Octrooiraad

(12) A **Terinzagelegging** (11) **8502640**

Nederland (19) **NL**

(54) **Front-End systeem.**

(51) Int.Cl⁴.: B41J 5/46, B41J 3/00, G03G 15/00, G03G 15/28.

(71) Aanvrager: Océ-Nederland B.V. te Venlo.

(74) Gem.: Ir. L.L.M. Bleukx c.s.
St. Urbanusweg 102
5914 CC Venlo.

(21) Aanvrage Nr. 8502640.

(22) Ingediend 27 september 1985.

(32) - -

(33) - -

(31) - -

(62) - -

(43) Ter inzage gelegd 1 april 1986.

De aan dit blad gehechte stukken zijn een afdruk van de oorspronkelijk ingediende beschrijving met conclusie(s) en eventuele tekening(en).

Uitvinders: T.H.M. Willems
F.H.J. Tunnissen

Océ-Nederland B.V., te Venlo

Front-End systeem

De uitvinding betreft een Front-end systeem voor het omzetten van grafische data en gecodeerde font-data in een seriële pixel-bit-stroom voor een raster-output-scanner (ROS) teneinde een afgedrukte pagina te verkrijgen volgens de aanhef van conclusie 1.

Dergelijke systemen zijn bekend en worden toegepast bij printers welke met serieel aangeboden data lijnsgewijs een volledige pagina kunnen afdrukken. Een typische vertegenwoordiger van dit soort printers is een laserprinter, waarbij een lichtstraal beeldmatig wordt gemoduleerd en waarbij deze gemoduleerde lichtstraal met behulp van een polygoonspiegel lijnsgewijs wordt afgebogen over een lichtgevoelig oppervlak, zoals een zinkoxide-binderlaag welke op een flexibele band is aangebracht. Op bekende wijze kan hierop een latent beeld worden geschreven met behulp van de gemoduleerde lichtstraal. Dit latente beeld kan op bekende wijze worden ontwikkeld en overgedragen op een ontvangstmateriaal zoals een blad papier.

De grafische data kunnen bijvoorbeeld verkregen zijn door met behulp van een scaninrichting een beeld af te tasten en de gegevens, al of niet via een coderingssysteem, op te slaan in een geheugen.
De gegevens voor een af te drukken pagina worden eerst, bijvoorbeeld met behulp van een grafisch werkstation, bewerkt tot een uiteindelijk gewenste lay-out welke tekst- en grafische data kan bevatten.
Wanneer een dergelijke samengestelde pagina moet worden afgedrukt, worden de al of niet gedecodeerde grafische informatie en de gecodeerde font-data, aan het front-end systeem toegevoerd met informatie omtrent hoogte, breedte en uiteindelijk gewenste plaats. Bovendien wordt aan het front-end de bit-map informatie van de af te drukken karaktertekens toegevoerd.
Nadat in het front-end al deze data in een geheugen zijn opgeslagen is het zaak deze data op de juiste plaats in een pagina-groot geheugen (bit-map geheugen) te zetten, en vervolgens dit pagina-geheugen serieel uit te lezen en de data aan de modulator van een laserprinter toe te voeren.

350 2640

Omdat de huidige laserprinters in staat zijn om met grote snelheid en een hoog oplossend vermogen pagina's af te drukken, zal een front-end systeem in staat moeten zijn de data voor deze pagina's met grote snelheid te verwerken en aan de printer aan te bieden.

Het doel van de uitvinding is dan ook om een front-end systeem te verschaffen dat deze gegevens met grote snelheid kan verwerken.

Dit doel wordt volgens de uitvinding bereikt in een front-end systeem volgens de aanhef doordat de raster-beeld-processor en het raster-beeld geheugen met een tweede bussysteem met elkaar zijn verbonden.

Hiermee wordt bereikt dat de front-end processor, welke dient voor het besturen van het front-end systeem en voor het binnenhalen van gegevens omtrent een samengestelde pagina, onafhankelijk van en tegelijkertijd kan werken met de raster-beeld-processor die voor expandering van gedecodeerde grafische- en font-data en plaatsing daarvan in het raster-beeld-geheugen zorg draagt, evenals voor het uitlezen van dit geheugen, waardoor de printer in real-time van gegevens kan worden voorzien.

Deze en andere voordelen zullen duidelijk worden gemaakt aan de hand van de hierna volgende beschrijving en met behulp van figuren, waarvan:

Fig. 1 een schematische weergave van het front-end systeem volgens de uitvinding is,

Fig. 2 de processen weergeeft die door de raster-beeld-processor kunnen worden uitgevoerd,

Fig. 3 een schematische weergave van een raster-beeld-processor is,

Fig. 4 een schematische weergave van een VME-bus interface is,

Fig. 5 een schematische weergave van een laser scan module interface is,

Fig. 6 een schematische weergave van een centralprocessing unit van de raster-beeld-processor is,

Fig. 7 een schematische weergave van een raster-beeld-bus interface is,

Fig. 8 een schematische weergave van een VME-master interface van het VME-bus interface is,

Fig. 9 de plaatsing van een karakter in het bit-map geheugen voorstelt,

Fig. 10 de resultaten van een aantal mogelijke bewerkingen in het bit-map geheugen weergeeft, en

Fig. 11 een schematische weergave van een raster-beeld-geheugen is.

Fig. 1 is een schematische weergave van een front-end systeem. Hierin is een front-end controller 10 (FEC) met een bedieningsconsole 19 verbonden en tevens met het besturingssysteem van een printer 20. Printer 20 is een raster output scanner, waarbij een lichtstraal beeldmatig wordt gemoduleerd en lijnsgewijs over het oppervlak van een lichtgevoelig element wordt afgebogen. Het lichtgevoelig element wordt loodrecht op de afbuigrichting van de lichtstraal voortbewogen teneinde een totaal beeld rastervormig te kunnen schrijven. Een voorbeeld van een rasteroutput scanner is een laserprinter, waarbij een gemoduleerde laserstraal met behulp van een roterende meervlaksspiegel over het oppervlak van een opgeladen fotogeleider wordt voortbewogen. De fotogeleider wordt hierbij beeldmatig uitbelicht, en het verkregen ladingsbeeld kan op bekende wijze met toner worden ontwikkeld, vervolgens overgedragen op een ontvangstblad en met warmte gefixeerd. De front-end controller 10 bevat een 16-bits microprocessorsysteem met een 68000 microprocessor van Motorola en fungeert in samenwerking met lokale ROM en een deel van een random-acces-geheugen 12 (RAM) als besturingssyteem voor het front-end. In een font-leesgeheugen 13 zijn de bitpatronen van een aantal lettertypen opgeslagen. Het front-end kan via een I/O processor 11, welke eveneens een 16-bits microprocessorsysteem met een 68000 microprocessor van Motorola bevat, met een schijfgeheugen, werkstation, computer en/of clustercontroller worden verbonden. De FEC 10, I/O processor 11, RAM 12 en font-geheugen 13 zijn met elkaar verbonden via een standaard VME-bus 14. Het font-geheugen 13 kan ook als RAM worden uitgevoerd of deel uitmaken van RAM12. Vanuit een schijvengeheugen of floppy-diskgeheugen worden de bitpatronen van de fonts dan in dit RAM geladen.

Een raster image processor 15 (RIP) is eveneens met de VME-bus 14 verbonden. Bovendien is de raster image processor 15 via een raster image bus 17 (RI-bus) met een pagina-groot bit-map geheugen 16, ook raster image memory genoemd (RIM), verbonden. De RIP 15 dient voor het beeldmatig vullen van het bit-map geheugen 16 (RIM) met lettertekens die uit het font-geheugen 13 worden opgehaald en op de juiste plaats in het bit-map geheugen 16 worden weggezet. Bovendien kan de RIP 15 grafische informatie uit geheugen 12 ophalen en eveneens op de gewenste plaatsen in het bit-map geheugen 16 wegzetten. Indien het bit-map geheugen 16 is gevuld kan dit weer via de RIP 15 worden uitge-

lezen, waarbij de uitgelezen gegevens als een seriële pixel-bit stroom via lijn 18 aan de modulator van de laserprinter worden toegevoerd.

Het beeld dat op de fotogeleider wordt geschreven wordt opgebouwd uit pixels van 0,05 x 0,05 mm, zodat voor het afdrukken van een A4 formaat zwart/wit beeld circa 4000 x 6000 pixels nodig zijn. Dientengevolge is bit-map geheugen 16 circa 24 Megabit of 3 Megabyte groot. Bij het uitlezen van het bit-map geheugen 16 is de pixel-bit-rate naar de modulator van de laserprinter via lijn 18, circa 25 Megapixels/sec., zodat een pagina van A4 formaat in circa 1 sec. kan worden geprint.

Alle gegevens omtrent een af te drukken pagina worden via de I/O processor 11 vanaf bijvoorbeeld een werkstation en onder besturing van de FEC 10, in het RAM 12 opgeslagen.
Hiertoe staan verschillende mogelijkheden ter beschikking.
Uitgaande van bijvoorbeeld een A4 pagina die in de "portretmode" moet worden geprint, worden circa 4000 deeltabellen gemaakt, overeenkomende met de circa 4000 scanlijnen die nodig zijn om een pagina te schrijven. In elke deeltabel worden lettercodes van die lettertekens of grafische tekens opgeslagen die op een bepaalde scanlijn hun startpunt hebben. Elke lettercode bevat bovendien gegevens omtrent de X-positie welke het teken op de scanlijn inneemt, gegevens omtrent het font-type alsmede gegevens omtrent de hoogte en de breedte van een bepaald teken. Ook bevat de lettercode gegevens omtrent een basisadres in het font-geheugen 13, waar de bit-representatie van dit teken in 16-bits woorden is opgeslagen. De verzameling van de aldus gevormde lijsten wordt de lijst der primitieven genoemd.

Zijn de gegevens omtrent een af te drukken pagina aldus in het RAM 12 opgeslagen, dan kan gestart worden met het vullen van het bit-map geheugen 16. Daartoe worden de lettercodes uit het RAM 12 één voor een door de RIP 15 opgehaald, en met de bijbehorende bit-representatie uit het font-geheugen 13, geëxpandeerd en op de juiste X en Y-positie in het bit-map geheugen 16 weggezet. Op overeenkomstige wijze worden alle tekens een voor een afgewerkt totdat het bit-map geheugen 16 geheel is gevuld met de pixel-representatie van de af te drukken pagina.
Het is niet noodzakelijk deze deellijsten te vormen. De gegevens omtrent een af te drukken pagina kunnen ook in een willekeurige

volgorde in het RAM 12 worden opgeslagen. Bij het vullen van het bit-mapgeheugen 16 wordt het RAM 12, waarin de primitieven in een willekeurige volgorde zijn geplaatst, achter elkaar uitgelezen, geëxpandeerd en op de bijbehorende plaatsen in het bit-map geheugen 16 geplaatst. Volgens weer een andere mogelijkheid worden alle op een pagina voorkomende tekens slechts één keer opgeslagen en voorzien van gegevens omtrent de verschillende posities die elk teken op de pagina inneemt. Zo worden de gegevens omtrent een veel voorkomend teken, zoals bijvoorbeeld de letter e, slechts één keer opgeslagen in het RAM12 en alle posities die deze letter op de pagina inneemt worden in een aparte tabel toegevoegd.

Gewoonlijk worden eerst de grafische tekens die in het RAM12 of font-geheugen 13 zijn opgeslagen in het bit-map geheugen 16 gezet en daarna pas de lettertekens.

De Raster Image Processor

In Fig. 2 zijn de processen weergegeven die door de RIP 15 kunnen worden uitgevoerd. Na het opstarten van het systeem (stap 24) wordt de RIP 15 geïnitialiseerd (stap 25)(INIT-commando) door een systeemreset of een INIT-commando van de front-end controller 10, waarna het "zelftest" proces (stap 26) wordt gestart. Deze "zelftest" omvat het testen van verschillende RIP functies en bit-map geheugen (RIM) functies. De RIP 15 mag geen toegang tot de VME-bus 14 hebben gedurende de zelftestprocedure, want de FEC 10 heeft de VME-bus 14 nodig voor het testen van het RAM 12 en font-geheugen 13. Wanneer de RIP 15 het "zelftest" programma met succes heeft uitgevoerd, wordt een interrupt signaal aan de FEC 10 doorgegeven, en de RIP 15 gaat in de wachtstand (stap 27). Wanneer het zelftestprogramma een fout heeft gedetecteerd gaat de RIP 15 ook in de wachtstand (stand 27), maar er wordt geen interrupt naar de FEC 10 gegenereerd. Op deze wijze weet de front-end controller 10 dat er een fout in de RIP 15 "zelftest" is geconstateerd. Een fout wordt tevens door het oplichten van een LED aan de bedienaar gesignaleerd.

Op een "RIP-diagnose" commando van de FEC 10 naar de RIP 15, start het RIP-diagnose proces. De RIP 15 voert een aantal interne testen uit en ook een aantal testen op het RIM 16. De resultaten van deze testen worden opgeslagen in het RAM 12 en kunnen worden doorgegeven aan, en zichtbaar gemaakt op bedieningsconsole 19. Het RIP-

diagnose proces test ook de VME-interfaces. Het RIP-diagnose proces heeft een meer uitgebreid karakter, terwijl het zelftest proces een meer functionele hardwaretest uitvoert. Na het beëindigen van het RIP-diagnose proces, slaat de RIP 15 status informatie op in het RAM 12, genereert een interrupt naar de FEC 10 en geeft de VME-bus 14 vrij.

Nadat de RIP 15 een "vul bitmap" commando ontvangt van de FEC 10, controleert de RIP 15 het datatransport op de VME-bus 14. De RIP 15 krijgt zo toegang tot het RAM 12, welke de primitieven bevat van de pagina die geprint moet worden.

De RIP 15 expandeert de primitievenlijst met behulp van de pixelrepresentaties van de fonts en zet deze weg in het bit-map geheugen 16 (RIM). De RIP 15 heeft toegang tot het RIM 16 via de RI-bus 17. Het RIM 16 bevat bovendien modificatie logica, welke de RIP 15 ondersteunt bij het uitvoeren van verschillende rekenkundige bewerkingen op data voor het bit-map geheugen 16, zoals bijvoorbeeld AND-, OR- en INVERT-operaties. Na het vullen van de bit-map (stap 28), slaat de RIP 15 status informatie op in het RAM 12, genereert een interrupt naar de FEC 10 en geeft de VME-bus 14 vrij.

De FEC 10 genereert vervolgens een "lees bitmap"-commando en de RIP 15 zal wachten op een pagina-synchronisatie signaal dat van de laserprinter afkomstig is via een besturingsinterface. Na deze pagina-synchronisatie start de RIP 15 met het uitlezen van het RIM 16 (stap 29) en genereert een seriële pixel-bit-stroom welke via een video interface aan de modulator van de laserprinter wordt toegevoerd. Na het beëindigen van het bit-map leesproces (stap 29) slaat de RIP 15 weer status gegevens op in het RAM 12, genereert een interrupt naar de FEC 10 en geeft vervolgens de VME-bus 14 vrij.

De RIP 15 (Fig. 3) is opgebouwd rond een intern bussysteem, de raster image processor bus 46 (RIP-bus), welke een synchrone bus is en ingericht om uitsluitend 16-bits woorden te transporteren. De RIP-bus 46 bevat datalijnen 47, adres- en controlelijnen 48 en conditielijnen 49. Via een VME-bus interface 41 is de RIP-bus met de VME-bus 14 verbonden en via een RI-bus interface 45 met de RI-bus 17. Deze RI-bus 17 bevat onder andere data-en adreslijnen 58, een busy-lijn 57, een RI-bus-address-available lijn 56, een klok-lijn 54 en modificatielijnen. Bovendien is de RIP-bus 46 verbonden met een laserscanmodule interface 44 (LSM-interface) en de eigenlijke Central

Processing Unit 43 (CPU) van de RIP 15. Het laserscanmodule LSM-interface 44 is verbonden met lijnen afkomstig van de printer, zoals een "start of scan" lijn 52 (SOS), waarover een synchronisatie signaal wordt aangevoerd om het begin van een te printen lijn aan te geven, een burst-lijn 53, waarover een signaal dat overeenkomt met de gewenste pixelfrequentie wordt toegevoerd en een videolijn 18, waarover de seriële pixel-bit-stroom bij het uitlezen van het bit-map geheugen 16 aan de modulator van de laserprinter wordt toegevoerd. De adres- en controlelijnen 48 en de conditielijnen 49 zijn ook nog met een pagina-synchronisatie interface 42 verbonden. Over lijn 50 wordt een "page-available" signaal (PAV) aan de besturingsinrichting van de printer toegevoerd, dat aangeeft dat een pagina in het bit-map geheugen 16 volledig is geformateerd en dat de RIP 15 met een "start-of-page" (SOP) via lijn 51, afkomstig van de besturingsinrichting van de printer, aan de uitlezing van het bit-map geheugen 16 kan beginnen.

VME-bus interface

In Fig. 4 is de VME-bus interface 41 schematisch nader weergegeven. Een master-interface 100, een slave-interface 101 en een interruptor 102 zijn met de VME-bus 14 verbonden. De datalijnen 47 van de RIP-bus 46 zijn met het master-interface 100 verbonden. De adres- en controlelijnen 48, evenals de conditielijnen 49 van de RIP-bus 46 zijn aangesloten op het master-interface 100, het slave-interface 101 en de interruptor 102. De VME-bus interface 41 heeft als taak om de RIP 15 af te schermen van de a-synchrone VME-bus 14. Het VME-master-interface 100 (VME-MI) bevat een intern besturingssysteem waarmee de aanwezig buffers en registers worden gestuurd en via deze interface kunnen accesscycles op de VME-bus 14 worden uitgevoerd. In de besturing is programmeerbare logica toegepast. Ook het slave-interface 101 en de interruptor 102 zijn voor de besturing van programmeerbare logica voorzien.

Het VME-MI 100 (Fig. 8) omvat ook data-transfer functies zoals data base-master (DTB master) en data base requestor (DTB requestor). Om de gewenste snelheid in het data transport te bereiken zijn in dit VME-MI 100 enkele extra functies toegevoegd.

De eerste functie is een adres up/down teller, gevormd door een

adres-hoog teller 132 en adres-laag teller 133. Bij het laden van het RIM 16 met de bit representaties van de verschillende lettertekens of grafische tekens die elk afzonderlijk met opeenvolgende adressen in het font-geheugen 13 of het RAM 12 zijn opgeslagen, worden de tellers 132 en 133 voor elk teken vóóringesteld met het basisadres van dat teken in bijvoorbeeld het RAM 12. Via buffer 134 en de VME-adresbus 141 van VME-bus 14 wordt dit basisadres aan het RAM 12 toegevoerd, en het eerste 16-bits woord op die betreffende geheugenplaats wordt via de VME-databus 142 van de VME-bus 14 aan een bi-directionele buffer 135 toegevoerd en vervolgens op de juiste plaats in het RIM 16 geplaatst. Het volgende adres voor het RAM 12 wordt gegenereerd door de teller 133 met één te verhogen en het tweede 16-bits woord wordt via het VME-MI 100 aan het RIM 16 toegevoerd. Op overeenkomstige wijze worden alle bij een bepaald teken behorende adressen gegenereerd, net zo lang tot het teken volledig in het RIM 16 is geschreven.

Op deze wijze wordt bereikt dat de CPU 43 per teken slechts één keer een basisadres hoeft te genereren, zodat tijdens het laden andere functies uitgevoerd kunnen worden, bijvoorbeeld pixel bewerkingen, RIM-adres bepaling, enz.

Nadat een teken is afgewerkt, wordt een nieuw basisadres voor een volgend teken aan de tellers 132 en 133 toegevoerd en de boven-beschreven cyclus wordt herhaald.

De tweede functie, de spiegelfunctie, wordt uitgevoerd met een spiegelschakeling 136, welke is opgebouwd met programmeerbare logica, zoals FPLA's of PAL's, en welke kan worden toegepast wanneer tekens 180^{o} geroteerd in het RIM 16 moeten worden gezet. De spiegelschakeling 136 verwisselt van een 16-bits woord bit-0 met bit-15, bit-1 met bit-14, bit-2 met bit-13, enz.

De CPU 43 genereert nu niet het basisadres, maar berekent uit gegevens omtrent lengte en breedte en het basisadres van een teken het hoogst voorkomende adres voor dat bepaalde teken in het RAM 12. Dit hoogste adres wordt in de tellers 132 en 133 geladen terwijl de tellers tevens door VME-MI controller 130 worden omgeschakeld tot down-tellers. Na elk geheugen access van RAM 12 wordt de inhoud van de teller 133 met één verlaagd en de 16-bits woorden uit RAM 12 worden in spiegelschakeling 136 gespiegeld en via het data-in register 137 in

het RIM 16 geplaatst. Deze cycli worden voortgezet totdat het oorspronkelijke basisadres van het teken is bereikt.
In het VME-MI 100 is ook een data-uit register 138 via datalijnen 47 met de RIP-bus 46 verbonden, om aldus data naar bijvoorbeeld de FEC 10 of naar RAM 12 te voeren.

De VME-MI controller 130 is via controlelijnen 48 en conditielijnen 49 met de RIP-bus 46 verbonden en bovendien via een buffer 131 met adres-, data-, en controlelijnen 139 en bus-arbitragelijnen 140 van de VME-bus 14.

De CPU 43 kan verschillende toestanden in het VME-MI 100 aanroepen, zoals "release-bus", "multiple access", "single access" en "change". Voordat de VME-MI 100 in de single-of multiple access toestand kan overgaan, moeten eerst nog de volgende gegevens worden gespecificeerd: lezen of schrijven, normaal of gespiegeld, het gewenste adres en de te verwerken data. Deze specificaties kunnen alleen veranderd worden tijdens de "release-bus" toestand en tijdens "change" toestand. De te verwerken data kan echter steeds worden gewijzigd. Dit wordt ook aangegeven door een "CHANGE ACKNOWLEDGE" lijn. Het register dat de gelezen data van de VME-bus 14 bevat kan steeds worden uitgelezen wanneer een "REGISTER FULL"-lijn actief is.

Na het aanroepen van een "release-bus" toestand zal de VME-MI 100 de VME-bus 14 vrijgeven. Dit betekent dat de VME-bus drivers gedisabled worden en een BBSY-signaal van de VME-bus inactief wordt gemaakt. Het vrijgeven van de VME-bus 14 kan slechts dan geschieden wanneer de laatste access cycle volledig is afgewerkt. Een "CHANGE ACKNOWLEDGE"-signaal geeft aan dat de "release-bus" toestand is ingetreden. In deze toestand van het interface kan geen access op de VME-bus 14 plaatsvinden. Na een "change" aanvrage wordt het VME-MI 100 geïnstrueerd bezit te nemen van VME-bus 14, indien dit nog niet het geval was. Dit wordt gerealiseerd met de bus-arbitrage lijnen 140. Via de "CHANGE ACKNOWLEDGE"-lijn wordt aangegeven dat de "change" toestand is bereikt. Een access op de VME-bus 14 kan dan plaatsvinden. Ook tijdens de "change" toestand kunnen de inhouden van de adres- en dataregisters worden veranderd. Met "change" wordt een mogelijkheid gegeven om accesses op de VME-bus tijdelijk te stoppen zonder dat de VME-bus wordt vrijgegeven. Een single access op de VME-bus kan gestart

worden door het aanroepen van een "single cycle" toestand. Wanneer de voorgaande toestand een "release-bus" toestand was, wordt pas bezit genomen van de VME-bus via een overeenkomstig actief signaal van de arbritrage logica. Hierna kan pas een woord access worden uitgevoerd op de VME-bus.

Een lees/schrijf indicator beslist of een lees- of een schrijfcyclus moet worden uitgevoerd.

Een leescyclus betekent dat data van de VME-bus 14 worden ingeklokt in het data-in register 137 via de spiegelschakeling 136, welke laatste geactiveerd kan worden met behulp van een normaal/gespiegeld indicator. Wanneer data in het data-in register 137 worden ingeklokt, wordt een REGISTER-FULL vlag gezet om aan CPU 43 te signaleren dat de transfer van data is afgelopen en dat de data zijn binnengekomen in het genoemde register. De REGISTER-FULL vlag wordt gezet op het moment dat de data in het data-in register 137 worden gelezen en na deze access wordt de inhoud van de adresteller met één verhoogd. In het geval van een enabled spiegelfunctie, wordt de inhoud van de adresteller met één verlaagd. Wanneer de REGISTER-FULL vlag nog is geactiveerd en data worden gelezen van de VME-bus, wordt de normale VME-cyclus verlengd totdat het data-in register 137 geheel is uitgelezen en nieuwe data in het data-in register 137 zijn ingelezen.

Een schrijfcyclus is in principe hetzelfde als een leescyclus. Het enige verschil is de richting van de datastroom. In een schrijfcyclus worden de data, welke zich in het data-uit register 138 bevinden, getransporteerd naar de VME-bus 14. De spiegelschakeling 136 verandert niets aan de geschreven data. Het data-in register 137 moet al reeds gelezen zijn om de REGISTER-FULL vlag te kunnen clearen.

De "multiple access" toestand vertoont veel overeenkomsten met de "single access" toestand. Een "single-access" is bedoeld voor het lezen en schrijven van commando's van, en status informatie naar de FEC. Een "multiple access" is vooral bedoeld voor het lezen van grafische- en font-data, waarbij door het VME-MI 100 automatisch een volgende access wordt gestart. Het nieuwe adres wordt gegenereerd door de adresteller. De enige actie die hierbij uitgevoerd moet worden is het lezen van het data-in register 137.

De hiervoor beschreven verschillende toestanden worden geselecteerd met de VME-MODE lijnen welke verbonden zijn met enkele

signaallijnen van de CPU 43. De lees/schrijfselector en de normaal/gespiegeld selector zijn eveneens verbonden met dergelijke signaallijnen. De CHANGE-ACKNOWLEDGE en VME-Register-Full signalen zijn afkomstig van de WAIT-lijnen van de CPU 43. Het VME adres wordt opgeslagen in 24-bit tellers132 en 133, de input- en output-data in twee 16-bit registers 137 en 138. "Adres-hoog"- en "adres-laag" van tellers 132 en 133 en het data-out register 138 worden geladen met behulp van register-kloklijnen. Het data-in register 137 kan gelezen worden met behulp van een register enable lijn afkomstig van de CPU 43.

Het LSM-interface

In Fig. 5 is het LSM-interface 44 nader schematisch weergegeven. Bij het uitlezen van het RIM 16 haalt de RIP 15 een 16-bits woord uit dit geheugen op en geeft dit via datalijnen 47 van de RIP-bus 46 door aan register 111. Het besturingsblok 110 geeft over lijn 115 een "load"-signaal af aan schuifregister 112 en de inhoud van register 111 wordt parallel in het schuifregister 112 geladen. De laserprinter geeft burst-pulsen af met een frequentie van circa 24 MHz, die over lijn 53 en via I/O buffer 113 aan het schuifregister 112 en het besturingsblok 110 worden toegevoerd. Met deze pulsen wordt de inhoud van het schuifregister 112 serieel uitgeschoven en via I/O buffer 113 over lijn 18 aan de modulator van de ROS toegevoerd.

De burst-pulsen worden in het besturingsblok 110 aan een 16-teller toegevoerd en wanneer 15 pulsen zijn geteld of tijdens de 16e telpuls, wordt een inmiddels nieuw in register 111 gezet woord, parallel aan schuifregister 112 doorgegeven en uitgeschoven. Vóór de uitschuifoperatie van dit 16-bits woord start, wordt het register 111 geladen met een nieuw 16-bits woord. Er wordt een "EMPTY"-vlag gezet wanneer data in het schuifregister 112 zijn gezet en nieuwe data in het register 111 mogen worden geschreven. De "EMPTY"-vlag is aangesloten op een "wachtlijn" van de CPU 43 van de RIP 15. Op deze wijze wordt achtereenvolgens een hele scanlijn aan de ROS doorgegeven. Het besturingsblok 110 geeft conditiesignalen af aan de CPU 43 over conditielijnen 49 van de RIP-bus 46. Nadat een scanlijn is afgewerkt en vóór een SOS-signaal via lijn 52 vanaf de ROS aan het besturingsblok 110 wordt toegevoerd, wordt het ophalen van data uit het RIM 16 door

de RIP 15 even gestaakt (wait-conditie). Gedurende deze tijd wordt het register 111 gecleared via lijn 114. Op het SOS-signaal wordt de eerder beschreven cyclus van vullen van register 111, doorgeven aan schuifregister 112, uitschuiven enz., weer voor een volgende scanlijn herhaald. Na het laden van een woord in register 111 wordt de "full" status eveneens via conditielijnen 49 aan de CPU 43 doorgegeven, waarbij deze met het ophalen van een nieuw woord wacht totdat de inhoud van het register 111 weer in het schuifregister 112 geladen is. Met behulp van een teller in de CPU 43 wordt na een PAV-signaal het aantal SOS-pulsen geteld, en hiermee kan worden vastgesteld wanneer een pagina volledig aan de ROS is doorgegeven.

De Central Processing Unit

In Fig. 6 is de CPU 43 van de RIP 15 nader schematisch weergegeven. Deze CPU is opgebouwd rond een microprogrammeerbare microprocessor, processor 74, type Am29116, en een bijbehorende address sequencer 70, type 2910A, beide van Advanced Micro Devices.

Op elke klokcyclus wordt de microinstructie die moet worden uitgevoerd in het micro-instructieregister 72 gezet. Deze micro-instructie is afkomstig van de micro-PROM 71 en deze wordt weer geadresseerd met behulp van de address sequencer 70. Elke functie in de processor 74 wordt bestuurd door een deel van de micro-instructie-bits. Deze micro-instructies kunnen worden verdeeld in bits voor de address sequencer 70, de processor 74, de sprong-adres besturings-eenheid 79, de conditieselector 75, de wachtselector 77 en de enables 78.

De volgorde van uitvoering van de micro-instructies die zijn opgeslagen in de micro-PROM 71, wordt eveneens gestuurd door de address sequencer 70. Naast de mogelijkheid van opeenvolgende toegang tot de adressen, kunnen ook conditionele spronginstructies naar elke micro-instructie in het 4096 grote micro-woordbereik van de micro-PROM 71 worden uitgevoerd. Een LIFO-stack voorziet in een micro-subroutine-return-koppeling en inlusmogelijkheden. De stack is negen stappen diep. Voor elke micro-instructie voorziet de address sequencer 70 in een 12-bits adres dat wordt geïnitieerd vanuit een van de vier navolgende bronnen:

- het micro-programma-adresregister (PC), welk gewoonlijk een adres

aangeeft met een adresverhoging van één, ten opzichte van het in behandeling zijnde adres. Echter, wanneer een "wacht" toestand wordt gegenereerd door wacht-selector 77, wordt de PC niet verhoogd.

- een externe ingang, aangesloten op lijnen 92, die zijn data krijgt van het sprong-adres besturingseenheid 79.
- een negen stappen diepe LIFO-stack, welke tijdens een voorafgaande micro-instructie wordt geladen met de inhoud van het micro-programma-adresregister (PC).
- een register/teller, welke de data die gedurende een voorafgaande micro-instructie vanaf een externe ingang zijn geladen, vasthoudt.

De processor 74 is een microprogrammeerbare 16-bit microprocessor, type Am 29116, met een instructieset die geoptimaliseerd is voor grafische toepassingen. De instructieset voor de processor 74 omvat met name single and double operand, rotate-n-bits en rotate & merge.

De processor 74 ontvangt zijn instructies voor het uitvoeren van een operatie van het micro-instructieregister 72 via bus 83 en een instructie-modificatieschakeling 73.

De instructie-ingang wordt ook als data-ingang toegepast voor "immediate"-instructies. Wanneer de "instructie-enable" (IEN) ingang van de processor 74 via lijn 94 wordt geactiveerd, worden de resultaten van de uitgevoerde instructie in de accumulator en het statusregister in de processor 74 vastgehouden. Wanneer een "output enable" (OE) via lijn 95 wordt geactiveerd, worden de datalijnen van de CPU 43 geschakeld als uitgangen en ze bevatten de inhoud van de ALU van processor 74. Omgekeerd, wanneer via lijn 95 de "output enable" inactief wordt gemaakt, fungeert de databus van de CPU 43 als 16-bits ingang, en kunnen gegevens welke op de RIP-bus aanwezig zijn via datalijnen 47 aan de processor 74 worden toegevoerd. Deze gegevens kunnen dan in een intern register worden vastgehouden. De databus van de processor 74 is rechtstreeks met de datalijnen 47 van de RIP-bus verbonden.

Op de "status bus" 87 van de processor 74 is tijdens elke cyclus de status van de ALU beschikbaar (bijvoorbeeld carry, negatief, nul, overflow). De instructie-modificatieschakeling 73 maakt het mogelijk om de in de micro-PROM 71 vastgelegde instructies aan te passen om bij instructies, zoals bijvoorbeeld "rotate n-bits", het aantal bits aan

te geven waarmee moet worden geroteerd. Dit aantal bits wordt dan gespecificeerd via een aantal lijnen (91) van de processor-datalijnen 47.

Wanneer een IEN signaal op de lijn 94 de instructie input van de processor 74 inactief maakt, kunnen dezelfde processor-instructie-bits die op bus 83 aan de processor 74 worden toegevoerd, via bus 84 ook aan de sprong-adres besturingseenheid 79 worden toegevoerd, en zo worden gebruikt om de address sequencer 70 naar een willekeurig ander adres te laten springen. Normaliter ontvangt de eenheid 79 zijn sprongadres uit de inhoud van een register dat via bus 90 met data van datalijnen 47 is gevuld.

De conditieselector 75 bevat een één-uit-acht multiplexer, en de uitgang hiervan is via lijn 89 met de address sequencer 70 verbonden. Eén van de acht mogelijke condities, welke van de conditielijnen 49 van de RIP-bus, of van de processor-status lijnen 87, via status buffer 76 en lijnen 88 aan de ingang van de conditieselector 75 worden aangelegd, kan worden gekozen. De geselecteerde conditie wordt gebruikt door de adress sequencer 70 om de gewenste conditionele instructie uit te voeren. Eventuele nieuwe condities kunnen in de statusbuffer 76 worden geladen door een selectie-enable signaal (SLE) via lijnen 85 aan de status buffer 76 toe te voeren.

De "wachtselector" 77 bevat eveneens een één-uit-acht multiplexer, welke in actieve toestand één van de acht "wacht"-lijnen 97 via lijn 93 met de address sequencer 70 verbindt. Een nul-niveau op een wachtlijn stopt de programmateller van het programma-adres-register. De wachtlijnen zijn verbonden met de conditielijnen van de RIP-bus.

Het enable blok 78 heeft verschillende functies en genereert bovendien alle signalen die nodig zijn voor de besturingslijnen op de RIP-bus. Het voert drie verschillende functies uit:

A. Het genereren van "enables".
   De enable signalen bepalen welke van de data registers die met hun uitgangen aan de RIP-bus zijn verbonden moeten worden geactiveerd. Voor elk register is één enable lijn voorhanden.

B. Het opwekken van registerklokken.
   De kloklijnen bepalen welke dataregisters die met hun ingangen aan

de RIP-bus zijn aangesloten, data moeten inklokken. Er is voor elk register één kloklijn aanwezig.

C. Het opwekken van andere signalen.
De signalen op de signaallijnen worden gebruikt als flags en functieselectoren op de interface modulen die met de RIP-bus zijn verbonden.

Raster Image bus interface

De verbinding tussen de RIP-bus 46 en de RI-bus 17 wordt gevormd door het RI-bus interface 45 (Fig. 7). Deze interface buffert de bidirectionele data, de aan te roepen adressen en de modificatiecode. Het bufferen wordt uitgevoerd met behulp van registers. De registers "data-uit" 120, "adres-laag" 122, "adres-hoog" 123 en het modificatieregister 124 kunnen geladen worden vanuit de RIP-bus 46. Het laden gebeurt onder besturing van adres- en controlelijnen 48 van het enable blok 78 van de CPU 43. Het "data-in" register 121 kan uitgelezen worden onder invloed van besturing met een enable lijn van het enable blok 78. Het "adres-hoog" register 123 bevat de meest significante bits van het adres. Het "adres-laag" register 122 bevat de minst significante bits. Na het laden van het "adres-hoog" register 123 wordt de RI-bus lees/schrijfcyclus automatisch gestart. Dit betekent dat de volgende processen door de controller 125 worden uitgevoerd:
cyclus 1- zet een adres op de RI-bus en activeert RAV,
cyclus 2- zet data-uit op de RI-bus en inactiveert RAV,
cyclus 3- leest de data op de RI-bus in het "data-in" register 121.
Vóór het starten van een RI-bus cyclus, moet de CPU 43 testen of de RI-bus-busy-lijn 57 inactief is. Deze busy-lijn 57 is verbonden met een van de wachtlijnen van de CPU 43.

De Raster Image bus (RI-bus)

De RI-bus 17 verbindt de RIP 15 met het bit-map geheugen 16 (RIM) en is opgebouwd uit 64 lijnen. Het omvat een 32-bits brede gemultiplexte adres/data-bus. Op de RI-bus 17 fungeert de RIP 15 als master. Het RIM 16, dat een of meer RIM borden omvat neemt zelf geen initiatief op de bus. Verder kunnen op de RI-bus 17 nog RI-bus-DMA devices aangesloten worden, die bij de RIP 15 een verzoek in kunnen dienen om beschikking te krijgen over de bus.

De RI-bus 17 is een synchrone bus. Door de RIP 15 wordt een

kloksignaal (BCLK) aan de RI-bus aangeboden. Alle acties op de bus worden op de flanken van de bifase klok uitgevoerd. Zo vinden alle akties van de RIP op de opgaande flank, en alle akties van het RIM op de neergaande flank van het kloksignaal (BCLK) plaats. De overige toestanden op de RI-bus 17 kunnen beschreven worden met behulp van drie signaalniveau's, hoog, laag en hoog-impedant (tri-state). Alle veranderingen in signaalniveau's vinden plaats nadat ze zijn ingeleid door een actieve flank van de bifase klok. Er zijn aldus drie groepen signalen: de kloksignalen, de adres/data signalen en de overige signalen.

De signalen die op de bus voorkomen zijn hierna gedefinieerd:

- BCLK: dit is een symmetrische klok, die door de RIP op de RI-bus wordt aangeboden.
- RAD 00 .. RAD 31 (RIP adres/data lijnen): dit is een gemultiplexed adres/data pad, dat door alle bus-devices aangestuurd wordt met tri-state drivers. Alle lijnen zijn "hoog" actief.
- RMC 0 .. RMC 3 (RIM modify code): op deze lijnen wordt door de RIP of een DMA device een code aangeboden aan het RIM bord. Deze code specificeert de "modify" functie die plaats vindt tijdens de logische operatie die op het RIM bord wordt uitgevoerd op de inhoud van het geadresseerde geheugenwoord. Ook deze signalen zijn tri-state uitgevoerd.
- RROFF (RI-bus refresh-off): met dit signaal wordt aangegeven dat de RIM borden de refresh uit kunnen schakelen om een minimale cyclustijd te kunnen halen. Om verlies van data te voorkomen wordt een speciale adresseringsvolgorde tussen de RIM borden en de RIP aangehouden.
- RBR 0, RBR 1 (RI-bus bus-request): met deze open collector signalen kunnen twee DMA devices aan de bus-arbiter om toegang tot de bus vragen. De devices hebben een verschillende prioriteit.
- RBG (RI-bus bus-grant): met deze lijn geeft de bus-arbiter aan dat de bus beschikbaar is voor het aanvragend device met de hoogste prioriteit.
- RBUSY (RI-bus bus-busy): met dit open collector signaal kan een geadresseerd RIM bord aangegeven, dat het bord gedurende een bepaalde tijd niet in staat is om een nieuwe buscyclus te verwerken.

- RAV (RI-bus address valid): dit tri-state signaal dat laag actief is, geeft aan dat op de RI-bus een geldig adres staat.

De RI-bus is uitgevoerd met een 32 bits breed data- en adres-pad dat gemultiplexed wordt op RAD 00 .. RAD 31. De toewijzing van deze lijnen is als volgt:

A 24 - D 16 : in deze situatie worden de adreslijnen RA00 .. RA23 gebruikt. De lijnen RAD 24 .. RAD 31 zijn dan don't care. Voor de datalijnen wordt RAD00 .. RAD15 gebruikt. De lijnen RAD16 .. RAD31 zijn op dat moment don't care. Datatransport vindt zo op basis van 16-bits woorden plaats en de adressen zijn 24-bits breed.

Een andere mogelijkheid om de 32 data- en adreslijnen te gebruiken is:

A24 - D16 - D16: Deze situatie is hetzelfde als de A24 - D16 situatie voor wat betreft de adreslijnen. Door het toevoegen van een tweede bord in dezelfde adresruimte, waarbij de data via de lijnen RAD16 .. RAD31 over de bus lopen, kan met twee borden die intern 16-bits breed zijn, een 32-bits brede databus gecreëerd worden. Op een RIM-bord kan ingesteld worden over welk gedeelte van de adres/databus de data getransporteerd worden.

Met behulp van de signalen RBR0, RBR1 en RBG wordt de toegang tot de RI-bus geregeld tussen de RIP en eventuele DMA devices. Deze arbitrage gaat volledig buiten het RIM 16 om.

Iedere cyclus op de bus bestaat uit een WRITE/READ cyclus. Indien de bus vrij is (RBUSY niet actief) kan de RIP een adres (ADR[n]) op de bus op lijnen (RAD00 .. RAD23) zetten. Dit gebeurt samen met het aanbieden van een RAV-signaal en een RI-bus modify code (RMcode) via lijnen RMC00 .. RMC03. Na het adres biedt de RIP zijn data (DATA0[n]) aan op de bus op lijnen (RAD00 .. RAD15).
Het door ADR[n] geadresseerde RIM-bord maakt het RBUSY signaal actief. Vervolgens gaat de RIP van de RI-bus af om het door ADR[n-1] geadresseerde RIMbord de mogelijkheid te geven DATAI[n-1] op de bus te zetten, zodat de RIP deze data kan inlezen. Twee opeenvolgende WRITE/READ cycli worden hierdoor als het ware in elkaar geschoven. Dit wordt nog verder geoptimaliseerd door de tijd, die de RIP nodig heeft om te beslissen of RBUSY inactief is geworden, samen te laten vallen met de laatste verwerkingsfase van het door ADR[n] geadresseerde RIMbord in de lopende cyclus. Dit is gerealiseerd doordat het RIMbord

RBUSY al inactief maakt, voordat het RIMbord al helemaal gereed is maar waarbij al zeker vast staat dat dit gereed zal zijn wanneer de RIP dit geconstateerd kan hebben. De eerste cyclus bevat dus ongeldige data en er is tevens een extra cyclus nodig om de laatste data uit het RIM op te halen.

Op bovenbeschreven wijze wordt een minimale cyclustijd op de bus gehaald. Met de minimale cyclustijd wordt bedoeld die timing volgorde van bustoestanden, waardoor een maximale transfer rate op de bus verkregen wordt.

Door de refresh van het RIM kan het voorkomen, dat een RIMbord niet in staat is de minimale cyclustijd te halen. De RIMborden maken dit kenbaar met behulp van het RBUSY signaal. Door dit RBUSY signaal met een bepaald aantal klokperiodes (BCLK) te verlengen, stelt de RIP zijn volgende access op het geheugen uit met een geheel aantal klokperiodes.

Eenzelfde situatie kan optreden indien de RIP nog niet klaar is met een bepaalde taak. De RIP geeft dit aan op de bus door het RAVsignaal een geheel aantal klokperiodes uit te stellen.

Raster image memory

Het RIM 16 (Fig. 11) omvat een 24 Mbit dynamisch geheugen 220, georganiseerd in 16-bits woorden en wordt als een paginagroot bitmap geheugen gebruikt. Elke geheugenplaats in het geheugen 220 stemt overeen met één exacte plaats op de uiteindelijke afgedrukte pagina. Het RIM 16 is via de RI-bus 17 met de RIP 15 verbonden en wordt door de RIP 15 gevuld met geëxpandeerde font-data en grafische data. Een belangrijk proces dat in het RIM 16 plaatsvindt is het modify proces, welk wordt uitgevoerd op een geadresseerd woord. Het modify proces omvat 16 verschillende logische operaties welke kunnen worden toegepast op de inkomende data en de alreeds op een bepaald adres aanwezige data. Eén bepaalde modify functie wordt geselecteerd door een RIM modify code op de lijnen RMC0 ... RMC3 223 van de RI-bus 17 aan te bieden.

Deze modify code wordt in het RMC register 222 gezet en toegevoerd aan de logische rekeneenheid 223 (ALU) die is opgebouwd met programmeerbare logica. De nieuwe data (NT) wordt via het DATA0 register 227 over datalijnen 225 aan de ALU 223 toegevoerd, terwijl de oude, reeds in het geheugen 220 aanwezige data (OD) via data out-

putlijnen 226 aan de ALU 223 wordt toegevoerd. Het resultaat van de bewerking (MD) in de ALU 223 wordt via lijnen 224 in het geheugen 220 geschreven.

In onderstaande tabel zijn enkele modify functies met de erbij behorende RM codes en de overeenkomstige logische functies weergegeven.

Modify functie	RMC 3	2	1	0	Logische functie
WRITE	0	0	0	0	ND
PAINT	0	0	0	1	ND.OR.OD
MASK	0	0	1	0	ND.AND.OD
ERASE	0	0	1	1	$\overline{ND}$.AND.OD
INVERT	0	1	0	0	$\overline{ND}$
INV.PAINT	0	1	0	1	ND.EXOR.OD
NOP	0	1	1	0	OD
CLEAR	0	1	1	1	ZERO
SET	1	X	X	X	ONE

ND = nieuwe data

OD = oude data

Omdat de RI-bus 17 een gemultiplexte bus is, moeten de afzonderlijke adressen en de data in registers worden ingeklokt. De RI-bus 17 is daartoe met een address/data busbuffer 228 verbonden en bij het aanbieden van een adres op de RI-bus 17 wordt dit adres via buffer 228 aan het address register 229 toegevoerd. Bij de aanbieding van data (een klokperiode later), worden deze data via buffers 228 in het DATO register 227 opgeslagen. Het data-in register 230 ("in" voor de RIP maar "uit" voor het RIM) is toegevoegd om de uit het geheugen 220 afkomstige data, welke behoren bij het vorig aangeboden adres, op de RI-bus 17 te kunnen zetten.

De besturing van de RIM 16 wordt verzorgd door het geheugenbesturingscircuit 231. Het geheugen besturingscircuit 231 bevat een bus state sequencer om op een RAV signaal een aantal akties te starten, welk bestaan uit het inklokken van een adres, het inklokken van

erbij behorende data, het inklokken van de modify code en het op de RI-bus 17 zetten van data behorende bij het voorgaande adres. Bovendien bevat het geheugenbesturingscircuit 231 een memory state sequencer, welke gesynchroniseerd is met de bus state sequencer. De memory state sequencer kan worden geïnitieerd door een refresh request of door een buscyclus voor een geheugen access. Wanneer een refresh cyclus wordt uitgevoerd, moet de eerstvolgende buscyclus worden opgeschort. Het circuit 231 is met programmeerbare logica geïmplementeerd.

Het geheugen 220 is opgebouwd met 256 K dynamische geheugenchips en is georganiseerd in zes "banks" van 256 K woorden van 16 bits. Bank selectie vindt plaats door middel van decodering van de adreslijnen A18, A19 en A20 in de address multiplexer 232. Het adresseren van een geheugenplaats in één bank vindt plaats met behulp van adreslijnen A0 - A7 en A16 en het genereren van een row-address strobe (RAS) vanuit het besturingscircuit 231, en vervolgens worden adreslijnen A8, A15 en A17 aan de geheugenadreslijnen via de address multiplexer 232 toegevoerd, en een column address strobe (CAS) wordt tevens door het besturingscircuit 231 gegenereerd.

Omdat dynamische geheugens zijn toegepast moeten alle geheugenplaatsen van het geheugen 220 tenminste één keer per 4 msec. een refresh ondergaan. Dit geschiedt door periodiek een "RAS-only" cyclus toe te voegen. Tijdens deze cyclus wordt aan alle banken een row-address aangeboden. Het refresh address in een rij wordt afgeleid met behulp van een 9-bit teller welke na elke refresh cyclus met één wordt verhoogd.

Door activering van de RROFF-lijn van RI-bus 17 wordt de normale refresh cyclus onderbroken en de RIP 15 zorgt ervoor dat er aan de minimale cyclustijd van de eerstvolgende cyclus wordt voldaan. Niet-geadresseerde banks van het geheugen gebruiken dan het adres op de RI-bus 17 om een refresh uit te voeren. Op de geadresseerde bank vindt refresh plaats door een access op het geselecteerde adres.

Bij het uitlezen van het RIM 16 zal, indien slechts één afdruk van een pagina moet worden gemaakt, op de RI-bus 17 de "CLEAR" modify code worden gezet omdat het RIM 16 na uitlezen geheel met nullen moet zijn gevuld. Indien de pagina moet worden bewaard om nog eens te worden geprint, zal op de RI-bus 17 de "NOP" modify code worden gezet.

Grafische instructies

De bitmap vuller 28 (Fig. 2) is ingericht om verschillende tekst- en grafische instructies uit te voeren, zoals CHAR, MCHAR, VLINE, HLINE, BLOCK FILL, AREAFILL, LINE en CIRCLE.

Al deze instructies, welke zijn opgeslagen in de micro-PROM 71 (Fig.6), zijn uitgevoerd als micro-instructies voor de CPU 43. De algorithmen voor deze instructies zijn zodanig geïmplementeerd dat een zo groot mogelijke bitmap vulsnelheid wordt verkregen.

CHAR: is een instructie voor het plaatsen van een teken op de juiste plaats in het bit-map geheugen 16. Aangezien de woordgrenzen van een teken meestal niet overeenkomen met de woordgrenzen van het bit-map geheugen is een verplaatsing nodig (zie Fig. 9). In het font-geheugen 13 is de bitmap representatie 200 van een teken 201 in 16-bits woorden opgeslagen. Een teken omvat gewoonlijk een aantal 16-bits woorden, waarvan enkele zijn aangegeven met 203, 204 en 205. Het hoekpunt 202 van het teken 201 wordt hier als voorbeeld als referentiepunt aangenomen, en het eerste 16-bits woord 203 bevat 16-bits waarvan het eerste bit met "0" is aangegeven, en het laatste bit met "F". De bit representatie van het eerste woord is aldus: 0000 0000 0001 1111.

Bij het plaatsen van dit teken 201 op de gewenste y-positie in het bit-map geheugen 16 zal de woordgrens 207 van het bit-mapgeheugen over het algemeen slechts zelden met de woordgrens 0' van het teken 201 overeenkomen. De uit te voeren operatie zal dus overeenkomen met het verschuiven van de bitmap representatie van het teken 201 over een aantal (n)bits, in de figuur aangegeven met Δ y.
Op micro-instructie niveau zijn hiervoor de volgende stappen uit te voeren:
rotate: van bit 0 tot bit F over Δ y (n: = Δ y)
merge : mask = 1 rotate
mask = 0 non-rotate

MCHAR: is een instructie voor het plaatsen van een teken in het bitmap geheugen 16 in gespiegelde vorm. Het uitlezen van de bit-map representatie van het teken door de VME-bus interface gebeurt in omgekeerde volgorde. Het verplaatsen van de woordgrenzen van het teken in het bit-map geheugen 16 geschiedt op identieke wijze als beschreven

bij CHAR. De schakeling voor het spiegelen is eveneens in de VME-bus interface ondergebracht.

Voor het schrijven van tekens in het bit-map geheugen 16 bezit het front-end een aantal overlay mogelijkheden (Fig. 10). Aan de hand van een letter V (210) zijn in Fig.10 deze mogelijkheden schematisch weergegeven. Een arcering 211 betekent dat de inhoud van het RIM 16 onveranderd is. Aangenomen wordt dat een "0" in het RIM "wit" oplevert en een "1" in het RIM "zwart" oplevert.

WRITE: de bestaande inhoud van het RIM 16 wordt "0" gemaakt en de bit- map representatie van een teken wordt met enen geschreven (212).

INVERT: de bestaande inhoud van het RIM wordt "1" gemaakt en de bit-map representatie van een teken wordt met nullen hierin geschreven (216).

PAINT: de inhoud van het RIM wordt niet gewist en de enen van het teken ondergaan met de inhoud van het RIM een "OR" functie (213).

MASK: de inhoud van het RIM wordt "0" gemaakt op die plaatsen waar het teken nullen bevat, en daar waar het teken enen bevat blijft de inhoud van het RIM gehandhaafd (214).

ERASE: de inhoud van het RIM wordt "0" gemaakt op die plaatsen waar het teken enen bevat, en daar waar het teken nullen bevat blijft de inhoud van het RIM gehandhaafd (215).

INVERTING-PAINT: de inhoud van het RIM blijft gehandhaafd waar het teken een "0" bevat en waar het teken een "1" bevat wordt de inhoud van het RIM geïnverteerd.

Een volgende instructie die het front-end uit kan voeren is:

BLOCK-FILL: het vullen van een vooraf bepaald gebied met een regelmatig patroon zoals blokken, arceringen, enz. De patronen zijn periodiek en de te vullen hoogte wordt uitgedrukt in hele woorden, terwijl ze voortzetbaar zijn in de X- en Y-richting.

AREA-FILL: het opvullen van een door lijnen omgeven gebied, met een repeterend patroon. Er zijn twee mogelijkheden, nl. uitgaande van grenzen die in vier richtingen zijn gesloten of grenzen die in acht richtingen zijn gesloten. Door vooraf de begrenzing van het gebied in het RIM in te lezen, en door de speciale acties die nader bij het RIM zijn beschreven, kan AREA-FILL op elk gebied worden uitgevoerd, ook indien de begrenzingen niet met de woordgrenzen overeenstemmen.

Voor het maken van lijnen, cirkels en cirkelbogen wordt gebruik

gemaakt van de meet- en regelmethoden, gebaseerd op het algorithme van Bresenham. Hierbij wordt uitgegaan van het theoretisch verloop van de lijnen en voor elke scanlijn wordt dat punt gekozen, dat deze gewenste lijn het dichtst benaderd. Het bekende algorithme, is bijvoorbeeld beschreven in ACM Transactions on Graphics, Vol. 1, no. 4, oct. 1972, p. 259-279 door Robert F. Sproull, met als titel "Using programm transformations to derive line-drawing algorithms".

Voor het printen van open of gesloten lijnen of lijnstukken welke een breedte hebben groter dan één pixel is een speciaal algorithme toegepast. Er wordt steeds uitgegaan van lijnstukken welke van ronde aansluitpunten worden voorzien, waarmee een ideale aanpassing aan andere lijnstukken ontstaat. Uitgaande van het cirkelalgorithme van Bresenham kan een ronde "plotterpunt" worden gegenereerd met een gewenste dikte die overeenstemt met een oneven aantal rasterpunten. De pixel representatie van deze "plotterpunt" wordt in het RAM 12 opgeslagen en kan verder als een karakter worden behandeld. De bit-map van een punt wordt evenals de bit-map van een letterteken gekenmerkt met hoogte- en breedte informatie van de omhullende rechthoek terwijl een hoekpunt van deze rechthoek het referentiepunt vormt dat gebruikt wordt om deze punt op de juiste plaats in het RIM 16 te plaatsen.

Het beginpunt van een lijnstuk moet het centrumpunt van de "plotterpunt" vormen en plaatsing in het RIM 16 geschiedt door op de positie van het centrumpunt een verschuiving over de halve breedte en ook over de halve hoogte van de "plotterpunt" toe te passen, waarmee het startpunt en dus ook het referentiepunt van de "plotterpunt" wordt verkregen.

Door nu met behulp van het Bresenham lijnalgorithme steeds het nieuwe centrumpunt voor de "plotterpunt" te berekenen en de bij de positie van dit centrumpunt behorende plotterpunt in het RIM 16 in te vullen kan een lijnstuk met een bepaalde dikte worden verkregen.

De uitvinding is niet beperkt tot de beschreven uitvoeringsvorm en vele wijzigingen zijn voor een vakman denkbaar. Echter al deze uitvoeringsvormen zullen vallen onder de hierna volgende conclusies.

CONCLUSIES

1. Front-end systeem voor het omzetten van grafische data en gecodeerde font-data in een seriële pixel-bit-stroom voor een raster output scanner (ROS) teneinde een afgedrukte pagina te verkrijgen, omvattende:
- een front-end controller(10), voor het besturen van het front-end systeem,
- een input/output processor (11) die verbonden kan worden met een computer of met een data-netwerk,
- een geheugen (12, 13) voor de opslag van via de input/output processor (11) toegevoerde grafische data en gecodeerde font-data voor een samen te stellen pagina en voor de opslag van programmagegevens voor de front-end controller (10),
- een raster-beeld-geheugen (16) (RIM) voor de opslag van de bitrepresentatie van een hele af te drukken pagina, waarbij elk geheugenelement overeenkomt met een plaats op de af te drukken pagina,
- een raster-beeld-processor (15) (RIP) voor het expanderen van de data voor een samen te stellen pagina en de plaatsing van de bitrepresentatie hiervan in het raster-beeld-geheugen (16) en voor het omzetten van deze opgeslagen bit-representatie van de af te drukken pagina in een seriële pixel-bit-stroom voor toevoer aan de raster-output-scanner (20),

waarbij de front-end controller (10), de input/output processor (11), het geheugen (12, 13) en de raster-beeld-processor (15) via een eerste bussysteem (14) met elkaar zijn verbonden, met het kenmerk, dat de raster-beeld-processor (15) en het raster-beeld-geheugen (16) met een tweede bussysteem (17) (RI-bus) met elkaar zijn verbonden.

2. Front-end systeem volgens conclusie 1, met het kenmerk, dat het tweede bussysteem (17) een synchroon bussysteem is.

3. Front-end systeem volgens conclusie 1 of 2, met het kenmerk, dat het tweede bussysteem (17) een aantal gemultiplexte adres/datalijnen omvat.

4. Front-end systeem volgens conclusie 1, 2 of 3, met het kenmerk, dat het tweede bussysteem (17) een refreshlijn omvat waarmee de refresh van het raster-beeld-geheugen (16) uitgeschakeld kan worden.

5. Werkwijze van een front-end systeem volgens een der voorgaande conclusies, met het kenmerk, dat door de raster-beeld-processor (15) op het tweede bussysteem (17) een buscyclus wordt uitgevoerd welke omvat:

- het op de bus zetten van een adres n,
- het vervolgens op de bus zetten van de bij dat adres n behorende data m
- en het tenslotte uit het raster-beeld-geheugen (16) ophalen van data die bij het voorgaande adres (n-1) behoren.

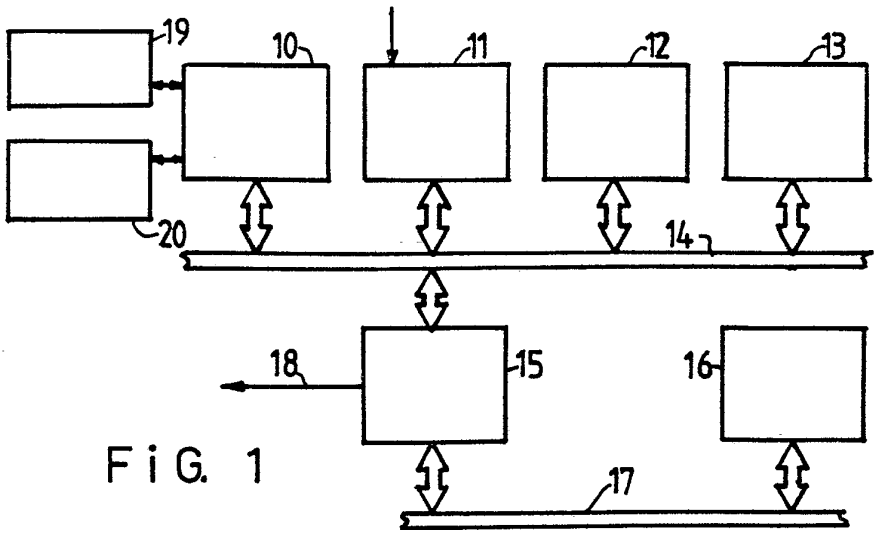

Fig. 1

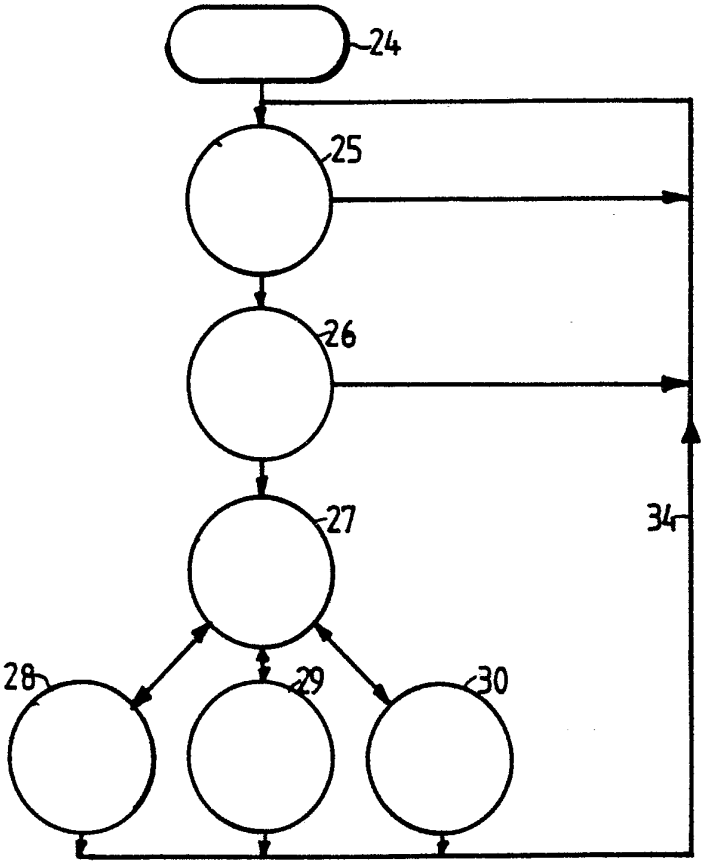

Fig. 2

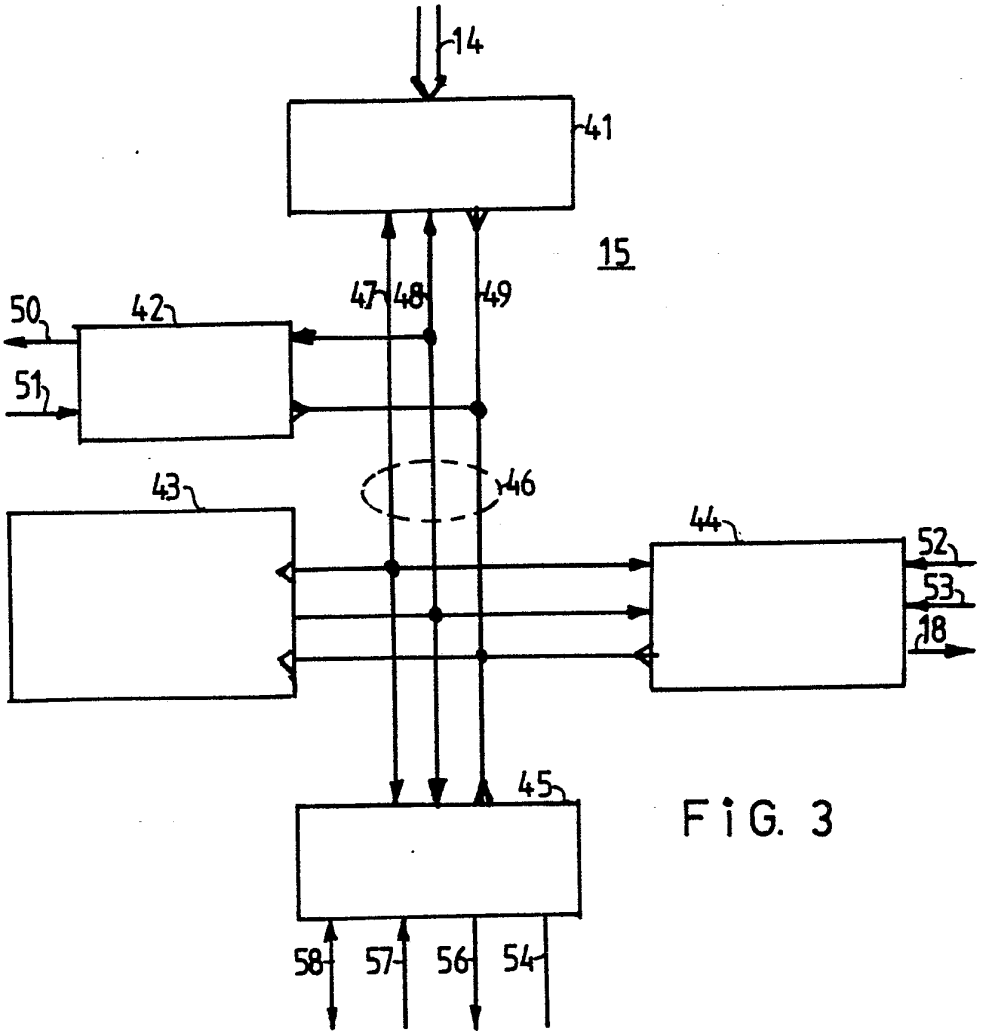
14
41
15
47
48
49
50
42
51
43
46
44
52
53
18
45
58
57
56
54

FiG. 3

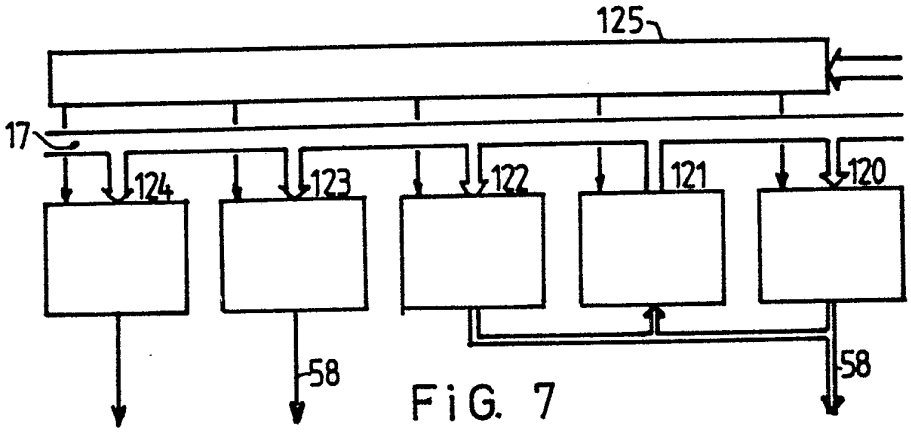
125
17
124
123
122
121
120
58
58

FiG. 7

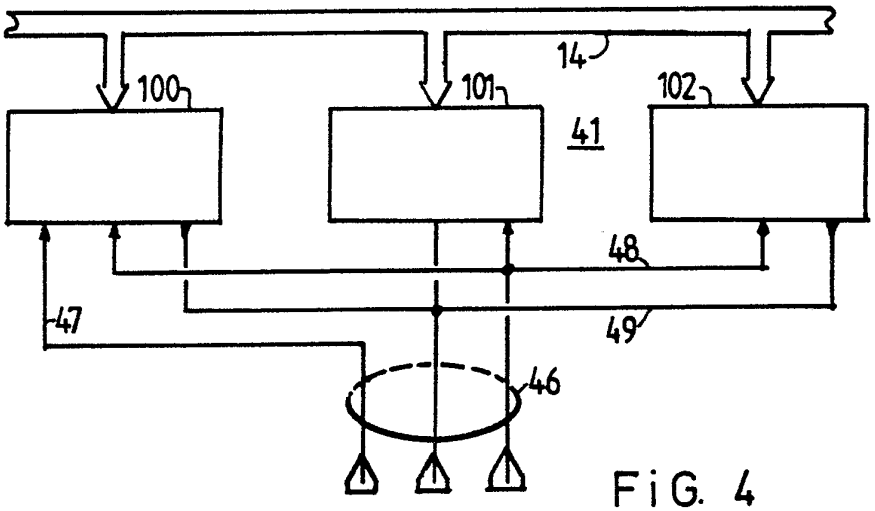
14
100
101
102
41
48
47
49
46

FiG. 4

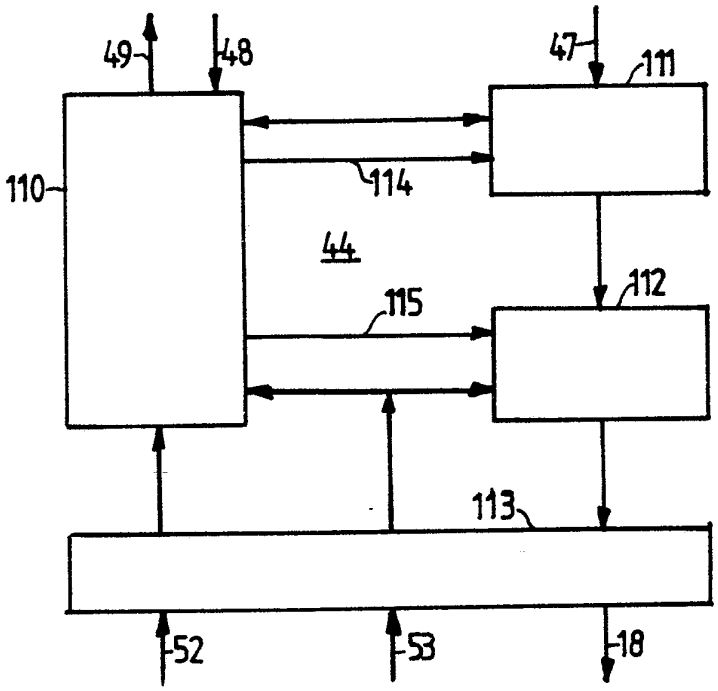
49
48
47
111
110
114
44
112
115
113
52
53
18

FiG. 5

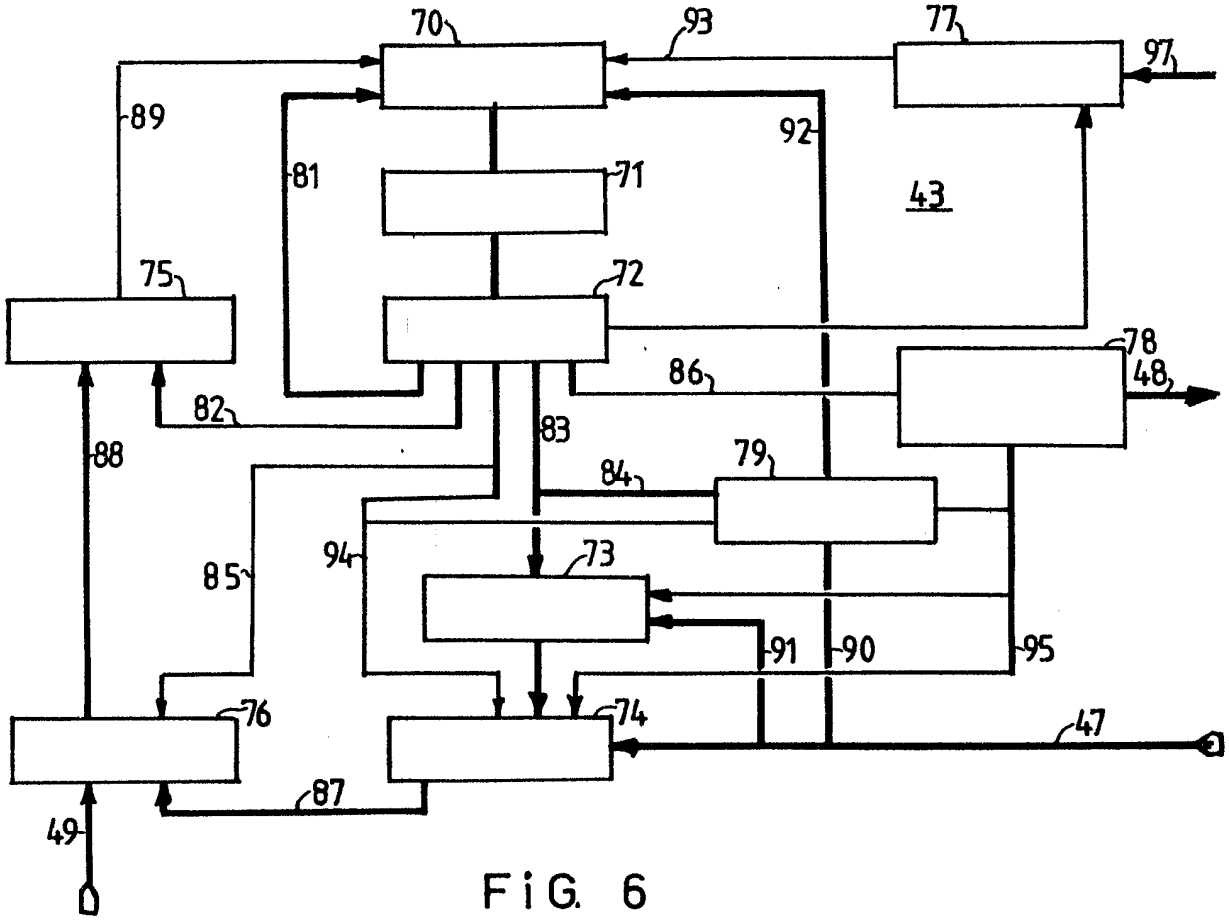

FiG. 6

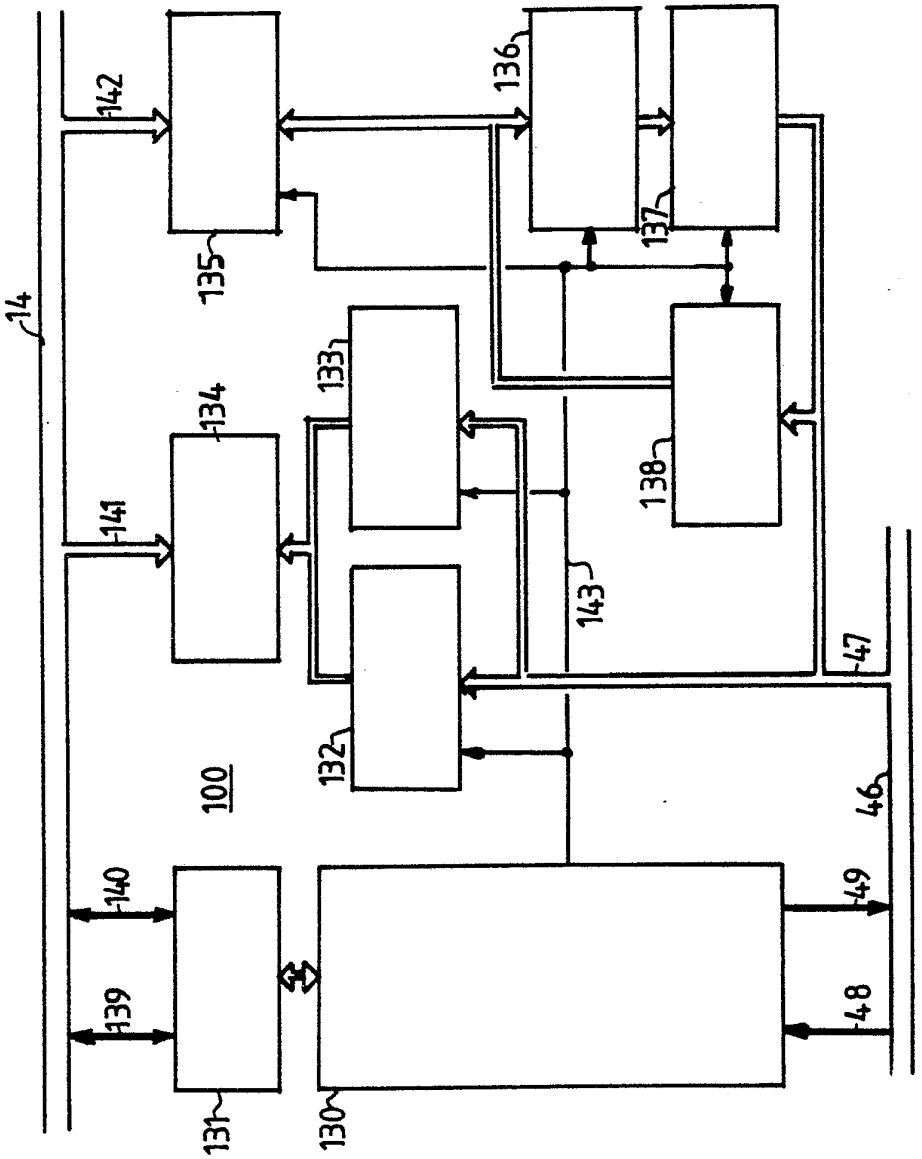
14
142
135
136
137
133
134
141
138
143
47
100
132
46
140
49
139
48
131
130

FIG. 8

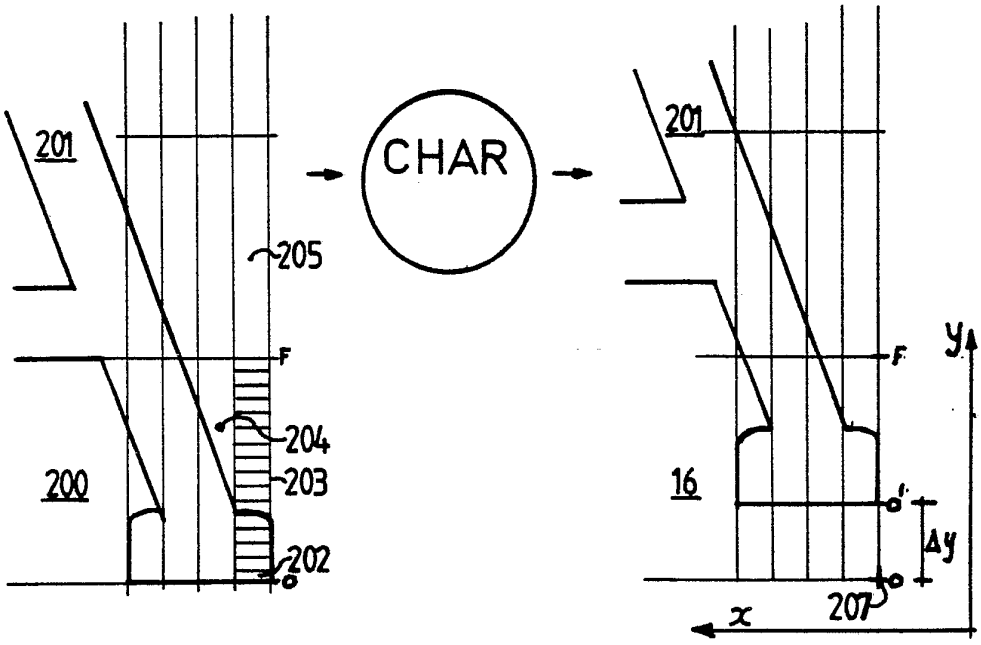
201
205
F
204
203
202
200
CHAR
201
F
y
16
Δy
x
207

FiG. 9

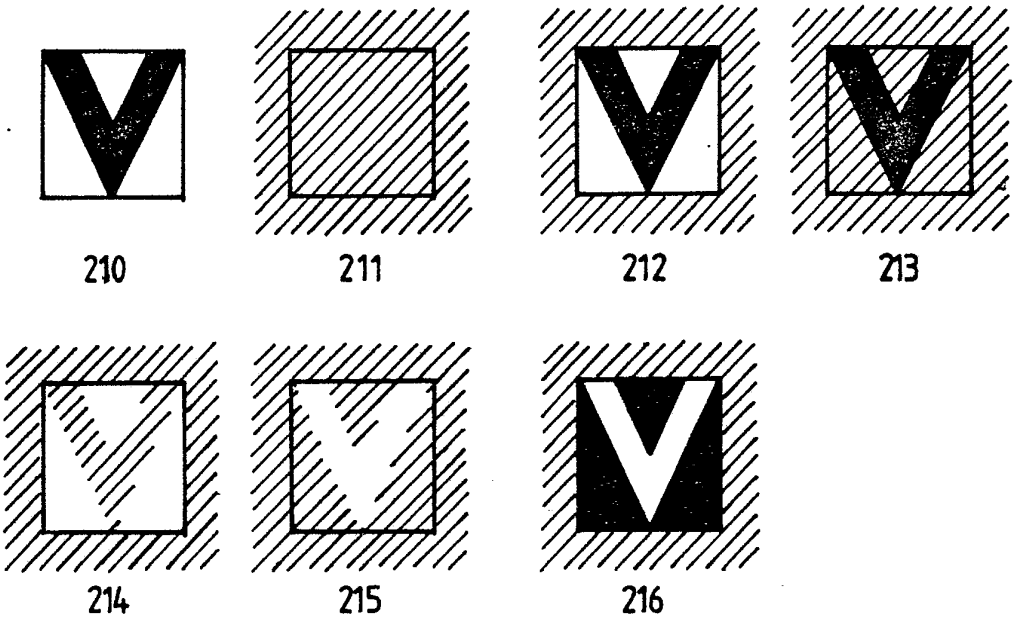
210
211
212
213
214
215
216

FiG. 10

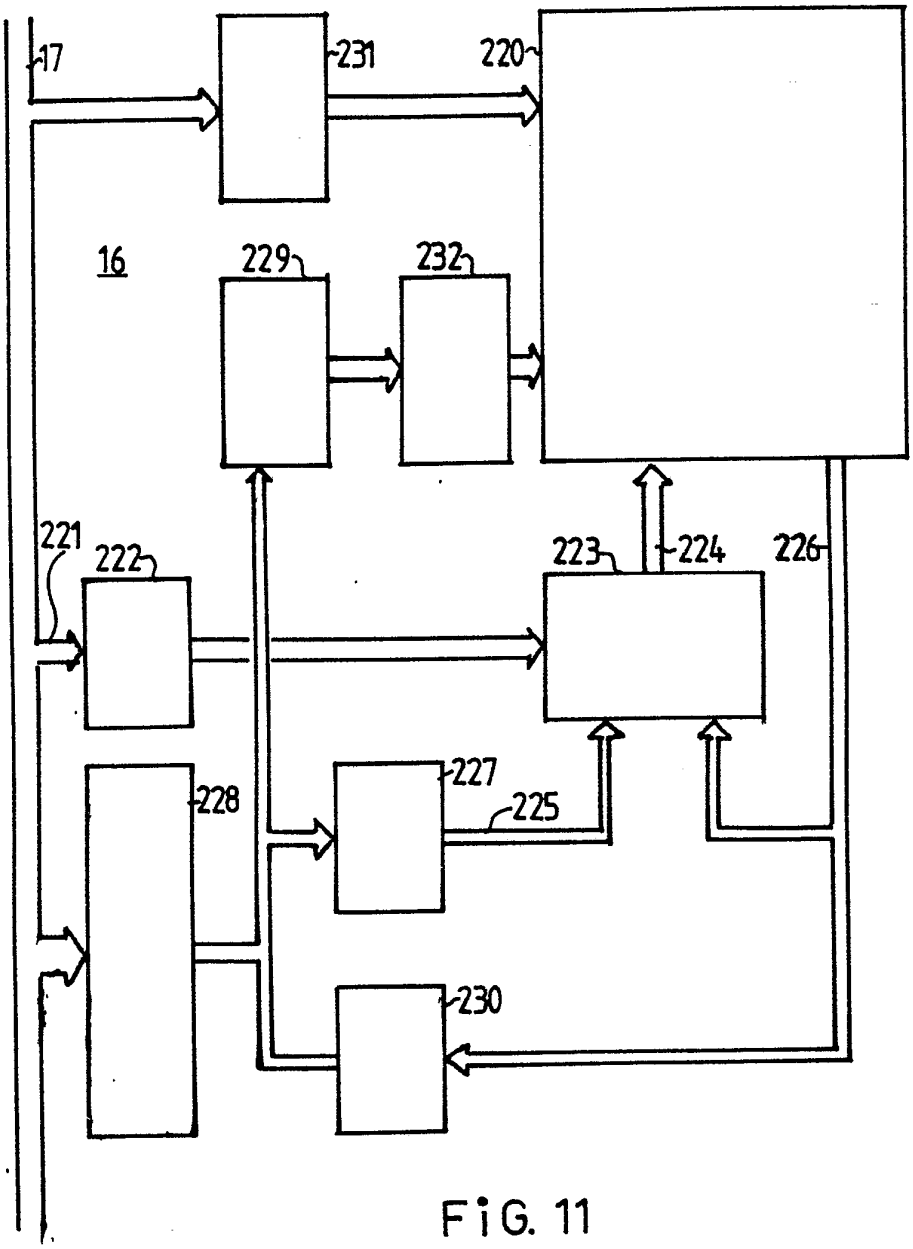

FiG. 11