(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl. [6]
H04N 5/45

(45) 공고일자 2002년08월19일
(11) 등록번호 10-0328801
(24) 등록일자 2002년03월05일

(21) 출원번호 10-1991-0006042
(22) 출원일자 1991년04월16일

(65) 공개번호 특1991-0019427
(43) 공개일자 1991년11월30일

(30) 우선권주장 511,269 1990년04월19일 미국(US)

(73) 특허권자 톰슨 콘슈머 일렉트로닉스, 인코포레이티드
미국 인디아나주 46290-1024 인디아나폴리스 노스 메리디안 스트리트 10330

(72) 발명자 에릭더글러스롬즈버그
미합중국,인디아나,인디아나폴리스,노스라잘리스트리트4353
케니드웨인메이즈
독일연방공화국,빌리겐,빌스토르프스트라쎄54
데이빗제이더필드
미합중국,인디아나,인디아나폴리스,노스브로드웨이6146
마이클스콧다이쓰
미합중국,인디아나,자이언스빌,인디언파이프레인1103
빌리웨즐리베이어2세
미합중국,인디아나,그린필드,우드크레스트드라이브6920
케빈엘리엇브릿지워터
미합중국,인디아나,인디아나폴리스,사우쓰뮤씽로드290

(74) 대리인 이병호

심사관 : 최훈

(54) 텔레비젼수상기용픽쳐-인-픽쳐시스템

요약

픽쳐-인-픽쳐 텔레비젼 수상기(100)에서 비디오 RAM(480)내에 기억된 디지털화된 비디오 영상 데이타는 픽쳐-인-픽쳐 텔레비젼 제어기(410)의 제어 하에 컴퓨터 인터페이스 회로(120)를 걸쳐 외부 컴퓨터(110)로부터의 디지털 데이타의 부가에 의해 수정된다.

대표도
도 4

명세서

도면의 간단한 설명

제 1 도는 본 발명으로 사용하기에 적합한 시스템을 그림 형태로 도시한 도면.

제 2 도는 본 발명으로 사용하기에 적합한 통신 인터페이스를 개략적인 형태로 도시한 도면.

제 3 도는 제 1 도의 외부 컴퓨터에 대한 제어 프로그램의 일부분을 도시하는 플로우챠트.

제 4 도는 제 1 도의 텔레비젼 수상기내에 내장된 본 발명의 실시예를 블록선도 형태로 도시한 도면.

제 5 도는 제 4 도의 마이크로컴퓨터에 대한 제어 프로그램의 일부분을 도시하는 플로우챠트.

제 6a 및 6b 도는 각각 그래픽 및 텍스트 문자로 픽쳐-인-픽쳐 영상의 변형도.

♣ 도면의 주요부분에 대한 부호의 설명 ♣

100 : 텔레비젼 수상기 110 : 외부 컴퓨터

120 : 인터페이스 유니트 402 : 튜너 어셈블리

404 : 튜너 제어 유니트 410 : 제어기

412 : 중앙 처리 유니트(CPU) 414 : 판독 전용 메모리(ROM)

416 : 랜덤 액세스 메모리(RAM) 420 : 키보드

425 : 원격 제어 유니트 430 : 처리 유니트

435 : 오디오 신호 처리기 유니트 450 : 픽쳐-인-픽쳐(PIP)처리 유니트

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

[발명의 분야]

본 발명은 픽쳐-인-픽쳐 기능(picture-in-picture capability)을 가진 텔레비젼 수상기 분야에 관한 것이다.

[발명의 배경]

다수의 최신 텔레비젼 수상기는 픽쳐-인-픽쳐(PIP, or PIX-IN-PIX)기능, 즉, 두개의 상이한 소오스로부터 비디오 신호들을 수신하여 디스플레이 될 때 주시청영역내의 제 1 프로그램과 동일 디스플레이 스크린의 제 2(삽입)시청 영역내의 제 2 프로그램을 포함하는 신호를 발생하도록 그 비디오 신호들을 결합시키는 기능을 갖고 있다.

비디오 랜덤 액세스 메모리(비디오 RAM 또는 VRAM)내에 디지털화된 형태로 " 스냅-쇼트형(snap-shot-like)" 스틸 영상(즉, 프리즈-프레임)을 기억하여 스틸 픽쳐를 발생시키는 것이 기술적으로 공지되어 있다. 이와 같은 시스템은 미합중국 인디아나주 인디아나폴리스에 소재하는 톰슨 컨슈머 일렉트로닉스사에 의해 시판된 RCA VPT-695 비디오 카세트 레코더로 공지되어 있다.

[발명의 개요]

픽쳐-인-픽쳐 텔레비젼 수상기에서 비디오 RAM에 기억된 기억 디지털화된 비디오 영상 데이타가 컴퓨터 인터페이스 회로를 걸쳐 외부 컴퓨터에 의해 픽쳐-인-픽쳐 제어기의 제어 하에 픽쳐-인-픽쳐 메모리의 선택된 메모리 위치에 수록된 그래픽이나 텍스트 문자의 부가에 의해 수정될 수 있다는 것이 본원 명세서내에 알려져 있다.

[실시예의 상세한 설명]

제 1 도를 참조하면, 픽쳐-인-픽쳐 기능을 가진 텔레비젼 수상기(100)는 디지탈 신호의 교환용으로 인터페이스 유니트(120)를 걸쳐 외부 컴퓨터(110)에 접속되어 있다. 외부 컴퓨터(110)는 현재 시중에서 쉽게 입수할 수 있는 퍼스널 컴퓨터중 어떤 특정한 것을 필요로 하지 않으나, IBM PC는 이하에 리스트된 데이타 전송 소오스 코드를 개선시키도록 사용되었다.

실제로 조립되고 평가되어진 본 발명의 실시예에서, 텔레비젼 수상기에 대한 통신 포트는 단일선 쌍방향의 직렬 통신 포트였다. 이러한 포트는 미합중국 인디아나주 인디아나폴리스에 소재하는 톰슨 컨슈머 일렉트로닉스사에 의해 생산된 RCA디멘시아 시스템(RCA DIMENSIA SYSTEM)에 사용된 통신 포트로부터 공지되어 있다. 인터페이스 유니트(120)는 디멘시아 버스 포맷과 대다수의 퍼스널 컴퓨터가 사용된 RS-232 포맷간의 신호를 변환한다. 인터페이스 유니트(120)는 텔레비젼 수상기내에 표준형 RS-232 통신 기능을 제공함으로써 생략될 수 있다.

간단히, 제 2 도에서, 퍼스널 컴퓨터(PC)(110)로부터의 발신 요청(REQUEST TO) 및 데이타 전송(TRANSMIT DATA) 신호는 다이오드 D1 및 D2에 의해 함께 OR화되고 오픈-콜렉터 트랜지스터 Q4를 걸쳐 디멘시아 버스 포노 접속기(DIMENSIA BUS phono connector) J1에 인가된다. 텔레비젼 수상기로부터의 데이타 신호들은 트랜지스터 Q3, Q2, 및 Q1을 걸쳐 DB9 접속기 P1의 수신 데이타(RECEIVE DATA) 단자에 인가된다. 데이타 세트 준비(DATA SET READY) 라인은 수신 데이타(RECEIVE DATA)라인에 연결된다.

외부 컴퓨터(110)는 제 3 도의 플로우챠트에 따른 프로그램의 제어 하에 비디오 RAM으로부터 데이타 전송을 초기화한다. 제 3 도를 참조하면, 제 3 도의 프로그램은 사용자에 의해 컴퓨터 명령의 엔터링에 응답하여 스텝(305)에서 엔터된다. 스텝(305)의 문구 파스명령 라인(PARSE COMMAND LINE)은 수신 받은 명령을 디코드화하는 것을 뜻한다. 그 명령은 실행 가능 명령인지를 결정하도록 스텝(310)에서 검사된다. 만약 실행 가능하지 않다면, 에러 메시지가 표시되고 그 프로그램은 종료된다(스텝 312). 만약 명령이 실행가능 하다면, 인터럽트가 가능케 되는 과정이 시작되어지고 텔레비젼 수상기와의 통신이 시도된다(스텝 315 내지 330). 디멘시아 버스(DIMENSIA BUS)는 CAV 버스라고도 불리 우고, 본 명세서에 사용된 바와 같은 이들 용어들은 동일한 의미를 갖는다. 텔레비젼 수상기(100)와의 통신이 성취되지 않는 경우에 있어서, 에러 메시지가 표시되고 프로그램은 종료된다(스텝 335). 그러나, 텔레비젼 수상기(100)와의 통신이 성취된다면, 이때, 텔레비젼(100)의 픽쳐-인-픽쳐 회로의 비디오 RAM에 기억 디지털화된 비디오 신호의 전송은 스텝(340 내지 355)에 따라 시작된다. 상기 전송이 완료될 때, 인터럽트가 리세트되고 프로그램은 스텝(360)에서 종료된다.

본 발명의 실시예에 대한 간이화한 블록선도가 제 4도에 도시되어 있다. 제 4 도를 참조하면, 무선 주파수(RF) 신호는

튜너 어셈블리(402)의 RF 입력 단자(400)에 인가된다. 튜너 어셈블리(402)는 와이어(403)를 걸쳐 튜너 어셈블리(402)에 동조 제어 신호를 인가하는 튜너 제어 유니트(404)의 제어 하에 개별 RF 신호를 선택하고, 제어 버스(403')를 걸쳐 밴드스위칭 신호(bandswitching signals)들을 인가한다. 튜너 제어 유니트(404)는 제어기(410)에 의해 제어된다. 마이크로 프로세서 또는 마이크로 컴퓨터인 제어기(410)는 중앙 처리 유니트(CPU)(412), 판독전용 메모리(ROM)(414)및, 랜덤 액세스 메모리(416)를 구비한다. 제어기(410)는 로컬 키보드(420)와 적외선(IR) 수신기(422)로부터 유저-엔터된 제어 신호들을 수신한다. IR 수신기(422)는 원격 제어 유니트(425)에 의해 전송된 원격 제어 신호를 수신하고 디코드화 한다.

튜너(402)는 중간 주파수(IF)로 신호를 발생하여 그 신호를, 비디오 IF(VIF) 증폭 스테이지, AFT 회로, 비디오 검출기 및, 음성 IF (SIF) 증폭 스테이지를 포함하는 처리 유니트(430)에 인가한다. 처리 유니트(430)는 베이스밴드 비디오신호(TV) 및, 음성 반송 신호를 발생한다. 음성 반송 신호는 오디오 검출기 및 스테레오 검출기를 구비하는 오디오 신호 처리기 유니트(435)에 인가된다. 오디오 신호 처리기 유니트(435)는 베이스밴드 좌 ·우 오디오 신호들을 발생하여 그 신호들을, 음성 재생용으로 한 쌍의 스피커(438)에 인가한다.

베이스 밴드 비디오 신호(TV)는 휘도 Y 신호와, U 및 V 칼라 성분 신호들을 발생하는 디코더(440)에 결합된다. AUX IN라고 라벨을 붙인 비디오 입력 단자(442)는 외부 소오스로부터 베이스밴드 비디오 신호를 수신하기 위해 제공되어 있다. 외부적으로 공급된 베이스밴드 비디오 신호는 휘도 Y 신호와, U 및 V 칼라 성분 신호들을 또한 발생하는 제2 디코더 유니트(444)에 인가된다. 유도된 Y, U 및 V 신호들은 제각기 픽쳐-인-픽쳐(PIP) 처리 유니트(450)의 입력 단자(A 및 B)들에 인가된다.

PIP 유니트(450)는 인가된 비디오 신호를 처리하기 위한 처리기 유니트(450a)와, 직렬 제어 버스(418)를 걸쳐 제어기(410)와 통신하기 위한 제어 버스 인터페이스 유니트(450b)를 구비한다. 직렬 제어 버스(418)는 주(또는 제 1)픽쳐 및 삽입(또는 제 2)픽쳐를 가진 디스플레이용 영상을 발생하도록 PIP 유니트(450)를 제어하는 제어기(410)로부터 명령 데이타를 결합한다. 직렬 버스(418)를 걸쳐 제어기의 명령 하에, PIP 기능이 가능케 될 수 있고, 무능력하게 될 수도 있으며, 삽입 픽쳐는 예컨대, 4개의 코너들 중 각각에 또는 다양한 다른 영역에 디스플레이될 수 있다. 또한, 직렬 버스(418)를 걸쳐 제어기의 명령 하에, PIP 유니트(450)는 주 및 삽입 텔레비젼 영상을 교환(swap)하도록 제어될 수 있다.

PIP 유니트(450)의 출력 Y, U 및 V 신호들은 비디오 신호 처리기 유니트(455)에 의해서 더 처리하기 위해 복합 비디오 신호로 그 신호들을 변환시키는 엔코더 유니트(452)에 인가되어 최후로 디스플레이 장치(도시되어 있지 않음)의 디스플레이 스크린상에 디스플레이 된다. 비디오 신호들은 수직 및 수평 동기 신호를 유도하는 동기 분리기 유니트(460)에 또한 인가된다. 상기 유도된 수직 및 수평신호는 제어기(410)와 요크 어셈블리(도시되어 있지 않음)에 인가하기 위한 편향신호 발생용의 편향 유니트(470)에 인가된다.

PIP 유니트(450)는 데이타 버스(482)에 의해서 비디오 메모리 유니트(비디오 RAM)(480)에 결합된다. 비디오 RAM(480)는 텔레비젼 픽쳐(즉, 필드 기억 메모리)의 한쪽 필드를 충분히 기억하도록 되도록이면 크게 한다. 본 명세서에 사용된 바와 같은 필드 기억 메모리라함은 각 텔레비젼 라인이 데이타의 8 비트의 512 바이트를 포함하는 데이타의 220 텔레비젼 라인을 기억할 수 있는 메모리를 의미한다. 제어기(410)는 직렬 제어 버스(418)을 걸쳐 적당한 명령을 발신함으로써, PIP 유니트(450)로 하여금 비디오 RAM(480)의 어떤 개별 메모리 위치를 어드레스 할 수 있다는 것이 주목하게 된다.

제어기(410)는 와이어(492)를 걸쳐 직렬 디지탈 통신 포트(490)와 결합된다. 본 발명은 제 1 도의 것과 비디오 RAM(480)으로부터 디지털화된 비디오 정보의 필드의 전송을 제어기(410)의 제어 하에 직렬 디지탈 통신 포트(490)를 걸쳐 외부 컴퓨터에 결합시키는 제어기(410)의 프로그램 명령과 관련된다.

제어기(410)에 대한 제어 프로그램의 RS-232 디코딩 부분은 제 5 도에 도시되어 있다. 상기 루틴은 스텝(500)에서 시작되고, 만약, 외부 컴퓨터가 RS-232 모드를 착수하도록 명령을 통신 받게 되는지를 결정하는 스텝(505)으로 진행한다. 만약, 그렇지 않다면, 상기 루틴은 디멘시아 시스템(DIMENSIA SYSTEM) 디코딩 루틴의 나머지로 스텝(530)에서 종료된다. 만약, 엔터 RS-232 모드(ENTER RS-232 MODE) 명령이 수신되어졌다면, 상기 프로그램은 명령의 제 2 바이트가 RS-232형 통신용으로 보오 속도(baud rate)를 설정하도록 사용되는 스텝(510)으로 진행한다. 스텝(515)에서, 300ms 타이머가 설정되고, 스텝(520)에서 타이머가 되풀이해서 체크되는(스텝 520) 루우프가 착수되며, 반면에, 프로그램은 RS-232 메시지의 시작을 위해 대기한다(스텝 525). 만약, 타이머가 만료된다면, 상기 루틴은 스텝(530)에서 종료된다. 만약 RS-232 메시지가 시작된다면, 상기 프로그램은 명령 파싱 루틴(command parsing routines)으로 진행한다(스텝 535, 540, 550, 560 및 565). 상기 수신 받은 명령이 비디오 RAM 수록(VIDEO RAM WRITE) 명령이라면, 상기 프로그램은 데이타가 기억되는 비디오 RAM(VRAM)내의 메모리 위치의 시작 어드레스 및 16 바이트 데이타를 수신한다(스텝 537). 상기 데이타는 스텝(539)에서 VRAM 에 수록되고 그 프로그램은 스텝(515)으로 리턴 한다. 스텝(540)에서, 상기 프로그램은 수신 받은 명령이 마이크로 RAM 수록(MICRO RAM WRITE) 명령인지를 알기 위해 체크한다. 만약 그렇다면, 상기 프로그램은 제어기(410)의 RAM(416)내에서 데이타 및 스타팅 어드레스를 수신한다(스텝542). 상기 데이타는 스텝(544)에서 RAM(416)에 수록된다. 스텝 (550)에서, 상기 프로그램은 수신된 명령이 서브루틴 호출인지를 결정한다. 만약, 서브루틴 호출이라면, 상기 프로그램은 제어기(410)의 CPU(412)내에서 레지스터의 설정을 위한 데이타와, 실행하기 위한 서브루틴의 ROM(414)내의 어드레스를 수신한다(스텝 557). 스텝(558)에서, 서브루틴이 호출된다. 스텝(559)에서, 서브루틴 호출의 결과로부터 어떤 데이타가 RAM(416)의 특정 영역내에 기억되어, 외부 컴퓨터가 마이크로 RAM 판독(MICRO RAM READ) 명령으로 그것을 액세스할 수 있게 된다. 스텝(560)에서, 상기 프로그램은 수신된 명령이 마이크로 RAM 판독 명령인지를 알기 위해 체크한다. 만약 마이크로 RAM 판독 명령이라면, 스텝(562)에서, 판독을 시작하도록 하는 RAM(416)내의 스타팅 위치의 어드레스와 판독될 다수의 바이트를 수신한다. 스텝(564)에서, 상기 데이타는 RAM(416)으로부터 판독하여 직렬 통신 포트를 걸쳐 외부 컴퓨터애 전송된다. 스텝(565)에서, 상기 프로그램은 수신된 명령이 비디오 RAM 판독 명령인지를 결정하며, 만약, 비디오 RAM 판독 명령이 아니라면, 상기 프로그램은 스텝(515)으로 리턴한다. 만약 비디오RAM 판독 명령이라면, 이때 스텝(567)에서, 상기 프로그램은 VRAM 판독 위치의 스타팅 어드레스와, 각각의 블록이 긴 16 바이트인, 판독될 다수의 블록 N 을 수신한다. 스텝 (569)에서, 데이타 블록은 VRAM으로부터 판독하여 직렬 통신 포트를 걸쳐 외부 컴퓨터에 전송한다. 스텝(570)에서, 모든 요청된 블록이 전송되어져 있는지를 알기 위해 체크된다. 만약 모든 요청 블록이 전송되어졌다면, 상기 프로그램은 스텝(515)으로 리턴한다. 만약 모든 요청 블록이 전송되지 않았다면, 판독 동작용으로 VRAM 스타팅 위치가 16 바이트마다 조정(스텝 575)되고 또 다른 데이타 블록은 모든 요청된 블록이 발신되어져 있을 때까지 판독(스텝 569)한다.

상술한 시스템을 이용하여, 사용자는 픽셀-바이-픽셀을 기초로 픽쳐-인-픽쳐텔레비젼 수상기의 VRAM내에 기억된 비디오의 프리즈된 필드를 수정할 수 있다. 상기 VRAM에 전송된 데이타는 텍스트 문자 또는 그래픽 중 어느 하나로 할 수 있다. 사용자는 비디오 프레임을 프리즈(freeze)하도록 픽쳐-인-픽쳐 텔레비젼을 사용할 수 있으며, 그것을 디지탈 통신 링크를 걸쳐 퍼스널 컴퓨터에 전송한다. 일단 비디오의 디지털화된 필드가 사용자의 PC에 기억되면, 사용자가 원할 때에 수정될 " 페인팅" 프로그램(즉, 그래픽 프로그램)내에 " 삽입" 될 수 있고 텔레비젼 수상기의 VRAM에 백전송될 수 있다. 상기 수정된 비디오 영상은 요구된다면 비디오 카세트레코드상에 기록될 수 있다. 후의 리콜, 픽쳐-인-픽쳐 텔레비젼 수상기에 백전송 및 디스플레이용 PC의 하드 디스크에 그들을 전송함으로써 비디오 영상의 수집을 창출하는 것이 또한 가능하다. 디지털화된 비디오 영상은 다른, 즉, 전화선에 연결된 표준 컴퓨터 모뎀을 걸쳐 원거리 컴퓨터로부터 수신받을 수 있고 디스플레이용 픽쳐-인-픽쳐 텔레비젼 수상기에 전송되는 것도 가능하다.

제 6a 도는 영상이 열십자로 그어진 큰 X자 형태로 그래픽 데이타의 첨가로 수정되어져 있는 삽입 비디오 영상을 가진 픽쳐 - 인 - 픽쳐 텔레비젼 수상기(600a)를 도시하고 있다. 제 6b 도는 텍스트 문자(즉, 숫자 8)의 첨가로 수정되어져 있는 삽입 비디오 영상을 가진 픽쳐 - 인 - 픽쳐 텔레비젼 수상기(600b)를 도시하고 있다. 제 6a 및 6b 도는 삽입 픽쳐의 수정을 도시하는 동안, 완전한 크기 영상은 상술한 바와 같이 또한 수정될 수 있거나, 또는 완성된 완전한 또는 감소된 크기 영상은 외부 컴퓨터로부터 완전히 제공될 수 있다는 것이 주목되어져 있다.

상기 텔레비젼 수상기가 픽쳐 - 인 - 픽쳐 모듈을 사용하지 않을 때 상기 데이타가 시청자에 의해 관찰된 픽쳐에 영향을 미침이 없이 VRAM에, 그리고 VRAM으로부터 전송될 수 있다는 것이 또한 주목하게 된다.

본 명세서에 사용된 바와 같이, 텔레비젼 수상기라 함은 VCRs와 같은, 디스플레이 장치를 가진 텔레비젼 수상기(일반적으로 텔레비젼 세트로 알려져 있음)와 디스플레이 장치가 없는 텔레비젼 수상기를 포함한다.

IBM PC용으로 수록된 그 다음의 소오스 코드는 비디오 데이타를 주제의 픽쳐 - 인 - 픽쳐 텔레비젼 수상기로부터 PC의 하드 디스크 유니트에 전송되게 된다.

제어기(410)용 소오스 코드는 히다찌 회사에 의해 제조된 HD63BO1YOP 마이크로 컴퓨터용 어셈블리 언어로 쓰여졌으며, IBM 소오스 코드 다음에 포함되어 있다.

```
program pip_get;                                                    RCA 84,221

uses
  async4u,dos;

const
  entry : word = 0;
  blocks_per_line = 32;
  bytes_per_block = 16;
  lines_per_file_block = 7;
  file_block_size = blocks_per_line * bytes_per_block * lines per file_block;
  blocks_to_get = lines_per_file block * blocks_per_line;

type
  anystr = string[255];
  pc_types = (XT, AT);

var
  baud_rate : byte;
  com_number : byte;
  port_number : char;
  file_name : string[255];
  position : word;
  temp : string[255];
  pc type : pc_types;
  timer_port : word;
  compatr_buffer : array[0..bytes_per_block-1] of byte;

PROCEDURE DisableInterrupts; inline(FA (cli) );    (MACROS)
PROCEDURE EnableInterrupts;  inline(FB (sti) );

procedure wait(length:word);
var
  junk : byte;
begin
  port[$42]:=$b0;
  port[$42]:=lo(length);
  port[$42]:=hi(length);
  while (port[timer_port] and $20)=0 do;
end; { wait }

procedure delay(length:word);
var
  index : word;
begin
  for index:=1 to length do wait(1160);
end; { delay }

{$L cavsend.ob}

function capture_cav_bus:boolean; External;

function send_cav_message(message:anystr):boolean; External;

function keypressed : boolean;
var
  reg : registers;
begin
  reg.ah:=1;
  intr($16,reg);
  keypressed:=(reg.flags and fzero)=0;
end; { keypressed }

function readkey : char;
var
  reg : registers;
begin
```

PIPGET.PAS 11:06a 4/18/1990 Page 2

```
  reg.ah:=1;
  intr($16,reg);
  readkey:=char(reg.al);
end; { readkey }

function place_cav_in_rs232:boolean;
var
  message : anystr;
  key : char;
begin
  place_cav_in_rs232:=false;
  disableinterrupts;
  while not capture_cav_bus do
  begin
    enableinterrupts;
    delay(100);
    if keypressed then
    begin
      key:=readkey;
      exit;
    end;
    disableinterrupts;
  end;
  message:=chr($ff)+chr(baud_rate);
  if not send_cav_message(message) then
  begin
    writeln('failed to send CAV message');
    enableinterrupts;
    exit;
  end;
  enableinterrupts;
  place_cav_in_rs232:=true;
  delay(100);
end; { place_cav_in_rs232 }

procedure reset_async;
var
  i : byte;
begin
  { read the RBR and reset any possible pending error conditions }
  { first turn off the Divisor Access Latch Bit to allow access to RBR, etc. }
  DisableInterrupts;
  Port[UART_LCR + Async_Base] := Port[UART_LCR + Async_Base] and $7F;
  { read the Line Status Register to reset any errors it indicates }
  i := Port[UART_LSR + Async_Base];
  { read the Receiver Buffer Register in case it contains a character }
  i := Port[UART_RBR + Async_Base];
  EnableInterrupts;
  Async_Buffer_Head := 0;
  Async_Buffer_Tail := 0;
  Async_Buffer_Overflow := FALSE;
end;

function pip_receive(horizontal_address,
                     vertical_address,
                     number_of_blocks:byte):boolean;
var
  counter : longint;
  message_length : word;
  m : byte;
  old_interrupt_mask : byte;
begin
  pip_receive:=false;
  reset_async;
  old_interrupt_mask := Port[IBOSS_IMR];  { get the interrupt mask register }
  m := (1 shl Async_Irq) xor $00FF;
  Port[IBOSS_IMR] := m; { disallow all but serial interrupts }
```



```
  if not async_send(chr((horizontal_ad  ss shl 3) or 4)) then exit;
  if not async_send(chr(number_of_blocks)) then exit;
  if not async_send(chr(vertical_address)) then exit;
  message_length:=number_of_blocks shl 4;
  counter:=$7fffff;
  while (async_buffer_head=message_length=3) and (counter<>0) do
  begin
    dec(counter);
  end;
  Port[IBOSS_IMR] := old_interrupt_mask and m;  { reenable other interrupts }
  if counter<>0 then pip_receive:=true;
end; { pip receive }

function compair(address1,address2,size:word):boolean;
inline($90/$90/$90/
        $59/            {       pop cx      ; size to scan      }
        $5E/            {       pop si      ; compair from ...  }
        $8F/            {       pop di      ; to ...            }
        $30/$C0/        {       xor al,al   ; assume failure    }
        $F3/$A6/        {       repe cmpsb  ; do the compair    }
        $75/$02/        {       jne done    ; jump if not ok    }
        $FE/$C0);       {       inc al      ; compair ok        }

procedure get_a_picture(file_name:anystr);
var
  line_counter : word;
  output_file : file;
  error : boolean;
begin
  assign(output_file,file_name);
  {$I-}
  rewrite(output_file,file_block_size);
  {$I+}
  if ioresult<>0 then
  begin
    writeln('Could not create the file ',file_name);
    exit;
  end;
  line_counter:=0;
  repeat
    error:=not pip_receive(blocks_per_line-1,line_counter+(lines_per_file_block - 1),1);
    move(async_buffer[3],compair_buffer,sizeof(compair_buffer));
    error:=not pip_receive(0,line_counter,blocks_to_get);
  until compair(ofs(async_buffer[3+(lines_per_file_block * blocks_per_line - 1) * bytes_per_block]),
                ofs(compair_buffer), sizeof(compair_buffer));
  blockwrite(output_file,async_buffer[3],1);
  line_counter:=line_counter + lines_per_file_block;
  until (line_counter=2223) or error;
  close(output_file);
  if error then
  begin
    writeln('An error occured while receiving a picture ... no file written.');
    erase(output_file);
  end;
end; { get_a_picture }

function write_byte(address : byte; byte_to_write : byte) : boolean;
begin
  write_byte:=true;
  if not async_send_string($90)+$FF+chr(address-$40)+chr(byte_to_write)) then exit;
end;

function read_byte(address : word; var byte_to_read : byte) : boolean;
var
  counter : word;
begin
```

PIPGET.PAS 11:06a 4/18/1990 Page 4

```
  read_byte:=false;
  reset_async;
  delay(10);
  async_buffer_head:=0;
  if not async_send_string(#002+#001+chr(lo(address))+chr(hi(address))) then exit;
  counter:=$ffff;
  while (async_buffer_head=0) and (counter=0) do
    dec(counter);
  if counter=0 then exit;
  byte_to_read:=ord(async_buffer[4]);
  if addr.=$48 then
  begin
    delay(1);
  end;
  read_byte:=true;
end;

function read_word(address : word; var word_to_read : word) : boolean;
var
  counter : word;
begin
  read_word:=false;
  reset_async;
  if not async_send_string(#002+#002+chr(lo(address))+chr(hi(address))) then exit;
  counter:=$ffff;
  while (async_buffer_head=4) and (counter=0) do
    dec(counter);
  if counter=0 then exit;
  word to read:=(ord(async_buffer[4]) shl 8) or ord(async buffer[5]);
  read_word:=true;
end;

procedure turn_off_vert_interrupts;
const
  max_count = 1000;
var
  error : boolean;
  status : byte;
  try_count : word;
begin
  try_count:=0;
  status:=0;
  repeat
    if place_cav_in_ra232 then
    begin
      error:=async_send_string(#002+#000+#000+#0f1+#000+#002+#0a1+#1e2);
      delay(1);
      if not read_byte($1e,status) then
        status:=$ff;
      if (status and $5f)=0 then delay(500);
    end;
    inc(try_count);
    if try_count>max_count then
    begin
      writeln('failed to turn off vert interrupts');
      async_close;
      halt;
    end;
  until (status and $5f)=0;
end;


function emulate_vert_interrupt : boolean;
const
  lowsize : array[0..6] of byte = (5, 6, 4, 6, 4, 4, 3);
  bit_test : array[0..6] of byte = (1,2,4,8,$10,$20,$40);
```



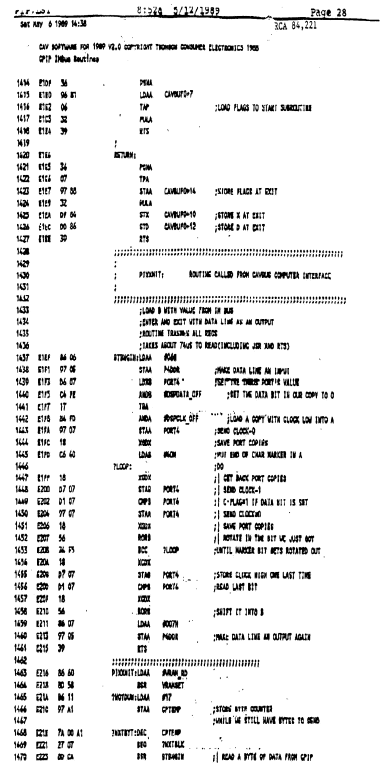
```
var
  cpipstat : byte;
  nextfield : byte;
  i : word;
  sendflags : byte;
  data : array[0..3] of byte;
  numsent : byte;
  x : word;
  cpipstat_test : byte;
begin
  inc(entry);
  emulate_vert_interrupt:=false;
  repeat
    if not read_byte($48,cpipstat) then exit;
    cpipstat_test:=cpipstat;
    if not read_byte($$84+cpipstat,nextfield) then exit;
    if not write_byte($48,nextfield) then exit;
    if not read_byte($$32+cpipstat,sendflags) then exit;
    for i:=0 to 3 do
      if not read_byte($$64+i*cpipstat+i,data[i]) then exit;
    numsent:=0;

    for i:=0 to 6 do
    begin
      if (sendflags and bit_test[i])<>0 then
      begin
        x:=$6302 + data[numsent];
        inc(numsent);
        if x<$5340 then
        begin
          if not read_word(x,x) then exit;
          if not async_send_string(#002+#000+#000+#000+#000+chr(lo(x))+chr(hi(x))) then exit;
          delay(2);
          if not read_word($84,x) then exit;
          if not async_send_string(#002+#000+chr(lowsize[i])+chr(i+$63)+#0a2+#000+#0a0+#0e2) then exit;
        end
        else
          if not async_send_string(#002+#000+chr(lowsize[i])+chr(i+$63)+chr(lo(x))+chr(hi(x))+#0a0+#0e2) then exit;
        delay(1);
      end;
    end;
    if not read_byte($48,cpipstat) then exit;
    if (cpipstat<(cpipstat_test+1)) or (cpipstat=0) then
    begin
      emulate_vert_interrupt:=true;
      if entry=2 then
      begin
        (delay(1));
      end;
      exit;
    end;
    if cpipstat>0 then
      if not write_byte($48,cpipstat-1) then exit;
    if not read_byte($48,nextfield) then exit;
    if nextfield=0 then delay(16*(nextfield+1));
  until cpipstat=0;
  emulate_vert_interrupt:=true;
end;

function unfreeze_a_picture:boolean;
var
  k : byte;
begin
  unfreeze_a_picture:=false;
  turn_off_vert_interrupts;
  if not read_byte($54,k) then exit;
  if not write_byte($54,(k and $6f)) then exit;
```

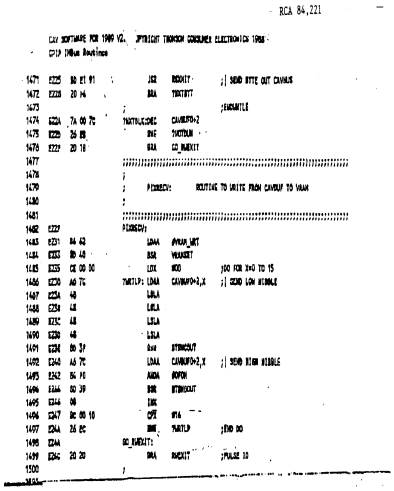
```
PICGET.PAS          11:06a  4/18/1990          Page 6
  if not write_byte(BAD,NOT) then exit;
  if not emulate_vert_interrupt then ex;
  unfreeze_a_picture:=true;                    RCA 84,221
end; { unfreeze_a_picture }

begin
  writeln;
  writeln('PICGet v1.3  Copyright 1990 THOMSON Consumer Electronics Inc.');
  port:=$3f8 !* port:=$3f0 or 1;
  if parmcount=0 then
  begin
    writeln;
    writeln('To capture a frozen pip picture type:');
    writeln('  PICGet file_name [/pcom_number]');
    writeln('  Where file_name = file name for captured picture.');
    writeln('        com_number = number of com port to use (default is com1).');
    halt;
  end;
  if parmcount>1 then
  begin
    position:=pos('/',parameter(2));
    if position=0 then
    begin
      writeln('Illegal parameter');
      halt;
    end;
    temp:=parameter(2);
    port_number:=temp[position+1];
    if (port_number<'1') or (port_number>'2') then
    begin
      writeln('Illegal port number');    - -
      halt;
    end;
    com_number:=ord(port_number)-ord('0');
  end
  else
    com_number:=1;
  if mem[$FFFF:$6A]<$FD then
    pc_type:=AT
  else
    pc_type:=XT;
  if pc_type=XT then
  begin
    baud_rate:=12;
    timer_port:=$42;
  end
  else
  begin
    baud_rate:=6;
    timer_port:=$43;
  end;
  file_name:=paramstr(1);
  if pos('.',file_name)=0 then file_name:=file_name+'.pic';
  if async_open(com_number,baud_rate div 4) then
  begin
    if not unfreeze_a_picture then
    begin
      writeln('Could not unfreeze the picture');
      async_close;
      halt(1);
    end;
    get_a_picture(file_name);
    async_close;
    delay(600);
  end
  else
    writeln('Failed to open serial port');
```

```
PICT.ASM            8:28a  5/12/1989          Page 25
Sat May 6 1989 14:38                         RCA 84,221
    CAV SOFTWARE FOR 1989 V2.0 COPYRIGHT THOMSON CONSUMER ELECTRONICS® 1988
    CPIP INDma Routines

1245                        \ ;;;
```

```
PIP.LST                    8:52a  5/12/1989                    Page 26
Set May  6 1989 14:38                                        RCA 84,221

    CAV SOFTWARE FOR 1989 V2.0 COPYRIGHT THOMSON CONSUMER ELECTRONICS 1988
    CPIP 1MBus Routines

1300                        ;
1301  0157  71 0F 03        RXRCV:  AIM     CLRCAVOUT,PORT2 ;FLOAT BUS IF IT WASNT
1302  015A  78 01 03        TLOOP6: TIM     CAVIN,PORT2
1303  015D  27 FB                   BEQ     TLOOP6          ;WAIT FOR BUS TO GO HIGH
1304  015F  CE 00 00                LDX     #0
1305  0162  09              TLOOP:  DEX
1306  0163  27 2A                   BEQ     TXRCERR         ;WAIT UP TO 300MS FOR CAVBUS TO GO LOW
1307  0165  78 01 03                TIM     CAVIN,PORT2
1308  0168  26 F8                   BNE     TLOOP
1309  016A  96 59           TBLFBIT:LDAA    TEMP    ;HALF BIT RXBITIME TAKES (T+5)*INT(TEMP/2) CYCLES
1310  016C  44                      LSRA            (ABOUT A HALF BIT + 4 CYCLES)
1311  016D  27 04                   BEQ     TRLFBAB
1312  016F  4A              TLP:    DECA
1313  0170  01                      NOP
1314  0171  26 FC                   BNE     TLP
1315  0173  06 60           TBLFERR:LDAB    #$60
1316  0175                  TRCVLP:
1317                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;DELAY FOR A BIT TIME -14 CYCLES
1318  0175  96 59                   LDAA    TEMP
1319  0177  4A              TLOOPE: DECA
1320  0178  27 07                   BEQ     TBOMB
1321  017A  01                      NOP
1322  017B  4A                      DECA
1323  017C  27 03                   BEQ     TBOMB
1324  017E  4A                      DECA
1325  017F  26 F6                   BNE     TLOOP2
1326  0181  19              TBOMB:  DAA
1327  0182  19                      DAA
1328  0183  01                      NOP
1329                        ;
1330  0184  96 03                   LDAA    PORT2           ;100
1331  0186  84 01                   ANDA    #CAVIN          ;
1332  0188  8B FF                   ADDA    #$FFH           ;[ SET CARRY TO MATCH STATE OF CAVIN
1333  018A  56                      RORB                    ;[ PUT BIT IN B
1334  018B  25 02                   BCS     RXCOTCH
1335  018D  20 E4                   BRA     TRCVLP          ;UNTIL FLAG BIT COMES OUT OF B
1336                        ;
1337  018F  38              TXSCEND:PULX                    ;PULL OFF CALLING ROUTINE'S ADDR
1338                                                        ;(ESCAPE ONE LEVEL OF RETURN)
1339  0190  39              RXCOTCH:RTS
1340                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1341                        ;       SEND BYTE IN B OUT THE CAV BUS
1342                        ;
1343  0191  96 59           RSENDB: LDAA    TEMP
1344  0193  4A              TLP1:   DECA                    ;DELAY ABOUT A BIT FOR STOP BIT
1345  0194  01                      NOP
1346  0195  26 FC                   BNE     TLP1
1347  0197  72 20 03                OIM     CAVOUT,PORT2    ;SEND START BIT
1348  019A  59                      SEC                     ;SET END OF CHARACTER MARKER
1349  019B  96 53           TRSTBIT:LDAA    TEMP            ;DELAY A FULL BIT -23 CYCLES
1350  019D  4A              TLOOP1: DECA
1351  019E  27 07                   BEQ     TBOM            ;4 CYCLES
1352  01A0  01                      NOP
1353  01A1  4A                      DECA
1354  01A2  27 05                   BEQ     TBOM            ;5 CYCLES
1355  01A4  4A                      DECA
1356  01A5  26 F6                   BNE     TLOOP           ;4 CYCLES
```

```
                           8:52a  5/12/1989                    Page 27
Set May  6 1989 14:38                                        RCA 84,221

    CAV SOFTWARE FOR 1989 V2.0 COPYRIGHT THOMSON CONSUMER ELECTRONICS 1988
    CPIP 1MBus Routines

1357  01A7  56              TBOM:   RORB                    ;WHILE THERE ARE STILL BITS IN THIS BYTE
1358  01A8  27 10                   BEQ     TXMTDUN
1359  01AA  24 07                   BCC     TZERO
1360  01AC  19                      DAA
1361  01AD  71 0F 03                AIM     CLRCAVOUT,PORT2 ;[ SEND THIS BIT, AND CLEAR MARKER
1362  01B0  0C                      CLC
1363  01B1  20 E8                   BRA     TRSTBIT
1364                        ;
1365  01B3  19              TZERO:  DAA
1366  01B4  72 20 03                OIM     CAVOUT,PORT2
1367  01B7  0C                      CLC
1368  01B8  20 E1                   BRA     TRSTBIT         ;DEPARTILE
1369                        ;
1370  01BA  71 0F 03        TXMTDUN:AIM     CLRCAVOUT,PORT2 ;FLOAT BUS BACK TO HIGH
1371  01BD  39                      RTS
1372                        ;;;;;;;;;;;;;;;;;;;
```





(57) 청구의 범위

청구항 1.

텔레비젼 장치에 있어서,

비디오 영상을 나타내는 비디오 신호를 수신하기 위한 제 1 수단과,

보조 영상을 나타내는 그래픽스 또는 텍스트 데이타와 메모리 어드레스 데이타를 포함하는 디지털 데이타를 외부 컴퓨팅 장치로부터 수신하기 위한 제 2 수단과,

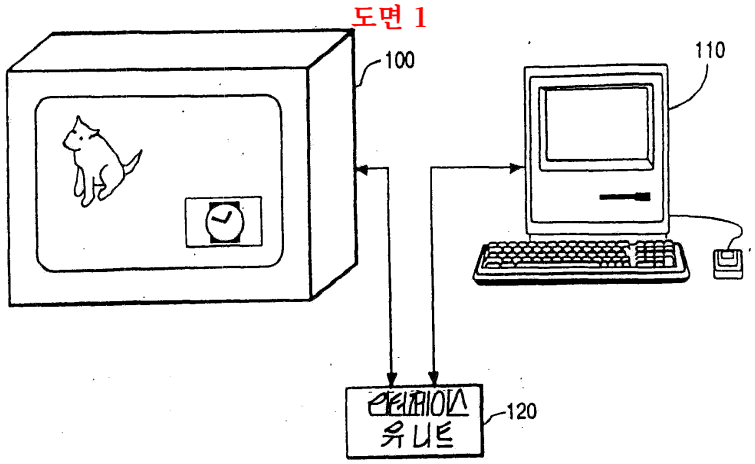
상기 비디오 영상의 디지털 표현을 기억하기 위한 메모리 수단을 갖는 신호처리 수단을 포함하며,

상기 신호 처리 수단은 상기 메모리 어드레스 데이타에 응답하여, 상기 그래픽스 또는 텍스트 데이타를 상기 메모리 수단에 전송하여, 상기 그래픽스 또는 텍스트 데이타와 상기 비디오 신호의 상기 기억된 디지털 표현을 상기 메모리 수단에서 복합시키고, 상기 메모리 수단에 기억되는 복합된 디지털 표현(combined digital representation)을 생성하고,

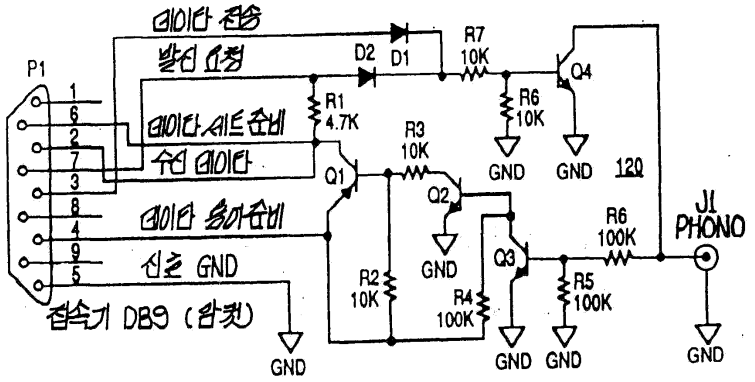
상기 신호 처리 수단은 상기 복합 디지털 표현을 디스플레이하기 위한 디스플레이 장치에 결합하는데 적절한 출력 신호를 형성하기 위해 상기 복합 디지털 표현을 처리하는 수단을 포함하는 텔레비젼 장치.

도면

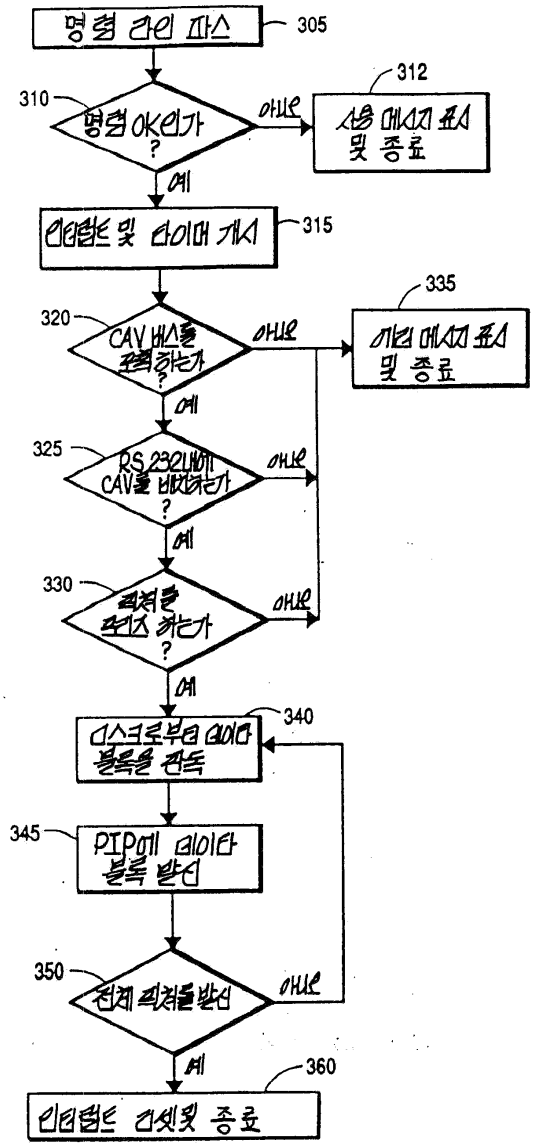
도면 1



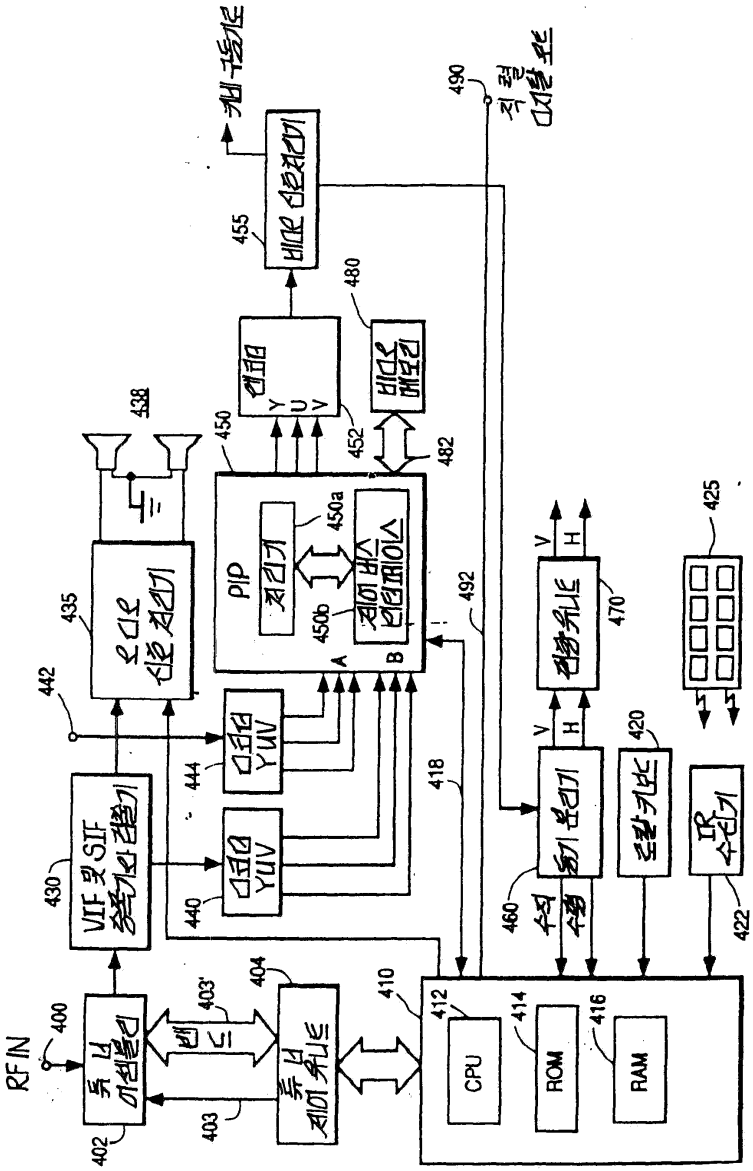
도면 2



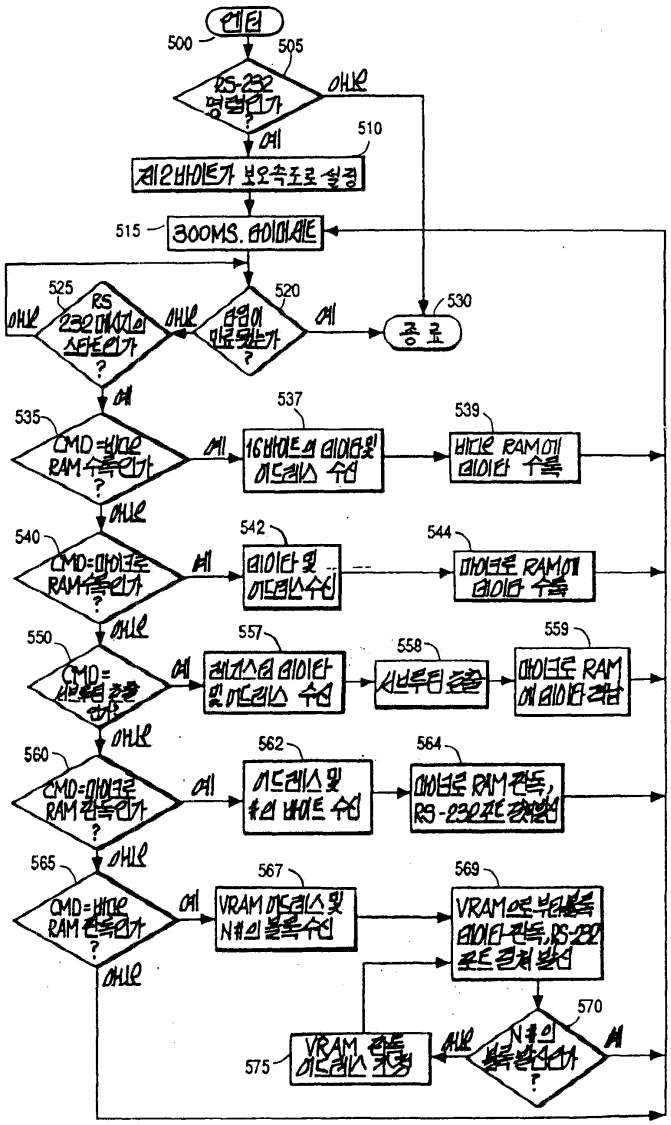
도면 3


명령 라인 파스
305
310
명령 OK인가 ?
아니오
312
사용 메시지 표시 및 종료
예
인터럽트 및 타이머 개시
315
320
CAV 버스를 포획하는가 ?
아니오
335
에러 메시지 표시 및 종료
예
325
RS 232내에 CAV를 배치하는가 ?
아니오
예
330
픽쳐를 주리즈 하는가 ?
아니오
예
340
디스크로부터 데이타 블록을 판독
345
PIP에 데이타 블록 발신
350
전체 픽쳐를 발신
아니오
예
360
인터럽트 리셋및 종료

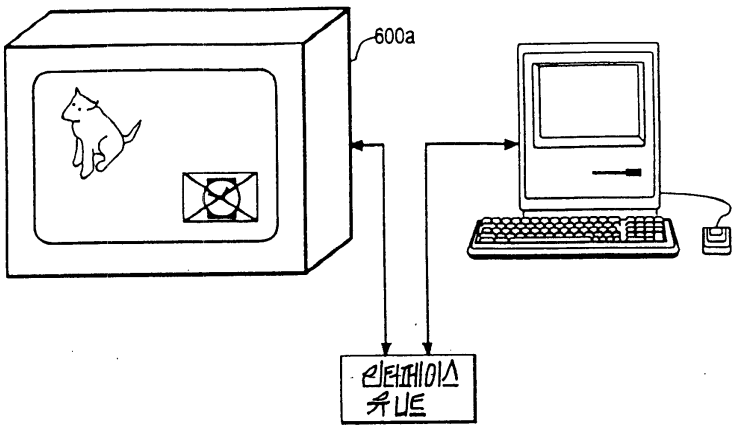
도면 4


RF IN
400
402
403
403'
404
410
412
CPU
414
ROM
416
RAM
418
420
422
425
430
VIF 및 SIF
440
444
442
435
438
450
450a
450b
PIP
A
B
452
480
482
455
490
492
460
470
V
H

도면 5


엔터
500
505
RS-232 명령인가?
아니오
예
510
제2바이트가 보오속조로 설정
515
300MS. 타이머세트
525
RS 232 메시지의 스타트인가?
아니오
520
타임이 만료됐는가?
아니오
예
530
종료
예
535
CMD=버퍼 RAM 수록인가?
예
537
16바이트의 데이타및 어드레스 수신
539
버퍼 RAM에 데이타 수록
아니오
540
CMD=마이크로 RAM수록인가?
예
542
데이타 및 어드레스수신
544
마이크로 RAM에 데이타 수록
아니오
550
CMD= 서브루틴 콜인가?
예
557
레지스터 데이타 및 어드레스 수신
558
서브루틴 콜
559
마이크로 RAM에 데이타 격납
아니오
560
CMD=마이크로 RAM 관독인가?
예
562
어드레스 및 4의 바이트 수신
564
마이크로 RAM 관독, RS-232포트 전송설정
아니오
565
CMD=버퍼 RAM 관독인가?
예
567
VRAM 어드레스 및 N개의 블록수신
569
VRAM으로부터 블록 데이타 관독, RS-232 포트 전송 발신
570
N개의 블록발신인가?
아니오
예
575
VRAM 관독 어드레스 계정

도면 6a



도면 6b

