. s. databuss och readysignal. Riktningen på databussen styrs av värddatorns output enable-signal. När man använder MPIF-kretsens data/increment funktion måste man tänka på att inkrementeringen styrs av chip

---

Krets	Adress	Längd
EPROM 0	0000 - 1FFF	8K bytes
EPROM 1	2000 - 3FFF	8K bytes
RAM	4000 - 47FF	2K bytes
MPIF	6000 - 6007	8 bytes
Zoomlatch	8000	4 bits (LSB)
VDP	A000 - A001	2 bytes

Tabell 5.2

---

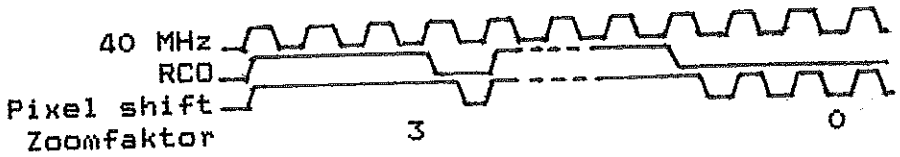
enable-signalen. Om man har flera CE-pulser för en instruktion kommer adresspekaren att räkna upp motsvarande antal steg. Observera även den "felaktiga" numreringen på MPIFs adress- och databussbitar.

Vi övergår nu till klockdrivningen för videodelen. För att skifta ut pixelbitarna till bildskärmen behöver vi en klocka med pixel-frekvensen 40 MHz (krets 24). Med denna frekvens får vi full upplösning, 1024 X 1024 pixels, med en bildfrekvens på ca 27 Hz (interlaced). Med upplösning på 1024 X 768 (4:3 som är förhållandet mellan sida och höjd för en normal bildskärm) ökar bildfrekvensen till ca 35 Hz vilket ger en flimmerfriare bild än den förra. Klockgeneratorn är en TTL-krets 74S124 och det frekvensbestämmande elementet är en kristall på 40 MHz.

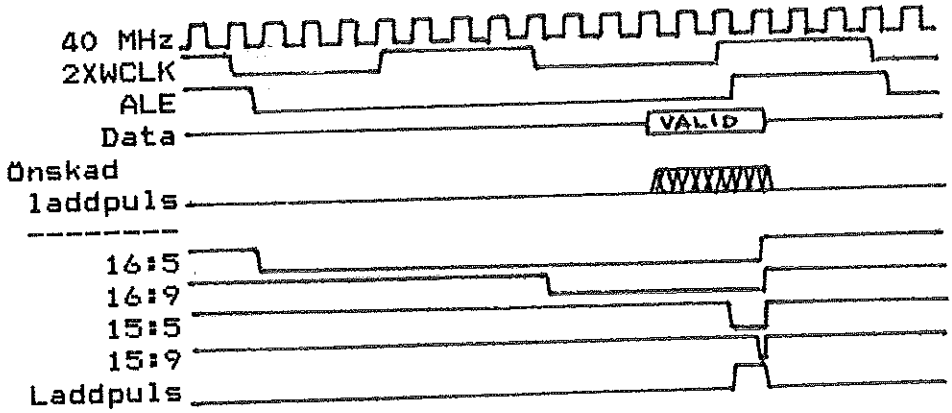
40 MHz dividerat med 8 ger 5 MHz som används till VDPns 2XWCLK-ingång. Detta är den högsta rekommenderade frekvensen och det medför att VDPn kan rita figurerna med maximal hastighet (= 800 ns per bit). Divisionen sker i kretsen 20 och utsignalen går via en buffert till ett antal kretsar. VDPn kräver att hög nivå på klockan är över 3.9 V, vilket kräver ett pull-up motstånd (24:1-18) till +5 V. Vidare finns en resistor (24:2-17) för att minska undershoten.

Om man inte vill använda sig av zoomningsmöjligheten kan man skicka ut 40 MHz-signalen direkt till skiftregistrens klockingångar. Om däremot zoomningen ska fungera måste pixelfrekvensen divideras ned med den aktuella zoomfaktorn. Detta sker genom att zoomfaktorn laddas in i krets 19, som egentligen är ett skiftregister men som här bara utnyttjas som låskrets. Zoomfaktorn laddas sedan in i en räknare som räknar ner till noll i takt med 40 MHz-klockan. Då går RCO (Ripple Carry Out) låg och genererar en pixelskiftpuls. Samtidigt laddas zoomfaktorn in i räknaren på nytt. Grindarna 13 och 7 behövs när zoomfaktorn är noll. I detta fall är RCO låg hela tiden och skulle utan grindarna inte ge några pulser. Se pulsschemat i figur 5.3.

När ett sextonbitars ord ska läsas ut från bildminnet till skiftregistren ska detta ske enligt den övre delen av figur 5.4. När ALE går

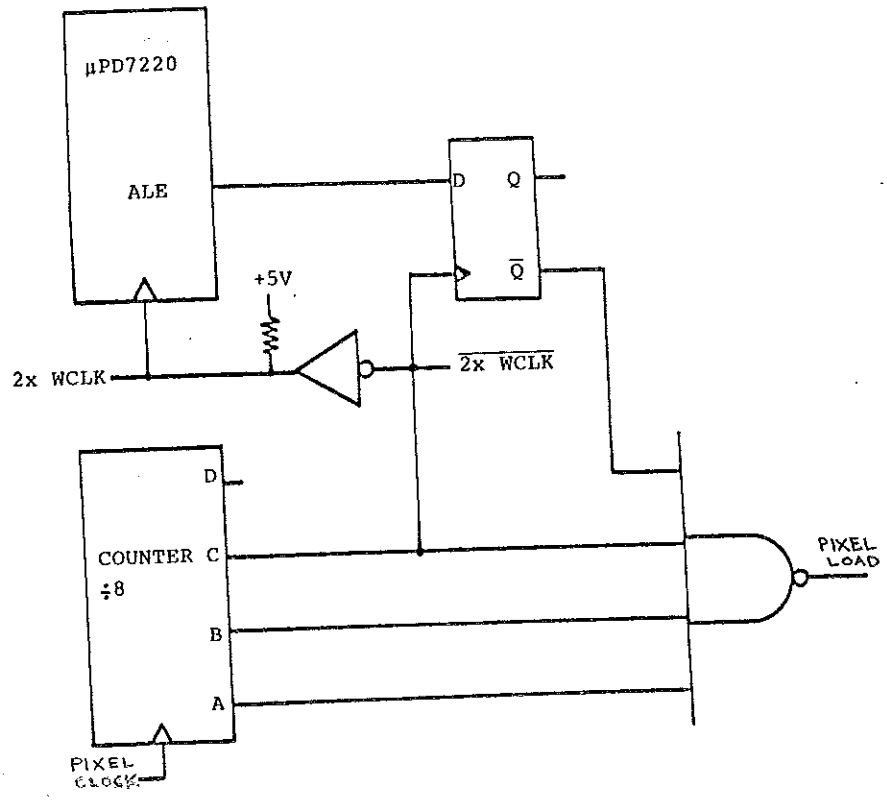


Figur 5.3



Figur 5.4

låg startar en minnesaccesscykel. Efter minnets accesstid finns dataordet klart på ingångarna till skiftregistren. Det finns kvar här tills ALE går hög plus ytterligare en tid på grund av fördröjningar i buffrar och adress-strobs-generering. Laddpulsen bör komma så sent som möjligt eftersom man då kan utnyttja långsammare minnen som är billigare. Den här valda lösningen utgörs av kretsarna 15 och 16. Dessa utgör totalt fyra D-vippor som via klockning av ALE och 2XWCLK ger en laddpuls enligt nedre halvan i figur 5.4.

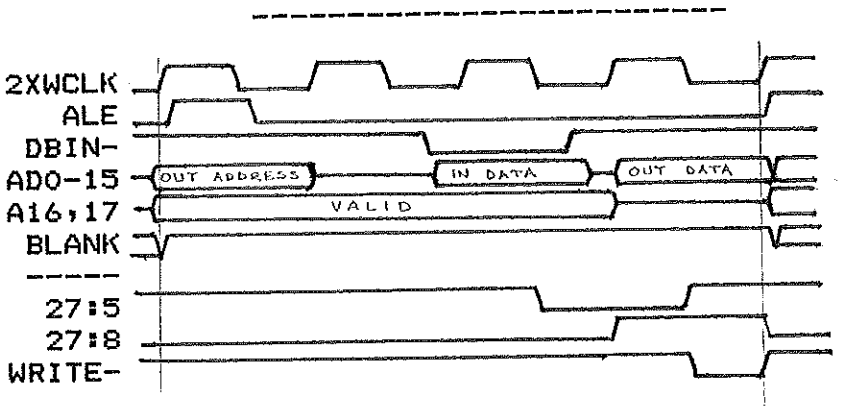


Figur 5.5

Laddpulsen kan också genereras med kopplingen i figur 5.5. Nackdelen är att pulsen kommer en pixelklockperiod tidigare än i den valda kopplingen, och eventuellt krävs därför snabbare minneskretsar.

När man vill ändra något i bilden sker detta via VDPn. Denna utför en read-modify-writecykel (RMW) enligt övre delen i figur 5.6. Anledningen till RMW-förfarandet är att man alltid arbetar med 16-bitars ord. De bitar som inte ska ändras måste ändå läsas in i VDPn och sedan skrivas tillbaks oförändrade. Hur modifieringen går till i VDPn framgår av appendix 1. Här ska vi se vad som händer utanför VDPn.

För att visa att VDPn vill göra en RMW-cykel låter den DBIN gå låg. Denna signal kan direkt styra en output enable-funktion på minneskorten



Figur 5.6

för inläsningen. Efter ytterligare en VDP-klockperiod läggs det modifierade dataordet ut på bussen och i detta skede måste man generera en write-puls externt. Eftersom enda gången man vill ha en write-puls är när DBIN precis tidigare varit aktiv använder vi DBIN för att styra två D-vippor (krets 27). Dessa klockas av 2XWCLK och hela tidsförloppet framgår av nedre delen av figur 5.6.

De två översta adressbitarna från VDPn (A16 och A17) används för att välja ut ett av de fyra minneskorten. Tyvärr är dessa bitar inte korrekta den sista klockperioden i RMW-cyklerna, varför man måste låsa dem i krets 12b. (Anledningen till b:et i numreringen är att kretsen lagts till i efterhand. I den första specifikationen på VDPn angavs nämligen felaktigt att A16 och A17 var korrekta hela cykeln.) Krets 12a är en avkodare som utför valet av minneskort.

Kretsarna 2 och 3 driver adress- och databussen till minneskorten. De kan stängas med en yttre enable-signal om man skulle vilja ha direkt åtkomst till bildminnet från någon annan enhet.

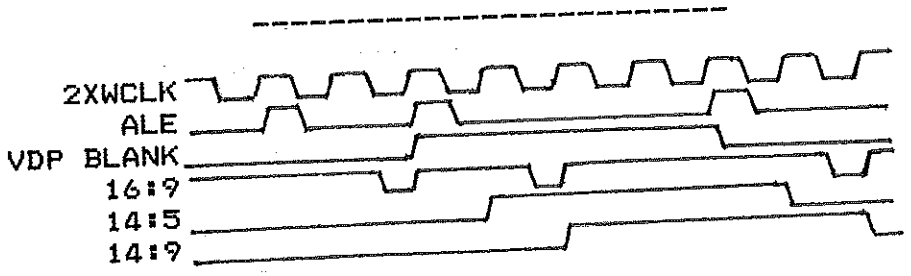
Det som återstår på videokortet är hanteringen av pixelbitarna. Dessa kommer, en från varje plan, i takt med pixelskiftklockan. De tre RGB-bitarna går först till en EXOR-grind. Denna utför en invertering av bitarna om biten från det fjärde planet (i schemat betecknad med X) är satt, och

man valt att utnyttja denna funktion genom att strappa 24:10-11. Vidare i behandlingsvägen sitter en NAND-krets som också den kan styras av X-signalen (via strap 24:9-11). Den styrs dessutom alltid av blankingsignalen från VDPn. Denna är aktiv under de mjukvarumässigt specificerade blank- och synkintervallerna.

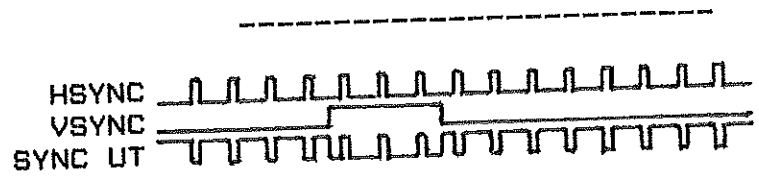
Utskiftningen av pixelbitarna ligger "ett steg efter" VDPns utläsning av dataordet och därför måste blankpulsen fördröjas med motsvarande tid. Detta görs med krets 14 vars tidsdiagram finns i figur 5.7.

Från NAND-grindarna går RGB-informationen till en låskrets (30) som klockas av pixelskiftpulserna. Denna krets förhindrar att eventuell olika fördröjning i tidigare steg blir märkbar på skärmen.

Som sista steg sitter för varje utgång en line driver av typ 74S140. Tillsammans med en resistor på 300 ohm ger dessa en utsignal på 0 till 2 volt vilket är standard för videosignalsöverföring med RGB-uppdelning. Synksignalen avslutas på samma sätt men utan resistor eftersom man här vill ha en nivå mellan 0 och 4 volt. Före slutsteget har de separata vertikala och horisontala synkutgångarna från VDPn mixats ihop i en EXOR-grind. Denna gör att man får inverterade horisontalsynkpulser inuti vertikalsynkpulsen. På så sätt får man en stabilare synkronisering av horisontalavlänkningen i monitorn. Se pulsschema i figur 5.8.



Figur 5.7



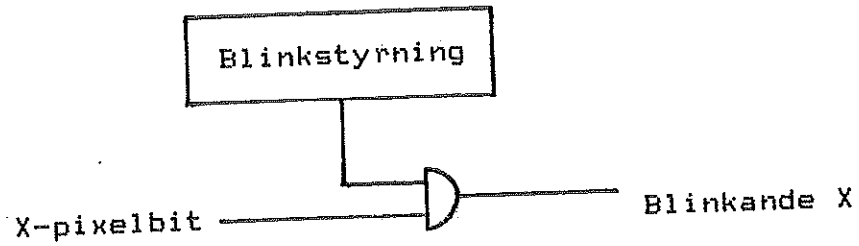
Figur 5.8

På schemat finns även en ingång för ljuspenna. Kravet för att VDPn ska registrera ljuspennans position är att samma punkt på bildskärmen i två efterföljande bildscan ska ge upphov till pulser på ingången. Denna funktion har ej testats.

Om vi återgår till informationen från det fjärde planet ska vi se på en annan möjlighet att utnyttja det på. Man kan låta detta plan markera blinkande fält i bilden. Hårdvarumässigt kan detta lösas på flera sätt, men grunden är en koppling enligt figur 5.9.

De ställen i bilden som ska blinka markeras med ettor i fjärde minnesplanet. Blinkstyrningen pulsar mellan ett och noll och utsignalen kan sedan styra en EXOR- eller NAND-grind precis som i den tidigare lösningen. Blinkstyrningen kan göras på olika sätt. En mycket flexibel lösning är att använda en latch som sätts och nollställs av mikroprocessorn. Detta gör att man kan välja blinkintervaller helt godtyckligt genom att ändra i mjukvaran. Nackdelen kan vara att man får ytterligare en sak att hålla reda på i sina program.

Man kan även ordna blinket helt med hårdvara. Då sätter man in en räknare som klockas av vertikalsynkutgången. Denna signal är normalt på ca 50 Hz och med en fyra eller fembits räknare får man då ca 3 respektive 1.5 Hz blinkfrekvens. Dessutom blir blinkningen synkroniserad med bildfrekvensen. Man kan också tänka sig att styra räknaren från mikroprocessorn på samma sätt som för zoomningsfunktionen och på så sätt få variabel blinkfrekvens.

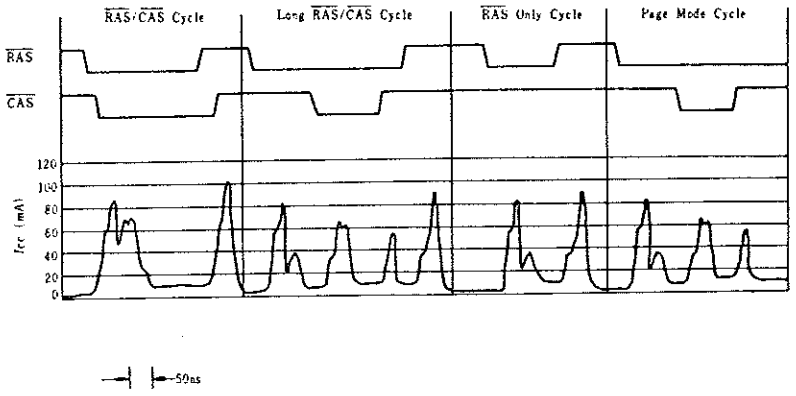


Figur 5.9

5.2.2 Minneskortet

Minneskortet är enklare att beskriva än videokortet, men i gengäld ska vi se på två något olika versioner. Först beskrivs den version som diskuterats tidigare, alltså ett kort med en fjärdedel av det totala minnet och där varje kort styr en grundfärg plus x-planet. Sedan ska vi se vilka förändringar som behövs för att köra systemet med bara ett minneskort som ändå rymmer fyra plan. (Naturligtvis blir det totala minnet också fyra gånger mindre och därmed blir upplösningen på skärmen hälften så stor. (Max 512 X 512 pixel)). Anledningen till den senare konfigurationen är att man får ett billigare och kompaktare system för de tillfällen då man kan nöja sig med lägre upplösning.

Vi börjar med att titta på den första varianten. Den mest dominerande delen är de sexton dynamiska minneskapslarna. Alla dessa har den multiplexade adressbussen på åtta bitar, RAS-, CAS-, och Write-signalerna gemensamma. De återstående pinnarna, data in och data out, är kopplade så att varje kapsel utgör en bit i sextonbitarsordet. En sak att notera är att varje minneskapsel har en kondensator kopplad över spänningsmatningen. Dessa kondensatorer finns där beroende på de dynamiska minnenas stora variationer i drivströmmen (Se figur 5.10).



Figur 5.10

De dynamiska minnenas datautgångar blir aktiva när CAS går låg och förblir så tills CAS åter går hög. Detta medför att under en RMW-cykels sista del (write) är både minnenas och VDPns datautgångar aktiva. Eftersom två aktiva utgångar inte kan kopplas samman sätter vi in en buffert (kretsarna 24 och 25) mellan dessa utgångar. Bufferten styrs av VDPns DBIN-signal så att dess utgångar bara är aktiva under VDPns read-faser. Ett annat sätt man kan lösa problemet på är att utnyttja de dynamiska minnenas "early write"-mod. Denna fungerar på så sätt att om write-ingången aktiveras före CAS blir datautgångarna aldrig aktiva under den minnescykeln. Vid RMW-operationer är denna metod svår att använda eftersom CAS måste gå låg redan när utläsningen görs. CAS skulle sedan behöva gå hög igen innan skrivdelen kunde göras enligt "early write"-sättet.

Dessutom måste man ändå ha en buffert om man har mer än ett minneskort kopplat till samma buss eftersom alla korten aktiveras vid pixeldata-utläsningen.

Mellan minneskapslarna och bufferten tar man ut pixeldatan till skiftregistren. Dessa består av två seriekopplade 74S299 TTL-kretsar, som är ett åttabitars skiftregister med parallella in- och utgångar och möjlighet att skifta i båda

riktningarna. I konstruktionen utnyttjas bara de parallella ingångarna och en seriell utgång samt skiftning åt ett håll. De har en garanterad övre skiftfrekvens på 50 MHz och har alltså en marginal till de 40 MHz som pixelskiftklockan är på. Skiftklocka och laddpulser framgår av tidsdiagrammet i avsnittet om videokortet.

Om vi nu övergår från datahantering till adresshantering ser vi först att dessa två delar kan kopplas samman med strapparna 26 och 27. Detta beroende på att kortet ska kunna användas till dels system med multiplexad adress- och data-buss, dels system med separata sådana. VDPn har adresser och data på samma buss så i detta fall använder vi strapparna och kopplar VDPn till en av bussarna på kortkontakten.

Adressen från VDPn går till TMS 4500-kretsen och är här kopplad så att den minst signifikanta halvan utgör radadress till minnena. Denna koppling gör att radadressen räknas upp ett steg i taget när datan utläses till skiftregistren. Eftersom radadressen också fungerar som refreshadress vid en minnescykel kommer därför minnet att vara helt refreshat efter 128 minnesaccesser. Med upplösningen 1024 pixels per linje har man 64 accesser per linje och det behövs alltså två linjer för fullständig refresh. Två linjer tar max 100 µs som ligger väl innanför max tillåten refreshtid som är 2 ms. Även vid vertikalblankingen, då ingen utläsning sker, klarar man tidskraven eftersom denna tar max 1.3 ms. Om man tänker använda lägre upplösning eller zoomning (då minnet inte scannas sekvensiellt) måste man kontrollera att detta refreshsätt fungerar, annars får man utnyttja den inbyggda refreshmöjligheten i VDPn.

Från TMS 4500 går adressen, halva i taget, ut till de sexton minnena där de latchas in med RAS- och CAS-pulserna som genereras i 4500. Denna krets kan direkt driva upp till 34 adressingångar och man behöver därför ingen speciell drivkrets häremellan.

Selekteringen av minneskorten måste kunna ske på två sätt. När man ändrar i bilden ska man kunna välja ut ett av de fyra korten och när pixeldatan ska läsas ut ska alla fyra kort aktiveras. Detta

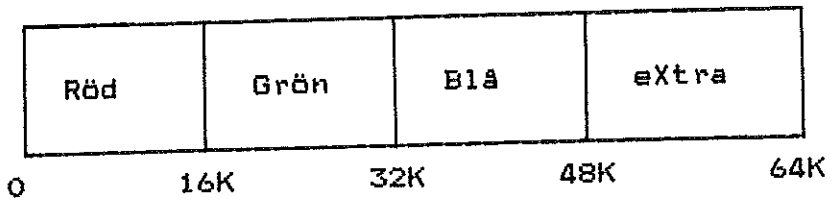
kan man lösa genom att låta ALE-signalen från VDPn starta en minnescykel på alla korten. Om den aktuella cykeln är en pixeldatautläsning laddas sedan skiftregistren på vanligt sätt. Om det är en RMW-cykel väljs ett av korten med card select-signalerna från videokortet. Som vi ser på kopplingschemat öppnar denna signal grindarna för DBIN och Write (krets 29). (Write-signalen går sedan till minnena via en buffert (krets 28) eftersom denna signal driver alla sexton minneskapslarna.) De övriga planen utför bara en pixeldatautläsning och denna har ingen betydelse eftersom blanksignalen från VDPn är aktiv under alla RMW-cykler. Detta förhindrar att oönskad data når skärmen.

När kortet används som ett vanligt minneskort i ett kortsystem har man normalt bara en signal att styra kortet med. Därför använder man strap 29:8-9 som kopplar samman ALE- och CS-ingångarna. En aktivering av en av dessa ingångar kommer då att både starta en minnescykel hos 4500-kretsen och öppna grindarna för DBIN och Write.

Den andra versionen av minneskortet finns uppritad på ett kopplingschema i appendix 2. Detta är inte fullständigt utan bara de förändrade delarna är redovisade. Resten är samma som i förra versionen.

Grundprincipen när vi nu ska använda ett kort är fortfarande att man har fyra pixelbitsplan, kan läsa ut 64 bitar på ett sätt så att en pixels alla bitar kommer fram samtidigt till videokortet och att minnesplanen är organiserade så att man kan utnyttja figurritningen i VDPn. Vi organiserar minnet enligt figur 5.11.

Nu kan vi utnyttja figurritningen precis som tidigare, eftersom VDPn för detta förutsätter ett sekvensiellt ordnat minne enligt figur 5.11. Vid pixeldatautläsning lägger VDPn också ut adresserna sekvensiellt, och på något sätt måste vi "lura" adresserna att hoppa mellan planen. Vi vill alltså kunna läsa ut orden i ordningen: Rött plans första ord, Grönts första, Blåtts första, eXtras första, Rötts andra, Grönts andra o. s. v.. Om vi tar bort de två lägsta adressbitarna ser vi att resten räknar upp efter denna ordning. För att välja plan använder vi de två lägsta bitarna (se tabell



Figur 5.11

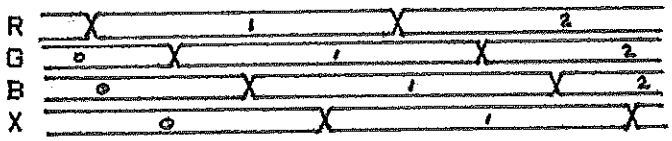
5.12).

I praktiken är detta lika med att skifta hela adressen två steg nedåt och flytta A1 och A0 till A15 resp. A14. Detta gör vi i konstruktionen med fyra fourbits data selectors (74LS157). Valet mellan de två adressmoderna styrs av blanksignalen från VDPn. Denna är alltid låg vid pixelutläsning och hög annars. Nu kan vi alltså läsa ut ett sextonbitarsord från varje plan. Dessa ord måste laddas in i skiftregistren, ett (16-bits) för varje plan. För att välja vilket register som ska laddas styr vi en decoder med den vanliga laddpulsen och de adressbitar som väljer ut planen (A0 och A1).

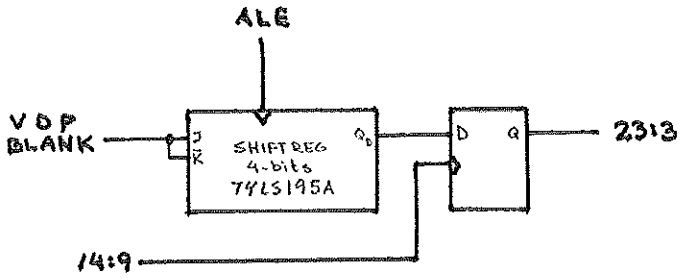
För att få ut en pixels alla bitar samtidigt fastän de laddas in vid olika tidpunkter, måste vi fördröja det först inladdade ordet tolv pixelklockpulser, det andra ordet åtta pulser och det tredje fyra. Detta görs med skiftregistren 30, 31, 32 och 33. Se figur 5.13. Dessutom måste fördröjningen av blankpulsen förlängas med tolv klockpulser. Detta görs på videokortet enligt figur 5.14. För att läsa ut 64 bitar och skifta ut dessa i grupper om fyra måste även pixelklockan divideras med fyra. Detta görs enkelt genom att sätta zoomlatchen till tre.

A15														A0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Röd första
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Grön första
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	Blå första
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	eXtra första
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	Röd andra
0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	Grön andra
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	. . . . .

Tabell 5.12



Figur 5.13



figur 5.14

6. Vad blev det ?

I detta kapitel ska vi se om det verkligen blev det vi ville ha av konstruktionen. Först testningen för att se om konstruktionen fungerar. Därefter ska vi se om det finns några detaljer som kunde konstruerats på ett bättre sätt. Till sist en sammanfattning av det hela.

6.1 Testning och mjukvara

Detta kapitel handlar om både testningen och mjukvaran av den anledningen att den mjukvara som skrivits har använts vid testningen. Mjukvaran består av dels program i assembly och BASIC som använts för att testa konstruktionens funktion och VDPns kommandouppsättning, dels av ett demonstrationsprogram skrivet i assembly som visar VDPns snabbhet och möjligheter att rita figurer.

Vid den första delen av testningen användes ett enkelt assemblyprogram för mikroprocessorn på videokortet. Detta finns återgivet i appendix 3. Programmet initierar VDPn och hämtar sedan eventuella vidare kommandon från MPIF-kretsen. Dessa kommandon är av VDP-struktur och de skickas i stort sett direkt till VDPn. Den enda avkodning av kommandon gäller ändring av zoomfaktor där zoomlatchen måste sättas till motsvarande värde, samt uppritning av tecken som finns lagrade i EPROM 1. Som värddator har en Commodore VIC-20 bordsdator använts. Ett interaktivt BASIC-program har härifrån skickat kommandona till MPIF-kretsen. Detta program finns ej listat i denna skrift, men principen är att koden för det valda kommandot POKEats i en minnesposition där MPIF finns (POKE A,X betyder lägg värdet X i minnesposition A).

Vid denna testning upptäcktes bland annat, som tidigare nämnts, att resetfunktionen fungerade tvivelaktigt. Vidare fanns det vissa problem med överföringen mellan VIC-20 och MPIF-kretsen, orsakade av en för lång kabel häremellan. I övrigt fungerade kretsar och kommandon som de skulle.

Vid den första delen av testningen gjordes all kommandohantering i BASIC och var därför tämligen långsam. För att få en uppfattning om snabbheten i uppdatering vid bildbyten och figurritning kördes därför ett rent assemblyprogram (appendix 3) på videokortet. Detta program består av ett antal subrutiner för linjedragning, areafyllning, teckenskrivning, tidsfördröjning m. m.. Dessa subrutiner kan kallas in från ett huvudprogram som hämtar kommandosekvensen från en tabell DEMTAB. Kommandona i denna tabell är av typ LINE,320,65,45,258,RED (dra en röd linje från (320,65) till (45,258)). Genom att ändra i DEMTAB kan man enkelt göra om eller lägga till bilder. Tillgängliga kommandon och deras parametrar finns också i appendix 3.

Demonstrationsprogrammet kördes på ett system med ett minneskort och upplösning mellan 300 och 500 pixels per linje och motsvarande antal linjer.

Denna senare del av testningen visade att en areafyllning av ett 320 X 360 pixels område tar omkring en halv sekund. På denna tid har VDPn alltså ändrat på samtliga bitar i den del av bildminnet som visas på skärmen. Om vi nu kommer ihåg det tidigare resonemanget om att man inte bör ha mer än 25 % av bildytan täckt med information skulle detta medföra att en bild kunde ritas upp på ca en tiondels sekund. Detta förutsätter att mikroprocessorn hinner med att ge VDPn instruktioner i motsvarande takt.

6.2 Vidareutveckling

Vi ska här se på några saker som kan utvecklas mer än den givna konstruktionen. För det första är uppbyggnaden på enkla europakort (160 X 100 mm)

diskutabel. Dessa kort placeras i en rack där även andra kort kommer att sitta. Alla förbindningar mellan korten (t.ex. adress- och databuss) sker i bakplanet av racken. Dessa förbindelser bör vara standardiserade för alla rackar och alla kort, så att man hårdvarumässigt inte behöver låsa ett in/ut-kort till en bestämd plats. Videokortet och minneskorten behöver en egen adress- och databuss vilket inte överensstämmer med nämnda filosofi. Därför bör man ha hela videosystemet på ett kort med standardkoppling till rackens bakplan.

Nu är det omöjligt att packa in alla kretsar på ett enkelt europakort, men på två dubbla europakort (233 X 160 mm) får allt plats. På det ena kortet har man då kretsarna motsvarande videokortet plus ett minneskort kopplat enligt den senare beskrivna metoden för bildminnet d.v.s. fyra bildplan på ett kort. Detta kort ska kunna köras ensamt med lägre upplösing. Det andra dubbla europakortet har de övriga tre minnesplanen och kan mekaniskt kopplas samman med det första kortet till en sandwich-konstruktion. Elektriskt kopplas korten samman med t.ex. flatkabel så att man inte behöver bryta upp kopplingarna i rackens bakplan.

En annan förbättring i konstruktionen gäller behandlingen av pixelbiten från fjärde planet i videodelen. För att välja funktion hos denna bit används nu strappar som måste flyttas mekaniskt. Ett bättre sätt är att ha en latch, styrd av mikroprocessorn, som väljer funktionen. Detta kan realiseras med en latch och några NAND-grindar. På detta sätt kan man mjukvarumässigt bestämma om man vill ha blinkande fält, inverterade fält e.dyl..

6.3 Sammanfattning

Tro aldrig på en försäljares leveranstider. Detta kan vara sensmoralen av examensarbetet efter alla de fördröjningar som varit. Nåväl, konstruktionen blev till sist klar och visade sig fungera som den var avsedd att göra. Inga speciella svåra och tidsödande problem dök upp i själva hårdvarudelen, medan det på programsidan

varit något besvärligare. Detta beror till största del på att det inte funnits någon emulator för den nya mikroprocessorn TMS 9995. Programmen har fått lagras i EPROM vilket betyder att det varit svårare att hitta och tagit längre tid att ändra fel i dem.

Prestandamässigt hävdar sig denna konstruktion väl i förhållande till tänkbara konkurrenter. Upplösningen är varierbar från noll till 1024 pixels i både vertikal och horisontell led och VDPn klarar av att rita upp en bild med högsta upplösning och 25 % täckning på mindre än en sekund. Tack vare mikroprocessorn på videokortet kan man hålla kommunikationen till enheten på en hög nivå vilket ger en låg belastning på värddatorn. Dessutom kan man enkelt ändra videoenhetens "kunnande" om så skulle vara önskvärt.

Innan denna videoenhet är klar att säljas på öppna marknaden krävs en avsevärd mjukvaruutveckling, dels för mikroprocessorn på videokortet, dels på nivåerna ovanför. Det är dock min förhoppning att detta med hjälp av den gjorda konstruktionen uppbyggnad och en lämplig nivåuppdelning inte ska bli alltför betungande.

APPENDIX

1. Beskrivningar på använda kretsar

NEC µPD 7220	: Ur Electronics 7 apr. 1981
	: Utdrag ur datablad
TMS 9995	: Ur Electronic Design 22 nov. 1980
TMS 99650	: Ur Electronic Design 12 nov. 1981
TMS 4500	: Utdrag ur datablad

2. Kopplingsscheman

Videokort höger del
Videokort vänster del
Minneskort 64K X 16 bits
Minneskort modifierat (endast förändringar)
Komponentlista videokort
Komponentlista minneskort

3. Program

Parameteröverföring för första testdelen
Demonstrationsprogram i assembly

4. Litteraturförteckning

APPENDIX_1

Beskrivningar på använda kretsar

NEC µPD 7220	: Ur Electronics 7 apr. 1981
	: Utdrag ur datablad
TMS 9995	: Ur Electronic Design 22 nov. 1980
TMS 99650	: Ur Electronic Design 12 nov. 1981
TMS 4500	: Utdrag ur datablad

μPD7220/GDC
GRAPHICS DISPLAY CONTROLLER
PRELIMINARY

Description

The μPD7220 Graphics Display Controller (GDC) is an intelligent microprocessor peripheral designed to be the heart of a high-performance raster-scan computer graphics and character display system. Positioned between the video display memory and the microprocessor bus, the GDC performs the tasks needed to generate the raster display and manage the display memory. Processor software overhead is minimized by the GDC's sophisticated instruction set, graphics figure drawing, and DMA transfer capabilities. The display memory supported by the GDC can be configured in any number of formats and sizes up to 256K 16-bit words. The display can be zoomed and panned, while partitioned screen areas can be independently scrolled. With its light pen input and multiple controller capability, the GDC is ideal for advanced computer graphics applications.

Features

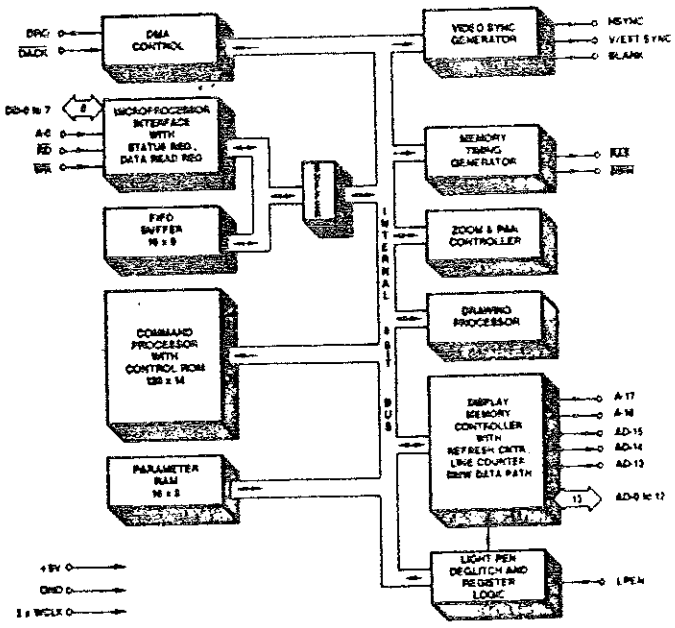
- ☐ Microprocessor Interface
  - DMA transfers with 8257 or 8237-type controllers
  - FIFO Command Buffering
- ☐ Display Memory Interface
  - Up to 256K words of 16 bits
  - Read-Modify-Write (RMW) Display Memory cycles in under 800ns
  - Dynamic RAM refresh cycles for non-accessed memory
- ☐ Light Pen Input
- ☐ External video synchronization mode
- ☐ Graphics Mode:
  - Four megabit, bit-mapped display memory
- ☐ Character Mode:
  - 8K character code and attributes display memory
- ☐ Mixed Graphics and Characters Mode
  - 64K if all characters
  - 1 megapixel if all graphics
- ☐ Graphics Capabilities:
  - Figure drawing of lines, arc/circles, rectangles, and graphics character in 800ns per pixel
  - Display 1024 by 1024 pixels with 4 planes of color or grayscale
  - Two independently scrollable areas
- ☐ Character Capabilities:
  - Auto cursor advance
  - Four independently scrollable areas
  - Programmable cursor height
  - Characters per row: up to 256
  - Character rows per screen: up to 100
- ☐ Video Display Format
  - Zoom magnification factors of 1 to 16
  - Panning
  - Command-settable video raster parameters
- ☐ Technology
  - Single +5 volt, NMOS, 40-pin DIP
- ☐ DMA Capability:
  - Bytes or word transfers
  - 4 clock periods per byte transferred

System Considerations

The GDC is designed to work with a general purpose microprocessor to implement a high-performance computer graphics system. Through the division of labor established by the GDC's design, each of the system components is used to the maximum extent through six-level hierarchy of simultaneous tasks. At the lowest level, the GDC generates the basic video raster timing, including sync and blanking signals. Partitioned areas on the screen and zooming are also accomplished at this level. At the next level, video display memory is modified during the figure drawing operations and data moves. Third, display memory addresses are calculated pixel by pixel as drawing progresses. Outside the GDC at the next level, preliminary calculations are done to prepare drawing parameters. At the fifth level, the picture must be represented as a list of graphics figures drawable by the GDC. Finally, this representation must be manipulated, stored, and communicated. By handling the first three levels, the GDC takes care of the high-speed and repetitive tasks required to implement a graphics system.

GDC Components

The GDC block diagram illustrates how these tasks are accomplished.



Microprocessor Bus Interface

Control of the GDC by the system microprocessor is achieved through an 8-bit bi-directional interface. The status register is readable at any time. Access to the FIFO buffer is coordinated through flags in the status register and operates independently of the various internal GDC operations, due to the separate data bus connecting the interface and the FIFO buffer.

Command Processor

The contents of the FIFO are interpreted by the command processor. The command bytes are decoded, and the succeeding parameters are distributed to their proper destina-

tions within the GDC. The command processor yields to the bus interface when both access the FIFO simultaneously.

DMA Control

The DMA control circuitry in the GDC coordinates transfers over the microprocessor interface when using an external DMA controller. The DMA Request and Acknowledge handshake lines directly interface with a μPD8257 or μPD8237 DMA controller, so that display data can be moved between the microprocessor memory and the display memory.

Parameter RAM

The 16-byte RAM stores parameters that are used repetitively during the display and drawing processes. In character mode, this RAM holds four sets of partitioned display area parameters; in graphics mode, the drawing pattern and graphics character take the place of two of the sets of parameters.

Video Sync Generator

Based on the clock input, the sync logic generates the raster timing signals for almost any interlaced, non-interlaced, or "repeat field" interlaced video format. The generator is programmed during the idle period following a reset. In video sync slave mode, it coordinates timing between multiple GDCs.

Memory Timing Generator

The memory timing circuitry provides two memory cycle types: a two-clock period refresh cycle and the read-modify-write (RMW) cycle which takes four clock periods. The memory control signals needed to drive the display memory devices are easily generated from the GDC's RAS and DBIN outputs.

Zoom & Pan Controller

Based on the programmable zoom display factor and the display area entries in the parameter RAM, the zoom and pan controller determines when to advance to the next memory address for display refresh and when to go on to the next display area. A horizontal zoom is produced by slowing down the display refresh rate while maintaining the video sync rates. Vertical zoom is accomplished by repeatedly accessing each line a number of times equal to the horizontal repeat. Once the line count for a display area is exhausted, the controller accesses the starting address and line count of the next display area from the parameter RAM. The system microprocessor, by modifying a display area starting address, can pan in any direction, independent of the other display areas.

Drawing Processor

The drawing processor contains the logic necessary to calculate the addresses and positions of the pixels of the various graphics figures. Given a starting point and the appropriate drawing parameters, the drawing processor needs no further assistance to complete the figure drawing.

Display Memory Controller

The display memory controller's tasks are numerous. Its primary purpose is to multiplex the address and data information in and out of the display memory. It also contains the 16-bit logic unit used to modify the display memory contents during RMW cycles, the character mode line counter, and the refresh counter for dynamic RAMs. The memory controller apportions the video field time between the various types of cycles.

Light Pen Deglitcher

Only if two rising edges on the light pen input occur at the same point during successive video fields are the pulses accepted as a valid light pen detection. A status bit indicates to the system microprocessor that the light pen register contains a valid address.

Programmer's View of GDC

The GDC occupies two addresses on the system microprocessor bus through which the GDC's status register and FIFO are accessed. Commands and parameters are written into the GDC's FIFO and are differentiated based on address bit A0. The status register or the FIFO can be read as selected by the address line.

Commands to the GDC take the form of a command byte followed by a series of parameter bytes as needed for specifying the details of the command. The command processor decodes the commands, unpacts the parameters, loads them into the appropriate registers within the GDC, and initiates the required operations.

The twenty commands available in the GDC can be organized into five categories as described in the following section.

GDC Command Summary

Video Control Commands

1. RESET: Resets the GDC to its idle state and specifies the video display format.
2. VSYNC: Selects master or slave video synchronization mode.
3. CCHAR: Specifies the cursor and character row heights.

Display Control Commands

1. START: Starts the display scanning process.
2. ZOOM: Specifies zoom factors for the display and graphics characters writing.
3. CURS: Sets the position of the cursor in display memory.
4. PRAM: Defines starting addresses and lengths of the display areas and specifies the eight bytes for the graphics character.
5. PITCH: Specifies the width of the X dimension of display memory.

Drawing Control Commands

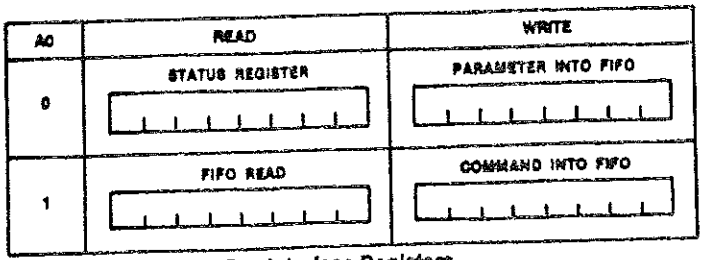
1. WDAT: Writes data words or bytes into display memory.
2. MASK: Sets the mask register contents.
3. FIGS: Specifies the parameters for the drawing processor.
4. FIGD: Draws the figure as specified above.
5. GCHRD: Draws the graphics character into display memory.

Data Read Commands

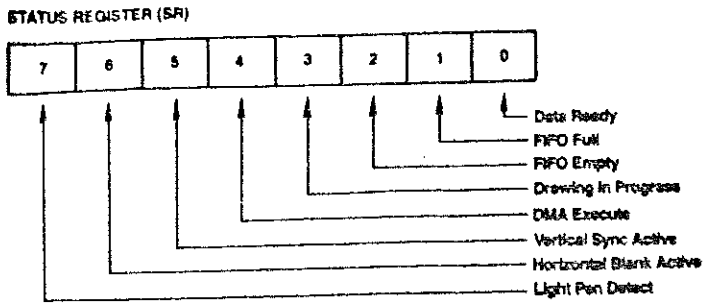
1. RDAT: Reads data words or bytes from display memory.
2. CURD: Reads the cursor position.
3. LPRD: Reads the light pen address.

DMA Control Commands

1. DMAR: Requests a DMA read transfer.
2. DMAW: Requests a DMA write transfer.



GDC Microprocessor Bus Interface Registers



Status Register Flags

SR-7: Light Pen Detect
When this bit is set to 1, the light pen address (LAD) register contains a deglitched value that the system microprocessor may read. This flag is reset after the 3-byte LAD is moved into the FIFO in response to the light pen read command.

SR-6: Horizontal Blanking Active
A 1 value for this flag signifies that horizontal retrace blanking is currently underway.

SR-5: Vertical Sync
Vertical retrace sync occurs while this flag is a 1. The vertical sync flag coordinates display format modifying commands to the blanked interval surrounding vertical sync. This eliminates display disturbances.

SR-4: DMA Execute
This bit is a 1 during DMA data transfers.

SR-3: Drawing In Progress
While the GDC is drawing a graphics figure, this status bit is a 1.

SR-2: FIFO Empty
This bit and the FIFO Full flag coordinate system microprocessor accesses with the GDC FIFO. When it is 1, the Empty flag ensures that all the commands and parameters previously sent to the GDC have been processed.

SR-1: FIFO Full
A 1 at this flag indicates a full FIFO in the GDC. A 0 ensures that there is room for at least one byte. This flag needs to be checked before each write into the GDC.

SR-0: Data Ready
When this flag is a 1, it indicates that a byte is available to be read by the system microprocessor. This bit must be tested before each read operation. It drops to a 0 while the data is transferred from the FIFO into the microprocessor interface data register.

FIFO Operation & Command Protocol

The first-in, first-out buffer (FIFO) in the GDC handles the command dialogue with the system microprocessor. This flow of information uses a half-duplex technique, in which the single 16-location FIFO is used for both directions of data movement, one direction at a time. The FIFO's direction is controlled by the system microprocessor through the GDC's command set. The microprocessor coordinates these transfers by checking the appropriate status register bits.

The command protocol used by the GDC requires the differentiation of the first byte of a command sequence from the succeeding bytes. This first byte contains the operation code and the remaining bytes carry parameters. Writing into the GDC causes the FIFO to store a flag value alongside the data byte to signify whether the byte was written into the command or the parameter address. The command processor in the GDC tests this bit as it interprets the entries in the FIFO.

The receipt of a command byte by the command processor marks the end of any previous operation. The number of parameter bytes supplied with a command is cut short by the receipt of the next command byte. A read operation from the GDC to the microprocessor can be terminated at any time by the next command.

The FIFO changes direction under the control of the system microprocessor. Commands written into the GDC always put the FIFO into write mode if it wasn't in it already. If it was in read mode, any read data in the FIFO at the time of the turnaround is lost. Commands which require a GDC response, such as RDAT CURD and LPRD, put the FIFO into read mode after the command is interpreted by the GDC's command processor. Any commands and parameters behind the read-evoking command are discarded when the FIFO direction is reversed.

Read-Modify-Write Cycle

Data transfers between the GDC and the display memory are accomplished using a read-modify-write (RMW) memory cycle. The four clock period timing of the RMW cycle is used to: 1) output the address, 2) read data from the memory, 3) modify the data, and 4) write the modified data back into the initially selected memory address. This type of memory cycle is used for all interactions with display memory including DMA transfers, except for the two clock period display and RAM refresh cycles.

The operations performed during the modify portion of the RMW cycle merit additional explanation. The circuitry in the GDC uses three main elements: the Pattern register, the Mask register, and the 16-bit Logic Unit. The Pattern register holds the data pattern to be moved into memory. It is loaded by the WDAT command or, during drawing, from the parameter RAM. The Mask register contents determine which bits of the read data will be modified. Based on the contents of these registers, the Logic Unit performs the selected operations of REPLACE, COMPLEMENT, SET, or CLEAR on the data read from display memory.

The Pattern register contents are ANDed with the Mask register contents to enable the actual modification of the memory read data, on a bit-by-bit basis. For graphics drawing, one bit at a time from the Pattern register is combined with the Mask. When ANDed with the bit set to a 1 in the Mask register, the proper single pixel is modified by the Logic Unit. For the next pixel in the figure, the next bit in the Pattern register is selected and the Mask register bit is moved to identify the pixel's location within the word. The Execution word address pointer register, EAD, is also adjusted as required to address the word containing the next pixel.

In character mode, all of the bits in the Pattern register are used in parallel to form the respective bits of the modify data word. Since the bits of the character code word are used in parallel, unlike the one-bit-at-a-time graphics drawing process, this facility allows any or all of the bits in a memory word to be modified in one RMW memory cycle. The Mask register must be loaded with 1s in the positions where modification is to be permitted.

The Mask register can be loaded in either of two ways. In graphics mode, the CURS command contains a four-bit dAD field to specify the dot address. The command processor converts this parameter into the one-of-16 format used in the Mask register for figure drawing. A full 16 bits can be loaded into the Mask register using the MASK command. In addition to the character mode use mentioned above, the 16-bit MASK load is convenient in graphics mode when all of the pixels of a word are to be set to the same value.

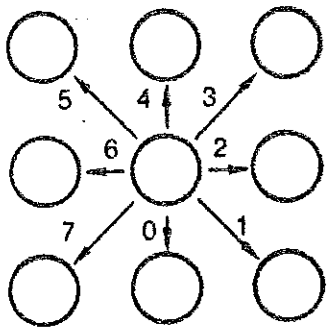
The Logic Unit combines the data read from display memory, the Pattern Register, and the Mask register to generate the data to be written back into display memory. Any one of four operations can be selected: REPLACE, COMPLEMENT, CLEAR or SET. In each case, if the respective Mask bit is 0, that particular bit of the read data is returned to memory unmodified. If the Mask bit is 1, the modification is enabled. With the REPLACE operation, the modify data simply takes the place of the read data for modification enabled bits. For the other three operations, a 0 in the modify data allows the read data bit to be returned to memory. A 1 value causes the specified operation to be performed in the bit positions with set Mask bits.

Figure Drawing

The GDC draws graphics figures at the rate of one pixel per read-modify-write (RMW) display memory cycle. These cycles take four clock periods to complete. At a clock frequency of 5MHZ, this is equal to 800ns. During the RMW cycle the GDC simultaneously calculates the address and position of the next pixel to be drawn.

The graphics figure drawing process depends on the display memory addressing structure. Groups of 16 horizontally adjacent pixels form the 16-bit words which are handled by the GDC. Display memory is organized as a linearly addressed space of these words. Addressing of individual pixels is handled by the GDC's internal RMW logic.

During the drawing process, the GDC finds the next pixel of the figure which is one of the eight nearest neighbors of the last pixel drawn. The GDC assigns each of these eight directions a number from 0 to 7, starting with straight down and proceeding counterclockwise.



Drawing Directions

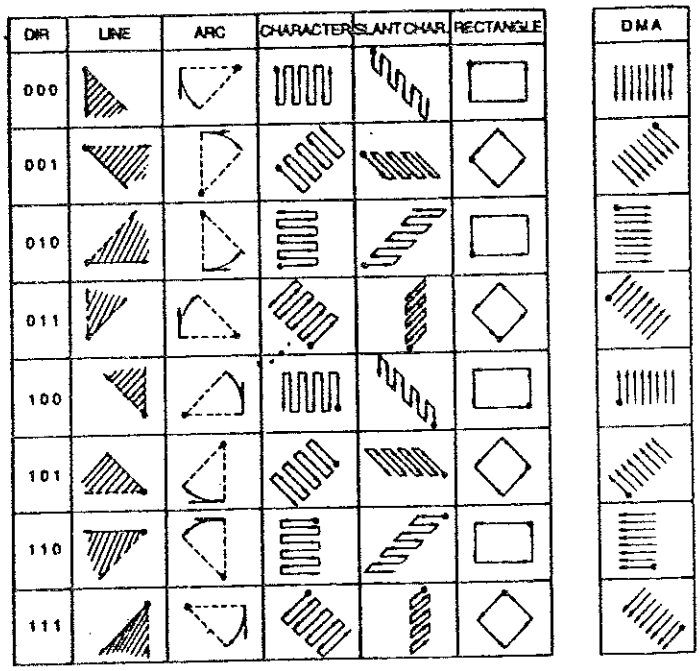
Figure drawing requires the proper manipulation of the address and the pixel bit position according to the drawing direction to determine the next pixel of the figure. To move to the word above or below the current one, it is necessary to subtract or add the number of words per line in display memory. This parameter is called the pitch. To move to the word to either side, the Execute word address cursor, EAD, must be incremented or decremented as the dot address pointer bit reaches the LSB or the MSB of the Mask register. To move to a pixel within the same word, it is necessary to rotate the dot address pointer register to the right or left. The table below summarizes these operations for each direction.

Whole word drawing is useful for filling areas in memory with a single value. By setting the Mask register to all 1s with the MASK command, both the LSB and MSB of the dAD will always be 1, so that the EAD value will be incremented or decremented for each cycle regardless of direction. One RMW cycle will be able to effect all 16 bits of the word for any drawing type. One bit in the Pattern register is used per RMW cycle to write all the bits of the word to the same value. The next Pattern bit is used for the word, etc.

DIR	OPERATIONS TO ADDRESS THE NEXT PIXEL
000	EAD + P → EAD
001	EAD + P → EAD dAD (MSB) = 1 : EAD + 1 → EAD dAD → LR
010	dAD (MSB) = 1 : EAD + 1 → EAD dAD → LR
011	EAD − P → EAD dAD (MSB) = 1 : EAD + 1 → EAD dAD → LR
100	EAD ÷ P → EAD
101	EAD − P → EAD dAD (LSB) = 1 : EAD − 1 → EAD dAD → RR
110	dAD (LSB) = 1 : EAD + 1 → EAD dAD → RR
111	EAD + P → EAD dAD (LSB) = 1 : EAD + 1 → EAD dAD → RR

Where P = Pitch, LR = Left Rotate, RR = Right Rotate
EAD = Execute Word Address
dAD = Dot Address stored in the Mask Register

For the various figures, the effect of the initial direction upon the resulting drawing is shown below:



Drawing Parameters

In preparation for graphics figure drawing, the GDC's Drawing Processor needs the figure type, direction and drawing parameters, the starting pixel address, and the pattern from the microprocessor. Once these are in place within the GDC, the Figure Draw command, FIGD, initiates the drawing operation. From that point on, the system microprocessor is not involved in the drawing process. The GDC Drawing Processor coordinates the RMW circuitry and address registers to draw the specified figure pixel by pixel.

The algorithms used by the processor for figure drawing are designed to optimize its drawing speed. To this end, the specific details about the figure to be drawn are reduced by the microprocessor to a form conducive to high-speed address calculations within the GDC. In this way the repetitive, pixel-by-pixel calculations can be done quickly, thereby minimizing the overall figure drawing time. The table below summarizes the parameters.

	DC	D	D2	D1	DM
Initial Value	0	δ	δ	−1	−1
Line	ΔX	2|ΔY|−|ΔX|	2|ΔY|−2|ΔX|	2|ΔY|	−
Circle	r/√2 ↑	r-1	2(r-1)	−1	0
Arc	r/√2 ↑	r-1	2(r-1)	−1	r sin θ ↓
Rectangle	3	A*	B*	−1	A*
Character	7	−	−	−	−
Area Fill	B**	A**	−	−	−
DMA	L	I	−	−	−
R/W DAT	W	−	−	−	−

Graphics Figure Drawing Parameters

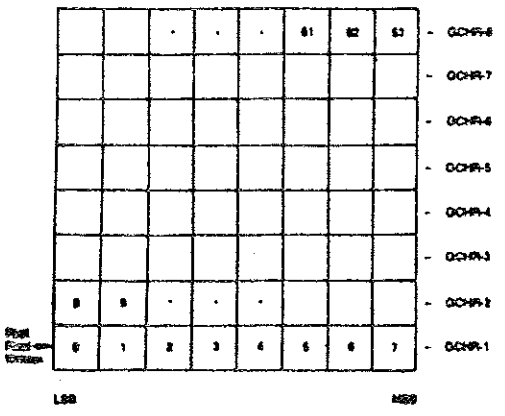
WHERE:

- ΔX = # of pixel positions in the line horizontally.
- ΔY = # of pixel positions in the line vertically.
- r = Radius of the curve, in pixels.
- I = Rounded up to the next higher integer.
- θ = The angle subtended from the octant axis to the arc.
- A* = The length − 1, in pixels, in the direction initially specified.
- B* = The length − 1, in pixels, in the direction at right angles to the initial direction.
- A** = # of pixels in the initial direction.
- B** = # of pixels in the right angle direction − 1.
- L = # of word address in the perpendicular direction for DMA.
- I = # of DMA transfers in the initially specified direction.
- W = # of word addresses to be written into.
- DC = Drawing Count.
- DM = Dots Masked Count.

Graphics Character Drawing

Graphics characters can be drawn into display memory pixel-by-pixel. The 8-by-8 character is loaded into the GDC's parameter RAM by the system microprocessor. Consequently, there are no limitations on the character set used. By varying the drawing parameters and drawing direction, numerous drawing options are available. In area fill applications, a character can be written into display memory as many times as desired without reloading the parameter RAM.

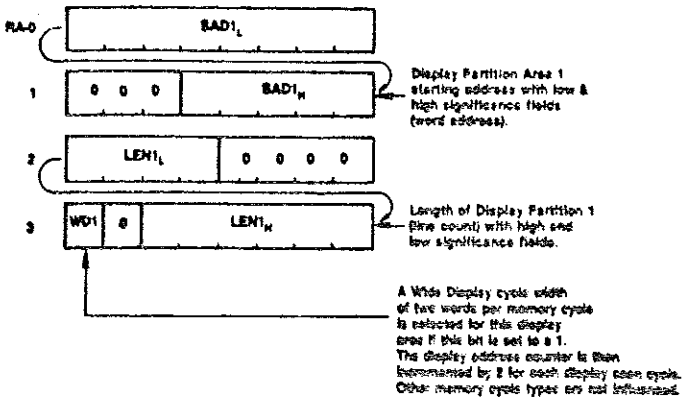
Once the parameter RAM has been loaded with the eight graphics character bytes by the appropriate PRAM command, the GCHRD command can be used to draw the eight bytes into display memory starting at the cursor. The zoom magnification factor for writing, set by the zoom command, controls the size of the character written into the display memory in integer multiples of 1 through 16. The bit values in the PRAM are repeated horizontally and vertically the number of times specified by the zoom factor. As seen in display memory, the writing is started in the lower left corner of the eventual 8-by-8 block and proceeds right to left and bottom to top. The LSBs of the stored character bytes are written first, and parameter RAM location 15 is the first byte to go out.

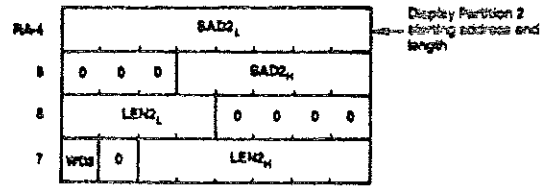


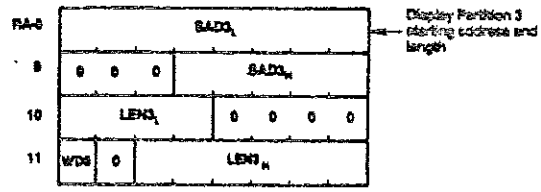
Display Memory

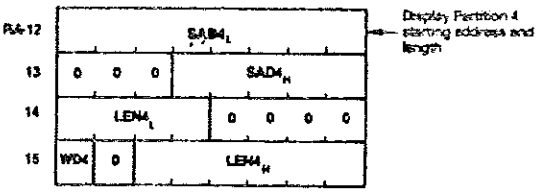
Parameter Ram Contents

Character Mode: Row Address (RA) 0 to 16

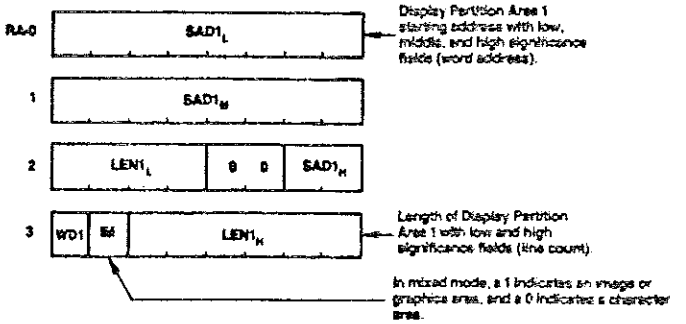


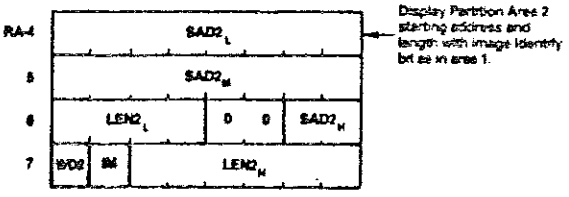






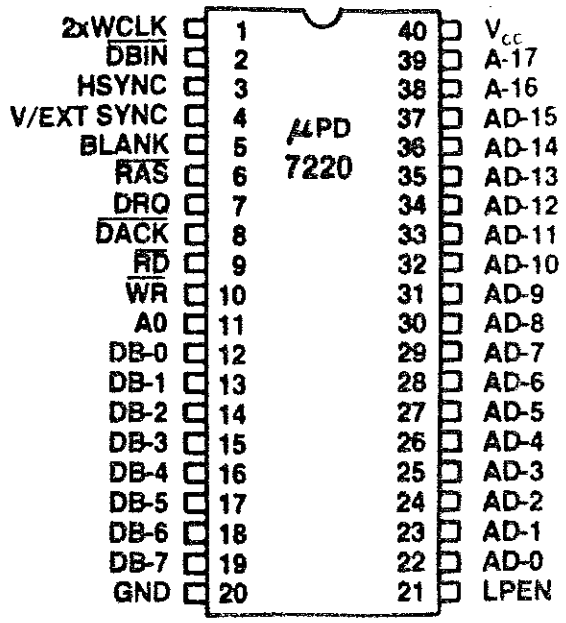
Graphics and Mixed Graphics and Characters Mode:



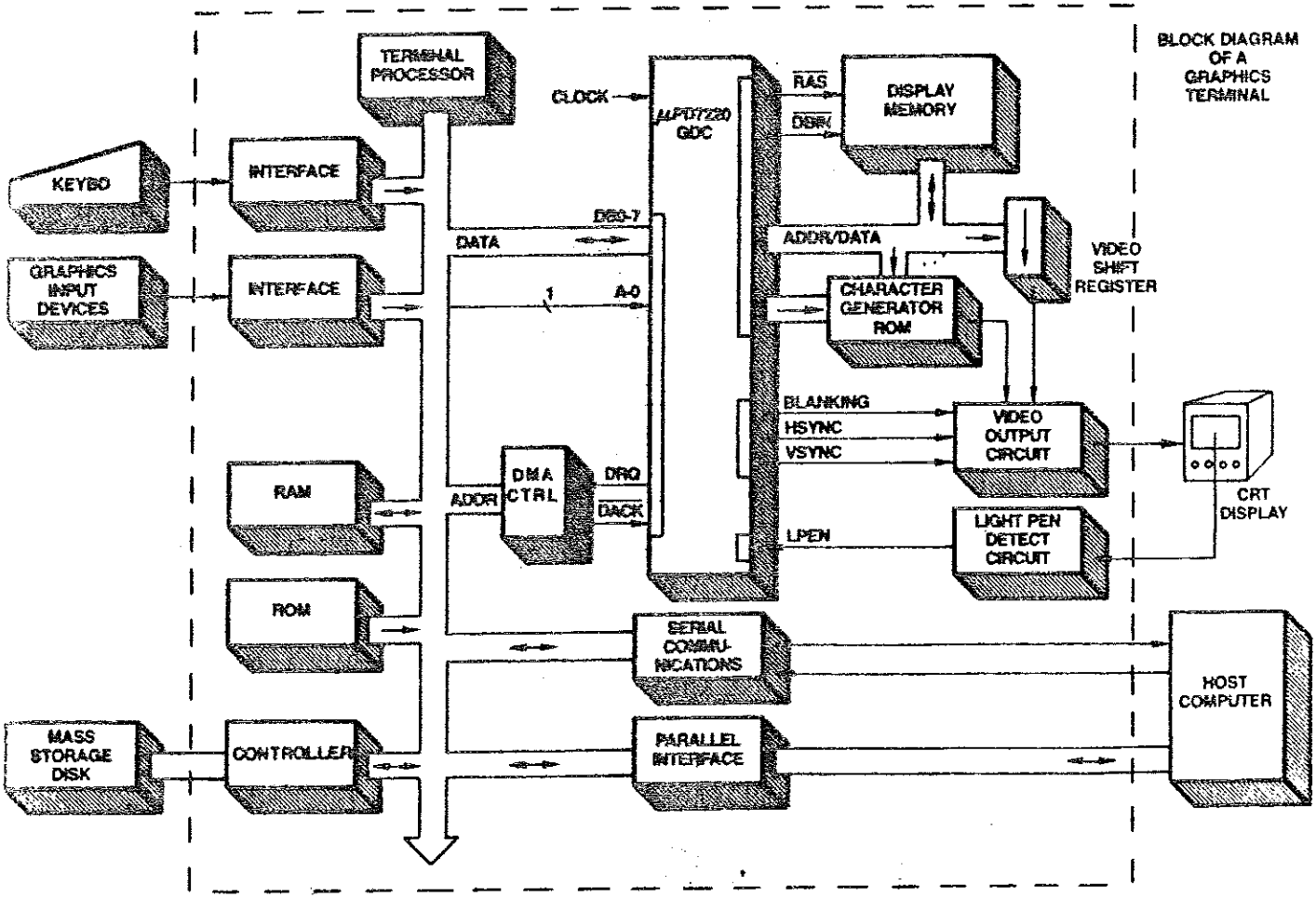


Pin #	Pin Name	Direction	Modes of Operation		
			Graphics	Character	Mixed
7	DRQ	OUT	DMA Request Output		
8	$\overline{DACK}$	IN	DMA Acknowledge Input		
12 to 19	DB-0 to 7	IN/OUT	Data Bus to Microprocessor		
11	A0	IN	Address Select Input for Microprocessor Interface		
9	$\overline{RD}$	IN	Read Strobe Input for Microprocessor Interface		
10	$\overline{WR}$	IN	Write Strobe Input for Microprocessor Interface		
3	HSYNC	OUT	Horizontal Video Sync Output		
4	V/EXT SYNC	IN/OUT	Vertical Video Sync Output or External Sync Input		
5	BLANK	OUT	CRT Beam Blanking Output		
1	2xWCLK	IN	Clock Input		
6	$\overline{RAS}$ (ALE)	OUT	Row Address Strobe and Address Latch Enable		
2	$\overline{DBIN}$	OUT	Display Memory Data Bus "Input" Read Cycle Output		
36	A-17	OUT	Address Bit 17 Output	Cursor Output	Cursor & Image Mode Flag
38	A-16	OUT	Address Bit 16 Output	Line Counter Bit 3	Attribute Blink & Clear Line Counter
35 to 37	AD-13 to 15	IN/OUT	Address Bits 13 to 15	Line Counter Bits 0 to 2	Address Bits 13 to 15
22 to 34	AD-0 to 12	IN/OUT	Address and Data Lines to Video Display Memory		
21	LPEN	IN	Light Pen Detect Input		
40	V_{CC}	IN	+5V		
20	GND	IN	Ground		

GDC PIN NAMES



PIN CONFIGURATION

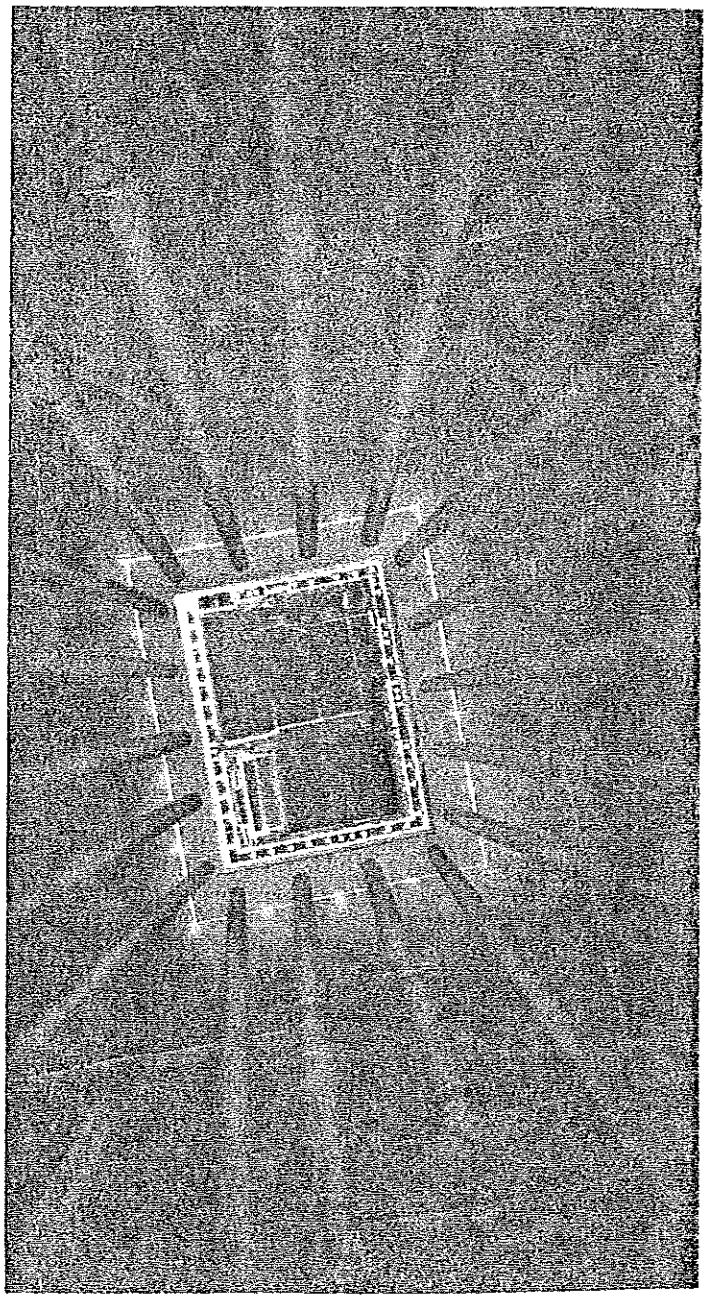


Design

As LSI gives way to VLSI, computers are getting squeezed into single chips. Fitting a 256-byte RAM in with the CPU and I/O circuits allows a one-chip μC to handle small, low-cost systems.

256-byte on-chip RAM turns μC chip into microcontroller

One of the advantages of memory-to-memory processor architecture is that memory cells can be integrated on-chip, giving "more performance bang for the buck," according to Walden C. Rhines, microcomputer department manager in Texas Instruments' Semiconductor Group (ELECTRONIC DESIGN, Oct. 11, 1980, p. 38 and Nov. 8, p. 131). The following article describes a microcomputer that packs 256 bytes of RAM on-board for the CPU—and provides three times the performance of earlier 9900-family microprocessors.



Complete microcomputers—the CPU, the I/O and even some memory—now fit comfortably on a single chip. Indeed, with VLSI imminent, denser memory cells and logic promise to add even more processing power to such "all-in-one" chips. Pointing to the future, the latest 16-bit microcomputer in the 9900 family—the TMS9995—provides on-chip 256-byte-deep RAM that eliminates the need for an external RAM chip in small-system applications needing limited data storage.

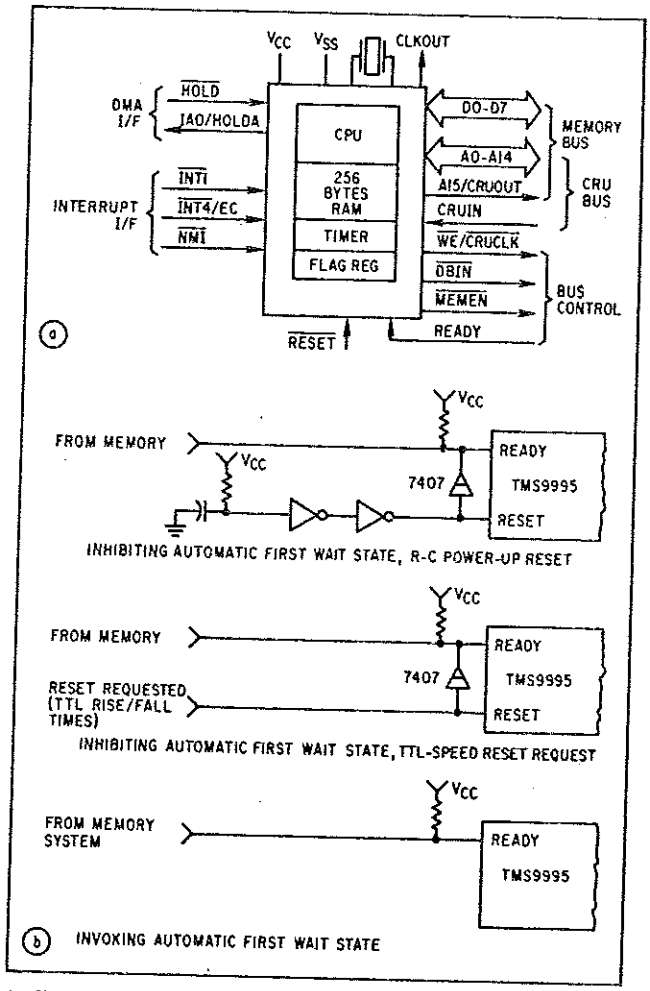
Though small compared to off-chip memories, the 256-byte on-chip RAM provides up to eight register files that operate at logic-gate rates. Other 16-bit processor chips, such as Intel's 8088, offer only one such on-chip register file. This enables the 9995, with its memory-to-memory architecture, to race with the fastest register-to-register minis (see "What's Memory-to-Memory Architecture?").

Principally intended for reducing the overall chip count and cost in small systems, the 9995 comes in a 40-pin package, operates from a single +5-V power supply, and includes the following additional on-chip features (Fig. 1):

- A user-selectable first-wait state for external memory, generated automatically without extra hardware

John V. Schabowski, Microprocessor Systems Engineer
Texas Instruments Inc.
8600 Commerce Park Dr., Houston, TX 77036

- An 8-bit data bus nonmultiplexed from the 16-bit address bus
- A synchronizer and latch for each external maskable interrupt
- A nonmaskable interrupt ($\overline{NMI}$)
- Bit-addressable I/O-capability enhancements (the communications-register unit includes a large number of addressable I/O bits and supports the generation of wait states during CRU cycles)
- A 16-bit flag register.



1. The latest addition to the 9900 family is the 16-bit TMS9995, in a 40-pin package (a). It features a 256-byte on-chip RAM, a 16-bit timer, a flag register, and separate address and data buses. Also, a user-selectable automatic first-wait state is invoked or inhibited by simple circuitry via the Ready line (b).

The 256-byte RAM is organized into 128 words of 16 bits each to match the 16-bit internal data paths. The RAM internally transfers 16-bit words to or from the processor (in only one clock cycle, as opposed to two cycles for an off-chip memory via the 8-bit data bus). Nevertheless, a write instruction may address only one byte of an internal RAM word. Then, just the selected byte location is written into the RAM; the other half of the word is left intact.

Even the 9995's off-chip memory capabilities are improved. Few of today's processor chips can take full advantage of the fast memories that are available now—not to mention the future speed bonuses VLSI technology has in store for memory. The 9995 is ready.

A memory-read transaction in the 9995 can take place in no more than 120 ns, and in 330 ns for a complete cycle. However, the 9995 also can automatically accommodate slow, low-cost external-ROM memories with the first-wait-state feature (Fig. 2) that allows accessing for reading in up to 450 ns, and a complete cycle in 667 ns.

Data and addresses on separate buses

Also helping reduce the number of chips are the 9995's separate data and address buses, which eliminate the need for demultiplexing circuitry. Moreover, the 8-bit data bus allows systems to use a memory package organized in single bytes, which enhances the μC's flexibility.

In addition, dedicated internal circuits solve the problems that often arise in handling asynchronous and pulse-type interrupt signals. Each external interrupt line (priority levels 1 and 4) features a synchronizer circuit to ensure that asynchronous interrupt signals reach the processor in proper relation to a clock signal. Thus, the possibility of a metastable (hangup) condition is minimized. (The output state of a clocked latch element can go to an undefined or metastable state for an indeterminate period of time, when the clock and input signals change simultaneously.)

Further, latches for the interrupt lines ensure that the processor sees interrupt-request signals as levels should they enter the chip as pulses. The appropriate latch is reset by the context switch for the requested interrupt. Thus, the synchronizer-plus-latch circuit on each interrupt line allows the 9995 to work reliably with either a multiple, wired-OR signal level or with a single-pulse signal.

Special emphasis is put on the design of the 9995's internal synchronizer and latch circuitry in anticipation of many interrupt-driven applications, such as in process-control systems. To this end, the interrupt signals are on dedicated pins, and therefore directly and quickly accessible without complica-

Table 1. CRU flag-register addresses

0000 ⋮ 1EDE	General use CRU address space	External (off-chip) CRU address space
1EE0	FLAG0	Flag register
1EE2	FLAG1	
1EE4	FLAG2	
1EE6	FLAG3	
1EE8	FLAG4	
1EEA	FLAG5	
1EEC	FLAG6	
1EEE	FLAG7	
1EF0	FLAG8	
1EF2	FLAG9	Internal (on-chip) CRU address space
1EF4	FLAGA	
1EF6	FLAGB	
1EF8	FLAGC	
1EFA	FLAGD	
1EFC	FLAGE	
1EFE	FLAGF	
1F00 ⋮ 1FD8	General use CRU address space	External (off-chip) CRU address space
1FDA	Mid flag	Internal (on-chip) CRU address space
1FDC ⋮ FFFE	General use CRU address space	External (off-chip) CRU address space

Table 2. Flag-register functions

Bit	CRU bit address	Description
FLAG0	1EE0	SET TO 0: DECREMENTER CONFIGURED AS INTERVAL TIMER SET TO 1: DECREMENTER CONFIGURED AS EVENT COUNTER
FLAG1	1EE2	SET TO 0: DECREMENTER DISABLED SET TO 1: DECREMENTER ENABLED
FLAG2	1EE4	LEVEL 1 INTERNAL INTERRUPT REQ. LATCH
FLAG3	1EE6	LEVEL 3 INTERNAL INTERRUPT REQ. LATCH
FLAG4	1EE8	LEVEL 4 INTERNAL INTERRUPT REQ. LATCH
FLAG5	1EEA	User-defined
FLAG6	1EEC	
FLAG7	1EEE	
FLAG8	1EF0	
FLAG9	1EF2	
FLAGA	1EF4	
FLAGB	1EF6	
FLAGC	1EF8	
FLAGD	1EFA	
FLAGE	1EFC	
FLAGF	1EFE	

What's memory-to-memory architecture?

Advances in μC technology have spawned a compatible architecture—the memory-to-memory structure. With it, the CPU needs just three programmable registers: a program counter, a workspace pointer, and a status register. Additional registers, stacks, and workspaces reside in the μC's more plentiful memory instead of using the relatively scarce logic gates available in the CPU. In addition, the memory-to-memory approach offers an enhanced operating speed, and also easy context switching, which facilitates processing multiple interrupts and simplifies multiprogrammed or multiprocessor operation.

In the 9995, the program counter (PC) holds a 15-bit address of the instruction to follow the one being executed. With this address, the processor fetches the next instruction from memory; while the new instruction is carried out, the PC is incremented. An instruction may alter the PC's content to make the program branch into a new direction. Thus, the PC figures in all context-switching, branching, and jumping operations.

The program status and results of programmed comparisons are stored in the status register (ST), which also supplies arithmetic-overflow and interrupt-mask-level information to the interrupt-priority logic. Each of the register's 16 bits is dedicated to a particular function or condition within the processor. Some instructions check for a prerequisite condition via the ST; other instructions can affect the ST contents either a bit at a time or all at one time. In addition to instructions, interrupts can modify the ST contents.

While instructions are being executed, the workspace pointer (WP) keeps track of workspaces in the RAM memory. Workspaces consist of blocks of 16 contiguous memory words, so special hardware registers for manipulating instruction operands are not needed. Individual memory locations can function as registers for operands, addresses, or index data, or even serve as accumulators. In different modes, the most or least-significant byte of any workspace word can be addressed directly and accessed or changed.

Servicing interrupts and subroutines in context switching in the older register-to-register arrangements requires storing some of the register-file contents in memory and then reloading the data into hardware registers. Besides the housekeeping overhead that is needed, this process must use one memory cycle to store and another to fetch each of many words. However, microcomputers based on the more advanced memory-to-memory architecture accomplish the same context switching faster and more easily by just interchanging the WP and PC-register contents. Context switching then takes just two fetch and three store cycles, during which the contents of PC and WP are exchanged and the contents of ST are saved; the data words stay in place.

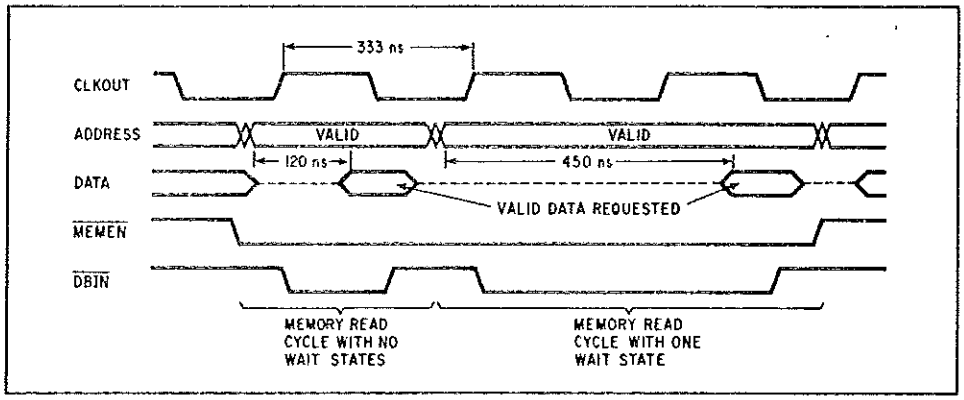
tions. For instance, the nonmaskable interrupt on a separate line is exempt not only from an interrupt-disabling mode after execution (which is characteristic of certain instructions) but also from the current value of the interrupt mask. Thus, the nonmaskable interrupt allows a quick check at any time of maintenance-panel functions such as single-stepping and breakpoints, or the introduction of user-defined functions.

Cutting the requirements for additional system chips even further is an on-chip communications register unit, which provides high bit-serial I/O addressing power. Instruction-driven, the CRU of the 9995 can address as many as 32,768 bits; the earlier 9900 chips address just 4096 bits. Three additional address lines mean that more peripheral devices can work directly with the CRU interface—all that's needed is to dedicate one address line to enable each peripheral unit.

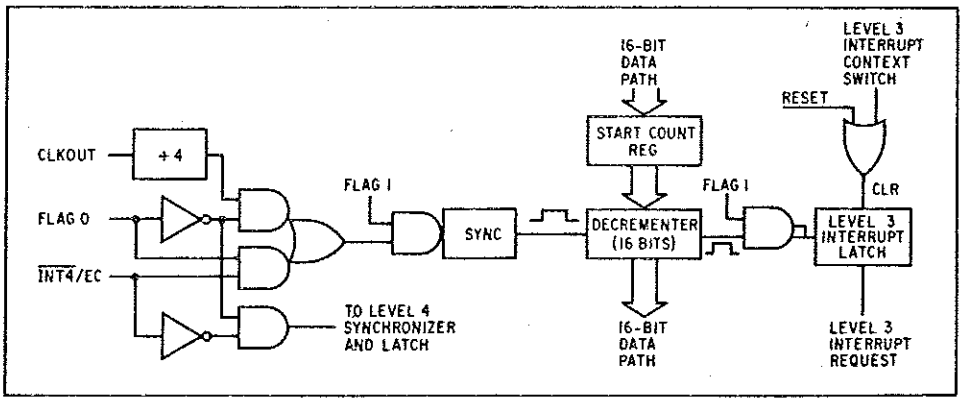
Internally extending CRU cycles with a wait feature instead of with external circuitry is another chip saver. An elongated CRU cycle can compensate for the propagation delays of terminals located far from the processor.

In addition, the CRU saves external circuitry with its 16-bit on-chip flag register (Table 1). Three of the flag bits (flags 2, 3, and 4) allow the CRU to check for interrupt stimuli that are being blocked by the status-register interrupt mask: The flags allow the software to reprioritize an interrupt set. Eleven of the bits (flags 5 through F) are general-purpose user-defined by the software, and can be used instead of bytes or even whole words from the memory (Table 2). The two remaining flag bits control the on-chip 16-bit decrementer, which is memory-mapped (Fig. 3). Flag 1 enables the decrementer, and flag 0 selects one of two operating modes: interval timing or event counting. The timer mode's resolution is 1.33 μs; the event-counter mode's maximum pulse-repetition rate is 1 MHz. The pulses need not be synchronous with the 9995's clock.

The 9995 can be clocked from internal or external



2. Versatile memory-access timing allows the 9995 to work efficiently with the new fast memories as well as with the lower-cost slow ones.



3. The on-chip 16-bit decrementer is configured either as a timer or as an event counter with flag bit number 0. In either mode, the decrementer is gated by flag bit number 1.

sources. The chip's internal clock oscillator requires an external crystal; all other clock circuitry is on the chip. The system's machine-state frequency is one-half the crystal's fundamental frequency.

Speed ranks high

Along with low cost and superior functional performance, processing speed ranks high in importance for programmable-controller use. The 9995 is from 1.5 to 3.8 times faster than previous 9900-family processors (Table 3).

One reason for this speed is the 9995's on-chip RAM, which allows a full 16-bit word access in one clock cycle, rather than the minimum two needed with off-chip memory via the 8-bit data bus. Also, all the addressing modes of all the 9995 instructions can access the on-chip RAM, which then can serve even as high-speed program space.

For example, when the op code is in an off-chip ROM and 16-bit registers are in on-chip RAM, a register-to-register Add instruction takes just 1.67 μs. But if both the op code and the registers are off the chip, the same Add instruction takes 0.67 μs longer—plus any time required for wait states.

Even more dramatic is the speeding up of Unsigned Multiply and Divide operations. The 9900 multiplies two 16-bit numbers in 17.33 μs and divides a 32-bit number by a 16-bit number in 41.33 μs; the 9995 performs the same in 7.67 and 9.33 μs, respectively—better than twice the previous speed for Unsigned Multiply and more than four times for Divide. Moreover, not only are the speeds of Add, Unsigned Multiply, and Divide improved, but execution speed is enhanced throughout the entire 9995 instruction set.

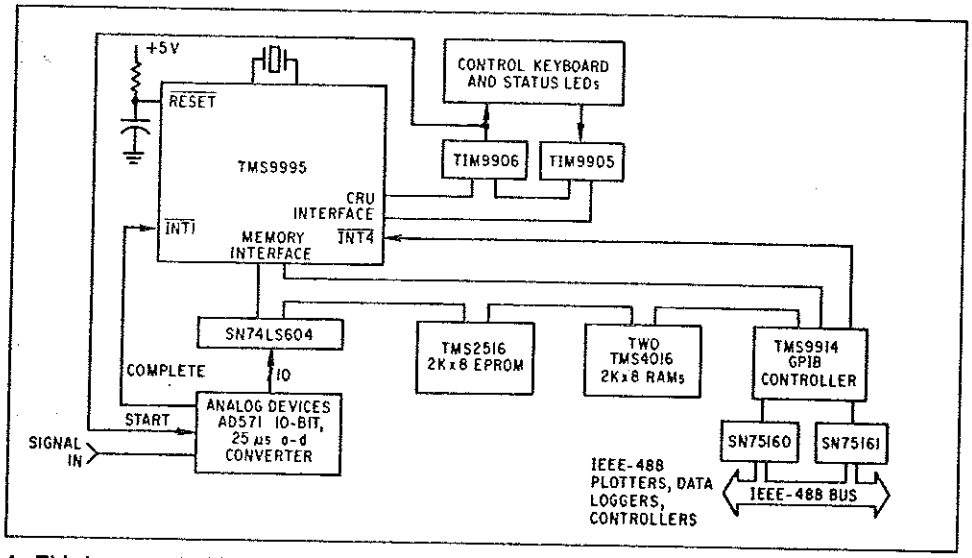
Also, a brand-new system-interrupt function—Arithmetic Overflow—helps speed math operations. This function automatically takes care of an overflow condition without software. Processors without the automatic overflow interrupt must ensure that operands are within the machine's range with time-consuming software.

In addition, four new instructions augment the 69 instructions of the TMS9900 to further speed performance:

- A Signed Multiply that can process two 16-bit operands in 8.33 μs
- A Signed Divide that can divide a 32-bit dividend

Table 3. Benchmark speed comparisons

	Move 275 bytes from buffer A to B	Convert 512 bytes from ASCII to EBCDIC	Add 25, 16-bit No. (32-bit accuracy)	Add 25, 32-bit No. (64-bit accuracy)
TMS9995	1653 μs (1×)	4270 μs (1×)	185 μs (1×)	303 μs (1×)
TMS9981	5460 μs (3.3×)	11,050 μs (2.6×)	709 μs (3.8×)	1034 μs (3.4×)
TMS9900				
3 MHZ	4200 μs (2.5×)	8500 μs (2×)	546 μs (3×)	795 μs (2.6×)
4 MHZ	3170 μs (1.9×)	6400 μs (1.5×)	409 μs (2.2×)	596 μs (2×)



4. This low-cost 20-kHz spectrum analyzer is made possible by the TMS9995's overall high throughput and high-speed signed-multiplication capabilities. While carrying out an FFT, the processor scans the keyboard and manages communication via the IEEE-488 bus. Applications include sonar, baseband radar, seismic processing, and acoustic analysis. A faster a-d converter could extend the instrument's frequency range considerably beyond the audio range.

planes is comfortably within the GDC's speed and addressing capability (Fig. 1).

In the controller's character mode, a word of display memory stores a character code and attribute bits. Up to 100 rows of characters can be displayed, each with a maximum of 32 lines. There can be up to 256 characters per row.

With a 7-by-10-dot area suitable for a 5-by-7-dot character and noninterlaced video, 40 rows of 80 characters can be displayed at 60 Hz without interlacing and over 20 such screens can be stored in display memory. With an 80-by-24-character format, up to 34 screens can be buffered.

Multiple GDCs may be synchronized in one system to expand display memory depth while maintaining its height and width. Two GDCs, for example, allow either alphanumerics superimposed over graphics or else more bit planes per pixel. For the fastest drawing speed, each plane can have its own GDC. True color is possible using several GDCs for display memories of up to 2,048 by 2,048 pixels.

Other innovations

Besides high resolution, the GDC introduces two outstanding features: the capability to process display-memory data in a single read-modify-write cycle and to draw graphics figures.

During a read-modify-write cycle, the controller reads the desired word from display memory, modifies the bit or bits, and writes the modified word back into memory. Its mask and pattern registers make possible multibit and multifield modifications and automatically dotted lines. The GDC's modification operations include set, clear, invert, and replace with a pattern.

Its graphics figure-drawing speed—for lines, arcs, circles, and rectangles at less than 800 ns per pixel—means that 20,000 pixels may be updated in one refresh period. Once the parameters are loaded into the controller and the drawing is initiated, no further attention is needed from the local processor. Parameters for the next figure can be prepared while the graphics controller calculates the addresses and draws the present figure using back-to-back read-modify-write cycles.

Local intelligence

An intelligent graphics display terminal can be built by adding a general-purpose microprocessor and possibly local mass storage. Graphics input devices, such as light pens, joysticks, trackballs, and tablets, should be handled locally by the microprocessor to achieve the fastest response times.

The terminal's microprocessor can maintain a vector-list representation of the objects to be displayed in order to effect rotation, scaling translation, and clipping of the objects. Communication with the host computer is required only to transfer high-level information such as vector list changes and viewpoint.

The graphics display controller significantly reduces the terminal processor's overhead by handling many of the most time-consuming tasks. It receives its commands through a first-in, first-out buffer to maximize the system's efficiency. Direct access to the display memory is

Address line 0	Read mode		Write mode
0	Read status register	Bit definitions	Write parameter into first-in, first-out buffer
	0	Data ready	
	1	FIFO full	
	2	FIFO empty	
	3	Drawing in progress	
	4	Execute direct memory access	
	5	Vertical synchronization	
	6	Horizontal synchronization	
	7	Light pen detected	
1	Read FIFO		Write command into FIFO

under the control of the GDC and proceeds at a rate of up to 1.25 million bytes per second. Figure 2 is a simplified block diagram of the GDC.

As shown in Table 1, the low-order address line, A_0, and the read-write line permit parameters and commands to be entered into the controller and status information, cursor position, and display-memory data to be extracted from it. Note the additional buffer isolating the CPU interface and the FIFO from the rest of the GDC to allow independent transfers between the FIFO and the terminal processor.

In the FIFO's read mode, data goes from the GDC to the CPU, so that it can read video memory, the position of the cursor, and the light-pen status. In the write mode, the CPU can send commands—each with a number of parameter bytes—asynchronously to the GDC until the FIFO is full.

Two bits in the status register reflect the FIFO's status as full or empty. It can be written into by the processor and responses can be taken from it, but not concurrently. The status register in the CPU interface may, however, be read independently of the buffer.

All 18 of the controller commands are listed in Table 2. These commands have been broken down into five basic categories: video control, display control, drawing control, data read, and direct-memory-access (DMA) control.

The two-phase clock in Fig. 2 has twice the display memory's word-access rate. This frequency is the basis of all timing in the GDC and can range from 0.5 to over 5 MHz. Any additional timing signals, like those needed for dynamic memories, can be easily generated from the clock with a shift register.

The GDC operates in one of three modes. The first, a full graphics mode, uses all 256-K words of memory and 16 data bits via the time-division–multiplexed display-memory interface pins. Another mode combines graphics and coded characters on the screen simultaneously; 16 of the lines are used for address and the other 2 control an external line counter to switch external circuitry between

TABLE 2: A SUMMARY OF DISPLAY CONTROLLER COMMANDS

Mnemonic	Operation(s)
Video control commands	
RESET	resets the graphics display controller to its idle state and specifies the video display format
VSYNC	selects master or slave video synchronization mode
CCHAR	specifies the cursor and character row heights
Display control commands	
START	starts the display scanning process
ZOOM	specifies zoom factors for the display and graphics characters writing
CURS	sets the position of the cursor in display memory
PRAM	defines starting addresses and lengths of the display areas and specifies the 8 bytes for the graphics characters
PITCH	specifies the width of the X dimension of display memory
Drawing control commands	
WDAT	writes data words or bytes into display memory
MASK	sets the mask register contents
FIGS	specifies the parameters for the drawing processor
FIGD	draws the figure as specified above
GCHRD	draws the graphics character into display memory
Read data commands	
RDAT	reads data words or bytes from display memory
CURD	reads the cursor position
LPRD	reads the light pen address
Direct-memory-access control commands	
DMAR	requests a DMA read transfer
DMAW	requests a DMA write transfer

graphics and character display. The third mode displays coded characters only and eliminates the need for an external line counter. Here 13 address and data lines control a display memory of up to 8-K words and the 13 bits per word are used for the character code and its attributes.

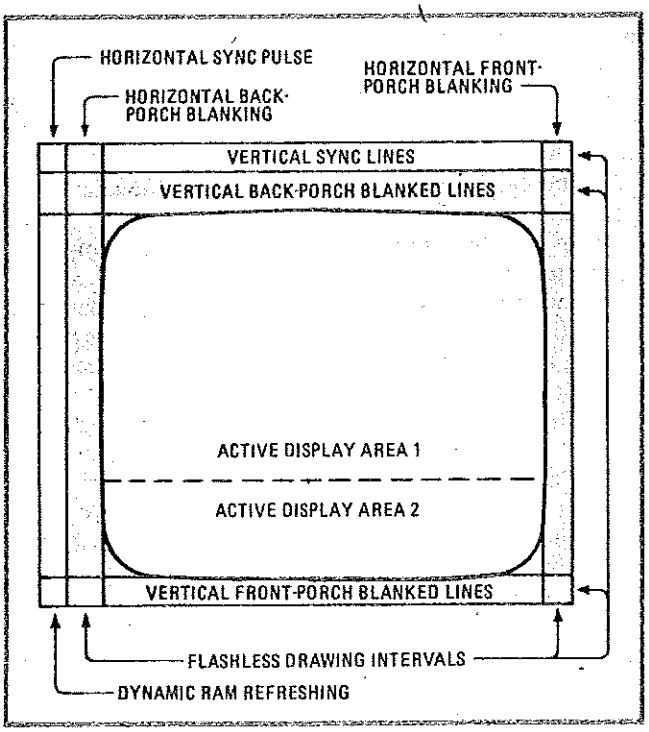
Different uses for lines

The GDC's interface with its memory uses TDM address and data lines. In addition, depending on the display mode, these lines are also used in other ways. In the graphics mode, all 18 pins supply address values, whereas the two character modes use either 13 or 16 lines respectively, for addressing. In any case, the address of the display memory is available early in the cycle for latching so that the rest of the memory cycle can use the lines for data. In the graphics mode, 16 of the lines are used for the data bus; in the character modes, 5 lines are used to indicate cursor position, line count, blink timing, and mode-switching information.

The parameter RAM holds 16 predefined variables. These variables are loaded through the CPU interface and referred to by other elements of the GDC as needed.

The drawing processor works with the parameter RAM and the display memory controller to compute the position of each pixel in graphics figures. Drawing can proceed uninterrupted while the next drawing command and parameters are loaded into the FIFO buffer.

The GDC has an unusually flexible video-raster format. It supplies separate horizontal and vertical synchronization signals, and the number of display blanking lines and words of active display per line can be any even number from 2 to 256.



3. Partitioning. The screen can be partitioned into four horizontal display fields, and each can be scrolled up and down independently. Drawing can be done at any time, but the display will not be disturbed if it is done during retracing periods.

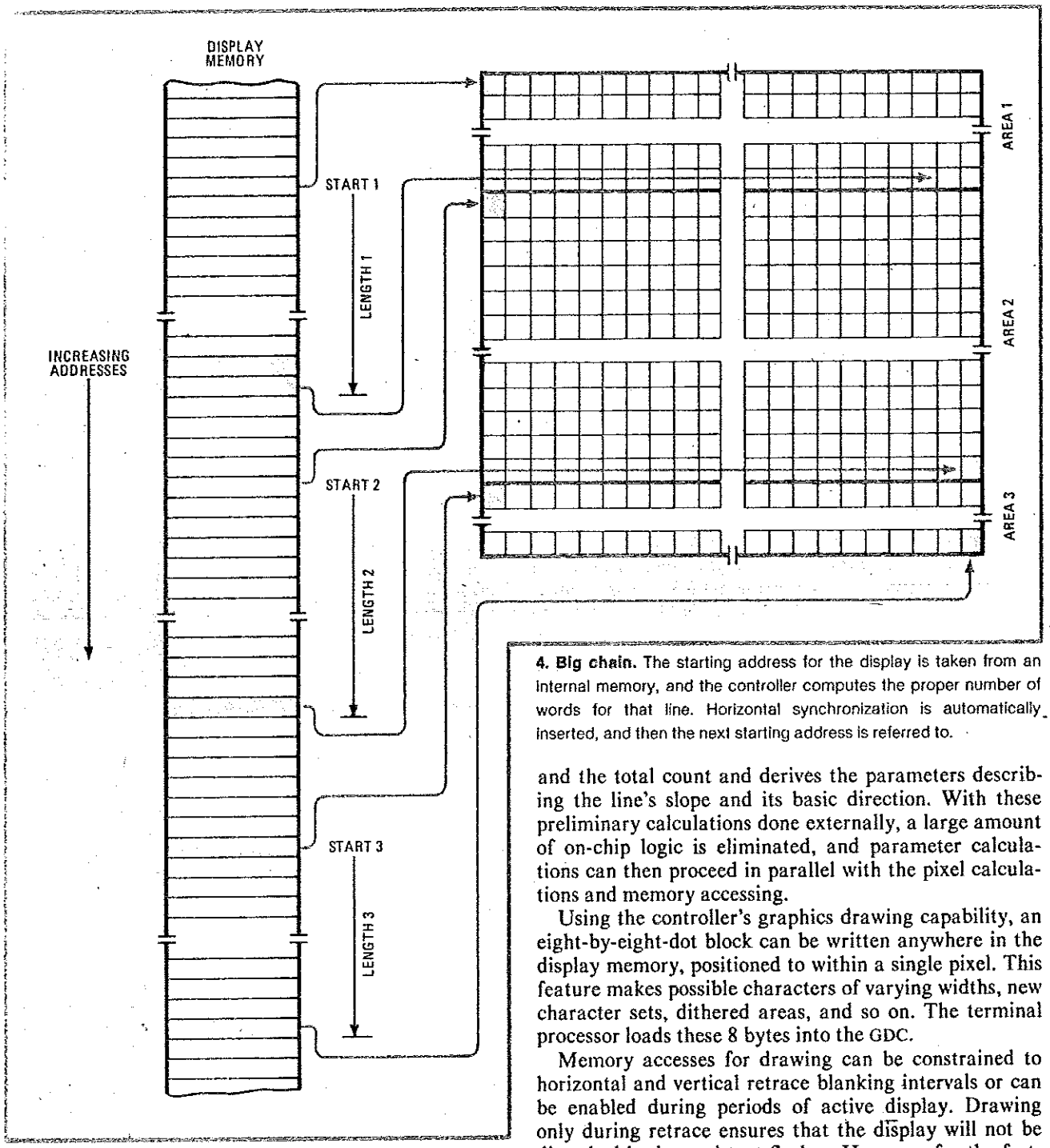
With a two-clock-cycle word, the horizontal front- and back-porch widths can be adjusted to up to 64 word periods, and the width of the horizontal synchronization pulse can range up to 32 words (Fig. 3). These same ranges apply to the vertical synchronization periods, but in terms of lines. In addition to noninterlaced video, both interlaced and repeat-field–interlaced video formats are available.

Display size may be set equal to or less than display memory size. Logic circuitry automatically takes the starting address from the parameter RAM in the controller and scans the number of words required for the line. The GDC then automatically inserts a horizontal sync pulse before going to the next line.

The count through the display memory continues from where it left off until the end of the field, when the starting address is again referred to. If the display area is partitioned, multiple starting addresses are stored in the GDC with their line counts. They are used no matter where they are in display memory (Fig. 4). The starting addresses can be individually modified to achieve the independent scrolling mentioned earlier.

Figure drawing

The drawing processor can calculate the word and dot addresses of the pixels in a figure in parallel with—and at the same speed as—the display-memory writing process. Patterns for various dotted, dashed, and solid figures are loaded into a pattern register by the terminal



4. Big chain. The starting address for the display is taken from an internal memory, and the controller computes the proper number of words for that line. Horizontal synchronization is automatically inserted, and then the next starting address is referred to.

processor, and the decision to modify a pixel of a figure is based upon the contents of this register.

A 16-bit mask register in the drawing processor selects the bits of the word for modification. Although this register is used automatically during drawing of the figures, it is also possible to load it with all logic 1s to write all 16 pixels of a word in one memory cycle in order to fill any area rapidly. In the character mode, the mask register can select the bits of the character or the attribute codes that are to be modified.

Parameters can be loaded into the GDC to draw lines, arcs, circles, or rectangles. With line drawing, for example, the microcomputer supplies the initial pixel address and the total count and derives the parameters describing the line's slope and its basic direction. With these preliminary calculations done externally, a large amount of on-chip logic is eliminated, and parameter calculations can then proceed in parallel with the pixel calculations and memory accessing.

Using the controller's graphics drawing capability, an eight-by-eight-dot block can be written anywhere in the display memory, positioned to within a single pixel. This feature makes possible characters of varying widths, new character sets, dithered areas, and so on. The terminal processor loads these 8 bytes into the GDC.

Memory accesses for drawing can be constrained to horizontal and vertical retrace blanking intervals or can be enabled during periods of active display. Drawing only during retrace ensures that the display will not be disturbed by intermittent flashes. However, for the fastest drawing speeds, the read-modify-write drawing cycles can be enabled at any point on the screen. The blanking signal from the GDC is asserted during such cycles to minimize display disturbances.

Direct access to display memory

The 7220's DMA-read and -write functions are useful in two basic situations. In the first, the GDC generates a graphics picture or receives video data—from a camera, a scanner, or other video source—that needs to be moved into its video display memory. In the second, it writes graphic figures into display memory that need to be sent out to a printer, microfilm, disk, or tape. □

With the MPIF, a universal 40-pin, two-port interface, different processors can communicate with one another as if each was a peripheral. On-board logic handles even asynchronous processors.

Chip blends microprocessors into a harmonious system

This is the second article in a series on peripheral chips in Texas Instruments' 99000 microprocessor family. The first article, which described a universal peripheral controller, appeared in the Oct. 29 issue, p. 113.

Multiprocessor intercommunication problems may be on the way out. A universal processor-to-processor interface IC developed by Texas Instruments radically simplifies the integration of multiprocessor systems. The chip is the 40-pin, two-port TMS99650, a member of the TMS99000 microprocessor family.

The multiprocessor interface (MPIF) contains control and status registers, message registers, and 256 bytes of dual-ported buffer RAM (Fig. 1). With MPIFs, processors can communicate with one another as if each were a peripheral device. Moreover, since the MPIF provides a slave-type interface that looks like a memory port, it mates easily with almost any processor, no matter which manufacturer made it.

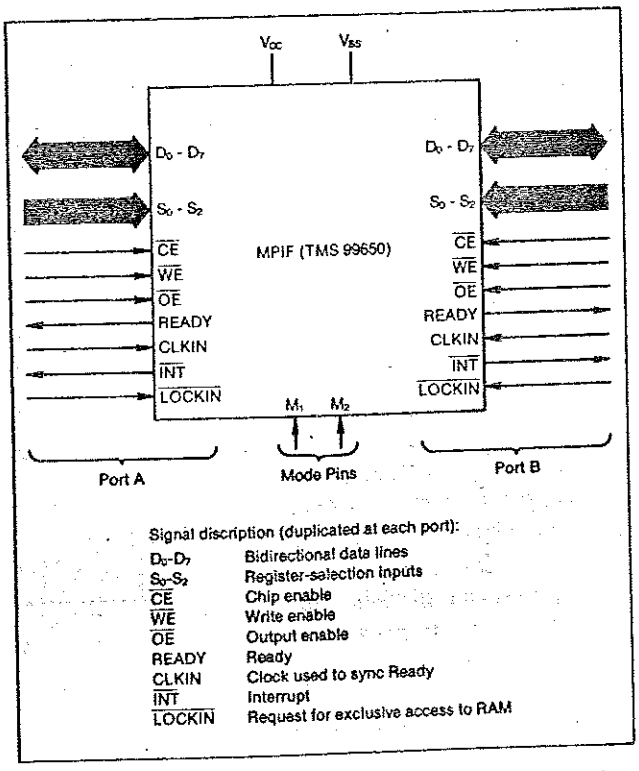
Synchronization by on-chip logic

Even asynchronous processors can be handled with the MPIF's on-board synchronization and arbitration logic. And not to be overlooked are the MPIF's n-channel silicon-gate technology, its single 5-V supply, and its TTL-compatible inputs and outputs. Thus, getting widely different microprocessor (or microcomputer) types to talk to each other formerly a most demanding discrete logic task—is no longer a problem.

The two ports present an identical interface to each of the mating processors. Data enter or leave the ports' eight registers via identical, independent 8-bit data lines. Also, each port is independently serviced by three register-address selection lines, an interrupt line, and six other dedicated-control lines (Fig. 2).

Four of these registers can access the MPIF's 256 × 8-bit static RAM. And all the registers can be mapped into eight locations of the I/O or memory-address space of virtually all 8- and 16-bit microprocessors.

Completing the MPIF's similarity to an external RAM, the external logic for the two ports consists mainly of decoding circuitry for generating the chip-enable ($\overline{CE}$) and write-enable ($\overline{WE}$) signals to the MPIF. This is similar to the requirements for a small



1. Each port of the MPIF has a set of identically designated input/output lines, except for terminals M_1 and M_2. These terminals establish the chip's mode: reset, stand-alone, master, or slave.

Jerry VanAken, Computer Systems Engineer
Texas Instruments, Inc.
8600 Commerce Park, Houston, TX 77036

block of memory. However, the MPIF produces its own synchronized ready and wait (READY) signal from an external clock input (CLKIN) signal. Data I/O buffers are fast enough to connect directly to the processor bus in most small multiprocessor systems.

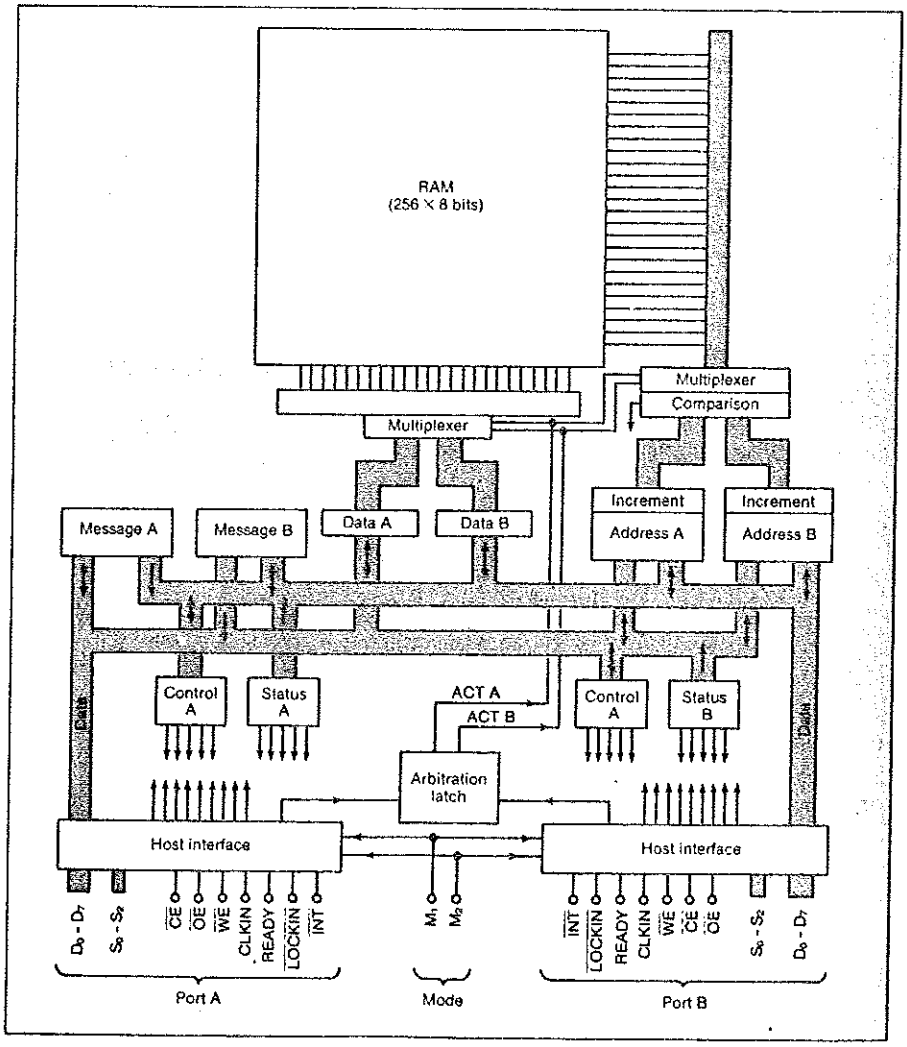
Besides the power terminals, V_{cc} V_{ss}, another pair, M_1 and M_2, are shared by both ports. They program four different modes: reset ($M_1 = M_2 = 0$), stand-alone ($M_1 = M_2 = 1$), master ($M_1 = 0$, $M_2 = 1$), or slave ($M_1 = 1$, $M_2 = 0$). Schmitt-trigger inputs permit the use of resistor-capacitor timing circuits to implement the reset. When reset, the READY output of each port is held in a high-impedance state.

Useful even in simplest case

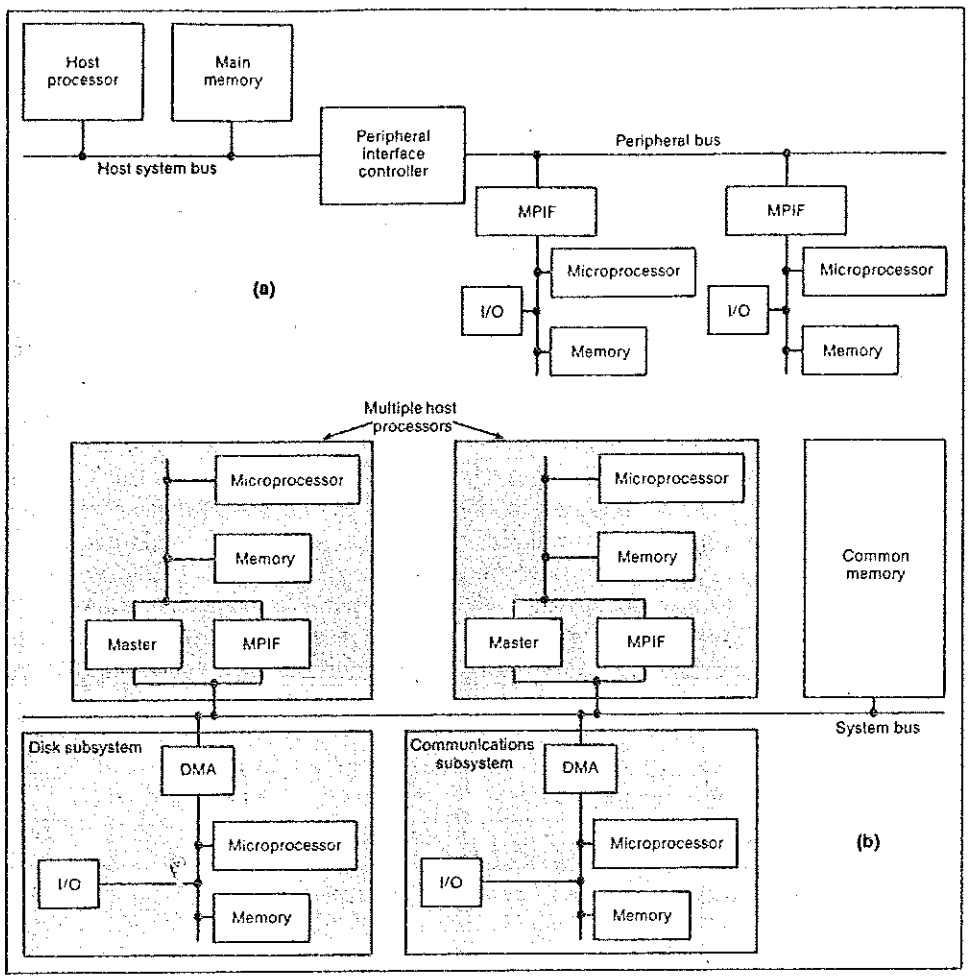
The simplest interface between a microprocessor and a peripheral is the single-word buffer; nevertheless, hardware is needed to generate interrupts and to program the peripheral. Also, the peripheral's internal clock (if any) must be synchronized with that of the host system. Of course, an MPIF can serve in such an application, even though most of its 256-byte buffering capacity remains unused. The great convenience of the MPIF's data buffer, programmable registers, interrupt logic, and competitive cost synchronization—with all the circuitry on a single chip—justifies its use.

Naturally, with multiword interfaces, where the MPIF's 256-byte RAM is more fully employed to buffer blocks of I/O data, the chip's effectiveness is even more apparent.

For handling large amounts of data, however, the peripheral interface in Fig. 3a requires a DMA capability to move the blocks of data directly to and from main memory with a minimum of processor overhead. Figure 3b shows a slave disk controller and communications subsystem with DMA capability. With DMA, data can be transferred directly to a host processor's MPIF rather than to a buffer in common memory. This operation decreases traffic on the bus. The same principle can be applied to the transfer of messages between hosts: one processor writes to



2. The multiprocessor interface, or MPIF, contains two identical ports with equal access to an on-chip 256-word × 8-bit RAM.



3. Peripheral subsystem processors of different manufacturers can join a common host system bus via MPIF chips (a) and look like peripheral devices to the system. When large blocks of data must be transferred from a peripheral to a host processor, the data may be written directly to the remote port of the host's MPIF (b) rather than to a buffer in common memory. In a similar fashion, the MPIF can buffer messages from the other host, because of the MPIF's two identical, independent ports.

the other's MPIF.

With or without DMA, data can enter the MPIF's on-chip RAM from either port via the two independent bidirectional buffers (nonstoring registers DATA A and DATA B) and two corresponding address-pointer registers (ADDR A and ADDR B). The mating processors can read their respective address-pointer registers or write into them. In addition, there are two message registers, one for each port. Each port can write into its respective message register, but can read only from the other port's register.

Of the remaining two registers, one (the control register) directs on-chip operation, such as keeping track of the enable bits that control the various sources of on-chip interrupt requests, and the other (the status register) indicates the state of the different interrupt sources.

The RAM can be accessed by only one host at a time, as determined by the outputs of arbitration latches ACTA and ACTB, which select DATA A and ADDR A or DATA B and ADDR B registers. ACTA latches when the data register of port A is addressed, unless ACTB is already latched. In other words, ACTA and ACTB are mutually exclusive: operation is on a first-come, first-served basis, except when a lockout mode is in effect.

On those occasions when both input ports address their data registers simultaneously, the arbitration latches assume indeterminate states temporarily. But a cross-coupling feedback action quickly resolves the conflict, essentially randomly. During this metastable state, threshold circuits ensure that the ports remain inactive, accepting no data inputs.

In addition to the first-come, first-served mode, either interface can request a lockout of the other with a flag bit for exclusive use of RAM. And if both ports try to assert a lockout simultaneously, the condition is handled on a first-come, first-served basis.

Following the concept of generality, the MPIF

registers that are addressed at the same "location" in each port serve the same function; thus, the ports can be interchanged. But for identification in the host's memory map, one port is called "local" and the other "remote." Table 1 lists the eight 8-bit registers at each port, their functions, and their selecting lines. S_0 through S_2.

Ports can read or write

Each port can read or write data for the MPIF RAM in two modes: Data and Data/Inc. In the Data/Inc mode, the local address-pointer register of that port is incremented at the completion of each memory cycle. No incrementing occurs when the memory operation takes place in the Data mode.

Also, data can be entered or read from either the local address-pointer register of a particular port (code 101) or the remote address-pointer register of the other port (code 111). By an initial setting of its own local pointer register, a port can determine the address in RAM where data entry starts. Or the management of the RAM can be under the control of the remote port, which sets both pointer registers. In either case, the address-pointer registers cycle repetitively from address 00_H through FF_H and back again.

Because of the extreme flexibility of the address-pointer register, certain operations that are possible in combination should be avoided. For example, an address-pointer register can be read while its value is being changed by a write operation from the other port. This may result in the reading of an erroneous value. Similarly, a port must not write to its remote address-pointer location if there is any possibility that the other port will perform a memory operation at the same time. This can produce not only an erroneous value in the pointer register, but also erroneous data in the RAM.

In contrast with the address pointers, the message functions have somewhat reduced flexibility. While a port provides both read and write modes in its message-out function (code 011), the port can only read the register of the opposite port during a message-in function (code 010). This read-only capability thus removes the possibility of overwriting and losing a remote message.

In addition, during a write operation to a Message-out register, the previous value of the register is stored in a latch. Should a read operation occur concurrently with a write at the remote port, the previous value of the register is read, thereby ensuring that hosts can poll their Message-in registers at any time without the possibility of reading an invalid byte.

However, hosts can read or write into their respective control registers at any time without conflict. When set, the bit assignments of the control registers in Table 1 are defined as follows:

- *LEA (Lockout on Equal Address Pointers).* When the values in both the local and remote address-pointer registers become equal, the port whose address-pointer register incremented last is locked out of access to the RAM. The incrementing can be the result of loading data from either port. LEA remains active until reprogrammed by its host or reset.
- *EN_1 through EN_5 (Interrupt Enable).* Any one activates the INT line, which goes active low.
- *SLOC (Software Lockout).* Software-programmable from the host, it directly locks out the remote port from the RAM. All the bits of the control register can be cleared by the reset function of mode terminals M_1 and M_2 (as previously explained).

Unlike the control registers, the status registers are read-only devices from the port terminals. They allow the host to inspect the status of on-chip parameters. When set, the status-bit assignments listed in Table 1 indicate the following conditions:

INT (Interrupt). When the $\overline{\text{INT}}$ bit is a ONE, the $\overline{\text{INT}}$ output line is low, and an interrupt is being requested.

MI (Message-in Interrupt). A remote port has loaded its Message-out register, with a byte and should be read. The MI bit is reset when the local port reads its message-in register. MI also is reset by the reset function.

MO (Message-out Interrupt). The local Message-out register is available. The MO bit is reset when a byte is written into the Message-out registers, and sets when the remote port reads its Message-in register. MO is set by the reset function.

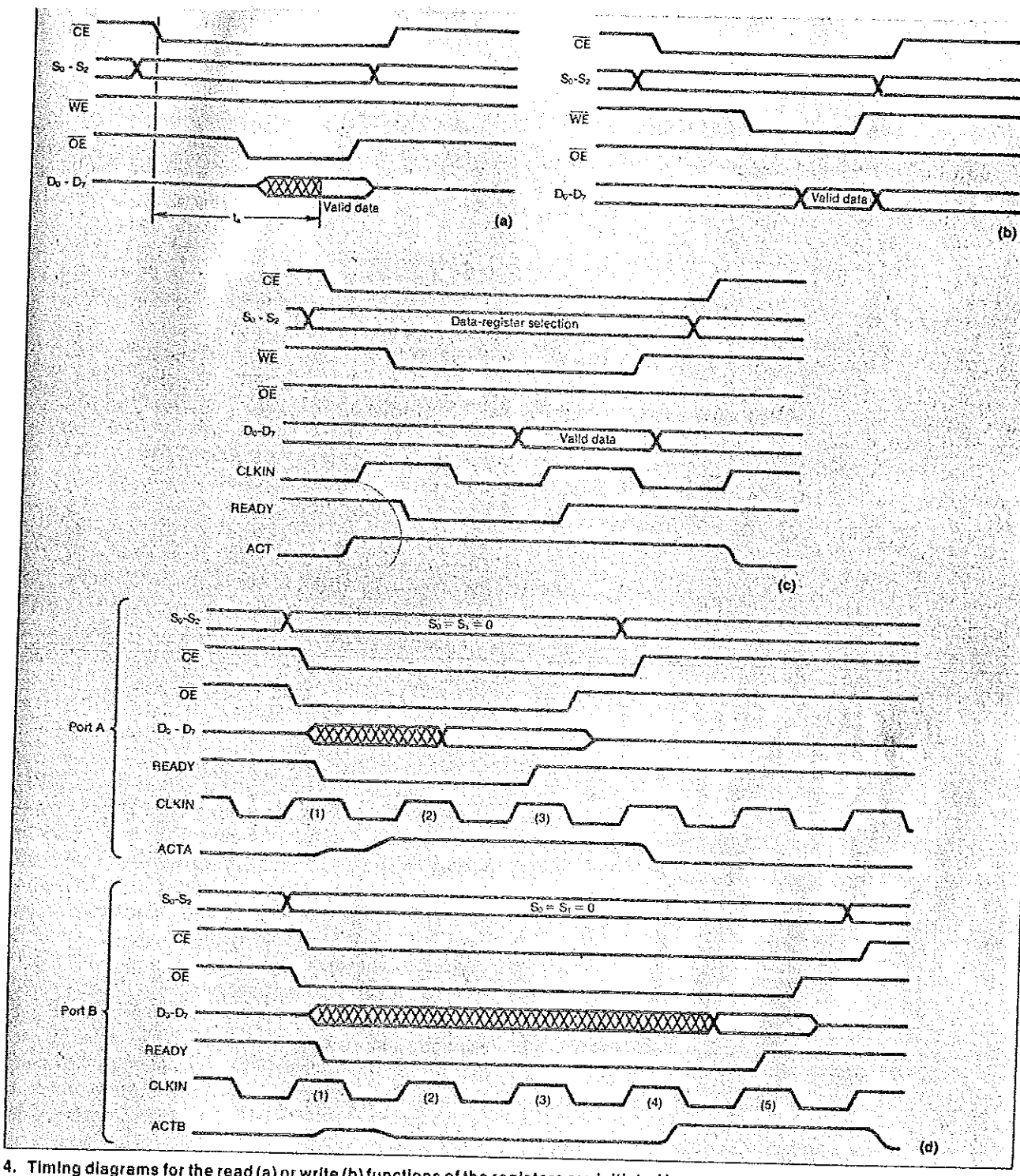
LPE (Local Pointer Equal). The value in the local address pointer equals that in the remote address pointer, and the local pointer was the last to change after being incremented or loaded from either port. LPE remains set as long as the quality condition remains.

RPE (Remote Pointer Equal). The value in the remote address pointer equals that in the local address pointer (the opposite of LPE), and the remote pointer was the last to change after being incremented or loaded from either port. RPE remains set as along as the equality condition remains.

LAK (Lockout Acknowledge). A software-lockout condition is acknowledged to the local port when the remote port is locked out from the RAM. LAK is cleared when the software lockout clears.

Reading and writing are basic

Thus reading from or writing to the eight registers are involved in all MPIF operations. Figures 4a and 4b show the timing diagrams for access to the



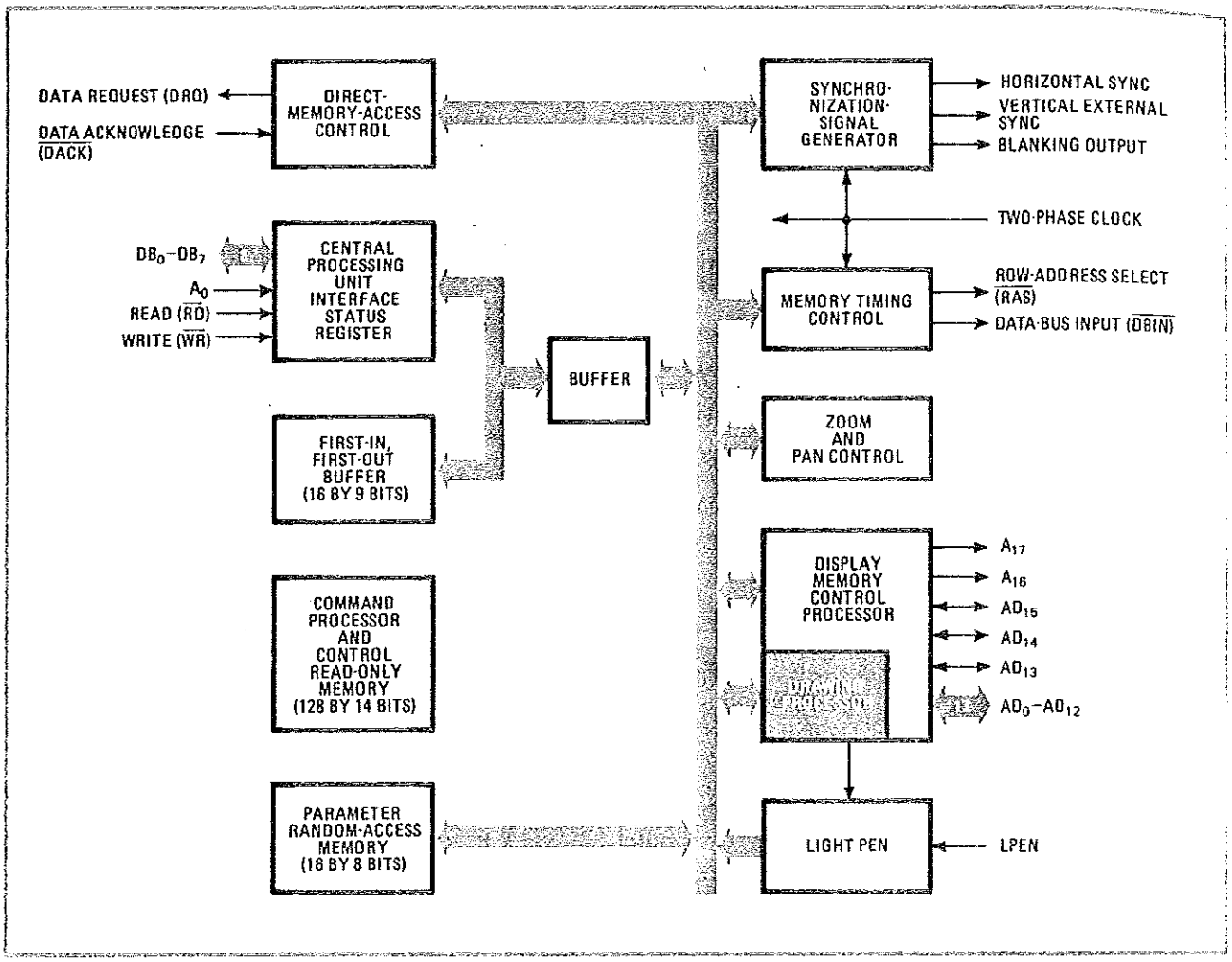
4. Timing diagrams for the read (a) or write (b) functions of the registers are initiated by OE or WE active-low signals, respectively. During access of the on-chip RAM, the READY signal is synchronized to the CLKIN (c). When both ports contend for an MPIF's RAM simultaneously, signals ACTA and ACTB decide at random which port gets access; otherwise, it's first come, first served (d).

registers.

Setting one of the eight codes (Table 1) on the register-select lines (S_0 through S_2) and pulling the chip-enable line ($\overline{CE}$) low selects a register. To write into the register, the host provides a write-enable signal ($\overline{WE}$) and stable data on lines D_0 through D_7. The data must stabilize on the input lines before the trailing edge of $\overline{WE}$ occurs. To read from the register, the output-enable ($\overline{OE}$) signal must be pulled low. Output data become valid after an access time (t_a) of 185 ns at most has elapsed. When the data lines are not being read, they assume a high-impedance state. This enables them to be paralleled with other lines on a bus without interference.

The MPIF's READY and CLKIN signals (Fig. 4c) are both necessary when concurrent data access to the interface chip is required by both ports. At this time the interface selection is under control of the arbitration latch, but the port that does not gain RAM access must put itself into a wait state with its own interface logic.

When a port addresses a data register with $\overline{CE}$ low



2. System on a chip. Inside the graphics display controller, elements are connected by means of an on-chip data bus. The small buffer permits independent transfers to take place through the first-in, first-out memory. The drawing processor constructs geometric objects.

and often adequate, but they consume considerable power and occupy significant board space. Controllers based on bipolar bit-slice processors have also been effective but require lengthy design work and are difficult to upgrade. The MOS microprocessor display controller offers lower package count and lower cost but is too slow to be considered a high-performance solution.

The GDC, however, fabricated with a 3-micrometer n-channel MOS process using over 13,000 transistors, simultaneously meets performance and cost requirements of sophisticated yet high-volume graphics terminals. It isolates the display memory from the microprocessor, freeing the processor to handle higher-level graphics calculations and communications with the terminal user and the host processor. It can work with almost any central processing unit to form a basis for a powerful computer graphics system with character capability at very low cost.

Flexible formats

How effective a graphics system is depends largely on the flexibility of the system's display controller. The GDC can control thousands of alphanumeric characters or graphics figures comprising millions of dots. These bit-mapped figures can be drawn so quickly that complex images can be created in one 16.7-millisecond video-frame period. Up to four character-display or two graphics-display areas may be horizontally split and scrolled independently, with little local processor overhead.

The display memory is often larger than the display area, so as to make possible double-buffered display frames, multiple-frame movies and panning and the use of lower-cost video circuitry and monitors. Zoom magnification factors of 1 to 16 may be selected under program control, and a light-pen detection circuit is included. The GDC facilitates the strobe generation for dynamic random-access memories, which it refreshes, even during high zoom magnification.

For graphics, the GDC's display memory can be organized as 2,048 pixels by 2,048 lines, or as 1,024 pixels by 1,024 lines with four bit planes per location, or in just about any other combination. Data can be moved from the display memory to the screen in 16-bit words in a minimum cycle of under 400 ns. Optionally, two 16-bit words can be moved simultaneously for a video pixel rate of 80 megahertz.

Allowing for RS-343 blanking, this rate yields a 60-hertz noninterlaced display of 1,024 pixels by 792 lines, for a 4:3 display aspect ratio. A 2:1 interlaced display of 1,024 by 1,024 dots with red, green, blue, and overlay

and the proper code on the S_0 through S_2 lines, the READY output signal immediately goes low, regardless of whether access is gained to the RAM. The READY signal stays low, and the host remains in a wait state until all uncertainties are resolved by the arbitration latch.

When one of the two arbitration latch outputs, ACTA or ACTB, rises above a threshold level, indicating that any conflict has been resolved, one of the ports can access the RAM. The output of the metastable threshold detector is sampled when CLKIN is high. When CLKIN goes low, a feedback circuit quickly consolidates the sampled value to convert a possible indeterminate sample into a valid high or low signal. A high ACT signal indicates that the memory cycle can proceed. Thereafter, the READY goes high after the next rising edge of CLKIN.

Since the majority of processors cannot accept an asynchronous READY signal, the CLKIN signal from a processor synchronizes the trailing edge of READY to the system clock. The leading edge is already synchronized by $\overline{WE}$ or $\overline{OE}$ from the host.

Figure 4d shows the detailed sequence of events when both ports attempt to read simultaneously and port A prevails. As the data registers are addressed, READY goes low on both ports, and ACTA and ACTB both begin to rise, producing the metastable condition. During a subsequent CLKIN pulse (pulse 2), a valid high-level ACTA is gated into the synchronizer, and the port-A READY signal goes high again on the rising edge of the third CLKIN pulse. The valid ACTA activates the data and local address-pointer registers of port A and connects them to the respective data and address lines of the RAM to accept valid data.

When port A completes its memory cycle, ACTA goes low, allowing ACTB to rise (pulse 4). Then port B's READY signal goes high after the next rising edge of CLKIN (pulse 5). With the high READY and ACTB, port B seizes the inputs to the RAM and supplies valid data until the memory cycle at port B is completed and ACTB goes low again.

Using the lockout capability

However, to obtain exclusive access to the MPIF RAM, the chip's lockout function is invoked. When lockout has been activated by a local port, either by pulling the LOCKIN input low or writing a ONE into the SLOC bit of the control register, the ACT signal of the remote port is held low and it cannot access the RAM. Instead, if the system bus applies CLKIN signals and accepts READY inputs, the remote port enters a wait state until the lockout is removed.

Nevertheless, a lockout asserted by a local port does not guarantee the immediate use of RAM. The lockout becomes effective only after any ongoing RAM access under control of the remote port is completed. Also, if a lockout had been asserted by the remote port, the lockout, too, must be cleared.

Even if the system does not employ READY and CLKIN signals on a port, the SLOC bit still can be used to guarantee local port access to the RAM. The LAK interrupt bit in the status register is set in response to SLOC as soon as the lockout becomes effective. To ensure exclusivity, LAK is set only after an ongoing memory cycle is complete and any lockout asserted by the remote port is cleared. After this, the local port has exclusive use of the RAM until the SLOC bit is cleared.□

MPIF register map

Register-selection lines (S_0, S_1, S_2)	Functions	Registers selected (Port A)	(Port B)
000	Data/increment mode	Data A	Data B
001	Data mode	Data A	Data B
010	Message-in buffer*	Message B	Message A
011	Message-out buffer	Message A	Message B
100	Control	Control A	Control B
101	Local-address pointer	Address A	Address B
110	Status*	Status A	Status B
111	Remote-address pointer	Address B	Address A

*Provides a read-only mode; all other functions have both read and write modes.

Control-register bit assignment

D_0 ---------- D_7

LEA IEN_1 IEN_2 IEN_3 IEN_4 IEN_5 SLOC X

Interrupt mask bits — X: Not used

Status-register bit assignment

D_0 ---------- D_7

INT MI MO LPE RPE LAK X X

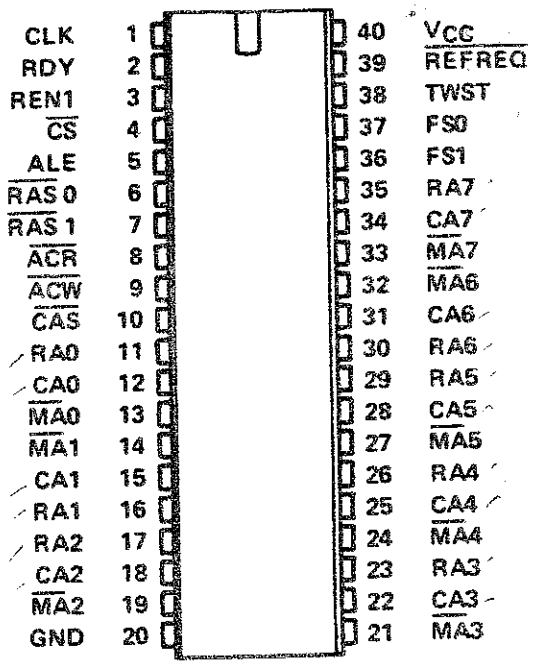
Interrupting status bits — X X: Not used

TMS 4500 NH
DYNAMIC RAM CONTROLLER

JULY 1981

- Controls Operation of 8K/16K/32K/64K Dynamic RAMs
- Creates Static RAM Appearance
- One Package Contains Address Multiplexer, Refresh Control, and Timing Control
- Directly Addresses and Drives Up to 256K Bytes of Memory Without External Drivers
- Operates from Microprocessor Clock
  - No Crystals, Delay Lines, or RC Networks
  - Eliminates Arbitration Delays
- Refresh May Be Internally or Externally Initiated
- Versatile
  - Strap-Selected Refresh Rate
  - Synchronous, Predictable Refresh
  - Selection of Distributed, Transparent, and Cycle-Steal Refresh Modes
  - Interfaces Easily to Popular Microprocessors
- Strap-Selected Wait State Generation for Microprocessor/Memory Speed Matching
- Ability to Synchronize or Interleave Controller with the Microprocessor System (Including Multiple Controllers)

TMS 4500
40-PIN PLASTIC
DUAL IN-LINE PACKAGE
(TOP VIEW)



description

The TMS 4500 is a monolithic DRAM system controller designed to provide address multiplexing, timing, control and refresh/access arbitration functions to simplify the interface of dynamic RAMs to microprocessor systems.

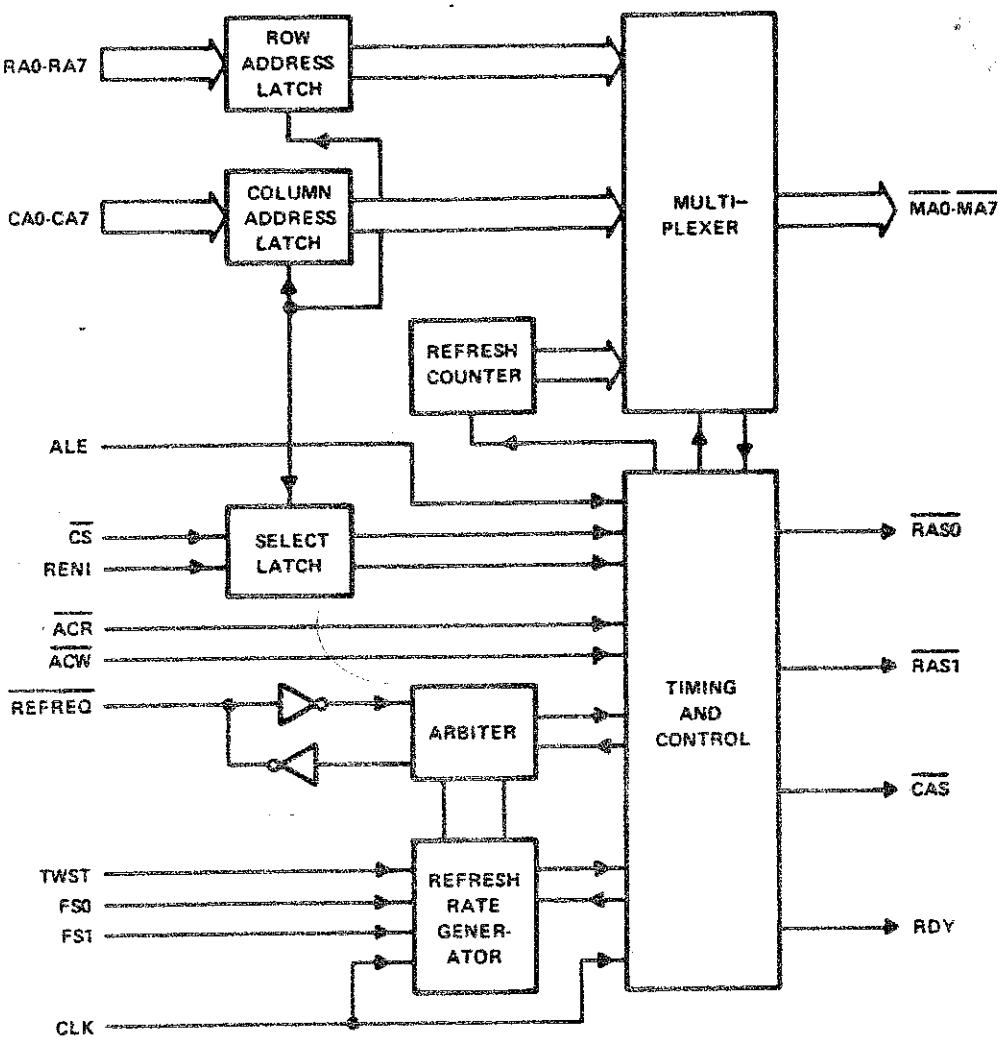
The controller contains a 16-bit multiplexer that generates the address lines for the memory device from the 16 system address bits and provides the strobe signals required by the memory to decode the address. An 8-bit refresh counter generates the 256-row addresses required for refresh.

A refresh timer is provided that generates the necessary timing to refresh the dynamic memories and assure data retention.

The TMS 4500 also contains refresh/access arbitration circuitry to resolve conflicts between memory access requests and memory refresh cycles. The TMS 4500 is offered in a 40-pin, dual-in-line plastic package and is guaranteed for operation from 0°C to 55°C.

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

BLOCK DIAGRAM



pin descriptions

RA0 - RA7	Input	Row Address – These address inputs are used to generate the row address for the multiplexer.
CA0 - CA7	Input	Column Address – These address inputs are used to generate the column address for the multiplexer.
$\overline{MA0}$ - $\overline{MA7}$	Output	Memory Address – These outputs are designed to drive the addresses of the dynamic RAM array.
ALE	Input	Address Latch Enable – This input is used to latch the 16 address inputs, $\overline{CS}$ and REN1. This also initiates an access cycle if chip select is valid. The rising edge (low level to high level) of ALE, $\overline{ACR}$ or $\overline{ACW}$, whichever occurs first, terminates the cycle by returning $\overline{RAS}$ and $\overline{CAS}$ to the high level.

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

pin descriptions (continued)

$\overline{CS}$	Input	Chip Select – A low on this input enables an access cycle. The trailing edge of ALE latches the chip select input.
REN1	Input	RAS Enable 1 – This input is used to select one of two banks of RAM via the $\overline{RAS}$ 0 and $\overline{RAS}$ 1 outputs when chip select is present.
$\overline{ACR}$, $\overline{ACW}$	Input	Access Control, Read; Access Control, Write – A low on either of these inputs causes the column address to appear on $\overline{MA0}$ - $\overline{MA7}$ and the column address strobe. The rising edge of $\overline{ACR}$ or $\overline{ACW}$ or ALE terminates the cycle by ending $\overline{RAS}$ and $\overline{CAS}$ strobes.
CLK	Input	System Clock – This input provides the master timing to generate refresh cycle timings and refresh rate. Refresh rate is determined by the TWST, FS1, FS0 inputs.
$\overline{REFREQ}$	Input/Output	Refresh Request – (This input should be driven by an open-collector output.) On input, a low-going edge initiates a refresh cycle and will cause the internal refresh timer to be reset on the next falling edge of the CLK. As an output, a low-going edge signals an internal refresh request and that the refresh timer will be reset on the next low-going edge of CLK. $\overline{REFREQ}$ will remain low until the refresh cycle is in progress and the current refresh address is present on $\overline{MA0}$-$\overline{MA7}$.
$\overline{RAS0}$, $\overline{RAS1}$	Output	Row Address Strobe – These outputs are used to latch the row address into the bank of DRAMs selected by REN1. On refresh both signals are driven.
$\overline{CAS}$	Output	Column Address Strobe – This output is used to latch the column address into the DRAM array.
RDY	Output	Ready – This output synchronizes memories that are too slow to guarantee microprocessor access time requirements. This output is also used to inhibit access cycles during refresh when in cycle-steal mode.
TWST	Input	Timing/Wait Strap – A high on this input indicates a wait state should be added to each memory cycle. In addition it is used in conjunction with FS0 and FS1 to determine refresh rate and timing.
FS0, FS1	Inputs	Frequency Select 0; Frequency Select 1 – These are strap inputs to select Mode and Frequency of operation as shown in Table 1.

TMS 4500 NH
DYNAMIC RAM CONTROLLER

arbiter

The arbiter provides two operational cycles: access and refresh. The arbiter resolves conflicts between cycle requests and cycles in execution, and schedules the inhibited cycle when used in cycle-steal mode.

timing and control block

The timing and control block executes the operational cycle at the request of the arbiter. It provides the DRAM array with $\overline{RAS}$ and $\overline{CAS}$ signals. It provides the CPU with a RDY signal. It controls the multiplexer during all cycles. It resets the refresh rate generator and decrements the refresh counter during refresh cycles.

TABLE 1 – STRAP CONFIGURATION

STRAP INPUT MODES			WAIT STATES FOR MEMORY ACCESS	REFRESH RATE	MINIMUM CLK FREQ. (MHz)	REFRESH FREQ. (kHz)	CLOCK CYCLES FOR EACH REFRESH
TWST	FS1	FS0					
L	L	L†	0	EXTERNAL	–	REFREQ	4
L	L	H	0	CLK ÷ 31	1.984	64 - 95‡	3
L	H	L	0	CLK ÷ 46	2.944	64 - 85‡	3
L	H	H	0	CLK ÷ 61	3.904	64 - 82§	4
H	L	L	1	CLK ÷ 46	2.944	64 - 85‡	3
H	L	H	1	CLK ÷ 61	3.904	64 - 80‡	4
H	H	L	1	CLK ÷ 76	4.864	64 - 77‡	4
H	H	H	1	CLK ÷ 91	5.824	64 - 88¶	4

† This strap configuration resets the Refresh Timer circuitry.

‡ Upper figure in refresh frequency is the frequency that is produced if the minimum CLK frequency of the next select state is used.

§ Refresh frequency if CLK frequency is 5 MHz.

¶ Refresh frequency if CLK frequency is 8 MHz.

Upon Power-Up it is necessary to provide a reset signal by driving all three straps to the controller low to initialize internal counters. A system's low-active, power-on reset ($\overline{\text{RESET}}$) can be used to accomplish this by connecting it to those straps that are desired high during operation.

functional description

The TMS 4500 consists of six basic blocks; address and select latches, refresh rate generator, refresh counter, the multiplexer, the arbiter, and the timing and control block.

address and select latches

The address and select latches allow the DRAM controller to be used in systems that multiplex address and data on the same lines without external latches. The row address latches are transparent, meaning that while ALE is high, the output at MA0 - MA7 follows the inputs RA0 - RA7 (with inversion of logic level.)

refresh rate generator

The refresh rate generator is a counter that indicates to the arbiter that it is time for a refresh cycle. The counter divides the clock frequency according to the configuration straps as shown in Table 1. The counter is reset when a refresh cycle is requested or when TWST, FS1 and FS0 are low. The configuration straps allow the matching of memories to the system access time.

refresh counter

The refresh counter contains the address of the row to be refreshed. The counter is decremented after each refresh cycle.

multiplexer

The multiplexer provides the DRAM array with row, column, and refresh addresses at the proper times. Its inputs are the address latches and the refresh counter. The outputs provide up to 16 multiplexed addresses on eight lines. The multiplexer can drive address inputs for up to 34 DRAM devices. (Note that the memory address outputs of the multiplexer are inverted with respect to the row and column addresses).

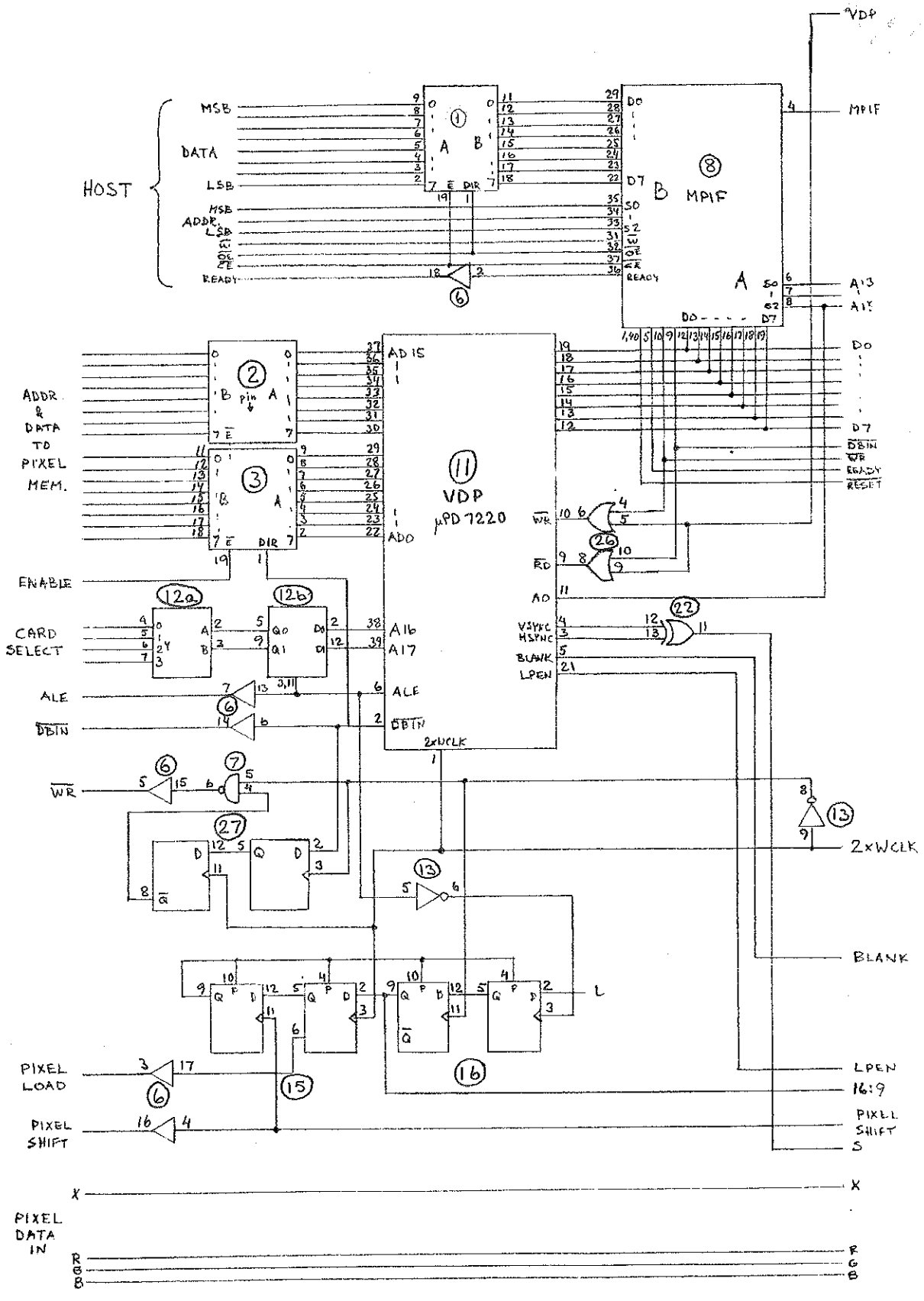
TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

APPENDIX 2

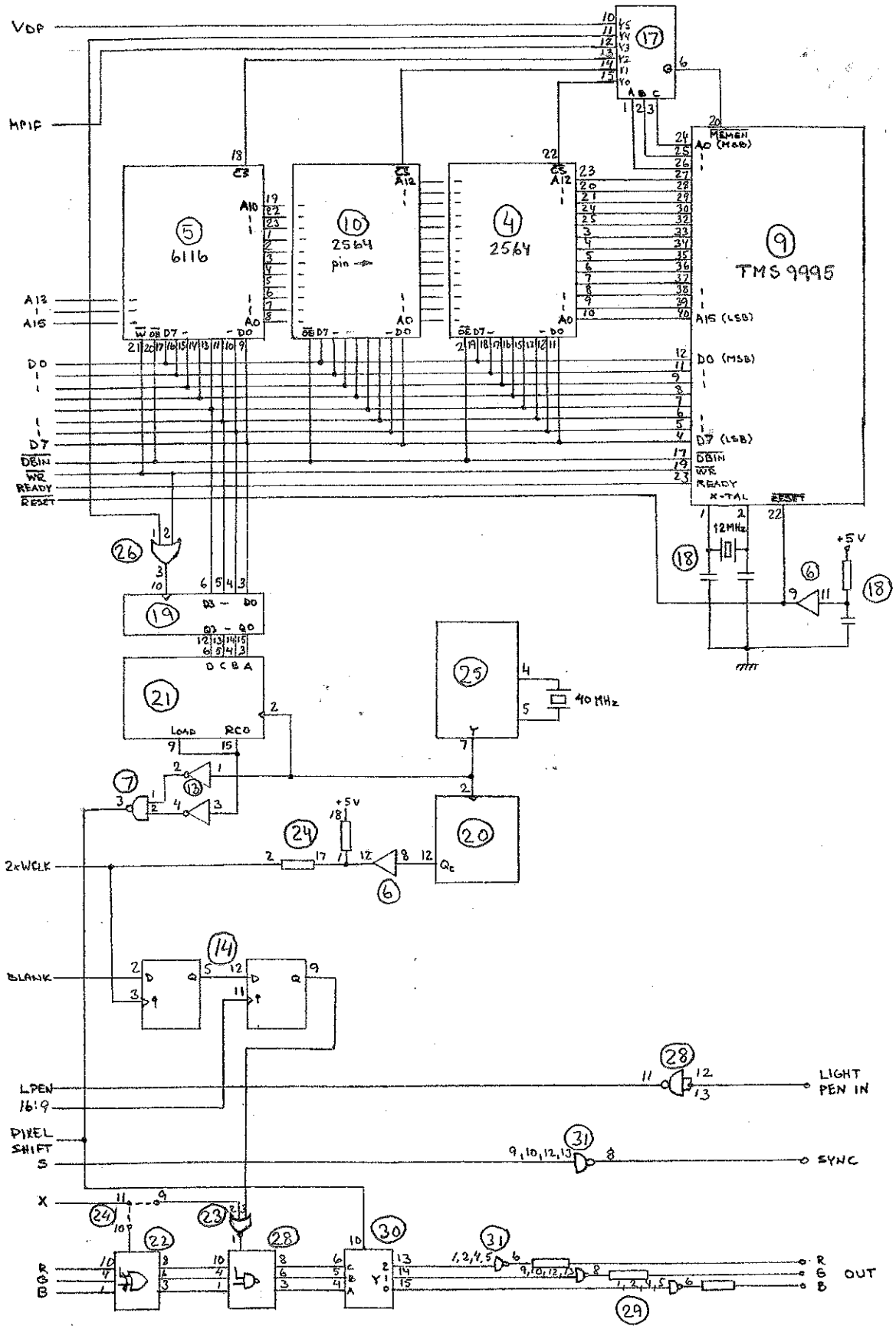
Kopplingsscheman

Videokort höger del
Videokort vänster del
Minneskort 64K X 16 bits
Minneskort modifierat (endast förändringar)
Komponentlista videokort
Komponentlista minneskort

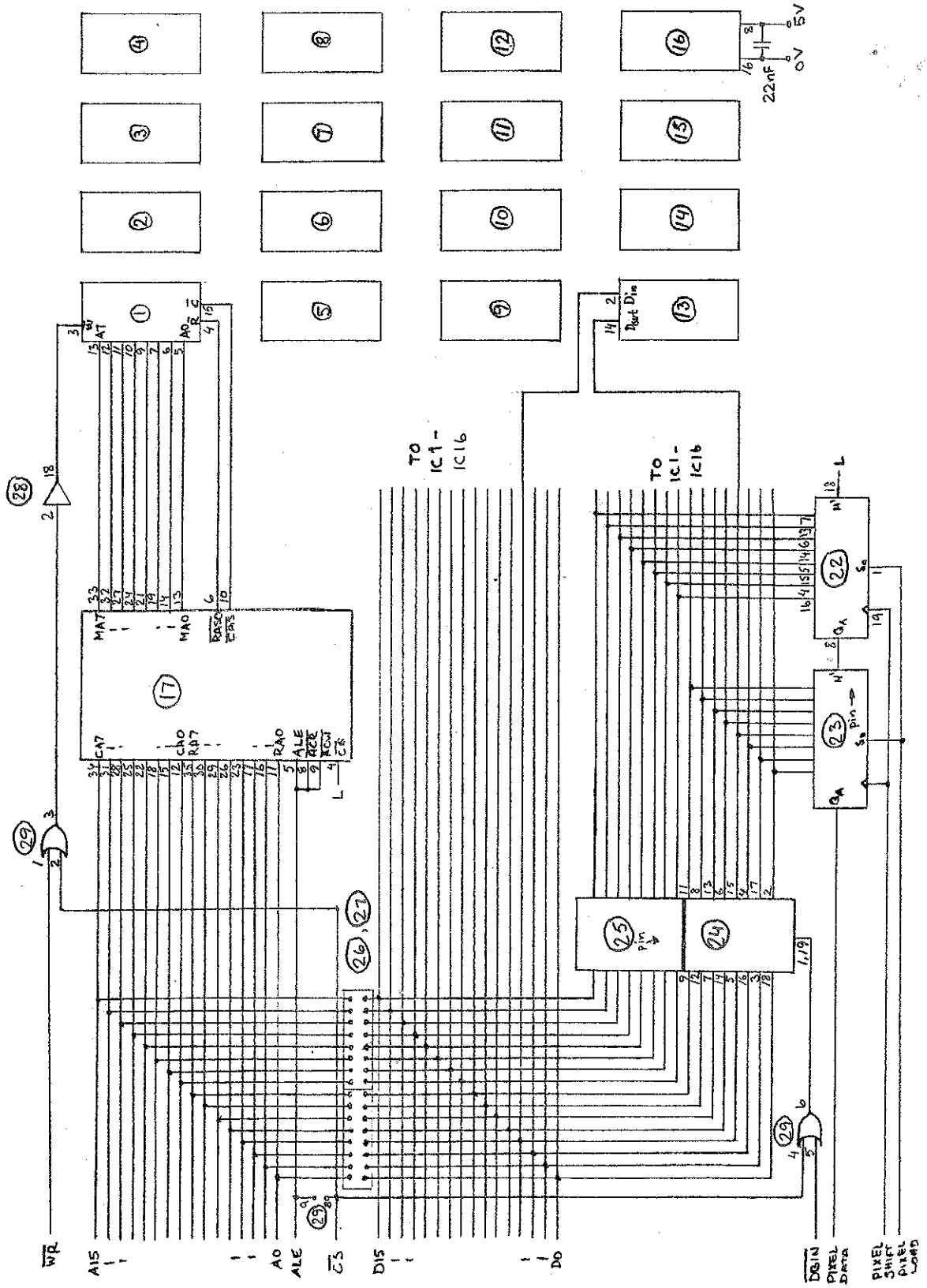
VIDEOKORT del vänster



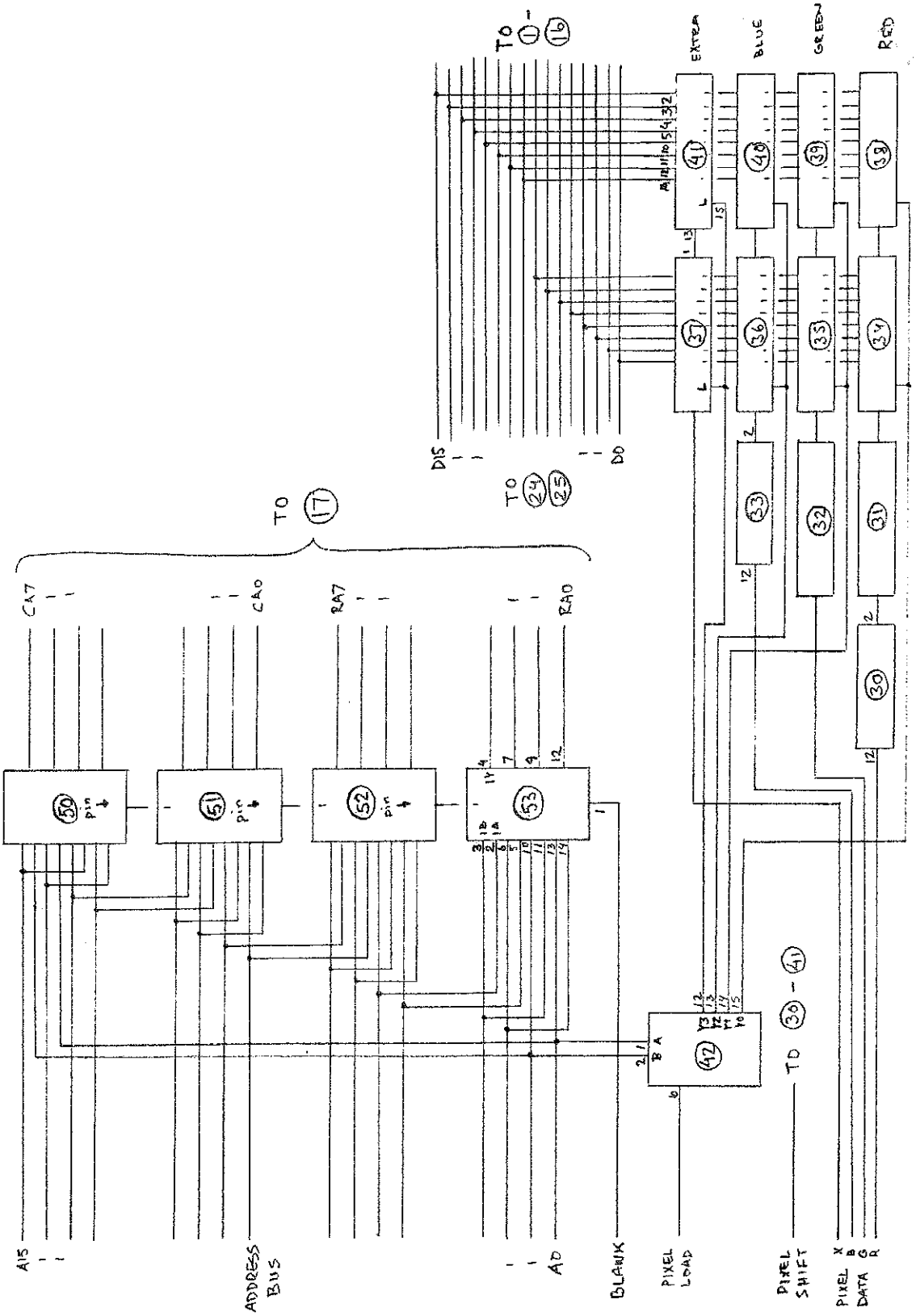
VIDEOKORT del höger



MINNESKORT 64 K X 16 bits



MINNESKORT Modifierat för fyra bildplan



Komponentlista videokort

Nummer	Krets	Funktion
1-3	74LS245	Bidir. buffer
4	2564	8K X 8 EPROM
5	6116	2K X 8 SRAM
6	74S244	Buffer
7	74S00	AND-gate
8	TMS 99650	MPIF
9	TMS 9995	µ-processor
10	2564	8K X 8 EPROM
11	µPD 7220	VDP
12a	74LS139	Data selector
12b	74LS74	Dual flip-flop
13	74S04	Hex inverters
14-16	74LS74	Dual flip-flop
17	74LS138	Data selector
18	Komponentbärare	
19	74LS395A	Latch (shift reg.)
20-21	74S169	Counter
22	74S36	XOR-gate
23	74S02	NOR-gate
24	Komponentbärare	
25	74S124	Oscillator
26	74LS32	OR-gate
27	74LS74	Dual flip-flop
28	74S00	NAND-gate
29	74S140	Line driver
30	74S195A	Latch
31	74S140	Line driver

Komponentlista minneskort

Nummer	Krets	Funktion
1-16	4164-20	64K X 1 DRAM
17	TMS 4500	Dynamic RAM Controller
18-21	----------	
22-23	74S299	8-bits shiftregister
24-25	74LS244	Buffer
26-27	Straps	
28	74LS244	Buffer
29	74LS32	OR-gate

Modifierad version

30	74LS195A	4-bits shiftregister
31-32	74LS166	8-bits shiftregister
33	74LS195A	4-bits shiftregister
34-41	74LS166	8-bits shiftregister
42	74LS139	Decoder
43-49	----------	
50-53	74LS158	Data selector

APPENDIX 3

Program

Parameteröverföring för första testdelen

Demonstrationsprogram i assembly
- constants
- interrupt vector
- pattern
- demo program
- DEMTAB
- commands
- subroutines
- beskrivning av kommandon

```
*       PARAMETERöVERFöRINGSPROGRAM FöR 9995
*
        IDT     'MOVE'
        DEF     START
        DEF     ARBWS
        DEF     DECINT
*
EPROM1  EQU     >2000           address mapping
MPIF    EQU     >5000           address mapping
ZOOM    EQU     >6000           address mapping
VDP     EQU     >7000           address mapping
ARBWS   EQU     >F000           address mapping
CHRTAB  EQU     EPROM1          address mapping
*
DATINC  EQU     MPIF            MPIF mapping
DAT     EQU     MPIF+1
MESIN   EQU     MPIF+2
MESOUT  EQU     MPIF+3
LAP     EQU     MPIF+5
STATUS  EQU     MPIF+6
*
VDPSTA  EQU     VDP             VDP mapping
VDPPAR  EQU     VDP
VDPCOM  EQU     VDP+1
*
MI      EQU     >4000           mask for message in
FIFOFU  EQU     >0200           mask for VDP fifo full
*
        PSEG
START   EQU     $
        LI      R2,2
        LI      R0,0
ST1     DEC     R0              tidsfördröjning
        SWPB    R1
        JNE     ST1
        DEC     R2
        JNE     ST1
        CLR     ZOOM            zoom factor = 0
*
VIDEO   EQU $
RES     CLR     R1
        MOVB    R1,VDPCOM       VDP reset
        LI      R2,PARTAB
PARAM   MOVB    *R2+,R3
        JEQ     COMMAN          if zero then end of parameters
        MOVB    R3,VDPPAR
        JMP     PARAM
COMMAN  EQU     $
        LI      R3,>6F00
        MOVB    R3,VDPCOM       VSYNC : internal generated
        LI      R3,>6B00
        MOVB    R3,VDPCOM       START
*
NEXTCO  EQU     $
        LI      R0,FIFOFU       fifo full bit mask
        LI      R8,>FF00
        MOVB    R8,MESOUT       ready for next command
        LI      R8,MI           message in bit mask
```

```
MESS    MOVB    STATUS,R6
        COC     R8,R6           wait for message =
        JNE     MESS               number of parameters
        CLR     R7
        MOVB    MESIN,R7        set message
        SWPB    R7
        CI      R7,127          if > 127 then character
        JLE     M3              jump if command
*
        AI      R7,-128
        SLA     R7,3            index in CHRTAB
        AI      R7,CHRTAB
        LI      R9,>7800        PRAM command
        LI      R1,>0400        fifo empty mask
M1      MOVB    VDPSTA,R6
        COC     R1,R6
        JNE     M1
        MOVB    R9,VDPCOM
        LI      R10,8
M2      MOVB    *R7+,VDPPAR     load character pattern
        DEC     R10
        JNE     M2
        LI      R9,>6800        GCHRD command
        MOVB    R9,VDPCOM
        JMP     NEXTCO
M3      CLR     R8
        MOVB    R8,LAP          LAP=0
        MOVB    DATINC,R8
        JEQ     M5              if reset skip test
        CI      R8,>4600
        JNE     M4              if zoom com change in ZOOM
*
        MOVB    DAT,R9          set zoom factor
        SRL     R9,4
        MOVB    R9,ZOOM
*
M4      MOVB    VDPSTA,R6
        COC     R0,R6           VDP ready ?
        JEQ     M4
M5      MOVB    R8,VDPCOM
        CI      R7,0
NEXTPA  JEQ     NEXTCO
M6      MOVB    VDPSTA,R6
        COC     R0,R6           VDP ready ?
        JEQ     M6
        MOVB    DATINC,VDPPAR
        DEC     R7
        JMP     NEXTPA
*
PARTAB  BYTE    >16             P1: mode of operation
        BYTE    120             P2: active display words
        BYTE    77              P3: vert s w (3), hor sync
        BYTE    8               P4: hor f p (6), v s w
        BYTE    10              P5: hor b p
        BYTE    1               P6: vert f p
        BYTE    104             P7: active display lines
        BYTE    41              P8: vert b p (6), a d l
        BYTE    >00             end of parameters
```

```
*       constants
*********************************
*
EPROM0  EQU     >0000           address mapping
EPROM1  EQU     >2000           (* differ from report
SRAM    EQU     >4000             description *)
MPIF    EQU     >5000
ZOOM    EQU     >6000
VDP     EQU     >7000
CHRTAB  EQU     EPROM1
*
*
*       VDP-COMMANDS
*
RESETC  EQU     >00             reset
VSYNCC  EQU     >6F             vertical sync : internal
BCTRLC  EQU     >0C             blanking control
CCHARC  EQU     >4B             cursor & char specify
STARTC  EQU     >6B             start display mode
ZOOMC   EQU     >46             set zoom factor
CURSC   EQU     >49             set cursor position
PRAMC   EQU     >70             load parameter RAM
PRAM8C  EQU     >78             load PRAM start at ad 8
PITCHC  EQU     >47             set pitch value
WDATC   EQU     >20             write data to display memory
WDATRC  EQU     >22             write data : reset bit
MASKC   EQU     >4A             set mask value
FIGSC   EQU     >4C             figure parameter specify
FIGDC   EQU     >6C             figure draw
GCHRDC  EQU     >68             graphic character draw
RDATC   EQU     >A0             read data from display memory
CURDC   EQU     >E0             read cursor position
LPRDC   EQU     >C0             read lightpen detect position
DMARC   EQU     >A4             DMA read
DMAWC   EQU     >24             DMA write
*
*********************************
*
*       VDP-status bit masks
*
DATARE  EQU     >0100           data ready
FIFOFU  EQU     >0200           FIFO full
FIFOEM  EQU     >0400           FIFO empty
DRAWIN  EQU     >0800           fig draw in progress
DMAEXE  EQU     >1000           DMA in progress
VERTSY  EQU     >2000           vertical sync active
HORBLA  EQU     >4000           horizontal blank active
LPENDE  EQU     >8000           light pen detect
*
*********************************
*
*       VDP-addresses
*
VDPSTA  EQU     VDP             VDP status
VDPPAR  EQU     VDP             VDP parameter write
```

```
VDPREA  EQU     VDP+1           VDP parameter read
VDPCOM  EQU     VDP+1           VDP command write
*
*******************************
*
*       Data references
*
        SREF    ARBWS
        SREF    SENTAB
        SREF    FIGTAB
        SREF    CURX
        SREF    CURY
        SREF    CURP
        SREF    COLOR
        SREF    SLINE
        SREF    CHRT
*

******  Interrupt vector
*
        IDT     'INTVEC'
        REF     ARBWS
        REF     START
        REF     DECINT
*
        PSEG
*
INTVEC  DATA    ARBWS,START             int 0
        DATA    ARBWS,START             int 1
        DATA    ARBWS,START             int 2
        DATA    ARBWS,DECINT            int 3
        DATA    ARBWS,START             int 4
FREE    BSS     44
XOPS    BSS     64
        END


******      PATTERNS  FOR  DEMOPROGRAM
*
*
*
PAT0    EQU     >FFFF
PAT1    EQU     >FFFF
PAT2    EQU     >FFFF
PAT3    EQU     >FFFF
PAT4    EQU     >FFFF
PAT5    EQU     >8FF1
PAT6    EQU     >AAAA
PAT7    EQU     >FFFF
PAT8    EQU     >FFFF
PAT9    EQU     >FFFF
PAT10   EQU     >FFFF
PAT20   EQU     >AA55
PAT21   EQU     >55AA
PAT22   EQU     >FFFF
PAT23   EQU     >FFFF
*
```

```
*       Demonstrationsprogram för videoenhet
*         ( färggrafik 320 X 360 pixels )
*
        IDT     'DEMO'
        DEF     START
        DEF     NEXTCO
        DEF     CHRTAB
*
        REF     DEMTAB
        REF     DTEND
        REF     SENDCP
*
        COPY    S.VDPCON
*
        PSEG
*
***   ++++++++++++++++++   Initializing      +++++++++++++++++
*
START   EQU     $
        CLR     ZOOM
        CLR     R0
        MOVB    R0,VDPCOM       VDP reset
        LI      R0,COMTAB
        LI      R4,NUMCOM
ST1     BL      SENDCP          send commands in COMTAB
        DEC     R4
        JNE     ST1
*
*   +++++++++++++++      Main program    ++++++++++++++++++
*
DEMO    EQU     $
        LI      R8,DEMTAB       R8 pointer in DEMTAB
NEXTCO  EQU     $
        MOV     *R8+,R0
        CI      R0,DTEND
        JEQ     DEMO            if end of DEMTAB then repeat
        B       *R0
*
*     ++++++++++++++++++++   COMTAB   ++++++++++++++++++++++++
*
NUMCOM  EQU     6               # of commands to send
*                                                       (50 Hz)
COMTAB  BYTE    RESETC          resetcommand            RESETC
        BYTE    8               # of parameters        8
        BYTE    >16             P1: mode of operation   >16
        BYTE    80              P2: active display words 120
        BYTE    138             P3: vert s w (3),hor sync 76
        BYTE    20              P4: hor f p (6),v s w   32
        BYTE    5               P5: hor b p             20
        BYTE    15              P6: vert f p            2
        BYTE    104             P7: active display lines 44
        BYTE    153             P8: vert b p (6),a d l  33
        BYTE    VSYNCC,0
        BYTE    CCHARC,3,0,0,0
        BYTE    ZOOMC,1,0
        BYTE    PRAMC,8,0,0,128,22,0,0,0,0
        BYTE    STARTC,0
*
        END
```

```
*       DEMTAB: innehåller kommandona för bild-definitionerna
*
        IDT     'DEMTAB'
        DEF     DEMTAB
        DEF     DTEND
        DEF     XONLY
*
        REF     AREA
        REF     LINE
        REF     DELAY
        REF     PATT8
        REF     PATT2
        REF     REKT
        REF     TEXT
        REF     SENDCP
*
*
DTEND   EQU     >FFFF           DEMTAB end marker
*
BLACK   EQU     0               color codes
RED     EQU     1
GREEN   EQU     2
YELLOW  EQU     3
BLUE    EQU     4
MAGENT  EQU     5
CYAN    EQU     6
WHITE   EQU     7
XBLACK  EQU     8               Xplane + color codes
XRED    EQU     9
XGREEN  EQU     10
XYELLO  EQU     11
XBLUE   EQU     12
XMAGEN  EQU     13
XCYAN   EQU     14
XWHITE  EQU     15
XONLY   EQU     16              only Xplane affected
*
SLANT   EQU     >0000           char & rekt direction
NORMAL  EQU     >0100
*
        COPY    S.PATTERN
*
*
*   ++++++++++++++++++++++      DEMTAB     ++++++++++++++++++++++++
*
        EVEN
DEMTAB  EQU     $
        DATA    DELAY,500
BILD1   DATA    PATT8,PAT0,PAT0,PAT0,PAT0
        DATA    AREA,0,0,320,360,BLACK,NORMAL
        DATA    TEXT,5,300,WHITE,1
        TEXT    'VÄLKOMMEN TILL DENNA DEMONSTRATION AV@'
        EVEN
        DATA    TEXT,55,270,RED,1
        TEXT    'HöGUPPLöSANDE FÄRGGRAFIK@'
```

```
        EVEN
        DATA    DELAY,5000
        DATA    TEXT,30,240,CYAN,2
        TEXT    'HÄR ÄR FÄRGERNA@'
        EVEN
        DATA    PATT8,PAT0,PAT0,PAT0,PAT0
        DATA    AREA,0,0,40,200,0,NORMAL
        DATA    AREA,40,0,40,200,1,NORMAL
        DATA    AREA,80,0,40,200,2,NORMAL
        DATA    AREA,120,0,40,200,3,NORMAL
        DATA    AREA,160,0,40,200,4,NORMAL
        DATA    AREA,200,0,40,200,5,NORMAL
        DATA    AREA,240,0,40,200,6,NORMAL
        DATA    AREA,280,0,40,200,7,NORMAL
        DATA    DELAY,8000
        DATA    TEXT,60,230,GREEN,1
        TEXT    'OCH NÅGRA OLIKA NYANSER@'
        EVEN
        DATA    PATT8,PAT20,PAT20,PAT20,PAT20
        DATA    AREA,0,0,320,20,0,NORMAL
        DATA    AREA,0,20,320,20,1,NORMAL
        DATA    AREA,0,40,320,20,2,NORMAL
        DATA    AREA,0,60,320,20,3,NORMAL
        DATA    AREA,0,80,320,20,4,NORMAL
        DATA    AREA,0,100,320,20,5,NORMAL
        DATA    AREA,0,120,320,20,6,NORMAL
        DATA    AREA,0,140,320,20,7,NORMAL
        DATA    DELAY,5000
        DATA    TEXT,32,178,XONLY,1
        TEXT    'TEXT I X-PLANET INVERTERAR DE@'
        EVEN
        DATA    TEXT,36,169,XONLY,1
        TEXT    'ÖVRIGA  FÄRGPLANENS  BITAR !@'
        EVEN
        DATA    DELAY,12000
        DATA    PATT8,PAT0,PAT0,PAT0,PAT0
BILD2   DATA    AREA,0,0,320,360,CYAN,NORMAL
        DATA    TEXT,0,300,BLACK,1
        TEXT    'NÅGRA OLIKA FIGURER SOM RITAS AV@'
        EVEN
        DATA    TEXT,258,300,RED,2
        TEXT    'VDP@'
        EVEN
        DATA    TEXT,306,300,RED,1
        TEXT    'N@'
        EVEN
        DATA    DELAY,2000
        DATA    PATT2,PAT0
        DATA    LINE,10,48,179,150,RED
        DATA    DELAY,1000
        DATA    PATT2,PAT5
        DATA    LINE,15,88,170,135,BLUE
        DATA    DELAY,1000
        DATA    PATT2,PAT6
        DATA    LINE,5,160,184,100,MAGENT
```

```
        DATA    DELAY,4000
        DATA    PATT8,PAT0,PAT0,PAT0,PAT0
        DATA    AREA,200,100,90,90,GREEN,NORMAL
        DATA    DELAY,2000
        DATA    AREA,195,150,60,60,RED,SLANT
        DATA    DELAY,2000
        DATA    PATT8,PAT20,PAT20,PAT20,PAT20
        DATA    AREA,190,10,80,140,BLUE,NORMAL
        DATA    PATT2,PAT0
        DATA    DELAY,3000
        DATA    REKT,20,200,40,40,0,SLANT
        DATA    REKT,30,200,40,40,1,SLANT
        DATA    REKT,40,200,40,40,2,SLANT
        DATA    REKT,50,200,40,40,3,SLANT
        DATA    REKT,60,200,40,40,4,SLANT
        DATA    REKT,70,200,40,40,5,SLANT
        DATA    REKT,80,200,40,40,7,SLANT
        DATA    DELAY,5000
        DATA    PATT8,PAT0,PAT0,PAT0,PAT0
        DATA    REKT,100,30,40,40,BLACK,NORMAL
        DATA    DELAY,3000
        DATA    AREA,101,31,39,39,YELLOW,NORMAL
        DATA    DELAY,3000
        DATA    AREA,111,41,19,19,BLUE,NORMAL
        DATA    DELAY,3000
        DATA    AREA,116,46,9,9,RED,NORMAL
        DATA    DELAY,4000
        DATA    TEXT,150,230,RED,1
        TEXT    '+ CIRKLAR OCH BÅGAR@'
        EVEN
        DATA    PATT8,PAT0,PAT0,PAT0,PAT0
        DATA    DELAY,5000
BILD3   DATA    AREA,0,0,320,360,WHITE,NORMAL
        DATA    TEXT,20,340,BLUE,1
        TEXT    'ETT EXEMPEL PÅ DIAGRAMRITNING@'
        EVEN
        DATA    DELAY,3000
        DATA    PATT2,PAT0
        DATA    LINE,30,28,30,300,MAGENT
        DATA    LINE,28,30,300,30,MAGENT
        DATA    LINE,50,28,50,32,MAGENT
        DATA    LINE,70,28,70,32,MAGENT
        DATA    LINE,90,28,90,32,MAGENT
        DATA    LINE,110,28,110,32,MAGENT
        DATA    LINE,130,28,130,32,MAGENT
        DATA    LINE,150,28,150,32,MAGENT
        DATA    LINE,170,28,170,32,MAGENT
        DATA    LINE,190,28,190,32,MAGENT
        DATA    LINE,210,28,210,32,MAGENT
        DATA    LINE,230,28,230,32,MAGENT
        DATA    LINE,250,28,250,32,MAGENT
        DATA    LINE,270,28,270,32,MAGENT
        DATA    LINE,290,28,290,32,MAGENT
        DATA    LINE,28,50,32,50,MAGENT
        DATA    LINE,28,70,32,70,MAGENT
```

```
DATA      LINE,28,90,32,90,MAGENT
DATA      LINE,28,110,32,110,MAGENT
DATA      LINE,28,130,32,130,MAGENT
DATA      LINE,28,150,32,150,MAGENT
DATA      LINE,28,170,32,170,MAGENT
DATA      LINE,28,190,32,190,MAGENT
DATA      LINE,28,210,32,210,MAGENT
DATA      LINE,28,230,32,230,MAGENT
DATA      LINE,28,250,32,250,MAGENT
DATA      LINE,28,270,32,270,MAGENT
DATA      LINE,28,290,32,290,MAGENT
DATA      TEXT,17,26,MAGENT,1
TEXT      '0@'
EVEN
DATA      TEXT,26,17,MAGENT,1
TEXT      '0@'
EVEN
DATA      TEXT,9,126,MAGENT,1
TEXT      '50@'
EVEN
DATA      TEXT,122,17,MAGENT,1
TEXT      '50@'
EVEN
DATA      TEXT,1,226,MAGENT,1
TEXT      '100@'
EVEN
DATA      TEXT,218,17,MAGENT,1
TEXT      '100@'
EVEN
DATA      DELAY,4000
DATA      PATT2,PAT6
DATA      LINE,40,90,45,92,BLUE
DATA      LINE,46,92,48,89,BLUE
DATA      LINE,49,90,55,92,BLUE
DATA      LINE,56,91,60,83,BLUE
DATA      LINE,61,84,65,100,BLUE
DATA      LINE,66,101,68,111,BLUE
DATA      LINE,69,111,82,113,BLUE
DATA      LINE,83,114,85,120,BLUE
DATA      LINE,86,120,92,115,BLUE
DATA      LINE,93,115,102,113,BLUE
DATA      LINE,102,113,104,120,BLUE
DATA      LINE,105,121,123,152,BLUE
DATA      LINE,124,152,129,160,BLUE
DATA      LINE,130,160,140,154,BLUE
DATA      LINE,141,154,146,159,BLUE
DATA      LINE,147,158,220,202,BLUE
DATA      LINE,221,202,228,192,BLUE
DATA      LINE,229,193,241,192,BLUE
DATA      LINE,241,193,248,212,BLUE
DATA      LINE,249,211,253,222,BLUE
DATA      LINE,254,223,270,292,BLUE
DATA      LINE,271,291,280,285,BLUE
DATA      DELAY,10000
DATA      TEXT,53,280,MAGENT,1
TEXT      'SLUT PÅ DEMONSTRATIONEN@'
EVEN
DATA      DELAY,15000
DATA      DTEND
```

```
*       Kommandon som används i DEMTAB.
*       Se förklaring av parametrar på sidan efter
*       dessa program. Se också VDPns datablad.
*
        IDT     'COMMAN'
*
*
        COPY    S.VDPCON
*
        REF     NEXTCO
        REF     FIGDRW
        REF     SENDCP
*
YRES    EQU     359
*
*   ++++++++++++++++++++++      AREA      ++++++++++++++++++++++++
*
        DEF     AREA            AREA,Sx,Sy,Lx,Ly,C,D
*
AREA    EQU     $
        LI      R0,ZOOMC*256+1
        MOV     R0,SENTAB
        CLR     SENTAB+2
        LI      R0,SENTAB
        BL      SENDCP          set zoomfactor = 0
        LI      R0,FIGSC*256+5  command and # of parameters
        MOV     R0,FIGTAB
        LI      R0,>1100        area fill specification
        A       10(R8),R0
        MOVB    R0,FIGTAB+2
        MOVB    5(R8),FIGTAB+5
        MOVB    4(R8),FIGTAB+6  A**
        MOV     6(R8),R0
        DEC     R0
        MOVB    R0,FIGTAB+4
        SWPB    R0              B**
        MOVB    R0,FIGTAB+3
        LI      R5,GCHRDC*256
        MOV     *R8,CURX
        LI      R0,YRES
        S       2(R8),R0
        MOV     R0,CURY
        MOV     8(R8),COLOR
        BL      FIGDRW          draw area
        AI      R8,12
        B       NEXTCO
*
*   ++++++++++++++++++++++    DELAY    ++++++++++++++++++++++++
*
        DEF     DELAY           DELAY,time
        DEF     SDELAY
        DEF     DECINT
*
DECREM  EQU     >FFFA
FLAGS   EQU     >1EE0
```

```
*
DELAY   EQU     $               delays execution specified time
        LI      R11,NEXTCO        in milliseconds.
        MOV     *R8+,R0
SDELAY  LI      R12,FLAGS       entry points for subroutines
        SBZ     0               select timer mode
        LI      R1,750          = 1 ms
        MOV     R1,DECREM
        SBO     1               start timer
WAIT    LIMI    3
        JMP     WAIT            wait for interrupt
DECINT  DEC     R0              interrupt routine
        JNE     WAIT
        SBZ     1               disable timer
        B       *11
*
*     ++++++++++++++++        LINE        ++++++++++++++++++++++
*
        DEF     LINE            LINE,Sx,Sy,Ex,Ey,C
*
DIRTAB  EQU     $
        BYTE    7,4,0,3,6,5,1,2
*
LINE    EQU     $
        CLR     R2
        MOV     6(R8),R0
        S       2(R8),R0        delta Y = Ey-Sy
        JLT     L1
        INC     R2
L1      MOV     4(R8),R1
        S       *R8,R1          delta X = Ex-Sx
        JLT     L2
        INCT    R2
L2      ABS     R0
        ABS     R1
        C       R1,R0
        JLT     L3
        AI      R2,4
        MOV     R1,R3
        MOV     R0,R1
        MOV     R3,R0           R0: independent axis
L3      MOVB    DIRTAB(R2),R2   R1: dependent axis
        AI      R2,8*256
        MOVB    R2,FIGTAB+2     line + dir specification
        MOVB    R0,FIGTAB+4
        SWPB    R0
        MOVB    R0,FIGTAB+3     I
        SWPB    R0
        SLA     R1,1            2*D
        MOV     R1,R2
        S       R0,R2
        MOVB    R2,FIGTAB+6     2*D - I
        SWPB    R2
        MOVB    R2,FIGTAB+5
        SWPB    R2
```

```
          S       R0,R2
          MOVB    R2,FIGTAB+8       2*D - 2*I
          SWPB    R2
          MOVB    R2,FIGTAB+7
          MOVB    R1,FIGTAB+10
          SWPB    R1                2*D
          MOVB    R1,FIGTAB+9
          LI      R0,FIGSC*256+9    command + # of parameters
          MOV     R0,FIGTAB
          LI      R5,FIGDC*256
          MOV     *R8,CURX
          LI      R0,YRES
          S       2(R8),R0
          MOV     R0,CURY
          MOV     8(R8),COLOR
          BL      FIGDRW            draw line
          AI      R8,10
          B       NEXTCO
*
*         ++++++++++++++++++++    TEXT     ++++++++++++++++++++
*
          DEF     TEXT                DATA   TEXT,Sx,Sy,C,Z
*                                     TEXT   'Xxxx xxx xx xxx@'
*                                     EVEN
*
TEXT      EQU     $
          MOV     *R8+,CURX
          LI      R0,YRES
          S       *R8+,R0
          MOV     R0,CURY
          MOV     *R8+,COLOR
          MOV     *R8+,R0           zoom factor
          MOV     R0,R9
          DEC     R0
          SWPB    R0
          MOVB    R0,SENTAB+2
          LI      R0,ZOOMC*256+1
          MOV     R0,SENTAB
          LI      R0,SENTAB
          BL      SENDCP            set zoom
          LI      R0,FIGSC*256+3    initialize FIGTAB
          MOV     R0,FIGTAB
          LI      R0,>1207          character specification
          MOV     R0,FIGTAB+2
          CLR     FIGTAB+4
NEXTCH    CLR     R0
          MOVB    *R8+,R0           get character
          CI      R0,'@'*256
          JEQ     TEXEND            end of string ?
          SRL     R0,5
          AI      R0,CHRTAB         address to char pattern
          MOV     *R0+,SENTAB+2       in CHRTAB
          MOV     *R0+,SENTAB+4
          MOV     *R0+,SENTAB+6
          MOV     *R0+,SENTAB+8
```

```
          LI        R0,PRAM8C*256+8
          MOV       R0,SENTAB
          LI        R0,SENTAB
          BL        SENDCP              load VDP PRAM
          LI        R5,GCHRDC*256
          BL        FIGDRW              draw character
          LI        R0,8
          MPY       R9,R0
          A         R1,CURX             advance cursor
          JMP       NEXTCH
TEXEND    DEC       R8
          ANDI      R8,>FFFE
          INCT      R8
          B         NEXTCO
*
*
*         +++++++++++++++     PATT8       ++++++++++++++++++++++
*
          DEF       PATT8               PATT8,P0*256+P1,..,P6*256+P7
*
PATT8     EQU       $
          LI        R0,PRAM8C*256+8
          MOV       R0,SENTAB
          MOV       *R8+,SENTAB+2
          MOV       *R8+,SENTAB+4
          MOV       *R8+,SENTAB+6
          MOV       *R8+,SENTAB+8
          LI        R0,SENTAB
          BL        SENDCP              load VDP PRAM
          B         NEXTCO
*
*
*         ++++++++++++++++++    PATT2       ++++++++++++++++++++
*
          DEF       PATT2               PATT8,P0*256+P1
*
PATT2     EQU       $
          LI        R0,PRAM8C*256+2
          MOV       R0,SENTAB
          MOV       *R8,SLINE
          MOV       *R8+,SENTAB+2
          LI        R0,SENTAB
          BL        SENDCP              load VDP PRAM
          B         NEXTCO
*
*
*       ++++++++++++++++++      REKT       ++++++++++++++++++++++
*
          DEF       REKT                REKT,Sx,Sy,Lx,Ly,C,D
*
REKT      EQU       $
          LI        R0,FIGSC*256+11
          MOV       R0,FIGTAB
          LI        R0,>4103            rektangle specification
          A         10(R8),R0
```

```
        MOV     R0,FIGTAB+2
        CLR     FIGTAB+4
        MOV     4(R8),R0
        MOVB    R0,FIGTAB+6
        MOVB    R0,FIGTAB+12
        SWPB    R0
        MOVB    R0,FIGTAB+5
        MOVB    R0,FIGTAB+11
        MOV     6(R8),R0
        MOVB    R0,FIGTAB+8
        SWPB    R0
        MOVB    R0,FIGTAB+7
        SETO    R0
        MOVB    R0,FIGTAB+9
        MOVB    R0,FIGTAB+10
        MOV     *R8,CURX
        LI      R0,YRES
        S       2(R8),R0
        MOV     R0,CURY
        MOV     8(R8),COLOR
        LI      R5,FIGDC*256
        BL      FIGDRW              draw rectangle
        AI      R8,12
        B       NEXTCO
*
        END
```

```
*       SUB-SUBROUTINES  FOR  USE  WITH  DEMOPROGRAM
*
*
        IDT     'SUBR'
*
*
        COPY    S.VDPCON        Copy VDP-constants
*
        PSEG
*
*     ++++++++++++++++++++   SENDCP   ++++++++++++++++++++++
*
        DEF     SENDCP          send command & parameters
*
SENDCP  EQU     $               com & par table pointer in R0
S1      MOVB    VDPSTA,R1
        ANDI    R1,FIFOFU
        JNE     S1
        MOVB    *R0+,VDPCOM
        CLR     R2
        MOVB    *R0+,R2
        JEQ     S3
        SWPB    R2
S2      MOVB    VDPSTA,R1
        ANDI    R1,FIFOFU
        JNE     S2
        MOVB    *R0+,VDPPAR
        DEC     R2
        JNE     S2
S3      B       *R11
*
*     ++++++++++++++++++   SENDC    ++++++++++++++++++++++
*
        DEF     SENDC           send command to VDP
*
SENDC   EQU     $               command in R0:msb
S4      MOVB    VDPSTA,R1
        ANDI    R1,FIFOFU
        JNE     S4
        MOVB    R0,VDPCOM
        B       *R11
*
*     +++++++++++++   PI120  &  PI30   ++++++++++++++++++++
*
        DEF     PI120           pitch set
        DEF     PI30
*
PI120   LI      R2,120*256
        LI      R0,(BCTRLC+1)*256
        JMP     P2              120 when displaying
PI30    LI      R2,30*256       30 when figure drawing
        LI      R0,BCTRLC*256
P2      LI      R1,PITCHC*256
P3      MOVB    VDPSTA,R3
```

```
          ANDI      R3,FIFOFU
          JNE       P3
          MOVB      R0,VDPCOM
P4        MOVB      VDPSTA,R3
          ANDI      R3,FIFOFU
          JNE       P4
          MOVB      R1,VDPCOM
P5        MOVB      VDPSTA,R3
          ANDI      R3,FIFOFU
          JNE       P5
          MOVB      R2,VDPPAR
          B         *R11
*
*     ++++++++++++++++++++    FIGDRW    ++++++++++++++++++++++++
*
          DEF       FIGDRW              figure drawing
          REF       XONLY
*
FIGDRW    EQU       $
          MOV       R11,R10             save return address
          CLR       R7                  plane counter
          MOV       COLOR,R6            color code
*
          CI        R6,XONLY            if only Xplane is to be changed
          JNE       F1
          LI        R7,3
          LI        R6,1
*
F1        BL        PI30
NEXTPL    MOV       R7,CURP             cursor pos in CURX, CURY, CURP
          BL        @CUR
          LI        R0,WDATRC
          MOV       R6,R1
          ANDI      R1,1
          A         R1,R0
          SWPB      R0
          BL        SENDC               WDAT: set/reset bit
          LI        R0,FIGTAB
          BL        SENDCP
          MOVB      R5,R0               FIGD or GCHRD command
          BL        SENDC
          SRL       R6,1
          INC       R7
          CI        R7,4
          JNE       NEXTPL
          BL        PI120
          B         *R10
*
*      ++++++++++++++++++    CUR     ++++++++++++++++++++++++
*
          DEF       CUR                 set cursor
*
PITCH     DATA      30
*
CUR       EQU       $
```

```
        MOV     CURX,R0
        MOV     CURY,R1
        MOV     CURP,R2
        MOV     R0,R3
        SLA     R3,12           dAD   XXXX 0000 0000 0000 R3
        SRL     R0,4
        SRC     R2,2
        A       R2,R0
        MPY     PITCH,R1
        A       R2,R0           EAD   XXXX XXXX XXXX XXXX R0
        LI      R1,CURSC*256+3
        MOV     R1,SENTAB
        SWPB    R0
        MOV     R0,SENTAB+2
        MOVB    R3,SENTAB+4
        LI      R0,SENTAB
        B       SENDCP
*
*
*
```

```
        EVEN
        DATA    TEXT,162,17,MAGENT,1
        TEXT    '70@'
        EVEN
        DATA    TEXT,9,186,MAGENT,1
        TEXT    '80@'
        EVEN
        DATA    TEXT,182,17,MAGENT,1
        TEXT    '80@'
        EVEN
        DATA    TEXT,9,206,MAGENT,1
        TEXT    '90@'
        EVEN
        DATA    TEXT,202,17,MAGENT,1
        TEXT    '90@'
        EVEN
        DATA    TEXT,1,226,MAGENT,1
        TEXT    '100@'
        EVEN
        DATA    TEXT,218,17,MAGENT,1
        TEXT    '100@'
        EVEN
        DATA    TEXT,1,246,MAGENT,1
        TEXT    '110@'
        EVEN
        DATA    TEXT,238,17,MAGENT,1
        TEXT    '110@'
        EVEN
        DATA    TEXT,1,266,MAGENT,1
        TEXT    '120@'
        EVEN
        DATA    TEXT,258,17,MAGENT,1
        TEXT    '120@'
        EVEN
        DATA    TEXT,1,286,MAGENT,1
        TEXT    '130@'
        EVEN
        DATA    TEXT,278,17,MAGENT,1
        TEXT    '130@'
        EVEN
        DATA    DELAY,10000
        DATA    TEXT,53,280,MAGENT,1
        TEXT    'SLUT PÅ DEMONSTRATIONEN@'
        EVEN
        DATA    DELAY,15000
        DATA    DTEND
*
        END
```

KOMMANDON FÖR DEMOPROGRAM

Dessa kommandon med parametrar enligt beskrivning används i demonstrationsprogrammets DEMTAB.

Koordinaterna anges i ett rätvinkligt system på skärmen där positiv y-axel är uppåt, positiv x-axel är åt höger och origo (0,0) är beläget i skärmens nedre vänstra hörn. Användaren måste själv kontrollera att koordinaterna håller sig inom skärmens område.

AREA

```
Anrop: DATA  AREA,Sx,Sy,Lx,Ly,C,D

där  (Sx,Sy) är koordinaten för nedre vänstra hörnet
               av arean.
     Lx,Ly   är areans utsträckning i X resp. Y-led.
     C       bestämmer färgen
     D       bestämmer areans läge, NORMAL resp SLANT
```

REKT

```
Anrop: DATA  REKT,Sx,Sy,Lx,Ly,C,D

där  se AREA
```

LINE

```
Anrop: DATA  LINE,Sx,Sy,Ex,Ey,C

där  (Sx,Sy) är linjens startpunkt
     (Ex,Ey) är linjens slutpunkt
     C       bestämmer färgen
```

TEXT

```
Anrop: DATA  TEXT,Sx,Sy,C,Z
       TEXT  'Xxxx xx xxxxxxx xx xxxxx@'
       EVEN

där  (Sx,Sy) är första tecknets nedre vänstra hörn
     C       bestämmer färgen
     Z       bestämmer zoomfaktorn ( 1 - 16 )
     @       markerar slut på texten
```

PATT8

Anrop: DATA PATT8,pat0,pat1,pat2,pat3

där pat0-pat3 är bitmönster som skrivs ut i 8 X 8 format

PATT2

Anrop: DATA PATT2,pat0

där pat0 är bitmönster till linjedragning

DELAY

Anrop: DATA DELAY,time

där time är tiden i millisekunder som fördröjning önskas

APPENDIX 4

Litteraturförteckning

Titel	Författare eller utgivare
1 Raster Graphics Handbook	Conrac Division
2 FOA 2 Rapport A 2538 -51	FOA
3 Principles of interactive computer graphics	Newman & Sproull, McGraw-Hill Inc., 1979

Diverse artiklar ur

Electronics
Electronic Design
Byte
Computer
Instrumentation Technology
Elteknik
Modern Elektronik

Datablad och applikationsrapporter för använda kretsar