**PCT** WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : H04L 7/00, 9/00 — A1

(11) International Publication Number: WO 95/01020

(43) International Publication Date: 5 January 1995 (05.01.95)

(21) International Application Number: PCT/US94/07004

(22) International Filing Date: 24 June 1994 (24.06.94)

(30) Priority Data:
082,313 25 June 1993 (25.06.93) US

(71) Applicant: XIRCOM, INCORPORATED [US/US]; 26025 Mureau Road, Calabasas, CA 91302 (US).

(72) Inventors: BIBA, Kenneth, J.; 1157 Stanyan Street, San Francisco, CA 94117 (US). BELANGER, Philip, H.; 17881 Los Felice Drive, Saratoga, CA 95070 (US). BAUGH, William, N.; 1051 Matterhorn Court, Milpitas, CA 95035 (US). ROSEN, David, B.; 481 Summit Knoll, Agoura Hills, CA 91301 (US). CRIPPS, Peter, K.; 20 Carrera Court, Redwood City, CA 94062 (US).

(74) Agents: KLIVANS, Norman, R. et al.; Skjerven, Morrill, MacPherson, Franklin & Friel, 25 Metro Drive, Suite 700, San Jose, CA 95110 (US).

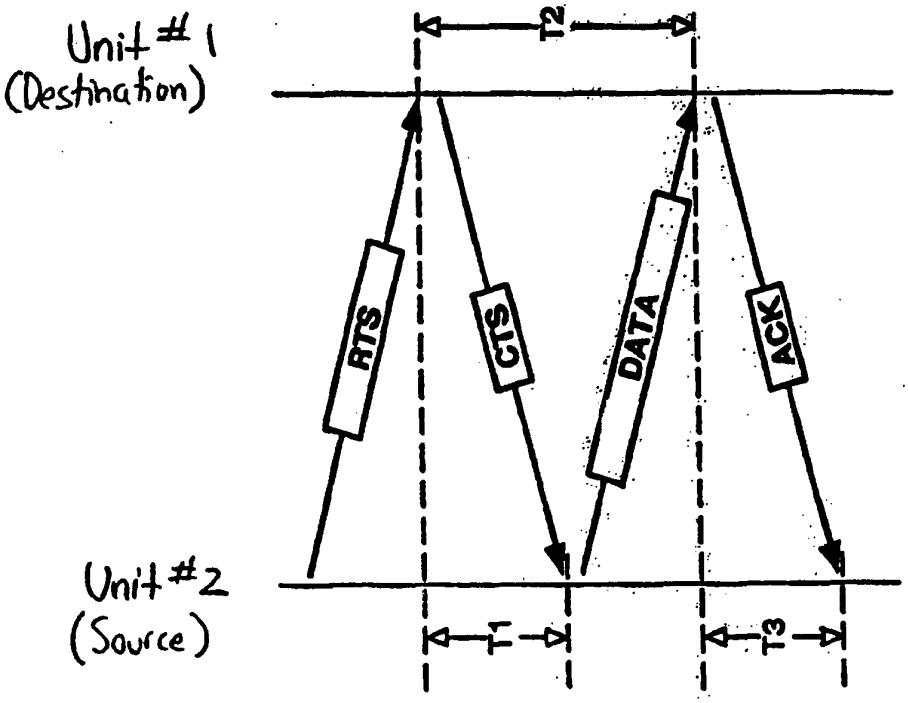
(81) Designated States: AU, CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

**Published**
*With international search report.*
*Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.*

(54) Title: VIRTUAL CARRIER DETECTION FOR WIRELESS LOCAL AREA NETWORK WITH DISTRIBUTED CONTROL

(57) Abstract

A wireless network has a distributed control protocol using carrier sense mutiple access. Each station in the network listens to traffic on the network communications channel (typically frequency hopping radio transmissions using spread spectrum). Based on the content of each received transmission, each station constructs its own perception of when the communications channel is going to be busy in the form of a network allocation vector (data structure). Each message transmission is by way of a four-way handshake involving two short control packets, the first being a request to send and the second being a clear to send that reserve the actual data transfer. The request to send includes an indication of the length of the subsequent data transmission. This length is used by the various receiving stations to reserve the communications channel for the period of time associated with that length of data transfer. The subsequent clear to send message echoes this data length for receipt by stations which are within range of the receiving station but not within range of the source station. This reservation increases the probability of subsequent successful transmission of the actual data.



*FOR THE PURPOSES OF INFORMATION ONLY*

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

| | | | | | |
|---|---|---|---|---|---|
| **AT** | Austria | **GB** | United Kingdom | **MR** | Mauritania |
| **AU** | Australia | **GE** | Georgia | **MW** | Malawi |
| **BB** | Barbados | **GN** | Guinea | **NE** | Niger |
| **BE** | Belgium | **GR** | Greece | **NL** | Netherlands |
| **BF** | Burkina Faso | **HU** | Hungary | **NO** | Norway |
| **BG** | Bulgaria | **IE** | Ireland | **NZ** | New Zealand |
| **BJ** | Benin | **IT** | Italy | **PL** | Poland |
| **BR** | Brazil | **JP** | Japan | **PT** | Portugal |
| **BY** | Belarus | **KE** | Kenya | **RO** | Romania |
| **CA** | Canada | **KG** | Kyrgystan | **RU** | Russian Federation |
| **CF** | Central African Republic | **KP** | Democratic People's Republic of Korea | **SD** | Sudan |
| **CG** | Congo | | | **SE** | Sweden |
| **CH** | Switzerland | **KR** | Republic of Korea | **SI** | Slovenia |
| **CI** | Côte d'Ivoire | **KZ** | Kazakhstan | **SK** | Slovakia |
| **CM** | Cameroon | **LI** | Liechtenstein | **SN** | Senegal |
| **CN** | China | **LK** | Sri Lanka | **TD** | Chad |
| **CS** | Czechoslovakia | **LU** | Luxembourg | **TG** | Togo |
| **CZ** | Czech Republic | **LV** | Latvia | **TJ** | Tajikistan |
| **DE** | Germany | **MC** | Monaco | **TT** | Trinidad and Tobago |
| **DK** | Denmark | **MD** | Republic of Moldova | **UA** | Ukraine |
| **ES** | Spain | **MG** | Madagascar | **US** | United States of America |
| **FI** | Finland | **ML** | Mali | **UZ** | Uzbekistan |
| **FR** | France | **MN** | Mongolia | **VN** | Viet Nam |
| **GA** | Gabon | | | | |

# VIRTUAL CARRIER DETECTION FOR WIRELESS LOCAL AREA NETWORK WITH DISTRIBUTED CONTROL

## BACKGROUND OF THE INVENTION

### Field of the Invention

This invention relates to local area networks and particularly to a wireless local area network having distributed control using time dimension multiple access and for preventing collisions of messages transmitted by network stations.

### Description of the Prior Art

Local area networks such as Ethernet are well known. Most local area networks are wired, so that each station (network node) is connected directly or indirectly to all the other stations by cabling or wires, thus providing full connectivity between all stations. Such local area networks avoid collisions and achieve efficient use of the communications channel (in this case the wire) by well known carrier sensing and collision avoidance schemes. Such schemes are typically not suitable for wireless networks, such as those communicating by radio or infrared.

Prior art wireless networks typically suffer from the problem that in a network having for instance stations A, B, and C, station A can communicate with B and station C can communicate with Station B, but stations A and C cannot communicate. Thus it is likely that transmissions from stations A and C will collide, resulting in

inefficient use of the network bandwidth and reducing network throughput.

In the prior art this is avoided by providing a network central controller which controls transmissions by each station. However, such a system is only operative if the central controller can communicate with every other station. This is a significant disadvantage both in terms of limiting communication distance to the communications radius of the controller, and in the need for the relatively expensive and complex central controller. Also, if the central controller for some reason is not available or accessible, all communications fails.

A similar problem obtains in a wired network having an extremely long wire between particular stations, so as effectively to lack full connectivity between those particular stations.

Therefore there is no way known in the prior art to have a wireless (lacking full connectivity) local area network with all stations sharing a common communications channel using time distributed multiple access, without central control, while achieving efficient use of the communications channel.

## SUMMARY OF THE INVENTION

In a local area network lacking full connectivity (such as a wireless network), virtual carrier detection using "listen before talk" allows distributed control of communications, while providing efficient use of the communications channel with minimal delay. Each station, including e.g. a communications adapter and a personal computer, portable personal computer or personal digital assistant-type device, is assigned a unique address and has associated with it a local area network adapter including the circuitry needed to connect to and maintain communications on the network. The stations include wireless "mobile units" and wired "access point units".

In one version the adapter is a PCMCIA adapter card with a standard connector for connecting to the appropriate port of a host portable computer.

The adapter includes memory for storing a network allocation vector (NAV) which is a data structure representing a series of time cells corresponding to times of future transmissions on the network. The NAV is implemented, e.g. as an array of Boolean flags (bits) each representing a "slot" in time; if one flag is true, this indicates the network is idle; if false, this indicates the network is busy. A memory index pointing to a particular flag indicates the present time. The NAV alternatively is an ordered linked list. Therefore in one version the NAV is a bit vector wherein each bit represents an instant in forward time from the present time, i.e. a quanta of forward time for keeping track of whether a particular adapter (station) believes that the network is going to be busy. Thus if a particular bit is indicated as being busy, that particular adapter believes the network is going to be busy at that particular time in the future. Thus each station constructs such a vector and continually updates it in present time by passively listening to the communications channel, when the particular adapter and its associated transceiver are not otherwise occupied. Each station independently maintains its own NAV based on the traffic it hears on the communications channel.

The transmission procedure for a particular adapter (station) is that the particular station determines the length of the message (computer data but also voice or video) which it wishes to transmit. Typically the message is received from the associated host computer at the station. The particular adapter determines from its own current NAV whether it believes that the channel will not be busy for the period of time necessary to transmit the message of the particular length. Unless the indication

is that the network will not be busy for that needed time, the adapter "backs off" and waits, as described below.

If the indication is that the network will not be busy, i.e. will be available for the particular length of time needed to transmit the message, then the adapter transmits via an associated radio transceiver included in the adapter a request to send "frame" (RTS) to the intended destination station. Each transmission (frame) of any station follows a general protocol having a fixed header format, as is conventional in network transmissions. In this case, the RTS, which is a transmission from a source to a destination, initiates a message transmission. The RTS in one version is a fixed length frame of a particular number of bytes, and includes fields having the destination address and the length of the associated message. This length is not necessarily the length of the entire transmission associated with the message but is an indication of the length of the actual data content of the message to be transmitted following the request to send. (Note that the other portions of each frame have a fixed length.) The RTS also includes other overhead data, as described hereinafter.

The intended destination station, upon receipt of the RTS, extracts the data length portion of the RTS and compares that to its current NAV to determine if it believes that the network will not be busy for the time required to transmit a message of that length including associated overhead.

If the indication according to the NAV at the destination station is that the network will be busy, then no response is made by the destination. If according to the NAV at the destination station the network will not be busy, then the destination station transmits a clear to send (CTS) including, in addition to the usual overhead data, a repetition of the data length portion of the RTS.

In the process of this exchange between the destination and source stations, every other station in the network which can receive messages from either the destination or the source stations updates its NAV to indicate that the network will be busy during the time indicated by the CTS.

Thus all stations which are in contact with either the destination or source station, i.e. the union of the stations in contact with the destination or source, has a simulated (virtual) indication of the busy status of the channel at the conclusion of the exchange of the RTS and CTS.

Those stations not in communication with either the destination or source stations will not have updated their NAV. However, this is not considered a problem because due to their not being able to receive either the RTS or CTS, it is likely that these other stations also will not be capable of transmitting so as to interfere with a subsequent message transmission from the source to the destination station. (It is assumed that the range of each station for receiving and transmitting is approximately equal although this is not a necessary condition.)

Upon receipt of the CTS by the source station, the source station then transmits the actual data "frame". The data frame transmission also typically includes the destination address, source address, and other overhead data.

Upon receipt of the data frame at the destination station, the destination station in one version transmits an acknowledgement (although this is not necessary).

The RTS and CTS are relatively short compared to a length of the typical data frame transmission, thus minimizing their impact on the capacity of the communications channel.

In one version the local area network is a frequency hopping radio network using U.S. FCC standard frequency hopping. Each frame transmission, i.e. RTS, CTS, data, and acknowledge includes an indication of (1) in which hop the transmitting station is; and (2) how far along in a particular hop each transmitting station is, for purposes of synchronizing transmission and receipt of the various RTS, CTS and data frames.

Also, each frame (transmission) has an associated sequence number which is incremented by the source station on transmission of each new message and is used for purposes of duplicate detection. This is to avoid duplicates when a particular message is retransmitted due to communications failure.

In the event of the destination station's failure to send a CTS or failure of the source station to receive a CTS, the source station assumes that a collision has occurred and schedules a retransmission of the particular message. In this case the source station waits a randomly determined period of time before resending in the RTS. The RTS is repeated twice at a particular frequency (in the frequency hopping version). After three failures at a particular frequency, the source station waits until the next (or a subsequent) frequency hop before attempting further RTS transmissions, to increase the chance of successful communications. This back off procedure allows the source station to delay its transmissions for a longer time when the network is very busy and eliminates the problem of clustering when multiple stations are waiting to transmit, and their back off timing elapses during transmission of a single long message.

Also provided is an announce frame which is a transmission sent out periodically by any station which has not received or transmitted any traffic for a particular period of time. The announce frame enables

faster synchronization of the stations to the hopping sequence, in the absence of other traffic.

Therefore, in accordance with the invention, a distributive control protocol uses carrier sense multiple access with a reservation procedure, without use of any central controller. All of the stations participating in a given network listen to the network communications channel, observe transmissions, and based on the content of the transmissions received build their own perception (the NAV) of when the communications channel is busy or not. This is implemented by the three-way handshake including the RTS, CTS and data frame transmissions. The RTS and CTS reserve time for the actual data (message) transfer and in one version an acknowledgement is transmitted following the data frame. The RTS and CTS control transmissions reserve the medium; the RTS includes the length (or an indication of the length) of the subsequent data transmission.

By use of the NAV which is updated according to all transmissions received by each station, the communications channel is reserved anywhere within the range of either a particular source or destination station. Therefore the probability that the message itself will be successfully transmitted is high.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a perspective view showing a mobile unit having a PCMCIA connector in accordance with one embodiment of the present invention.

Figure 2 is a perspective view of a mobile unit having a parallel port connector in accordance with one embodiment of the present invention.

Figure 3 is a perspective view of an access point unit in accordance with one embodiment of the present invention.

Figure 4 is an exploded perspective view of a mobile unit having a PCMCIA connector in accordance with one embodiment of the present invention.

Figure 5 is a view of the top of a populated printed circuit board of an embodiment of a mobile unit in accordance with the invention as illustrated in Figure 4.

Figure 6 is a view of the bottom of a populated printed circuit board of an embodiment of a mobile unit in accordance with the invention as illustrated in Figure 4.

Figure 7 is a schematic of a digital portion of an embodiment of a mobile unit in accordance with the invention as illustrated in Figures 5 and 6.

Figure 8 is a schematic of radio circuitry of an embodiment of a mobile unit in accordance with the invention as illustrated in Figures 5 and 6.

Figures 9A-D comprise a high level block diagram of an embodiment of the radio controller ASIC U6 of the schematic of Figure 7.

Figure 10 is a schematic diagram illustrating a parallel port embodiment of field programmable gate array component U2 of the schematic of Figure 7.

Figure 11 is a schematic diagram illustrating a PCMCIA embodiment of field programmable gate array component U2 of the schematic of Figure 7.

Figure 12 is a diagram illustrating a directed MPDU communication and a multicast MPDU communication between two units.

Figure 13 is a diagram illustrating a directed MPDU communication between two units.

Figure 14 is a diagram illustrating a general format of a header.

Figures 15-19 are diagrams illustrating a format of an RTS frame, a CTS frame, a DATA frame, an ACK frame, and an announce frame, respectively.

Figure 20 is a diagram illustrating a multicast MPDU communication between two units.

Figures 21-24 are diagrams illustrative of a synchronization of units in accordance with one embodiment of the present invention.

Figure 25 is a diagram illustrating roaming.

Figure 26 is an instance diagram of one embodiment of software executing in an access point unit in accordance with the present invention. The instance diagram is associated with the roaming illustrated in Figure 25.

Figure 27 is a diagram illustrating the relationship of the asynchronous service and the time bounded service to higher and lower level protocols.

Figure 28 is an illustration of an inbound MPDU of the time bounded service in accordance with one embodiment of the present invention.

Figure 29 is an illustration of an outbound MPDU of the time bounded service in accordance with one embodiment of the present invention.

Figures 30-33 are diagrams illustrating a format of a RTSI frame, a CTSI frame, a RTSO frame, and a CTSO frame, respectively.

Figure 34 is a diagram illustrating access point units managing bandwidth in accordance with the time bounded service in accordance with the present invention.

Figure 37 is a diagram illustrating the operation of time bounded service in overlapping service areas in accordance with the present invention.

Figures 38 and 39 are block diagrams of a radio receiver/transmitter circuit in accordance with the present invention.

Figures 40A-M comprise a schematic of an embodiment of an access point unit in accordance with the present invention.

Figure 41 is a block diagram of an embodiment of an access point unit in accordance with the present invention.

Figures 42, 43A-43B, and 44 are instance diagrams illustrating an operation of access point unit software during a registration of a mobile unit, a handling of a user message, and a discovery of a new access point unit by a mobile unit, respectively.

DETAILED DESCRIPTION OF THE INVENTION

A wireless local area network in accordance with the present invention may comprise one or more mobile units as well as one or more access point units. A mobile unit may, for example, be coupled to a portable personal computer ("host") so that the personal computer can form part of a wireless local area network. In the event the personal computer comprises a PCMCIA port, a mobile unit having a PCMCIA connector may be inserted into the PCMCIA port of the personal computer and thereby be coupled to the personal computer. If, on the other hand, the personal computer is provided with a parallel port, then a mobile unit having a parallel port connector may be coupled to the parallel port of the personal computer.

Figure 1 is a perspective view illustrating mobile unit 1 having a PCMCIA form factor as well as a PCMCIA connector 2. Mobile unit 1 also provides, however, an antenna extension 3. A first light emitting diode (LED) 4 and a second LED 5 are provided on the antenna extension 3. The mobile unit illustrated in Figure 1 may be inserted into a PCMCIA slot of a personal computer (not shown) so that the connector 2 will couple to a corresponding PCMCIA connector of the personal computer. Antenna extension 3 as well as LEDs 4 and 5, in some embodiments, extend beyond the mouth of the PCMCIA slot in the housing of the personal computer so that LEDs 4 and 5 as well as the antenna extension 3 are not concealed from view or significantly electromagnetically shielded by the personal computer housing.

Figure 2 is a perspective view of another embodiment 6 of the mobile unit in accordance with the present invention. This embodiment 6, rather than having a PCMCIA connector 2, has a connector 7 for connecting to a standard parallel port of the personal computer. The embodiment 6 of the mobile unit illustrated in Figure 2 also has a first LED 8 as well as a second LED 9 disposed on a antenna extension 10.

Figure 3 is a perspective view of an access point unit 11 in accordance with the present invention. Access point unit 11 may, in some embodiments, comprise a telephone jack (not shown) for connecting to a standard wall mounted telephone jack and an associated land line. In the embodiment of Figure 3, the access point unit 11 comprises a mobile unit 1 which is inserted into a slot 12 in a housing of the access point unit 11. The mobile unit provides a radio transmitter/receiver for communicating with mobile unit. Figure 3 illustrates that an antenna extension 13 extends from the enclosure of the access point unit 11 in similar fashion to the way an antenna extension 3 of a mobile unit extends from the housing of a host personal computer.

In accordance with one embodiment of the present invention, an access point unit 11 is in radio communication with a mobile unit 1 such that a portable personal computer coupled to mobile unit 1 may be part of the local area network to which the access point unit 11 provides access. In some embodiments of the present invention, there are multiple access point units coupled to a single hardwired local area network. This hardwired local area network may comprise mechanical connections between desktop personal computers, other computers, printers, plotters, fax machines, and other peripherals. The access point units are located throughout an area so that a personal computer provided with a mobile unit can access the hardwired local area network through one or

more of the access point units which are within radio communication range of the mobile unit. A mobile unit may be moved throughout the operating area of the hardwired local area network such that the mobile unit moves out of a radio communication range with a first access point unit but moves into a radio communication range with a second access point unit. In this situation the mobile unit is said to be "roaming". Control of the mobile unit may be passed from the first access point unit to the second access point unit in order to maintain communication between the mobile unit and the hardwired local area network to which the access point units are coupled.

In accordance with another embodiment of the present invention, no access point units are required to form a local area network. When a first mobile unit enters a communication range of another mobile unit, the two mobile units may form an "ad hoc" local area network. An ad hoc local area network may include many mobile units, some of which are not in direct radio communication with other of the mobile units. As long as each mobile unit of the ad hoc network can communicate with each other mobile unit of the network either directly or indirectly through other intermediary mobile units, all the mobile units can be part of the same ad hoc network. In this way, a user of a personal computer having a mobile unit is able to enter a communication range of either another mobile unit or an access point unit and join the local area network without having to establish any mechanical electrical (wired) connections to the local area network whatsoever. When the user of the personal computer removes the personal computer and the attached mobile unit from the radio communication range of the wireless network, radio communication is broken. The local area network continues operating without the departed mobile unit.

MOBILE UNIT HARDWARE

Figure 4 is an exploded perspective view of an embodiment of the mobile unit 1 of Figure 1 having a PCMCIA form factor and a PCMCIA connector 2. A printed circuit board 40 is sandwiched between a top cover member 41 and a bottom cover member 42. The printed circuit board 40 is held in place and somewhat sealed into the PCMCIA form factor by a plastic gasket 43 to which the top cover member 41 and bottom cover member 42 adhere. The plastic gasket 43 comprises first and second metal electrostatic discharge (ESD) protection contacts 44 and 45 disposed on the edges of the gasket member 43 such that high voltage electrostatic charge on, for example, the hand of a person will be discharged when the person grasps the mobile unit in a natural way between the thumb and forefinger. The gasket member 43 has two holes 46 and 47 in a plastic antenna extension 48. These openings 46 and 47 are positioned so that first and second LEDs 4 and 5 disposed on the printed circuit board 40 can be viewed from the outside of the assembled PCMCIA form factor enclosure. The PCMCIA connector 2 is soldered to one end of the printed circuit board 40 as illustrated in Figure 4. Details of the electrical components, integrated circuits, and traces of the printed circuit board assembly are not illustrated in Figure 4 for ease of illustration.

Figure 5 is a top down view of the top of populated printed circuit board 40 of the mobile unit of Figure 4. The circuit components to the right of the designators A in Figure 5 comprise radio circuitry for both transmitting and receiving information. The circuitry and components to the left of the line designated A comprise digital circuitry associated with coupling the radio circuitry to an PCMCIA port such as a PCMCIA port of a host personal computer (not shown). A substantially T-shaped antenna trace 50 is illustrated on an antenna extension 51 of the printed circuit board 40.

Figure 6 is a view of the bottom of the populated printed circuit board 40 of Figure 4. The components and circuitry disposed to the right of the line designated A comprise part of the radio circuitry whereas components and circuitry to the left of line A comprise part of the digital circuitry which couples the radio circuitry to the PCMCIA port of the host computer. All components are surface mount components.

Figure 7 is a schematic illustrating the digital portion of the mobile unit illustrated in Figures 5 and 6. Figure 8 is a schematic of the radio circuitry of the mobile unit illustrated in Figures 5 and 6. The PCMCIA connector illustrated in Figures 5 and 6 is designated on Figure 7 by the reference numeral P1. The circuitry illustrated in Figure 7 comprises PCMCIA connector P1, a field programmable gate array (FPGA) U2, an 80C188-10 microcontroller U4, a 128K x 8 flash memory U3, a 32K x 8 static RAM U5 and a application specific integrated circuit (ASIC) radio controller U6. The connections designated with circles to the right of radio controller U6 in Figure 7 designate electrical connections to correspondingly labeled connectors on the radio circuit illustrated in Figure 8.

In the event the mobile unit is to provide a PCMCIA interface for coupling to a host computer, then the circuitry of Figure 11 is programmed into the field programmable gate array U2 of Figure 7. If, on the other hand, the mobile unit is to communicate with a host computer via a parallel port, then the circuitry illustrated in the schematic of Figure 10 is programmed into the field programmable gate array U2 of Figure 7 and PCMCIA connector P1 is replaced with a suitable parallel port connector.

Figures 9A-9D comprise a high level block diagram schematic of the contents of the radio controller ASIC U6. The Radio Controller ASIC U6 is realized in approximately

11,000 gates in a one micron CMOS NEC gate array in an 80-pin thin quad flat pack package. External pins of the radio controller ASIC are illustrated on Figure 9A and are designated either with arrowheads, arrow tails, or both. Arrowheads designate output pins, arrowtails designate input pins, and nets having both an arrowhead and an arrowtail designate bidirectional pins. The squares in the schematic of Figure 9A illustrate net connections connecting various identically labelled nets on Figures 9A through 9D. The output pins illustrated in a column and labelled TXSCRAM through DBRXCRC are diagnostic pins usable to determine information about the operation of the radio controller ASIC.

The block labelled Bus Interface in Figure 9B contains a plurality of registers for communicating to and from the radio controller ASIC. Once a register is written with an appropriate value, the Bus Interface block causes an appropriate signal to be asserted indicative of the particular activity or operating mode the ASIC is to perform or assume. The signals to the right of the Bus Interface block labelled RTX1 through RCK0 are these signals. The registers are accessed by address pins A0-A4 of Figure 9A and address/data pins D0-D7 of Figure 9A.

The block labelled Interrupt Controller in Figure 9B provides an interrupt interface for interrupting the 80C188 microcontroller 04. Each interrupt is capable of being masked at the control of either microcontroller U4 of the host via lines IMR0 through IMR7 which extend from the Bus Interface block. The interrupt request output of the Interrupt Controller which is coupled to the 80C188 microcontroller U4 via pin IRQ0 is labelled HINT. When the Interrupt Controller block interrupts the microcontroller U4 with line HINT and pin IRQ0, the microcontroller can cease program execution and determine the cause of the interrupt by reading lines ISR[7:0] via the Bus Interface block.

The block labelled TX/RX DMA Controller in Figure 9B allows the radio controller ASIC to operate as a slave DMA device at the command of the DMA controller of the 80C188 microcontroller U4. Accordingly, the ASIC may be instructed to supply byte after byte of information from a 5-byte FIFO in the TX/RX DMA Controller block to the radio circuit and to assert a DMA request signal DRQ back to the DMA controller of the microcontroller U4 when the ASIC requires additional bytes to forward to the radio circuit. In such a situation, the TX/RX DMA Controller block is able to determine that a DMA request on line DRQ should be asserted because the ASIC has been programmed with information indicating how many total bytes should be transmitted. Alternatively, assertion of the DMA request line DRQ by the DMA Controller block when the ASIC is receiving data from the radio circuit indicates to microcontroller U4 that it should relinquish control of the address bus RA[16:0] and address/data bus AD[7:0] and indicates to the host that the host should empty the FIFO of the DMA Controller in the ASIC.

The Clock Tree Generator block in Figure 9B comprises a plurality of clock buffers to supply clock signals throughout the ASIC. The clock signal CLK10 is a one megahertz clock signal which may be increased if required to enable higher data bit rates.

The Clock/Power Control Unit block of Figure 9B allows the mobile unit to be placed into a sleep mode to conserve power and allows the unused ports of the radio circuit to be unpowered to conserve power. When data is being transmitted from the transmitter portion of the radio circuitry, the radio receiver portion of the radio circuitry is not being used. The Clock/Power Control Unit can therefore cut power to the receiver portion of the radio circuit to conserve power. Also, when a radio transmission is being transmitted from the radio transmitter portion of the radio circuit, the digital

circuitry in the ASIC associated with receiving an incoming radio signal is not used. This digital circuitry is therefore not clocked so as to conserve power. Similarly, when a radio transmission is being received, the transmitter portion of the radio circuit is not powered and the digital circuitry on the ASIC associated with generating a bit stream to transmit is not clocked. A register in the Bus Interface block may also be written to change the clock frequency of CLK10 output by the Clock/Power Control Unit block to the Clock Tree Generator block.

The LED Interface block of Figure 9B causes the two LEDs on the antenna extension of the mobile unit to light under appropriate conditions. One LED blinks when information is either being transmitted or received by the mobile unit. This LED may therefore be considered to be an indication of "activity". The other LED is turned on to indicate that the mobile unit has registered with an access point unit, and has been authenticated. This LED therefore indicates that the mobile unit is "linked" into a hardwired local area network by an access point unit.

The blocks of Figure 9C can be grouped into a data flow group and a control group. The data flow group comprises the blocks labelled Transmit FIFO, Parallel-Serializer, TX CRC Generator, TX Serial Mux and Data Scrambler. The control blocks comprise the blocks labelled Pre-Transmit FSM, Transmit Core FSM, Transmit Down Counter, Post-Transmit Logic and TX Enable Mux. The letters FSM are an acronym for finite state machine. When the mobile unit is transmitting, bytes of information to be transmitted are supplied one by one into the five byte Transmit FIFO block. Bits from the Transfer FIFO block are then serialized one at a time by the Parallel-Serializer block. Serial bits are supplied out of the Parallel-Serializer block on the signal line indicated TXSOUT. The serial data bit stream is supplied through

the TX-Serial Mux to the Data Scrambler. After an appropriate number of serial bits have been supplied from the Parallel-Serializer, through the TX Serial Mux and to the Data Scrambler, the TX Serial Mux is switched so that a CRC code which is generated for the preceding serial bit stream is appended to the end of the serial bit stream. The CRC bits are therefore also supplied to the Data Scrambler. The Data Scrambler contains a linear feedback shift register for generating a pseudorandom sequence of bits from an incoming sequence of bits based on a key received on bus lines KEY[15:0]. KEY[15:0] is controlled by writing an appropriate register in the Bus Interface block of Figure 9B. The other blocks illustrated in Figure 9C control the five data flow blocks in accordance with the frame formats described below to generate the output serial bit stream on the conductor labelled TXD of the Data Scrambler Block. The Pre-Transmit finite state machine is activated by the signal TXFRAME, TXFRAME is asserted by writing an appropriate register in the Bus Interface block. TXFRAME is an indication that something is to be transmitted. The Pre-Transmit state machine therefore asserts TX1 through the TX Enable Mux block to switch the radio circuit to transmit as opposed to receive. Some time later, the Pre-Transmit finite state machine will assert TX2 high that will turn power on to the transmitter of the radio circuit. After a period of time, TX3 is asserted to start modulating information into the radio transmitter.

After the Pre-Transmit FSM block has prepared the radio circuit for the transmission, the Transmit Core FSM operates to read the FIFO of the Transmit FIFO block and to load the Parallel-Serializer every eight microseconds. The Transmit Down Counter block is decremented once upon serialization of each byte. After all the bytes of a frame to be transmitted are serialized, the Post-Transmit Logic block enables the Tx CRC Generator to append the CRC

bits and then issues an end of frame ENDOFFRAME interrupt to the 80C188 microcontroller indicating that the frame has been transmitted.

Figure 9D illustrates a data path and associated control for receiving a serial bit stream into the ASIC from the radio circuit. The incoming serial bit stream is supplied to a Frame Detector/Descrambler/Parallelizer block on the conductor labelled RXD. Frame Detector Descrambler/Parallelizer synchronizes a block to a preamble sequence of incoming alternating ones and zeroes, detects a frame delimiter bit sequence, determines from the header of the incoming frame whether or not data bits follow, determines where in the frame those data bits are located, descrambles any following data bits based on a key which is supplied on bus KEY[15:0], and assembles the resulting bits into parallel bytes which are output from the block on the data bus labelled RXDATA[7:0]. A clock signal is also received from an incoming stream of bits which is described in further detail in the copending U.S. patent application entitled "NRZ Clock Recovery With Instant Lock" by Swaminatha V. Vasudevan, filed June 25, 1993.

The RX CRC Generator block checks the sequence of incoming bits with the incoming CRC bits. The Parser block receives the sequence of data bytes RXDATA[7:0] from the Frame Detector/Descrambler/Parallelizer and parses the various bytes of the frame being received. The Parser also determines whether the frame being received is an Announce frame, an RTS frame, a CTS frame, a Data frame, or an ACK frame. A 13-byte deep Receive FIFO is provided to buffer successive bytes received on the bus RXDATA[7,0] so that they can be read at on bus RXFIFO[7:0]. The Receiver Core FSM writes a byte of parallel information once every eight microseconds into the Receive FIFO block. The Receive FIFO is 13 bytes long as opposed to 5 bytes long like the transmit FIFO so that when the

microcontroller is very busy, the radio controller ASIC will be able to receive and store an entire RTS, CTS or ACK frame without suffering a FIFO overflow.

In operation, the host (i.e. portable personal computer connected to a mobile unit) may wish to send a frame comprising a number of bytes over the radio link to another mobile unit or to an access point unit on a local area network. Bytes of the frame are written by the host one by one into the field programmable gate array U2 in a memory mapped fashion over the bus of the host and through connector P1. Once the field programmable gate array contains the bytes of information, the field programmable gate array U2 requests a DMA transfer by asserting line DRQ0 on pin 90. The 80C188 microcontroller U4 receives the DMA request from the field programmable gate array on pin 79 and relinquishes control of the address bus RA[16:0] as well as the address/data bus AD[7:0]. In this condition, data bytes are written from the field programmable gate array U2 directly into the transmit FIFO block of Figure 9C of radio controller ASIC U6 under control of a DMA controller on the microcontroller U4.

Similarly, bytes of information received from the radio circuit may need to be removed from the Receive FIFO block of the radio controller ASIC U6 of Figure 9D to make room for additional information. The Tx/Rx DMA Controller of Figure 9B of ASIC U6 may assert a DMA request DRQ by generating a DMA request from pin 53 to the DRQO pin 76 of microcontroller U4. The DMA controller of microcontroller U4 will then read bytes of information from the Receive FIFO of ASIC U6 and write those bytes to the field programmable gate array U2.

## SOFTWARE EXECUTING IN THE MOBILE UNIT

When DMA operations are not taking place on the address bus RA[16:0] and the address/data bus AD[7:0], the microcontroller U4 executes software residing in flash

memory U3 and/or RAM U5. File RESTART.ASM contains code that executes initially to set up 80C188 microcontroller U4, to do housekeeping, and to initialize hardware of the mobile unit including registers in the radio controller ASIC U6 and registers in the field programmable gate array U2 which interfaces with the host. File HWDEF.INC contains definitions used by RESTART.ASM to initialize hardware. Other tasks also use the HWDEF.INC file to change operating parameters of the 80C188 microcontroller U4. The file TASKEXEC.ASM is a task executive program which allows multiple tasks to run separately in a multithreaded environment. It also facilitates intertask communication and synchronization so that different tasks can set events for one another, wait for events that are set by other tasks, send messages to other tasks, suspend themselves pending messages from other tasks, and suspend themselves pending events that are triggered by interrupt routines.

There are multiple task files in the system: HOSTISR.ASM, HSTTXTSK.ASM, HSTRXSK.ASM, MACTXTSK.ASM, MEDIAISR.ASM and SCHEDULE.ASM. The file HOSTISR.ASM contains a task that handles the host interface. Both parallel port and PCMCIA field programmable gate array interfaces are supported. HOSTISR.ASM comprises an interrupt service routine that performs the necessary handshaking to allow the host to read information including status information from the programmable gate array U2 which interfaces to the host. The file HSTTXTSK.ASM contains a task which receives an Ethernet-type packet from the HOSTISR.ASM, and fragments the packet as necessary into smaller units. The task MACTXTSK.ASM converts the fragmented packets into frames which are consistent with a communication protocol. MEDIAISR.ASM controls the radio controller ASIC U6 and the associated radio circuit of Figure 8 to transmit the frames generated by MACTXTSK.ASM. MEDIAISR.ASM comprises an interrupt

service routine that initiates the transmission of frames from the radio controller ASIC U6, handles reception of frames from the radio controller ASIC U6, and handles transmit and receive errors.

When MEDIAISR.ASM controls the radio controller ASIC U6 to receive a frame, the frame is passed to MACTXTSK.ASM. MACTXTSK.ASM decodes and dissembles the frames in accordance with a communication protocol into packets (small units). These units are then passed to HSTRXSK.ASM. HSTRXSK.ASM then assembles the small units back into either host Ethernet-type packets or another format in which the host is expecting to receive information. HOSTISR.ASM then handles the communication of the packets to the host via the field programmable gate array U2.

HOSTDEF.INC contains definitions including register addresses within the field programmable gate array U2 and bit definitions of the registers within the field programmable gate array U2. MEDIADEF.INC comprises a plurality of definitions including register addresses within the radio controller ASIC U6 and bit definitions of the registers inside the radio controller ASIC U6. SCHEDULE.ASM comprises an interrupt routine which controls frequency hopping and ad hoc initialization. ENVIRONMENT.INC defines the events that tasks can set by one task for another task. MACDEF.INC contains definitions related to the protocol which defines the format of frames including the lengths of the various frames and the meanings of the various fields within the various frames. GLOBALS.INC contains a list of functions and variables that are shared between tasks. DIAG.INC contains a number of diagnostic routines for testing.

## PROTOCOL

Figure 12 illustrates two units, one of which may be an access point unit, properly synchronized with each

other in both time and frequency so that they are part of the same wireless local area network. Communication between units number 1 and number 2 is divided into hops. Figure 12 illustrates two hops, hop number X and subsequent hop number X+1. Each of the hops is 100 milliseconds in duration. In order for the two units to communicate, the unit transmitting and the unit receiving must be tuned to the same frequency. Figure 12 illustrates units 1 and 2 tuned to frequency A in hop number X and tuned to frequency B in subsequent hop number X+1. The units of a network transmit by repetitively sequencing through 82 hops, each of the 82 hops having a different frequency and having a duration of 100 milliseconds. Each frequency is used exactly once during one repetition of a hopping sequence.

There are two protocol data units (MPDU) formats for communicating information between units of the wireless network, a directed MPDU and a multicast MPDU. Hop number X in Figure 12 illustrates a directed MPDU. The directed MPDU comprises a request to send (RTS) frame transmitted from unit number 2 to unit number 1, a clear to send (CTS) frame transmitted from unit number 1 back to unit number 2, a data frame (DATA) transmitted from unit number 2 to unit number 1, and an acknowledge (ACK) frame transmitted from unit number 1 to unit number 2. There may be numerous such directed MPDU communications between units in a given 100 millisecond hop. The bit rate of information transmitted during each hop is 1 megabyte so each bit has a duration of 1 microsecond.

Figure 13 illustrates a directed MPDU communication in greater detail. A directed MPDU is sent to a specific unit from a specific unit. Each directed MPDU generally consists of four frames: a RTS frame, a CTS frame, a DATA frame, and a ACK frame. RTS is used to reserve the communication channel for transmission of the DATA frame. RTS is able to reserve the communication channel because

the RTS frame contains a destination address uniquely identifying the unit to which a following DATA frame is to be sent as well as a data length field. Other units receiving the RTS frame but which are not identified as the destination therefore are able to compute the amount of time required for the MPDU communication to be completed. The other units will therefore not attempt to use the communication channel during this period and the channel will be reserved. CTS is a transmission from the destination unit which is used to indicate that the destination unit can receive the DATA frame. The DATA frame is used to transmit the data of the MPDU communication to the designated unit. The ACK frame is used to acknowledge that the DATA frame was received without error.

The radio circuit illustrated in Figure 8 requires about 50 microseconds to switch from a transmit mode to a receive mode. The minimum time between successive frames is therefore at least 50 microseconds. The maximum value of timeouts T1, T2 and T3 are as small as possible. T1 (the CTS timeout period) is the maximum time that a second unit will wait for a CTS frame after sending an RTS frame. T3 (the ACK timeout period) is the maximum time that a sending unit will wait for an ACK frame after transmitting the DAT frame. T2 (the DATA timeout period) is the maximum time that a receiving unit will wait for a DATA frame (250 microseconds maximum). T1 and T3 each have a duration of 50 to 100 microseconds.

Figure 14 is a diagram of the general header format used in all asynchronous service MPDU frames. A 32 bit preamble of alternating ones and zeros 10101010... starts the frame. This preamble enables the radio controller ASIC U6 to extract clock information from the incoming bit stream so that the individual bits of the frame can be clocked into the radio controller ASIC U6. An unique 8-bit Start Frame Delimiter (SFD) having a specific pattern

of 10101011 follows the preamble and is used to determine that the received radio signal is in fact a frame to be received and not random noise that happened to cause the radio receiver to lock and to provide a demarkation of the beginning of the information to follow. The header is the following seven bytes comprising the TYPE, CONTROL, DATA LENGTH, HOP TICK, NETID, and SEQNUM fields. These fields are transmitted with the most significant byte first and with most significant bit first. A cyclic redundancy check (CRC) is also appended to each frame.

The TYPE field indicates whether the frame is an RTS, CTS, DATA, ACK, or ANNOUNCE frame. The TYPE field also indicates the MPDU type. The CONTROL field is an 6-bit field which describes the type of service and the structure of the wireless network. The DATA LENGTH is a 10-bit field which indicates the number of bytes in the DATA frame of the MPDU. The HOP TICK field indicates the number of ticks remaining in the current hop. Each tick represents 1/256 of the hop length. HOP TICK may be used by other units to synchronize their 100 millisecond hops in time so as to correspond 100 millisecond hops of the transmitting unit. The NETID field is a 16-bit number that identifies the particular wireless network to which the sending unit belongs. Multiple different wireless networks may therefore operate at once in overlapping service areas provided that each network uses a different NETID. A NETID of 0FFFFh indicates that the sending unit does not belong to any network. A group of units can therefore communicate without agreeing on a NETID. The field SEQNUM is an 8-bit sequence number used for duplicate detection. The sending unit increments this field on transmissions of each number MPDU.

The TYPE field in turn has two subfields: Frame Type and MPDU Type. Frame Type is the high order 4 bits and MPDU Type is the low order 4 bits. The most significant two bits of the MPDU Type subfield of the TYPE field are

reserved. The next most significant bit is the ENCRYPT bit and the least significant bit is the SYSPKT bit. An ENCRYPT bit of "1" indicates that the data portion of the MPDU is encrypted. An ENCRYPT bit of "0" indicates that the data portion of the MPDU is not encrypted. A SYSPKT bit of "1" indicates that the current MPDU should be processed by the microcontroller of the mobile unit. A SYSPKT bit of 0 indicates that the current MPDU data should be passed to the host. The Frame Types designated by the first four bits of the TYPE field are as set forth in Table 1 below:

Table 1

Frame Types are:

| | | |
|---|---|---|
| RTS | = 0; | Request To Send. |
| CTS | = 2; | Clear to Send. |
| ACK | = 4; | Acknowledge. |
| DATA | = 8; | Data. |
| RTSI | = 1; | Request To Send Inbound, time bounded service. |
| CTSI | = 3; | Clear to Send Inbound, time bounded service. |
| RTSO | = 5; | Request To Send Outbound, time bounded service. |
| CTSO | = 7; | Clear to Send Outbound, time bounded service. |
| ANNOUNCE | = 6; | Announce MPDUs are one frame that says: "I'm here." |

The bits of the CONTROL field also is broken down into subfields. The bits, listed from highest order to lowest order, comprise: a TB bit, a HIERARCHICAL bit, an AP bit, a RETRY bit, two reserved bits, and the two most significant bits of the DATA LENGTH field. A TB bit of "0" indicates that this MPDU is for asynchronous service and not for time bounded service. A TB bit of "1" indicates that this MPDU is for time bounded service which defines a special format of the rest of the frame. A HIERARCHICAL bit of "1" indicates that the MPDU will only be received by access point units. Access point units always clear the HIERARCHICAL bit before relaying an MPDU.

The HIERARCHICAL bit is used to force asynchronous traffic through an access point unit. An AP bit of "1" indicates that the current MPDU was sent by an access point unit. The AP bit allows wireless units to distinguish between an MPDU sent directly and one that has been forwarded by an access point unit. A RETRY bit of "1" indicates that the current MPDU is a retransmission of an earlier MPDU.

Figure 15 is a diagram showing the various fields comprising an RTS frame. RTS, or Request-To-Send, is a transmission from a source unit to a destination unit which initiates the MPDU. It is a fixed length frame of 14 bytes (not counting the preamble and SFD byte). The field TYPE has a format of 0x, where x is the MPDU Type. HOP TICK is the HOP TICK of the transmitting unit. The DATA LENGTH field is not the length of this frame but rather it is the length of the payload of the DATA frame that will be sent if the source unit receives a CTS frame. DESTINATION ADDRESS is the 48-bit unique identification code which uniquely identifies the destination unit. CRC 8 is an 8-bit cyclic redundancy check which is computed on the RTS frame.

Figure 16 is a diagram showing the various fields comprising a CTS frame. CTS, or Clear-To-Send, is a transmission from the destination unit to the source unit granting permission to transmit the DATA frame. CTS is not used for flow control in this context. CTS indicates that the destination unit can hear the source unit and is ready to receive the DATA frame. There is no implication about the availability of buffers in the receiving unit. CTS is important for "carrier sense" in a wireless network because it conveys the length of the data frame to those wireless units that can hear the destination unit but not the source of the MPDU. In accordance with Table 1, the TYPE field has the format of 2x, where x is the MPDU Type. CRC 8 is an 8-bit cyclic redundancy check computed on the CTS frame.

Figure 17 is a diagram showing the various fields comprising a DATA frame. DATA is a transmission from the source unit to the uniquely identified destination unit that contains the data "payload" of the MPDU. The payload may be from 1 to 576 bytes in length as specified by the DATA LENGTH field, not counting the preamble and SFD byte. The total length of the DATA frame is: 7 + 12 + 4 + DATA LENGTH bytes. Seven bytes are provided for the header, 12 bytes for SOURCE ADDRESS and DESTINATION ADDRESS, and 4 bytes for the CRC 32 field. In accordance with Table 1, the TYPE field has the format of 8x, where x is the MPDU type. DATA is a variable length field specified in the DATA LENGTH field of the RTS, CTS, DATA and ACK frames. The minimum DATA LENGTH is 1 byte. The maximum DATA LENGTH is 576 bytes. CRC 32 is a thirty-two bit IEEE 802 standard cyclic redundancy check sequence on the DATA frame.

Figure 18 is a diagram showing the various fields comprising a ACK frame. ACK is a transmission from the destination unit to the source unit that acknowledges receipt of the DATA frame with a correct MPDU check sequence. Negative acknowledgments are not used in the protocol. An ACK frame always indicates successful receipt of the DATA frame. The ACK is a fixed length frame of 8 bytes not counting the preamble and SFD byte. In accordance with Table 1, the TYPE field is of the form 4x, where x is the MPDU Type.

Figure 19 is a diagram showing the various fields comprising a ANNOUNCE frame. Announce is a single frame MPDU. It is used to indicate that there is a unit using a particular NETID and hopping sequence. It conveys the NETID of the network of the unit transmitting the ANNOUNCE frame and HOP TICK of that unit. A unit joining the network for the first time listens to determine where other mobile units and access point unit are in their hopping sequence. If a network is idle, ANNOUNCE frames

are sent periodically on each hop by member of the network. In accordance with Table 1, the field TYPE is of the form 61h. 61h designates an ANNOUNCE frame and SYSPKT. The CONTROL field equals 0 or 20h if an access point unit sent the ANNOUNCE frame. DATA LENGTH is always 0.

Whereas hop number X of Figure 12 illustrates an example of a directed MPDU, hop number X+1 of Figure 12 illustrates an example of a multicast MPDU. A multicast MPDU consists only of two frames: an RTS frame and a DATA frame.

Figure 20 illustrates a multicast MPDU communication in greater detail. A multicast MPDU is an MPDU which is destined for more than one receiving unit. A multicast MPDU may have a group address in which case it is directed to a group of receiving units. In the alternative, a multicast MPDU may be called a broadcast MPDU which has an address directed to every receiving unit. Multicast MPDUs do not have a specific destination unit to provide return CTS or ACK frames. If all receiving units transmitted return CTS and ACK frames, there would likely be collisions. The unit sending a multicast MPDU inserts a delay of time T4 between the transmission of the RTS frame and the transmission of the DATA frame. T4 is the interframe gap (IFG) time plus the time required to transmit a CTS frame. At a one megabyte per second data bit rate, T4 is 154 microseconds. The frame formats for multicast RTS and DATA frames are the same as the frame formats for RTS and DATA frames of directed MPDUs.

A positive acknowledge protocol is used. Low level acknowledgments are used to improve the reliability of the wireless medium. When a directed MPDU is sent, the source unit expects an ACK after the DATA frame. If an ACK is not received within 100 microseconds, the ACK is assumed lost and the MPDU is scheduled for retransmission. This process is repeated 9 times. If no ACK is received after

the ninth attempt, a send failure is reported. Broadcast and multicast MPDUs do not expect ACKs and are never retransmitted. The retransmission method is enhanced to work better with the spread spectrum frequency hopping of the radio circuit. Sending stations retransmit directed MPDUs up to three times on a particular frequency. If no ACK is received after the third attempt, a sending unit will wait until its radio hops to the next frequency before sending additional retransmissions. This minimizes the effect of narrow band interference because a sender will attempt to transmit on a least three different frequencies before aborting a transmission.

Whenever frames are retransmitted, there is a possibility of duplicates. The communication protocol employed detects and filters out duplicate MPDUs and conserves channel bandwidth. Transmitting units do not send a new MPDU during retransmission attempts of a previous MPDU. There is only one MPDU outstanding per sending unit. The receiving unit may receive MPDUs from other units interleaved with retransmissions from one unit.

Each wireless unit maintains a list of recently received MPDUs. MPDUs are identified by a 16 bit number (MPDU ID) that is the SEQNUM field concatenated with the low byte of the 48-bit source address field. Whenever a directed MPDU is accepted by a receiving unit, its MPDU ID is stored in the list. The list contains 128 entries. When a new MPDU ID is added to the list, the oldest one in the list is discarded.

Sending units set the RETRY bit (in the CONTROL field of the frame header) whenever they transmit an MPDU more than once. If an RTS for an MPDU is received with the RETRY bit set, the destination unit will send the CTS (if ready) and prepare to receive the DATA frame. After the DATA frame is received, the MPDU ID list is scanned to see if there are any matching entries. If no match is found,

the MPDU is not a duplicate and reception proceeds as usual. If, however, a match is found in the MPDU ID list, then the MPDU is a duplicate. The destination unit will send an ACK frame, but will discard the MPDU because it was already delivered to the higher level software. If the ACK frame is received, the source unit will stop retransmitting the MPDU. The following duplicate detection rules are applied: 1) MPDU IDs are stored only when a directed MPDU is received with a matching NETID and a matching DESTINATION ADDRESS; and 2) The MPDU ID list is scanned for potential duplicates only when an MPDU is received with a matching NETID and a matching DESTINATION ADDRESS, and the RETRY bit is set.

The carrier is sensed by: 1) detecting the start of a frame using hardware of the radio controller ASIC U6; and 2) maintaining a logical NetBusy flag. All units maintain the NetBusy flag by detecting RTS and CTS frames. A unit will transmit only when NetBusy is not true and FrameDetect is not true. Whenever a valid RTS or CTS frame is received, NetBusy is set to TRUE and a NetBusy timer is set to clear NetBusy after transmission of the corresponding DATA and ACK frames. Receiving units therefore can compute the total time NetBusy is TRUE because the DATA LENGTH field in RTS and CTS frames specifies the length of the DATA frame. Other MPDU frames are fixed length so the time NetBusy is TRUE can also be determined for these frames. When the time elapses, the unit clears the NetBusy flag (NetBusy = FALSE).

Every receiving unit has a time out for receiving a DATA frame. The data frame timeout T2 is illustrated in Figure 13 as T2. Every unit that receives an RTS frame will set its NetBusy and the NetBusy timer. If a corresponding DATA frame is not received within 250 microseconds, the NetBusy flag will be cleared immediately. This limits the reservation time of an RTS frame that is not followed by the rest of the MPDU such as

conditions when a unit attempts to transmit to another unit that is no longer on the network.

CHANNEL CONTENTION AND COLLISIONS

Before sending, units check NetBusy and FrameDetect and defer to any transmissions in progress. This is known as Listen Before Talk or "LBT". It is possible that more that one unit can sense the channel, conclude that the channel is not busy, and begin transmitting. When more than unit simultaneously start to transmit, a collision occurs. Collisions may cause frames to be garbled, but it is possible that some units may receive one of the transmissions correctly. Collisions cause a frame to be lost and waste bandwidth. Bandwidth is wasted because the garbled frames are transmitted to completion and there is no useful data transfer until the frames are finished. CSMA/CD protocols such as Ethernet reduce this wasted bandwidth by detecting the collision early in the frame by using a Listen While Talk (LWT) protocol and aborting the transmission as soon as a collision is detected. To minimize the effect of collisions in accordance with one embodiment of the present invention, the above-described RTS/CTS exchange is used to resolve channel contention. After an RTS/CTS exchange, any unit that "heard" either frame will not attempt a transmission until the rest of the MPDU is finished. This eliminates collisions during the DATA frame. If collisions do occur, they will occur during the subsequent RTS frame when multiple units determine that the channel is free and determine to initiate a transmission at the same time. However, because an RTS frame is only 13 bytes long, wasted bandwidth is minimal. If an RTS frame is garbled, no CTS frame will be sent for colliding RTD frames and the source unit will be able to determine that the transmission must be sent again.

When a CTS frame is not received by a source unit, the source unit assumes that a collision occurred and schedules a retransmission of the MPDU. An ACK frame not being received by the source unit indicates that either the DATA frame was damaged or that the ACK frame itself was damaged. In either case, the source unit must retransmit the entire MPDU.

Retransmissions from different units must, however, not be sent at the same time or they will cause another collision. Each unit therefore delays its retransmission for a random amount of time in order to reduce the probability of another collision. The delay before a retransmission is called the backoff time. On the first transmission attempt, there is no backoff time. The source unit retransmits as soon as the channel is detected to be free. For all retransmission attempts of the same MPDU, however, the source unit waits random backoff time before sending the RTS frame.

To implement the backoff time feature, a special backoff timer is used. The backoff timer only counts down when the NetBusy flag is false. The sending unit computes a random number, scales this number, and uses it to set the initial value of the backoff timer. The backoff time is a multiple of the "slot size". The slot size is the time required to reserve the channel for the transmission of a DATA frame. In some embodiments, the time required to reserve the channel is the length of an RTS frame, plus the length of a CTS frame, plus some overhead for the interface gap (IFG).

The following pseudocode illustrates a method of determining the backoff time.

```
int backoff()
     int WaitTime;
{
     WaitTime = (Random() MOD MaxSlots) * SlotSize;
     REPEAT
          IF (NOT NetBusy) WaitTime = WaitTime - 1;
     UNTIL (WaitTime <= 0);
}
```

In this example, the variable SlotSize has a value of 360 microseconds which equals the length of an RTS and CTS frame plus overhead. After the statement WaitTime = (Random() MOD MaxSlots) * Slotsize;, WaitTime is the number of microseconds to wait. The variable MaxSlots has a value of 16. When the network is idle, units will backoff up to 5.8 milliseconds per retry. When traffic exists on the network, the backoff time may be much longer. Random() is a function that returns a random number. The unit ID number is used as the seed for the random number generator. After the backoff timer elapses, the source unit senses the channel again and retransmits as soon as the channel is available.

This method of determining a backoff time has two desirable properties. First, source units delay their transmissions for a longer time when the network is busy. The network therefore automatically remains stable under heavy loads because the WaitTime will not elapse unless the network is idle. Second, the method eliminates the "clustering" which may occur when multiple units are waiting to transmit and their backoff timers elapse during the transmission of a long frame. Even though those backoff timers may elapse at different times, the transmission attempts will be clustered at the same time, just after the end of the long frame. Because the backoff timer set forth above will not count down during the transmission of a frame, clustering is avoided. There are no retransmissions for broadcast or multicast packets.

The following pseudocode describes the operation of the wireless network protocol used by mobile units and

access point units in more detail. In the example, hardware and software are assumed to be synchronized and the pseudocode is executed exactly as fast or as slow as required. Many of the software timer loops in the examples are implemented by hardware in the preferred embodiment illustrated in Figure 7.

```
   /* XWM pseudo code type declarations                              */

   typedef unsigned char byte;
   typedef unsigned int word;

   struct header {        /* header is the frame header format  */
      byte ftype;         /* frame type                         */
      byte control;
      word datalen;       /* payload length of DATA frame       */
      byte hoptick;       /* sender's tick count                */
      word netid;
      byte seqnum;
      byte destadr[6];    /* 48 bit unique destination ID       */
      byte srcadr[6];     /* 48 bit source address              */
      };

/* XWM pseudo code constants                                     */
   MaxSlots = 16; /* the maximum number of slots to back off    */
   SlotSize = 361;/* 360 microsecond slot size                  */
   MaxTries = 9;  /* maximum number of send attempts per MPDU   */
   HopMax = 3;    /* send 3 times per hop maximum               */
   HopLen = 100;  /* 100 millisecond per frequency hop          */

   CTSWait = 100; /* max # microseconds to wait for CTS, T1     */
   ACKWait = 100; /* max # microseconds to wait for ACK, T3     */
   DATAWait = 250;/* max μsecs to wait for DATA frame, T2       */


   /* transmit() return codes                                    */
   SentOK = 0;    /* successful send                            */
   GaveUp = 0x80; /* gave up after MaxTries send attempts       */

   /* sendMPDU() return codes                                    */
   LostAck = 1;   /* timed out waiting for ACK                  */
   LostCTS = 2;   /* timed out waiting for CTS                  */

/* XWM pseudo code global variables                              */
   boolean gotACK;      /* received the ACK I was waiting for   */
   boolean gotCTS;      /* received the CTS I was waiting for   */
   boolean netBusy;     /* network busy                         */
   boolean frameDet;    /* frame detect                         */
   boolean iamSending;  /* True when this node is sending MPDU  */
```

```
   byte sstat;        /* hardware serial status register          */
   byte myAddr[6];    /* node address for this station            */
   word myNetID;      /* the netID this node is using             */
   word myTick;       /* the hop tick counter for this node       */
 5 int curhop;        /* the current frequency number             */
   header tframe;     /* tframe is the frame to be transmitted    */
   header rframe;     /* rframe is the frame just received        */

   word duplist[128]; /* list of mpduIDs used to filter dupes     */
   byte cd;           /* current duplist index                    */

10 /* standard functions                                          */
   int random();  /* returns random number between 0 and 65535   */

   int setbusy(byte ftype, word len);
      /* sets the netBusy timer for the appropriate time         */

   int clrbusy(word when);
15    /* clears the netBusy flag after when time expires         */

   int sendframe(header *fp, byte ftype);
      /* sends the proper type and length frame from     *fp     */
      /* fp is a pointer to the frame to send                    */
      /* sendframe also sets fp->hoptick to current myTick       */

20 int duplicate(word id);
      /* returns True if id matches an mpduID in duplist         */
      /* returns False if no match found                         */

   int rcv indication();
      /* informs the host that a good packet has been received   */


25                MPDU TRANSMIT OPERATION PSEUDOCODE

   /* sendMPDU() sends exactly one MPDU
         it waits for the CTS and ACK frames
         returns SentOK, NoACK, or NoCTS
         sendMPDU() is called by the transmit() routine below    */
30 int sendMPDU()
         {
         word timeout;
         iamSending = True;
         WHILE ((netBusy) OR (frameDet))
35          {};      /* defer to current transmissions          */
         IF (tframe.destadr[0] >= 0x80)/* if multicast bit == 1
            { /* if broadcast, no need to wait for ACK or CTS   */
            gotCTS = True;
            gotACK = True;
40          }
         ELSE /* directed MPDU must wait for CTS and ACK         */
            {
            gotCTS = False;
            gotACK = False;
```

```
            }
         sendframe(&tframe, RTS);
         timeout = CTSWait;
         WHILE ((NOT gotCTS) AND (timeout > 0))
            timeout = timeout - 1; . . . . . .   /* Wait for CTS */
         IF gotCTS
            {
            sendframe(&tframe, DATA);
            timeout = ACKWait;
            WHILE ((NOT gotACK) AND (timeout > 0))
               timeout = timeout - 1;  . . . .   /* Wait for ACK */
            iamSending = False;
            IF (gotACK)
               RETURN(SentOK);
            ELSE
               RETURN(NoACK);
            }
      ELSE
            {
            iamSending = False;
            RETURN(NoCTS);
            }
      }

   /* transmit() is the high level transmit routine.
         it is called as a result of a send command from the host
         transmit includes all of the retry and backoff logic    */
   int transmit()
         }
         int tries;
         int hoptry;
         int sendhop;
         int sendstat;

         /* Format MPDU.                                          */
         tframe.netid = myNetID;
         tframe.seqnum = tframe.seqnum + 1;

         tries = 0;
         sendhop = curhop;
         hoptry = 0;
         sendstat = sendMPDU();
         WHILE ((sendstat != SentOK) AND (tries < MaxTries))
            {
            IF (curhop == sendhop)  /* if still on the same hop  */
               hoptry = hoptry + 1;
            ELSE
            hoptry = 0;
            IF (hoptry>= HopMax) /* if sent too many times on    */
               {                 /* this hop wait till next hop  */
               WHILE (sendhop == curhop) {};
               hoptry = 0;
               curhop = sendhop;
               }
```

```
            tries = tries + 1;
            tframe.control.retry= 1;/*set retry bit for dup detect*/
            backoff();
            sendstat = sendMPDU();
            {
      IF (tries >= MaxTries) sendstat = GaveUp;
      RETURN(sendstat);
      }
```

MPDU RECEIVE OPERATION PSEUDOCODE

```
/*Every wireless station processes all of the frames it
   receives, and always sets the NetBusy flag.  The receive()
   routine is the highest priority task in the mobile unit
   firmware.  A frame detect interrupt will interrupt any other
   firmware function and cause the receive() routine to run.    */

    int receive()
       {
       word mpduID;

    /* assumes all of the received header is in memory
       set netBusy flag regardless of netid or CRCok            */
    setbusy(rframe.ftype, rframe.datalen);

    IF (sstat != CRCok)
       RETURN(sstat);  . . . . . .   /* abort if bad CRC */
    IF (rframe.netid != myNetID)
       {
       /* if RTS, clear netBusy if DATA frame never comes       */
       IF (rframe.ftype == RTS) clrbusy(DATAWait);
       RETURN(sstat);                  /* abort if not myNetID */
       }
    SWITCH (rframe.ftype)
       }
       CASE RTS:
          myTick = rframe.hoptick;/* adjust my tick counter     */
          IF (rframe.destadr == myAddr)
             sendframe(&rframe, CTS); /* send the CTS          */
          /* clear netBusy if DATA frame never comes           */
          clrbusy(DATAWait);
          BREAK;

       CASE DATA:
          IF (rframe.destadr == myAddr)
             {          /* if addressed to me                  */
             sendframe(&rframe, ACK); /* send the ACK          */
             /* save MPDUid in duplist                         */
             mpduID = rframe.seqnum + rframe.srcadr[6]*0x100;
             duplist[cd] = mpduID;
             cd = (cd+1) MOD 128;
             IF (rframe.control.retry)
                   /* this is a retry, must check duplicate    */
                IF (NOT duplicate(mpduID))
```

```
                    rcvindication(); /* report successful rcv   */
                 ELSE /* not a retry, so can't be a duplicate     */
                    rcvindication();/* report successful receive  */
                 }
           ELSE /* not directed to me                            */
              IF (rframe.destadr[0] >= 0x80)/* if multicast      */
                 rcvindication(); /*report successful receive   */
           }
           BREAK;
     /*
        receive() routine continued...
     */
           CASE CTS:
              myTick = rframe.hoptick;/* adjust my tick counter*/
              IF (rframe.seqnum == tframe.seqnum)
                 IF (iamSending) gotCTS = True;
              BREAK;
           CASE ACK:
              IF (rframe.seqnum == tframe.seqnum)
                 IF (iamSending) gotACK = True;
              BREAK;
           /* for CTS and ACK frames
              could also check (rframe.datalen == tframe.datalen)
           */
           }
        RETURN(sstat);
           }
```

INITIALIZATION WITH AN ACCESS POINT

In order for a wireless network to be established in accordance with the present invention, all mobile units and all access point units of the network are synchronized to frequency hop at the same times and to the same frequencies. Figure 12 shows such a synchronization between two units. When a first unit is brought into radio communication range with a second unit that is already part of a network so that the first unit can join the network of the second unit, however, the first unit may be tuned to a different frequency than second unit and the 100 millisecond hop periods of the two units may be offset in time with respect to each other.

Figure 21 illustrates hops of a mobile unit and an access unit. The mobile unit wishes to join the network of the access point unit. The access point unit is a unit operating on an existing network. The two units are not

yet synchronized in frequency and time in the first 100 millisecond period of the mobile unit illustrated in Figure 21. In one specific embodiment, there are 82 different sequences which may be used by a network to pass through the eighty-two hop frequencies: 2401, 2402 ... 2482 MHz. Sequences may be constructed as hyperbolic hop codes as described in "Frequency Hop Codes with Nearly Ideal Characteristics..." by Marin and Titlebaum, IEEE Transactions on Communications, Sept. 1992. Each network, however, must use the same sequence of frequencies so that all the units of a network will move from frequency to frequency in unison. The access point unit in Figure 21 is illustrated as having a sequence including a subsequence of frequency A, frequency B, and frequency C. Providing multiple hoping sequences allows for multiple different wireless networks to operate in the same radio communication area. Each unit therefore must be able to determine the particular sequence being used by the particular network of which it is to be a part.

In the first 100 millisecond period illustrated in Figure 21, the mobile unit receives a frame from the access point unit. This frame may, for example, be a frame addressed by the access point unit to another unit of the network using sequence Y. Because all frames contain the field NETID, the mobile unit is able to determine from the NETID that the particular hoping sequence being used by the access point unit is hopping sequence Y. If an unregistered mobile unit receives a frame transmitted by an access point, the mobile unit will adopt the NETID of the access point rather than the access point adapting the NETID of the mobile unit, because mobile units do not control access points. Therefore, once the mobile unit in Figure 21 receives the NETID and determines that it would like to join the that network, the mobile unit adopts the hopping sequence Y of the network.

In order for the mobile unit to detect a frame from which the NETID can be determined, it is necessary for the mobile unit to have its receiver tuned to the frequency at which the frame is transmitted. In some embodiments, a unit may simply tune its receiver to one of the 82 hop frequencies until a frame transmission is received. Preferably, however, a new mobile unit tunes its radio receiver to receive on all 82 frequencies over multiple hop periods in a total amount of time period equalling less than eighty-two 100 millisecond periods. This reduces the maximum amount of time required for a new unit to detect a frame transmission. Such scanning may be accomplished by cycling through a default randomly chosen hopping sequence, lingering on each frequency for 20 milliseconds or until a NETID is received. After the NETID of the unit which transmitted a frame is received, a lookup table may be used to determine the hopping sequence used by the transmitting unit. Adapting the correct hopping sequence does not, however, necessarily result in the mobile unit's hops being synchronized in time with the hops of the access unit.

As illustrated in Figures 14-19, each frame transmitted also contains a HOP TICK field. In each unit, the value HOP TICK is loaded with the value 255 at the beginning of each hop. Every 1/256 of a hop, the unit decrements HOP TICK by one. When HOP TICK reaches zero, the unit tunes its radio to the frequency of the next hop, and loads HOP TICK with a value of 255. Units transmit in the HOP TICK field of the frames they transmit the current value of their HOP TICK. Accordingly, the mobile unit of Figure 21 which is attempting to join a network of which the access point is a member, can determine from the HOP TICK field of a received frame where the access point unit is located in its current 100 millisecond frequency hop. The mobile unit is therefore able to synchronize with the access point unit (as well as other units which are part

of a network with the access point unit) by adopting the value of HOP TICK transmitted from the access point unit and by using the hopping sequence determined from NETID. The mobile unit is illustrated in Figure 21 switching to the sequence Y of unit number 2 after the reception of a frame in the first 100 millisecond period of the access point unit so that the mobile unit is using sequence Y and frequency B by the second 100 millisecond period of the access point unit.

There must be some traffic on the network during each hop in order to ensure that units remain in synchronization and to allow new units to join the network. When there is an access point unit in the network, the access points transmit announce frames every 10 milliseconds. When there is no access point unit in the network, all units maintain a count of the MPDUs they have received during a hop. At the beginning of each hop, every unit computes a random number R. If a unit has heard less than 4R frames when the HOP TICK matches R, the unit transmits an announce frame. This method of the mobile units transmitting announce frames does not reduce the efficiency of the network because it is only evoked when there is little or no useful data traffic being sent over the network.

## AD HOC NETWORK INITIALIZATION

Figure 28 illustrates a situation in which two mobile units, neither of which is part of a network, are brought into radio communication range to form a network. Because no access point unit is involved, the network to be formed as described above is called an "ad hoc" network. Mobile unit 1 and mobile unit 2 are initially transmitting on different frequencies and are out of synchronization in time. The two units are also initially using different hop sequences.

Both mobile unit 1 and mobile unit 2 transmit announce frames (indicated as arrows in Figure 23) at least once every e.g. 5 milliseconds because no network activity has been received by either mobile unit in the past. If both units were to remain in one frequency hop of the 82 frequency hops transmitting and receiving, it is likely that the two units would never receive each other's announce frames. Similarly, if each of the two units were simply to transmit and receive at the frequency of its current hop, then it would also be likely that the two units would not receive the announce frames from one another. Mobile unit 1 could be transmitting and receiving in hop 1 at frequency A while mobile unit 2 would be transmitting and receiving in hop 3 at frequency C. When mobile unit 1 switched to hop 2 and frequency B, mobile unit would switch to hop 4 and frequency D.

Accordingly, each mobile unit receives at multiple different frequencies (called scanning) during each hop. Because announce frames are guaranteed to occur at least every 5 milliseconds if no other transmission activity is present, and if other activity is present the typical frame duration is 5 milliseconds, a frequency is scanned for a period of 5 milliseconds. In the embodiment illustrated in Figure 22, four different frequencies are scanned at the end of each 100 millisecond period, each for a period of 5 milliseconds. This is the situation described above in which multiple frequencies are scanned each hop so that all 82 hop frequencies can be scanned in less that eighty-two 100 millisecond hop periods. Because the scan periods of the hops of mobile unit number 1 coincide with the announce frames transmitted by mobile unit number 2, and because the scan periods of mobile unit number 2 coincide with the announce frames transmitted by mobile unit number 1, one of the mobile units will receive a frame from the other and the NETID and HOP TICK can then be used to form an ad hoc network.

Figure 23 illustrates a situation in which each of the two such mobile units would not, however, receive a frame from the other within eighty-two 100 millisecond periods. In this situation, the two mobile units happen to be substantially synchronized with each other in time so that the scan periods of the two mobile units exactly coincide. Neither mobile unit would be transmitting during the scan periods of the other.

Therefore, in accordance with one embodiment of the present invention, the mobile units do not scan and/or transmit identically during all hops. Each mobile unit may, for example, use a table such as set forth below to determine whether or it is to scan or to transmit in the current hop.

Table 2

| Hop Number | Scan/Transmit | Hop Number | Scan/Transmit |
|---|---|---|---|
| 0 | 1 | 42 | 1 |
| 1 | 0 | 43 | 0 |
| 2 | 1 | 44 | 0 |
| 3 | 1 | 45 | 0 |
| 4 | 0 | 46 | 0 |
| 5 | 0 | 47 | 1 |
| 6 | 1 | 48 | 0 |
| 7 | 1 | 49 | 0 |
| 8 | 0 | 50 | 0 |
| 9 | 1 | 51 | 0 |
| 10 | 0 | 52 | 0 |
| 11 | 0 | 53 | 1 |
| 12 | 0 | 54 | 1 |
| 13 | 1 | 55 | 0 |
| 14 | 0 | 56 | 1 |
| 15 | 0 | 57 | 1 |
| 16 | 0 | 58 | 0 |
| 17 | 1 | 59 | 0 |
| 18 | 0 | 60 | 0 |
| 19 | 1 | 61 | 0 |
| 20 | 0 | 62 | 0 |
| 21 | 0 | 63 | 0 |
| 22 | 1 | 64 | 0 |
| 23 | 1 | 65 | 0 |
| 24 | 0 | 66 | 1 |
| 25 | 0 | 67 | 1 |
| 26 | 1 | 68 | 0 |

| | | | | |
|---|---|---|---|---|
| | 27 | 1 | 69 | 0 |
| | 28 | 1 | 70 | 0 |
| | 29 | 1 | 71 | 1 |
| | 30 | 1 | 72 | 1 |
| | 31 | 1 | 73 | 0 |
| | 32 | 0 | 74 | 1 |
| | 33 | 1 | 75 | 1 |
| | 34 | 1 | 76 | 0 |
| | 35 | 1 | 77 | 0 |
| | 36 | 0 | 78 | 0 |
| | 37 | 0 | 79 | 1 |
| | 38 | 1 | 80 | 1 |
| | 39 | 1 | 81 | 0 |
| | 40 | 1 | | |
| | 41 | 0 | | |

In Table 2, a "1" in the Scan/Transmit column designates a scan whereas a "0" designates a transmit. Each hop number corresponds with one of the eighty-two frequencies at which the units may receive and transmit. The scan values of a table such as Table 2 may be selected to have a maximum number of situations where one mobile unit is scanning and other mobile unit is transmitting for each possible spacing of hops between two mobile units. If, for example, the mobile unit 1 is two frequency hops ahead of the mobile unit 1 as illustrated in Figure 24, mobile unit 1 may be in hop number 80 and mobile unit 2 may be in hop number 78. This is a situation where one mobile unit is scanning and the other is transmitting because the scan/transmit entry for hop number 80 is "1" and the scan/transmit entry for hop number 78 is "0". Subsequently, when mobile unit 1 is in hop number 81 and mobile unit 2 is in hop number 79, mobile unit 1 will be announcing and mobile unit 2 will be scanning. The total number of scan/transmit pairs for all the different hop numbers for this hop separation of two is made to be a maximum. In a similar fashion, the total number of scan/transmit pairs for all the different hop numbers of the two mobile units for a hop separation of three is also made to be a maximum. The table used may be made to have

a maximum number of scan/transmit pairs for all the different hop numbers for each possible hop separation.

That the two mobile units can now receive frames from one another and that the two mobile units have adopted the same NETID and HOP TICK does not, however, determine which of the two mobile units will adopt the hop frequency of the other where the two hop frequencies differ. Where one unit is a unit mobile unit and the other unit is an access point unit, the mobile unit simply adopts the NETID and HOP TICK of the access point unit. If, however, a mobile unit is not able to form a network with an access point, then the mobile unit attempts to form an ad hoc network with another mobile unit that is not already associated with an access point. But where two mobile units come into contact, it must be determined which hopping sequence and HOP TICK will be adopted for the ad hoc network.

In accordance with one embodiment of the present invention, the mobile unit having the smaller hop number will adopt the HOP TICK and hop frequency of the mobile unit having the larger hop number. This determination of the hop frequency of which mobile unit will be adopted must, however, always result in the same hop frequency being determined to be the hop frequency of choice by all mobile units at all times in the sequence.

Figure 24 illustrates a situation where the hop numbers of mobile unit 1 and mobile unit 2 are offset with respect to each other by 2. When mobile unit 1 is in hop number 81 and mobile unit 2 is in hop number 79, mobile unit 1 would have the hop number of choice because hop number of 81 would be the greater hop number of the two mobile units. 100 milliseconds later in time when mobile unit 1 would have a hop number of 0 and when the mobile unit 2 would have a hop number of 80, however, mobile unit 2 would have the hop number of choice because hop number 80 is the greater hop number.

In order to avoid such a situation where the hop frequency of choice changes depending on where in the hopping sequence the two mobile units are located, the determination is made based upon the hop numbers of the two mobile units at a location in the sequence where the separation in hop number between the two mobile units is smallest. Alternatively, the determination could be made based upon the hop numbers where the separation is the largest. In Figure 24, the two mobile units have a hop separation of 2 when mobile unit 1 is in hop number 81 and have a separation of 80 when mobile unit 1 is in hop number 0. The hop frequency of mobile unit 1 is therefore determined to be the hop frequency of choice because in the hops of smaller hop separation, mobile unit 1 has the greater hop number. After making this determination, mobile unit 2 will adopt the hop frequency of mobile unit 1.

In the event that there are multiple mobile units in a network and the new mobile unit is determined to have the hop frequency of choice, then the multiple mobile units all change their hop frequencies to the hop frequency of the new mobile unit. In this way, a large number of mobile units attempting to form an ad hoc network will all eventually coalesce to a single NETID and HOPTICK.

Appendix A, which is a part of the present disclosure, is a listing of a pascal program which generated the scanning table of Table 2 set forth above.

## TAG CENTERED ROAMING

Roaming is a collaborative effort. A set of access points and mobile units cooperate to make roaming as fast and smooth as possible. Each access point unit constructs a table that contains information about other access point units in the system, with the help of the mobile units. The information in the tables describes to a mobile unit

which access point units are good candidates for roaming and exactly how to tune to and synchronize with each access point unit. Every time an access point unit enters the network, it announces its presence and finds out about all the other access point units that are already in the network. Access points coordinate amongst themselves to ensure that they do not use the same NETID and hopping sequence.

When a mobile unit enters a network with infrastructure (a network with a hardwired interconnect), it finds an access point unit by scanning the frequencies. The mobile unit then attempts to register with that access point unit. If the access point unit authenticates the mobile unit it sends a scrambling key to the mobile unit. The mobile unit loads that key into its memory, and then, the mobile unit's data frames are scrambled as they are transmitted. The mobile unit also receives a copy of the access point table (APTable) if it registers successfully.

When the mobile unit is idle, it attempts to tune to the other access point units in its table to verify the information in the table and to determine which access point units are the best candidates for roaming. This verification is very brief, about 20 milliseconds per check. After a short time, the mobile unit will have a short list of (e.g. 2 or 3) access point units that it could roam to if necessary.

If the mobile unit cannot send messages reliably to its current access point unit, it registers with a new access point unit. The new access point unit will contact the old access point unit over the wired portion of the network and inform the old access point unit that the mobile unit has roamed away. If any messages were waiting for the mobile unit, they are transferred to the new access point unit and then relayed to the mobile unit. If there are no buffered messages, this handoff is very quick, about 20 milliseconds. Once a mobile unit has been

authenticated by one access point unit, it does not have to be authenticated again.

The mobile units first listen. Specifically, they scan the frequencies in search of an access point unit. As soon as they find an access point unit, they adopt its NETID (network ID number) and hopping sequence and attempt to register. If registration is successful, the mobile unit automatically learns about the other NETIDs that are in use in the same system and can roam to a different access point unit if necessary. If the mobile unit is unable to register with any access point unit, it will attempt to form an ad hoc group (explained in detail above).

Access point units communicate with each other over the wired LAN using the access point unit-to-access point unit protocol (carried within the conventional bridge-to-bridge protocol of 802.1D). Before an access point unit starts up its wireless interface, the access point unit will: 1) pick a random NETID (the first one it tries could, for example, be the NETID it used previously) and hopping sequence, 2) find all other access point units connected to the same wired LAN system, 3) multicast a message indicating that a new access point unit is about to use that hopping sequence, and 4) listen to hear any access point units that respond with a conflict message. If there is a conflict, the access point unit repeats the process until it finds a free NETID and hopping sequence. Overlapping the communication ranges of individual access point units on a single network therefore is possible by using different hopping sequences.

Each NETID maps into a hopping sequence. Every network unit (access point unit or mobile unit) has a table of all hopping sequences. If a unit wants to communicate with other units using a particular NETID, it must listen until it hears an MPDU with a matching NETID. One method for quick scanning is to cycle through the

hopping sequence backwards, lingering on each frequency for 20 milliseconds or until a matching NETID is found. When a matching MPDU is heard, the other information necessary to synchronize, the HOPTICK is also heard. The HOPTICK indicates when to move to the next frequency in the hopping sequence. Units must always receive at least one MPDU before they start transmitting on an active NETID.

A mobile unit registers with an access point unit through a message exchange as defined above. The mobile unit sends a RegisterFirst message containing information about the identity of the mobile unit. The access point unit responds with a RegisterResponse message. If the registration was successful, the access point unit responds with a RegisterResponse message containing an APTable which describes the wireless environment. If the registration was not successful, the RegisterResponse message contains a NACK rather than the APTable. The RegisterFirst and RegisterResponse messages are the only messages that are transmitted in unscrambled form. All other interactions with the access point unit take place through scrambled messages.

Once a mobile unit successfully registers with an access point unit, the access point unit with which the mobile unit has registered will relay messages from the wired LAN for the mobile unit and will relay messages to the wired LAN for the mobile unit. A registered mobile unit is able to access the wired LAN through the access point unit and unscramble any messages sent using the access point unit's NETID.

If a mobile unit is not registered, it is unable to send or receive messages from the wired LAN. It is also unable to interpret messages sent by the access point unit other that the RegisterResponse. An unregistered mobile unit can, however, search for other mobile units that want to operate in an ad hoc network.

A mobile unit decides to switch to a new access point unit by a decision made automatically by the mobile unit. A mobile unit decides to switch to a new access point unit when its quality of service through the current access point unit degrades. Quality of service is measured by the number of retries needed to send an MPDU. A "pull" method is used for a handoff of a mobile unit from an access point unit to another access point. When a mobile unit wishes to roam, it contacts a new access point unit and attempts to register. If registration is successful, the new access point unit contacts the old access point unit over the wire (using the access point unit-to-access point unit protocol) and "pulls" the context of the mobile unit to the new access point unit. The roaming mobile unit contacts the new access point that is on a different hopping sequence by accessing the information that it needs in the APTable.

Mobile units receive a copy of the APTable each time they register or roam to an access point unit. For each access point unit in the APTable, its hop offset and tick offset are stored. The APTable therefore tells the mobile unit where the new access point unit is relative to the mobile unit's hopping sequence. The information in the APTable allows the mobile unit to tune directly to the new access point unit's current frequency without scanning. Because all access point units on the same network use the same scrambling key, a roaming mobile unit is able to interpret messages from any access point unit on the same network. A mobile unit that has not been authenticated will not be able to roam because it will not understand the scrambled messages from access point units.

The MPDU TYPE field has a bit that indicates User or System Packet. If the bit is zero, the MPDU contains user data and the payload is passed to the host. If the bit is one, it is a System Packet. This bit is used to differentiate between MPDUs that should be processed by

the unit firmware and MPDUs that should be passed to the host.

Power management is realized by a two message exchange. A mobile unit that will be operating in low power mode sends a SleepRequest message to its access point unit. If there are messages pending in the access point unit for that mobile unit, the access point unit will reply with a SleepNotOk message and will forward the remaining messages. If there are no messages pending and resources are available, the access point will respond with a SleepOk message. The SleepRequest message contains a parameter that indicates the duration of the requested sleep time. While the mobile unit is "sleeping", the access point unit buffers any messages that are intended for the mobile unit. When the mobile unit wakes up, the mobile unit sends a WakeUp message to the access point unit and normal operation resumes. There is also a ForcedSleep message that the mobile units may send in an emergency situation such as when they are about to lose power and have no control over the loss of power. There is no response from the access point unit for a ForcedSleep message.

In most file server network environments, the client workstation initiates the data transfers. The client therefore can control when its network interface will be idle. The only unexpected messages received by the client are watchdog packets from the server. These watchdog packets are used by the server to verify that the client is still there. If the client does not respond within a specified amount of time, the server will terminate the session. If such a file server is operating on a network comprising an access point unit and an associated mobile unit, the access point unit may respond to watchdog packets addressed to the sleeping mobile units instead of simply buffering them and allowing the server to terminate the session.

The APTable is an array containing elements of the following structure:

```
struct APinfo {
  ushort netID;
  addr nodeID;              //1148 bit node ID of AP
  uchar hopSeq;             //11 the hopping sequence used by
                                AP
  uchar hopOffset;          //11 hop offset WRT my hop index
  uchar tickOffset;         //11 tick offset WRT to my
                                HOPTICK
  uchar neighborStatus;     //11 near, far, or unknown...
  }
```

Access point units keep track of each handoff that they are involved in (to or from this access point unit) during roaming. After a short period of time, an access point unit has a complete list of all of the other access point units that it is possible to roam to--the "likely neighbor" list. This list will include access point units that are out of radio range of the source access point unit and even some access point units that are reachable only by going through a dead area. The "likely neighbor" list is encoded in the APTable by setting the neighborStatus field to "near."

Each time a mobile unit registers with an access point unit, it gets a copy of that access point unit's APTable. This occurs when the mobile unit first enters the system as well as each time there is a handoff. Mobile units therefore quickly learn about their changing environment as they move around the system. Access point units alone can not provide the hopOffset and the tickOffset, because they are not able to hear all of the other access point units over the wireless medium. This dynamic information must be supplied to the access point units by the mobile units.

Mobile units scan the radio frequencies to determine the relative timing of the other access point units. This is also known as the "break" approach. Mobile units

occasionally go off line when they are idle and listen to other frequencies. They linger on each frequency until they hear any MPDU from or forwarded by an access point unit, or until 10 milliseconds has passed. Eventually, they will hear all of the access point units that are within range. Mobile units occasionally do a "complete scan" of all of the frequencies in the manner described above during a single "break" to look for new access point units coming within communication range. It is not guaranteed that every access point unit within range will be heard because access point units continue to hop during a scan and each will hop eight times during a complete scan. A single, "complete scan" will not usually find all of the available access point units. Several short scans work as well as a single long scan.

Mobile units also periodically do "short scans" to look for known neighboring access point units. After several short scans, they will have heard every possible known access point unit. It has been found that 100 milliseconds is a convenient duration for the scan, since a mobile unit could simply skip a hop while scanning. The mobile units do not send messages when they scan; they simply receive. If a scanning mobile unit receives any MPDU from an access point unit, it can determine the hopOffset and tickOffset.

Units using a given hopping sequence visit each frequency exactly once during a hopping sequence and always go through the sequence in the same order. Every unit has the hoptable[] which is the array of hopping sequences. There is a function or a table for mapping NETID into a particular hopping sequence. Every frame contains NETID. When a mobile unit scans other frequencies and hears an MPDU, the mobile unit determines which hopping sequence is being used. Knowing the frequency and the hopping sequence, the mobile unit can compute the hop index (or hop number). The hop index is

the entry in the hopping sequence array that the access point unit is using.

The hopOffset for a given NETID is the difference between my hop index and the hop index observed on the current frequency for that NETID. If myNETID is on hop hoptablet[myNETID,i] (that is the ith entry in the hopping sequence for myNETID) and another access point unit is on hop hoptable[hisNETID,i+5], then the hopOffset for hisNETID is 5. One second later (or 10 hops), I will be on hop hoptable[myNETID,i+10] and the other access point unit will be on hop hoptable[hisNETID,i+15]. This offset will remain the same until one of the access points using myNETID or hisNETID is reset.

The determination of tickOffset is simpler. When mobile units scan, they continue to maintain their original HOPTICK timer. As soon as they hear any frame on a scanned frequency, they can determine the tickOffset. tickOffset = hisHOPTICK - myHOPTICK.

The mobile units record the information they gather during each scan in their local copy of the APtable. If any new information was collected, the mobile units send an update message to their access point units at the end of the scan. Whenever an access point unit changes its APTable, it broadcasts an update to all of its mobile units. At initialization, the APTable contains only the netID and NETID of the other access point units. hopOffset and tickOffset are zero and neighborStatus is "unknown".

Then, when the first mobile unit registers with an access point unit and subsequently hears other access point units during a scan, any information it gathers comprises a change and therefore cause an update to be sent to the access point unit of the mobile unit. As more mobile units enter the system and scan other frequencies, more APTable[] entries are filled in. After all of the information in the APTable is filled in, there will be very few changes and therefore very few updates.

This system is a learning system. The longer the access point units have been functioning in a network, the better the information in their APTables. The more mobile units in a network and the more handoffs, the better the information about the network.

Figure 25 is an diagram of a mobile unit designated Mu which is roaming. The circle labeled 1 in Figure 25 represents an area within which access point number 1 (not shown) can communicate with a mobile unit. Access point number 1 will be able to communicate with a mobile unit Mu if mobile unit Mu is located within area 1. The circles labeled 2-5 are similar areas of other access points (not shown) of the illustrated network. The arrow illustrates the roaming travel of mobile unit Mu.

Figure 26 is an instance diagram illustrating the operation of software associated with the roaming of mobile unit Mu of Figure 25. The table below is a corresponding list of function calls with may be performed by software in an example of roaming. The circled numbers in the instance diagram of Figure 26 correspond with the numbers labelling the function calls in the table below.

Table 3

Function Calls

```
   1    insertMsg();    /* last driver description */
   2    getMsg();       /* it's a RegisterRoam */
   3    lookup();       /* lookup returns 0 */
   4    processMsg();   /* AP needs to build an MuFsm */
   5    ctor();         /* arg says it RegisterRoam */
   6    ctor();         /* the MuFsm creates it deque */
   7    insertMsg();    /* handoffReq msg */
        eventually, the old Access Point Unit responds by the
             hardwired network...
   8    getMsg();       /* it's a HandoffData */
   9    lookup();       /* finds the roaming MuFsm */
  10    processMsg();   /* the MuFsm enqueues the data */
  11    getMsg();       /* it's a HandoffComplete */
  12    lookup();       /* gets the roaming MuFsm */
  13    processMsg();   /* still in the Roaming state
  14    insertMsg();    /* RegisterRoamAck */
  15    insertMsg();    /* the forwarded data */
```

The function call software flow begins with the mobile unit Mu roaming out of communication range of access point unit 1. When the mobile unit senses that it has lost contact with access point unit 1, the mobile unit Mu transmits a message to access point unit 3 requesting to register with access point unit 3. The mobile unit Mu knows that access point 3 is a likely now to be in communication range from the APTable storing likely access points for roaming. When the wireless driver of access point unit 3 receives the message, the insertMsg() function is called which is represented in Table 3 with number 1. As illustrated in Figure 26, this is a function of the deque. Sometime later, the dispatcher instructs the deque to retrieve the identification number of the mobile unit that sent the message. This is represented in Table 3 with number 2. The dispatcher then reads the identification number out of the deque and performs a lookup of the identification number. This is represented Table 3 with number 3. The lookup returns a value of zero indicating that the mobile unit is a new mobile unit from which access point unit 3 has not yet heard. Process control is then handed off to the core software of access point unit 3 as represented by number 4. A software object is then created for the new mobile unit to be managed by access point unit 3. The access point unit then builds a state machine to manage messages between the new mobile unit and access point unit 3. The Ethernet driver then outputs a handoff request message for access point unit 1 which is represented in Table 3 by number 7. Eventually the old access point unit, access point unit 1, responds across the Ethernet infrastructure and forwards any Ethernet messages it was storing for mobile unit Mu which had roamed from its communication range. The dispatcher of access point unit 3 then instructs the queue to supply the dispatcher with the next message. The message is determined to be handoff data as represented in

Table 3 as number 8. The dispatcher then looks up the point identification number as represented in Table 3 by number 9. Because the mobile unit has been heard from before, the dispatcher instructs the finite state machine associated with the new mobile unit to process the message. This is represented in Table 3 as number 10. The forwarded data is queued for transmission to the mobile unit. The finite state machine then determines that the message indicates that the hand off from access point 1 to access point 3 is complete. This is represented in Table 3 as number 11. The finite state machine therefore formulates a RegisterRoamAck message and does an insertMsg() into the outgoing wireless driver. This is illustrated in Table 3 as number 14. The RegisterRoamAck message is then sent as represented in Table 3 as number 15.

## TIME BOUNDED SERVICE

The above described protocol is a Media Access Control (AC) layer (see Figure 27) protocol with enhancements for improved reliability, support of hidden units, and associated wireless LANs. There may be two types of service provided to the higher level logical link control (LLC) layer protocol: an asynchronous service and a time bounded service. The asynchronous service as described above is a Listen Before Talk (LBT) protocol that provides low-delay, asynchronous delivery of MAC Protocol Data Units (MPDUs) between any units within direct radio communication range of each other. The time bounded service is used to transfer data between a wireless unit and an access point unit at a constant rate. Time bounded traffic and asynchronous traffic can share the same medium. Time bounded service is optional, but every wireless access point unit and mobile unit supports the asynchronous service and the infrastructure extensions.

The time bounded service protocol is physical layer independent. It works equally well with other wireless physical (PHY) layers. The time bounded service supports e.g. real time voice traffice or video traffic. The time bounded service operates under the assumptions that: 1) it is generally better to drop a voice "packlet" than to delay it for a long time, 2) voice traffic does not generally have to be as reliable as data traffic, 3) it is generally better to deliver a voice packlet in error than to delay it for a long time, and 4) voice traffic always flows through the access point unit. The time bounded service transfers data over a point-to-point connection between an access point unit and a wireless node (mobile unit) that are within range (defined by the physical layer) of each other. The protocol minimizes the delay variance between successive frames sent over the connection.

The time bounded traffic is controlled by access point units. Mobile units wishing to send time bounded messages negotiate with their access point unit to set up a voice connection using the asynchronous service. The access point unit grants the connection request only if there is enough available bandwidth to support a new call. If the access point unit accepts the connection, it assigns a local identifier that is used in subsequent packet headers for that connection. One of the parameters of the call setup is quality of service. This parameter dictates the timing between time bounded packlets for a connection. Time bounded data is transferred in fixed length "packlets". A time bounded data packet contains an ATM cell (usually containing digitized voice) and a wireless MAC header. After a call is established, the access point unit sends time bounded packlets to the wireless node at fixed intervals. The access point unit and wireless node (mobile unit) exchange a series of modified RTS and CTS frames with data attached to transfer

the time bounded data. There are two types of MPDUs for time bounded service: inbound MPDUs and outbound MPDUs. All time bounded MPDUs are directed; there is no time bounded multicast. Each time bounded MPDU is a two frame exchange.

Inbound MPDUs (see Figure 28) cause data to flow from a wireless node (mobile unit) to an access point unit. This is a flow of data into the wired infrastructure. The access point unit sends an RTSI frame and the wireless node (mobile unit) responds with a CTSI frame that includes the time bounded data. The access point unit sends RTSI frames at fixed intervals, t2, for the duration of the connection. Each time the access point unit receives an RTSI, it responds with an immediate CTSI frame. There are no ACKs (acknowledgement) and no retransmissions. If one of the frames is lost, it is ignored. The gap between the Inbound MPDUs, t3, is available for asynchronous traffic or other time bounded MPDUs.

Outbound MPDUs (see Figure 29) cause data to flow from an access point unit to a wireless node. This is a flow of data out from the wired infrastructure. The access point sends an RTSO frame which includes the data. The wireless node responds with a CTSO frame. RTSO frames are sent at a constant interval, t2, as illustrated in Figure 29.

All time bounded service frames have the same header format. The header is always preceded by a preamble of 32 bits and a Start Frame Delimiter (SFD) which is eight bits long. Frames such as RTSI and CTSO append an 8 bit CRC to the header. The time bounded header comprises several fields. The VCID field contains a virtual circuit identifier which is assigned by the access point unit each time a time bounded connection is established. VCID is unique within a basic service area defined by the transmission range of the PHY layer. The NETID/VCID

combination is unique within a domain. The VCID field contains an 8 bit number. The MORE field indicates the number of times (e.g. 4) in the future to reserve the medium for MPDUs from this connection. The field GAPTIME indicates the time between successive MPDUs from this connection measured in slot times.

Figure 30 is a diagram of an RTSI frame. RTSI frames are sent from an access point unit to a mobile unit and are used to maintain timing between MPDUs of the connection. The field TYPE is equal to 1x, where x is the MPDU type. The field CONTROL = {TB = 1, HIERARCHICAL = 0, AP = 1, RETRY = 0}. The field CRC-8 is an 8 bit CRC computed on the entire frame.

Figure 31 is a diagram of an CTSI frame. CTSI frames are sent from the mobile unit to the access point unit in response to an RTSI frame. CTSI frames contain the data payload for the MPDU. The field TYPE = 3x, where x is the MPDU type. The field CONTROL = {TB = 1, HIERARCHICAL = 1, AP = 0, RETRY = 0}. The field DATA is the payload of the inbound MPDU. The field CRC-32 is a 32 bit CRC computed on the entire frame.

Figure 32 is a diagram illustrating a RTSO frame. RTSO frames carry the data for outbound connections. RTSO frames are sent from an access point unit to a mobile unit. The field TYPE = 5x, where x is the MPDU type. The field CONTROL = {TB = 1, HIERARCHICAL = 0, AP = 1, RETRY = 0}. The field DATA is the payload of the outbound MPDU. The field CRC-32 is a 32 bit CRC computed on the entire frame.

Figure 33 is a diagram illustrating a CTSO frame. CTSO frames are sent from the mobile unit to the access point unit and are used to propagate timing information for an outbound connection. The field TYPE = 7x, where x is the MPDU type. The field CONTROL = {TB = 1, HIERARCHICAL = 1, AP = 0, RETRIAL = 0}. The field CRC-8 is an 8 bit CRC computed on the entire frame.

The length of the payload for time bounded MPDUs is a fixed length. A digital distribution system for time bounded connections may be based on ATM cell switching. One option for IEEE 802.11 is to specify the payload of the time bounded MPDUs as an ATM cell or multiple ATM cells. Using this assumption about the payload, it may be possible to simplify the frame check sequence for time bounded MPDUs.

The time bounded service minimizes the MPDU delay variance. One way to achieve minimum delay variance is to provide guaranteed bandwidth for time bounded connections through a central point coordination function. The access point units are a location for this centralized coordination function.

The time bounded service operates in conjunction with the asynchronous service which uses a distributed control, multiple access protocol. Mobile units operating in the asynchronous mode do not have to check with any point coordination function before transmitting. If a mobile unit perceives that the network is idle, it may initiate an asynchronous transmission, but this is a problem for the time bounded service. Once a call is established, time bounded connections should not contend for the media for each MPDU during a connection. Because of the minimum delay variance requirement, time bounded MPDUs cannot defer to other traffic once the connection has been started.

A solution is a two step process that is a combination of distributed and centralized control. Call set up and bandwidth allocation are managed by a point coordination function. All time bounded connections are set up through the access point unit. Access point units are aware of traffic level for their service area. Access point units know many calls are in progress and the number of asynchronous nodes in their area and they manage the bandwidth accordingly. The access point units also

control the timing of the connection and initiate all time bounded MPDUs. Once a connection is granted, the access point unit sends the first MPDU at a time when the media is idle and coordinated with other connections that are ongoing. After the first MPDU, subsequent MPDUs for that connection are sent at fixed intervals. The access point unit then manages the bandwidth according to a predetermined policy.

Figure 34 illustrates two two examples of access points managing bandwidth. As illustrated in the upper line in Figure 34, the access point unit can group all of the time bounded voice connections together (VC1 and VC2 in this example) in order to keep maximum size gaps available for asynchronous traffic. Alternatively, as illustrated in the lower line in Figure 34, the access point unit spaces the voice connections evenly in an attempt to minimize latency for the asynchronous service. The operation of the MAC layer may be identical regardless of the particular predetermined policy used by access point units.

The second part of the time bounded protocol is a distributed coordination function to reserve the bandwidth for time bounded MPDUs and to inform all of the mobile units in the service area to prevent contention with time bounded frames. This is done by an extension to the above described carrier sense method of the asynchronous service. Each time bounded MPDU has a fixed length. Any node that receives a time bounded frame can determine the length of time the network will be busy for that MPDU exchange. There is a GAPTIME field in all time bounded frames that specifies when the next MPDU for this connection will be sent. GAPTIME is used to reserve bandwidth for subsequent frames of the same voice connection. Any unit that receives a time bounded frame assumes that the network will be busy for the length of one timed based MPDU after GAPTIME time has passed. This

"reserve ahead" mechanism prevents other nodes within range of either end of the voice connection from contending for the network during the transmission of time bounded frames for this voice connection. An additional field in the header called MORE which indicates the number of intervals the reservation covers. With MORE and GAPTIME, it is possible to reserve ahead up to e.g. four transmission opportunities. This makes the time bonded service mechanism more reliable because each voice connection will actually reserve a specific transmit opportunity up to four times.

After an access point unit and a mobile unit have successfully exchanged the first MPDU of a time bounded voice connection, the medium is reserved at both ends of the voice connection. However, the access point units initiate the timing of a voice connection without knowing the state of the reserved ahead state of the mobile unit. To resolve this potential problem, the response to the call setup message and the first MPDU of the voice connection are timed as if they were successive MPDUs from an established voice connection. The mobile unit will respond with a confirmation message if its network allocation vector indicates the network will be available for the first MPDU transfer. Otherwise, the mobile unit will not respond and the access point unit will attempt to initiate the voice connection at a different time.

Most of the complexity of the time bounded service is in the point coordination function which is the access point unit. However, there are enhancements to the distributed coordination function in every mobile unit to support time bounded service. In particular, the carrier sense mechanism is expanded. A wireless mobile unit that supports the asynchronous service only needs to know about the current transmission in order to implement carrier sense. When time bounded service is added, however, each

mobile unit must keep a map of what will happen in the future.

The net allocation vector (the reserve ahead mechanism as implemented in software) must be able to record network availability for some period of time into the future. The transmit procedure for asynchronous MPDUs is slightly more complex. A sending mobile unit must find an idle period that is long enough for transmission of the entire MPDU (rather than the first idle period) before contending for the media. These changes are not an unreasonable burden on the MAC implementation.

The time bounded service works well with collocated or overlapping service areas. All nodes maintain their own net allocation vector based on the frames they "hear" on the network. This includes frames with matching NETID and those frames from different service areas with different NETIDs. Network reservations for time bounded connections are automatically propagated only to those mobile units that are within range of one of the ends of a connection.

Figure 37 is a diagram illustrating four mobile units a, b, c and d (designated as square boxes), as well as two access point units AP1 and AP2 (designated as rectangular and rounded boxes). Mobile unit c can hear both access points AP1 and AP2. Mobile unit c's net allocation vector will include reservations for all of the time bounded traffic for AP1 and AP2. Mobil unit d can only hear AP1 and mobile unit c. If mobile unit c is registered with AP1 and has established a time bounded connection, then mobile unit d's net allocation vector simply includes any asynchronous traffic and all of the time bounded traffic from AP1. Traffic from AP2 and mobile units a and b will not affect mobile unit d.

If mobile unit c is registered with AP2 instead, mobile unit d's net allocation vector will include everything described above plus the reservations for any

time bounded traffic between AP2 and mobile unit c. As soon as the first time bounded MPDU is sent from AP2 to mobile unit c, mobile unit d will learn about the relevant time bounded traffic from the other service area.

In actuality, the boundaries of service areas are not as well defined as illustrated in Figure 37. Consider the situation in which mobile unit b is moving towards AP1. When mobile unit b initiated a time bounded connection with access point AP2, there were no conflicting reservations in its network allocation vector. As it crosses the boundary into access point unit AP1 service area, mobile unit b may encounter new traffic with reservations that conflict with its connection with access point unit AP2. If the conflict is for a single MPDU, mobile unit b could choose to ignore the conflict and remain silent during the conflicting period. The result would be a dropped MPDU, but the connection would continue unhindered after the first successful new MPDU exchange. If the conflict is with another time bounded connection, mobile unit b must abort its connection. It does so by sending a CTSO or CTSI with GAPTIME and MORE = 0. At that point, mobile unit b can initiate a new connection with access point unit AP2 or associate with access point unit AP1 and establish a connection there.

## RADIO CIRCUITRY

Figure 8 is a detailed schematic of a radio transmitter for transmitting at 1 Mbits/sec as well as a radio receiver for receiving transmissions at 1 Mbits/sec in accordance with Part 15.247 of the Federal Communications Commission (FCC) rules which govern Industrial, Scientific and Medical (ISM) bands. To support spread spectrum frequency hopping in the 2.4 GHz ISM band, the transmitter and receiver can be tuned to one of 82 different frequencies spaced at 1 MHz intervals: 2401, 2402, ...2481 and 2482 MHz. Range between a

transmitter of a unit and a receiver of another unit is approximately 150 feet in a typical office building where a typical office building is assumed to have some sheetrock walls such that the attenuation is approximately a square law power fall off for the first 3 meters and a 4th law fall-off thereafter. Transmitter power output of one specific embodiment is approximately +17dBm and receiver sensitivity is approximately -85 dBm.

Figure 38 is a block diagram of a low cost, low power and small size radio receiver and transmitter circuit. The incoming signal is received on a printed circuit tab antenna 3800, is filtered by a printed circuit filter 3801, and supplied by a switch T/R switch 3802 to the receiver low noise amplifier 3803. The receiver LNA 3803 uses two cascaded bipolar amplifiers. Overall frequency response is tailored to provide substantial attenuation below 1.5 GHz in order to minimize problems causes by cellular portable telephones in the 825-845 MHz range. The output of the receiver LNA 3803 is supplied through a 90° splitter 3804 to two mixers 3805 and 3806 which are connected in image-reject configuration with a local oscillator signal 3807 coupled in phase and with RF inputs 3808A and 3808B coupled in quadrature. The mixer outputs 3809A and 3809B are passed to a low noise preamplifier. A quadrature network 3810 combines the outputs of these preamplifiers and the resulting signal is passed to a three pole LC filter 3811 centered at 4.75 MHz. The remainder of the intermediate frequency (IF) signal gain required is provided by an limiting IF amplifier 3812 in integrated circuit form. The limiting IF amplifier has an additional single pole filter for control of the overall noise bandwidth. Quadrature detection is used to demodulate the incoming FSK signal to generate the output signal RXDATA of the radio receiver. A DC coupled buffer amplifier drives a post-demodulation filter and then a slicer 3813 which provides a continuous 0 or 1 logic

output bit stream. The slicer threshold is set automatically by signal PACDET which is a pass filtered version of output of the post-demodulator filter.

When there are no incoming bits, the threshold filter time constant is short, about 20 microseconds. This allows the threshold to adjust rapidly to offsets caused by frequency errors. Once the radio controller ASIC U6 (see Figure 7) has determined that a valid frame is being received, then the threshold filter is switched to operate as a hold circuit to maintain the correct DC threshold to receive the remainder of the frame. This technique allows rapid adaptation to frequency offsets while at the same time allowing any data pattern to appear in the data field of a data frame. The frequency control block of Figure 38 is illustrated in greater detail in Figure 39. In figure 38, incoming bits to be transmitted are supplied to the input TXDATA of the frequency control block 3814. The the output of the frequency control block is supplied to a voltage doubler 3815. The output of the voltage doubler 3815 is filtered and then amplified by a transmitter power amplifier (PA) 3816 in three bipolar transistor stages. Overall frequency response is tailored to provide a rejection notch at the VCO frequency. The output of the power amplifier is coupled through T/R switch 3802 by a PIN diode (which is switched to its low resistance state), through printed filter 3801, and to tab antenna 3800 which is printed on the printed circuit board of the mobile unit. Coupling of the power amplifier 3816 output to the antenna through the PIN diode and coupling the PIN diode to the antenna 3800 via a quarter wave line ensures that almost all the power passes to the antenna. The printed circuit filter 3801 provides bandpass response at 2.4 GHz and band reject notches at 4.8 GHz and 7.2 GHz.

On receive, power amplifier 3816 is biased off. A voltage is, however, applied to the collector of the power amplifier 3816 resulting in the output of the power

amplifier being a primarily capacitive impedance. The PIN diode of the T/R switch 3802 is biased to its high impedance state. The quarter wave line is therefore connected to an input matching network of the receiver LNA. Nearly all the received signal power is therefore coupled to the receiver LNA 3803 resulting in an overall 2-pole bandpass response.

Figure 39 is a more detailed diagram of the frequency control block 3814 in Figure 38. The frequency control block 3814 comprises two phase-locked loops. The first phase-locked loop, called the low frequency (LF) phase-locked loop 3900, generates a reference signal for the second phase-locked loop, called the high frequency (HF) phase-locked loop 3901. The HF phase-locked loop includes a prescaler 3902 and controls a voltage controlled oscillator (VCO) 3903. The prescaler 3902 is set to divide by 512 on receive and by 513 on transmit. The IF varies slightly across the band because the IF is effectively 1/513 of the transmit frequency. All frequencies are derived from a single 20 MHz crystal. The parameters for transmitter/receiver operation in each of the 82 frequency hops is set forth below:

Table 4

| Parameter | Value | Characteristic |
|---|---|---|
| Frequency hops | 2401,2402....2482 MHz | 82 hops, spaced at 1 MHz intervals |
| Frequency error | +/-150 kHz maximum | for less than 3 dB degradation in error rate |
| Data rate | 1 Mbps | fixed data rate |
| Data polarity | Non-inverting | data "1" at radio input produces data "1" at radio output |

| Parameter | Value | Characteristic |
|---|---|---|
| Modulation | GFSK, 700 kHz prefilter +/-175 kHz deviation | data "1" corresponds to negative frequency deviation |
| Transmitter power output | +17 dBm | 20 dB power bandwidth is less than 1 MHz; meets Part 15.247 of FCC rules |
| Receiver sensitivity | -85 dBm | for 1 in $10^6$ error rate |
| Error rate floor | better than 1 in $10^7$ | equivalent to 1 in 2000 frame loss rate for maximum length frames |
| Maximum frame length | 5 ms | equivalent to 625 bytes at 1 Mbps |
| Line coding | NRZ, no bit stuffing or other manipulation required | modem frequency response extends to DC |
| Decision threshold | adapts to +/- 5% error within 15 $\mu$s | corrects for TX/RX frequency errors; fixed after-frame detect |
| TX/RX turnaround time | less than 50 $\mu$s | MAC protocol defines at least 50 $\mu$s gap between radio frames |
| Slot change time | less than 5 ms | hops are fixed length, nominally 100 ms |
| Image response | -20 dB | 9.5 MHz below receive hop |
| Other receiver spurious | better than -60 dB | |

| Parameter | Value | Characteristic |
|---|---|---|
| Transmitter harmonics | better than -30 dBc | meets Part 15.247 of FCC rules |
| Other transmitter spurious | better than -60 dBc | |

Switching from transmit to receive involves only the HF phase-locked loop 3901 which settles in less than 25 microseconds. Changing from hop to hop requires settling the LF phase-locked loop 3900 to a different frequency which may take up to 2 milliseconds of settling time. Synthesizer R and N values are carefully selected for each hop to maximize the reference frequency. After settling is complete, the LF track-and-hold 3902 is set to hold for the remainder of the hop.

During transmission, the HF phase-lock loop 3901 is first set to the correct frequency by switching the prescaler 3902 to divide by 513. The correct tuning voltage is then held for the remainder of the transmit frame. VCO pulling caused by imperfect power supply regulation and RF pickup is eliminated by switching all the transmitter stages on before the tuning voltage is held. A separate VCO control input 3904 is used to apply transmit modulation. During receiving, this input 3904 is unused and therefore is biased to 2.5 voltage and tri-stated. On transmit, the TXDATA transmit data input drives the input to 0 volts or 5 volts for logic 0 and logic 1, respectively. Tri-stating is removed only after the HF loop has been held. This technique provides a modulation bandwidth down to DC resulting in accurate modulation for any data pattern. Droop caused by leakage in the VCO hold capacitor 185 and associated circuitry during transmission is not significant for up to 5 milliseconds of frame duration.

At the start of each new hop, all transmit and receive operations are suspended by the 80C188 microcontroller U4 (see Figure 7) via the radio controller ASIC U6. The LF phase-locked loop is programmed for the new hop and the LF track-and-hold 3902 is enabled. After the phase-lock loop has settled, the 80C188 microcontroller resumes normal operation. When a frame is transmitted, the HF loop is switched to the correct frequency by the radio controller ASIC driving the TXEN1 input to the radio circuit to a logic 1 voltage level. After 16 microseconds, the radio controller ASIC applies a logic 0 to the TXEN2 input (see Figure 8) of the radio circuit to switch on all the transmitter stages. At this point, some pulling of the HF VCO frequency can be expected, so a further 16 microseconds of settling time is allowed before the radio controller ASIC applies a logic 0 TXEN3 input to the radio circuit. The TXDATA input to the radio circuit is then driven by the radio controller ASIC. At the end of the transmit frame, the radio controller ASIC returns all radio control signals to their receive states to conserve power. The only sequencing requirement is that TXEN2 should be deactivated before TXEN1.

Interface between the radio controller ASIC U6 and the radio circuitry is set forth below:

Table 5

| Name | I/O | Description |
|---|---|---|
| LFLOCK | In | Controls LF synthesizer lock. CMOS logic 1 locks the LF loop, logic 0 holds the LF VCO control voltage. Sets LF synthesizer for each hop. |
| RXEN | In | Switches on the RX LNA, IF amplifiers, and output comparator. CMOS logic 0 enables the RX, logic 1 disables. Uses for power management. |

| Name | I/O | Description |
|---|---|---|
| LFEN | In | Switches on the LF VCO, synthesizer, and crystal oscillator. CMOS logic 0 enables the receiver, logic 1 disables. Used for power management. |
| HFEN | In | Switches on the HF loop, including the prescaler and HF VCO. CMOS logic 0 enables the receiver, logic 1 disables. Used for power management. |
| RXDATA | Out | Derived from an open collector comparator via a resistor network. For a CMOS load, logic low is less than $0.3*V_{cc}$ and logic high is greater than $0.7*V_{cc}$. Data 1 corresponds to logic 1. RXDATA is not clocked by the radio. It represents the best estimate of the current received data state. |
| TXDATA | In | CMOS logic 1 is data 1, which produces a deviation below the nominal carrier frequency. The radio requires TXDATA to be tristated when the transmitter is not operating. |
| RSSI | Out | Received signal strength indication. This is an analog level which indicates the strength of the received signal. One possible application is to drive a Schmitt trigger to provide an energy-detect LBT function. High output corresponds to signal present. |
| TXEN3 | In | Controls HF loop lock. CMOS logic 1 locks the loop, logic 0 holds the VCO control voltage. Logic 0 is required before modulation is applied. |
| PACDET | In | Switches the receiver slicer threshold time constant. CMOS logic 0 must be asserted while a frame is being decoded, that is, after 8 bits of dotting followed by the 16 bit frame sync word have been correctly received. CMOS logic 1 is required at all other times. |
| ACT | In | CMOS logic 0 lights activity LED |
| LNK | In | CMOS logic 0 lights link integrity LED |
| TXEN1 | In | Switches the frequency control system between transmit and receive. CMOS logic 1 is for receive, logic 0 for transmit. |

| Name | I/O | Description |
|---|---|---|
| SYNDATA | In | Sets the divider ratios in the frequency synthesizer. See Fujitsu data book for a description of synthesizer programming. |
| SYNDATA CLK | In | Clock SYNDATA into the synthesizer |
| SYNLE | In | Load enable signal for the synthesizer |
| TXEN2 | In | Switches transmitter on. CMOS logic 0 for transmit, 1 for receive |

The voltage signals at the outputs of the mixers 3805 and 3806 are very weak signals. The signal at the input of the IF amplifier may, for example, be in the microvolt level. Second order mixers are generally used in conventional receivers. In a possible receiver configuration, $f_{IF}=f_S-f_{LO}$, where $f_{IF}$ is the frequency of the IF signal, $f_S$ is the frequency of the incoming signal being received on the antenna, and $f_{LO}$ is the frequency of the signal supplied to the second order mixers by a local oscillator. Using a prescaler to generate a local oscillator having a frequency 512 times the frequency of a reference signal $f_{REF}$ requires a reference signal frequency of $f_{REF}=f_{LO}/512$. If, for example, $f_s$ were 2436.75 MHz, then $f_{REF}$ would be about 4.75 MHz. 4.75 MHz is, however, the frequency of the IF signal on the input of the IF amplifier because $f_{IF}=f_S-f_{LO}$. The conventional design therefore results in shielding requirement due to reference signal noise from the local oscillator appearing on the input to the IF amplifier and overwhelming the small amplitude intelligence signal.

In accordance with one embodiment of the present invention, third order mixers 3805 and 3806 are used in combination with a switched prescaler 3902 to build a very low cost and very compact frequency control system which is capable of rapid switching of the radio between transmitting and receiving. No special shielding between

the frequency control block 3814 and the IF amplification circuitry is needed. This allows the radio circuit to be inexpensively placed on one printed circuit board in a PCMCIA form factor. The third order mixers 3805 and 3806 may be HSMP2822 mixers made by Hewlett-Packard. Component U4 of the prescaler of Figure 8 may a HB507PF made by Fujitsu.

Using third order mixers eliminates the need during receiving for a local oscillator operating at the IF frequency. In accordance with one embodiment of the present invention, $f_{IF}=f_S-2f_{LO}$, where $f_{IF}$ is the frequency of the IF signal on the input of IF amplifier 3812, where $f_S$ is the frequency of the incoming signal to be received on antenna 3800, and where $f_{LO}$ is the frequency of the local oscillator signal supplied to mixers 3805 and 3806 from the frequency control block 3814.

In accordance with one embodiment of the present invention, it may be desired to receive an incoming signal having a frequency $f_S$ of 2436.75 MHz. Because the IF frequency $f_{IF}$ amplified by the IF amplifier 3812 is 4.75 MHz, $f_{LO}$ is 1216 MHz due to the use of third order mixers 3805 and 3806. Because the prescaler 3902 is set to divide by 512 on receive, the reference signal frequency $f_{REF}$ output by the LF phase locked loop is 2.375 MHz, not 4.75 MHz. The 2.375 MHz reference signal output of the LF phase locked loop therefore does not interfere with the weak 4.75 MHz IF signal being received at the input of the IF amplifier.

During transmit, the 2.375 MHz reference signal results in local oscillator frequency $f_{LO}$ of 1218.375 MHz on the output of the frequency control block 3814 due to the prescaler being set to divide by 513 during transmissions. This results in a signal being output to the antenna having the frequency $f_S$ of 2436.75 MHz due to frequency doubler 3815. The radio transmitter/receiver circuit can therefore be rapidly switched between receive

and transmit modes by switching the prescaler using the TXEN1 signal. To tune to different hop frequencies, the LF phase locked loop is controlled to change the frequency of the reference signal $f_{REF}$.

## ACCESS POINT UNIT HARDWARE

Figures 40A-M are a schematic of one embodiment of an access point unit in accordance with the present invention. Component U1 is a switching voltage regulator. Component U2 is a supervisory circuit which provides power on reset and watch dog timer functions. Component U3 is a 32-bit RISC SPARC processor made by Fujitsu. Component U4 is a programmable device containing "glue" logic, a burst mode DRAM controller, a bus controller, and an interrupt controller. The glue logic is realized as a Xilinx XC3120-5 component. Component U4 reads its configuration data from nonvolatile flash memory U11-U14. Components U6 and U7 comprise 256K of 32 bit wide DRMA. The DRAM are NEC 42S4260 256K x 16 components. Component U5 is an address MUX for the DRAM components U6 and U7. Components U8 and U9 are static random access memories (SRAM). Component U10 is a buffer to reduce loading due to flash memories U11-U14. The flash memory is realized as AMD 28F020 256K x 8 components. Component U15 is also a testability slot. Component U16 is a configuration memory in which various parameters related to roaming and other system functions are stored. This memory is maintained over power outages by a battery. Component U17 is a bus buffer. Component J12 is a PCMCIA connector. Component U16 is a buffer for the address bus whereas component U19 is a buffer for the data bus. Component U20 is an AMD 79C940 Ethernet UART chip which produces Ethernet packages from the contents of memory or converts Ethernet packages into information which is written into memory. The access point unit provides three types of connectors for coupling to a hardwired network connector. J13 is a 10 base T

connector (such as a common telephone land-line jack), component J14 is a BNC connector, and component J15 is an AUI connector. Components LED1, LED2, LED3, and LED4 are LEDs to indicate Ethernet traffic, wireless traffic, 10 base T link integrity and access point unit integrity. The PCMCIA header J12 shown in Figure 40H is a slot through which the access point unit controls a mobile unit. This PCMCIA header therefore allows the access point unit to use the radio circuit illustrated in Figure 8. Figure 41 is a block diagram of the access point unit schematic of Figure 40A-M.

## ACCESS POINT UNIT SOFTWARE

One embodiment of software executing in the access point unit of Figure 40A-M is described. Figure 42 is an instance diagram of an illustrative scenario of a mobile unit registering with an access point unit. The numbered function calls of the table below correspond with the circled numbers appearing in the instance diagram Figure 42.

Table 6

Function Calls

```
1    getMsg();
2    processMsg();
3    MuFsm::MuFsm();          /* constructor */
4    Deque::Deque();          /* constructor */
5    getNeighbors();
6    insertMsg();             /* RegisterAck */
7    restart();
8    getMsg();
```

Figures 43A-B comprise an instance diagram of another illustrative scenario of an embodiment operation of the software executing in the access point unit of Figure 40A-M. The function calls in the numbered lines of the table below correspond with the circled numbers appearing in the instance diagram of Figures 43A-B.

Table 7

Function Calls

```
1    insertMsg();
later ...
2    getMsg();
3    lookup();
4    processMsg();
5    insertMsg();
6    restart();
7    getMsg();
asynchronously ...
8    insertMsg later (the next dispatcher look at the
queue ... ();
9    getMsg();
10   lookup();
11   processMsg();
12   insertMsg();
13   restart();
14   getMsg();
```

Figure 44 is an instance diagram of yet another illustrative scenario of the operation of software executing in the access point unit of Figures 40A-M. The function calls in the numbered lines of the table below correspond with the circled numbers appearing in the instance diagram of Figure 44.

Table 8

Function Calls

```
1    getMsg();
2    processMsg();
3    MuFsm::MuFsm();      /* constructor */
4    Deque::Deque();      /* constructor */
5    getNeighbors();
6    insertMsg();         /* RegisterAck */
7    restart();
8    getMsg();
```

Although the present invention has been described in connection with specific embodiments for purposes of illustrating the disclosed invention, the invention is not limited thereto. Various modifications, adaptations and

substitutions of various features and elements disclosed may be practiced without departing from the scope of the invention as defined in the appended claims.

APPENDIX A

```
(*
-------------------------------------------------------------------------*)
(* scantbl.pas --- program to generate scaning table for                 *)
(* wireless network adapter.                                             *)
(*
-------------------------------------------------------------------------*)
program scantbl;
uses crt;
const
      NUM_HOPS=82;
      NO_SCAN=0; SCAN=1; UNASSIGNED=2;
type  stat_def = record
                  min, max, avg     : integer;
            end;
var
      offset, n                                                    :
integer;
      keep_going                                                   :
boolean;
      right_stats, left_stats                                      :
stat_def;
      scan_table, rights, lefts                                    :
array[0..81] of integer;


(*
-------------------------------------------------------------------------*)
(* procedure to display the table                                        *)
(*
-------------------------------------------------------------------------*)
procedure display_table(right_stats, left_stats : stat_def);
var
      n, x, y : integer;

begin
for n := 0 to (NUM_HOPS-1) do
      begin
      y := (n mod 21) + 2;
      x := (n div 21) * 16 + 1;
      gotoxy(x, y);
      if scan_table[n] = NO_SCAN then
            write('no scan')
      else if scan_table[n] = SCAN then
            write('scan   ')
      else if scan_table[n] = UNASSIGNED then
            write('-------')
      end;
gotoxy(1, 24);
write('right: min=', right_stats.min);
write('  max=', right_stats.max);
write('  avg=', right_stats.avg);
gotoxy(40, 24);
write('left: min=', left_stats.min);
write('  max=', left_stats.max);
write('  avg=', left_stats.avg);
end;


(*
-------------------------------------------------------------------------*)
(* procedure to write the table to disk                                  *)
(*
-------------------------------------------------------------------------*)
procedure write_table;
```

```
var
      filename          : text;
      low, high   : integer;

procedure write_a_line(start, finish : integer);
var
      n                       : integer;

begin
write  (filename, '             db      ');
for n := start to finish-1 do
      if scan_table[n] = NO_SCAN then
            write(filename, '0, ')
      else if scan_table[n] = SCAN then
            write(filename, '1, ');
if scan_table[finish] = NO_SCAN then
      writeln(filename, '0')
else if scan_table[finish] = SCAN then
      writeln(filename, '1')
end;

begin
assign(filename, 'scantbl.inc');
rewrite(filename);
writeln(filename, 'scan_table   label   byte');
low :=  ((NUM_HOPS div 4) * 0);
high := ((NUM_HOPS div 4) * 1) - 1;
write_a_line(low, high);
low :=  ((NUM_HOPS div 4) * 1);
high := ((NUM_HOPS div 4) * 2) - 1;
write_a_line(low, high);
low :=  ((NUM_HOPS div 4) * 2);
high := ((NUM_HOPS div 4) * 3) - 1;
write_a_line(low, high);
low :=  ((NUM_HOPS div 4) * 3);
high := ((NUM_HOPS div 4) * 4) - 1;
write_a_line(low, high);
low := ((NUM_HOPS div 4) * 4);
high := NUM_HOPS - 1;
write_a_line(low, high);
close(filename)
end;


(*
-------------------------------------------------------------------------*)
(* procedure to evenize the table                                        *)
(*
-------------------------------------------------------------------------*)
procedure evenize_table(offset, right_min, left_min : integer);
var
      x, y : integer;

begin
for x := 0 to (NUM_HOPS-1) do
      begin
      if (x+offset) < NUM_HOPS then
            y := x+offset
      else
            y := x+offset-NUM_HOPS;
      if (scan_table[x]=SCAN) then
            if (rights[offset] <= right_min) and
(scan_table[y]=UNASSIGNED) then
                  begin
```

```
                    scan_table[y] := NO_SCAN;
                    rights[offset] := rights[offset] + 1
                    end;
        if (scan_table[x]=NO_SCAN) then
            if (lefts[offset] <= left_min) and
    (scan_table[y]=UNASSIGNED) then
                    begin
                    scan_table[y] := SCAN;
                    lefts[offset] := lefts[offset] + 1;
                    end
        end
    end;


    (*
    ------------------------------------------------------------------------*)
    (* procedure to analyze the table                                       *)
    (*
    ------------------------------------------------------------------------*)
    procedure analyze_table;
    var
        offset, x, y : integer;

    begin
    for offset := 1 to (NUM_HOPS-1) do
        begin
        rights[offset] := 0;
        lefts[offset] := 0;
        for x := 0 to (NUM_HOPS-1) do
            begin
            if (x+offset) < NUM_HOPS then
                y := x+offset
            else
                y := x+offset-NUM_HOPS;
            if (scan_table[x]=SCAN) and (scan_table[y]=NO_SCAN) then
                    rights[offset] := rights[offset] + 1;
            if (scan_table[x]=NO_SCAN) and (scan_table[y]=SCAN) then
                    lefts[offset] := lefts[offset] + 1;
            end
        end
    end;


    (*
    ------------------------------------------------------------------------*)
    (* procedure to get the right stats                                     *)
    (* analyze table must be called before this function                    *)
    (*
    ------------------------------------------------------------------------*)
    procedure set_right_stats(var rs : stat_def);
    var
        n : integer;

    begin
    rs.min := 1000;
    rs.max := 0;
    rs.avg := 0;
    for n := 1 to (NUM_HOPS-1) do
        begin
        rs.avg := rs.avg + rights[n];
        if rights[n] < rs.min then
            rs.min := rights[n];
        if rights[n] > rs.max then
            rs.max := rights[n]
```

```
      end;
rs.avg := rs.avg div NUM_HOPS
end;


(*
 5 ------------------------------------------------------------------------*)
(* procedure to get the minimum number of lefts                           *)
(* analyze table must be called before this function                      *)
(*
------------------------------------------------------------------------*)
10 procedure set_left_stats(var ls : stat_def);
var
      n : integer;

begin
ls.min := 1000;
15 ls.max := 0;
ls.avg := 0;
for n := 1 to (NUM_HOPS-1) do
      begin
      ls.avg := ls.avg + lefts[n];
20    if lefts[n] < ls.min then
            ls.min := lefts[n];
      if lefts[n] > ls.max then
            ls.max := lefts[n]
      end;
25 ls.avg := ls.avg div NUM_HOPS
end;


(*
------------------------------------------------------------------------*)
(* main program body                                                      *)
30 (*
------------------------------------------------------------------------*)
begin
clrscr;
for n := 0 to (NUM_HOPS-1) do                          (* initialize
35 arrays                                  *)
      begin
      rights[n] := 0;
      lefts[n] := 0;
      scan_table[n] := UNASSIGNED
40    end;                                                          (*
we must pre assign at least   *)
scan_table[0] := SCAN;                                      (* one
element of this array         *)
repeat
45    for n := 1 to (NUM_HOPS-1) do
            begin
            offset := NUM_HOPS - n;
            analyze_table;
            set_right_stats(right_stats);
50          set_left_stats(left_stats);
            display_table(right_stats, left_stats);
            if (rights[offset] <= right_stats.min) or (lefts[offset]
<= left_stats.min) then
                  evenize_table(offset, right_stats.min,
55 left_stats.min);
            end;
      keep_going := FALSE;
      for n := 0 to (NUM_HOPS-1) do
            if scan_table[n] = UNASSIGNED then
```

```
                    keep_going := true
  until not keep_going;
  analyze_table;
  set_right_stats(right_stats);
  set_left_stats(left_stats);
  display_table(right_stats, left_stats);
  write_table;
  end.
```

CLAIMS

We Claim:

1. A method of operating a local area network having a plurality of stations, each station having address, with distributed control and incomplete connectivity to a communications channel, comprising the steps of:

at each station, locally storing in an associated memory a future busy state of the channel;

before a source station begins a data transmission to a destination station, determining at the source station that the channel is not busy for a time sufficient to transmit the data;

transmitting via the communications channel from the source station a request to send including the address of the destination station and an indication of a length of the data;

upon receipt of the request to send, each receiving station revising the local indication of the future busy state according to the indication of the length of the data;

the destination station, upon receipt of the request to send, transmitting via the communications channel a clear to send including the indication of the particular length of the data, if the future busy state stored in the associated memory of the destination station indicates sufficient time is available to transmit the data; and

upon receipt of the clear to send at the source station, transmitting the data.

2. The method of Claim 1, wherein the request to send further includes a sequence number incremented by the source station for each data transmission.

3. The method of Claim 1, further comprising the step of transmitting an acknowledgement from the destination station to the source station upon receipt of the data at the destination station, indicating receipt of the data.

4. The method of Claim 1, further comprising the steps of:

for each station, determining if no request to send, clear to send, or data have been transmitted or received for a predetermined time period; and

then, transmitting an announcement from the individual station.

5. The method of Claim 4, wherein the announcement includes an indication of a particular current transmission frequency and of how long the individual station has been transmitting on the particular current frequency.

6. The method of Claim 1, wherein if the source station does not receive the clear to send, further comprising the steps of:

waiting a randomly determined time period, and transmitting again the request to send;

repeating the step of waiting and transmitting at least once, if no clear to send is received; and, if after the step of repeating, no clear to send is received, sending the request to send on a frequency different from that of prior requests to send.

7. The method of Claim 1, wherein the communications channel is a wireless channel.

8. The method of Claim 7, wherein the communications channel is a radio frequency hopping channel.

9. The method of Claim 2, further comprising the step of the destination station comparing the sequence number for each request to send to a sequence number of previously received requests to send, and then if two compared sequence numbers are the same, disregarding a most recently received data transmission.

10. The method of Claim 8, further comprising the step of including in the request to send and clear to send how long respectively the source and destination stations have been transmitting on a particular hop frequency.

11. A communication adapter for a station in a local area network having distributed control, and a plurality of stations, for communications over a communication channel lacking full connectivity between the stations, each station having an address, the adapter comprising:

memory for storing a data structure indicating a future busy state of the channel;

a communications transceiver for sending and receiving on the channel;

means for determining from the stored data structure that the channel is not busy for a time sufficient to transmit a data originating in a station;

means for assembling a request to send to be transmitted by the transceiver, the request to send including the address of a destination and an indication of a length of the data;

means for altering the data structure according to an indication of a length of data received by the transceiver;

means for determining if the channel is not busy for a time sufficient to receive data of the length indicated in a received request to send;

means for assembling a clear to send to be transmitted by the transceiver in response to a received request to send, the clear to send including the indication of a length of the data in the received request to send; and

a connector for connecting to and receiving data from an associated host.

12. A radio transmitter/receiver circuit for a wireless local area network, comprising:

a first circuit generating a first signal, said first signal having a first frequency $f_1$ which is switchable between a plurality of frequencies;

a second circuit receiving said first signal and generating a second signal on an output lead, said second signal having a second frequency $f_2$ which is switchable between a first multiple of said first frequency $Af_1$ and a second multiple of said first frequency $Bf_1$;

an image reject mixer circuit comprising a first third order mixer and a second third order mixer, said image reject mixer circuit having a first input lead, a second input lead and an output lead, said first input lead of said image reject mixer circuit being coupled to an antenna, said second lead of said image reject circuit being coupled to said output lead of said second circuit;

an IF amplifier, an input lead of said IF amplifier being coupled to said output lead of said image reject mixer circuit; and

a frequency multiplier circuit, an input lead of said frequency multiplier circuit being coupled to said output lead of said second circuit, an output lead of said frequency multiplier circuit being at least selectively coupled to said antenna.

13. The radio transmitter/receiver circuit of claim 12, wherein said first circuit comprises a phase locked loop and wherein said second circuit comprises a phase locked loop.

14. The radio transmitter/receiver circuit of claim 12, wherein said first frequency $f_1$ is switched periodically between said plurality of frequencies to perform frequency hopping spread spectrum communication, said second frequency $f_2$ being switched between $Af_1$ and $Bf_1$ to switch said radio transmitter/receiver circuit from a transmit mode to a receive mode.

15. The radio transmitter/receiver circuit of claim 14, wherein a signal on said input lead of said image reject mixer circuit is transformed into a signal on said input lead of said IF amplifier, said signal on said input lead of said image reject mixer circuit having a frequency substantially different from $f_1$.

16. The radio transmitter/receiver circuit of claim 15, wherein said second circuit comprises a phase locked loop, said phase locked loop comprising a track and hold circuit having a track/hold input lead.

17. The radio transmitter/receiver circuit of claim 15, wherein said second circuit comprises a phase locked loop, said phase locked loop comprising a voltage controlled oscillator having a first input lead, a second input lead and an output lead, said first input lead being coupled to receive data for transmission.

18. The radio transmitter/receiver circuit of claim 15, further comprising a transmit/receive switch and a transmitter output amplifier, a first lead of said transmit/receive switch being coupled to an output lead of

said transmitter output amplifier, an input lead of said transmitter output amplifier being coupled to an output lead of said frequency multiplier circuit, said transmit/receive switch also having a control terminal for controlling an impedance between said first and second leads of said transmit/receive switch.

19. The radio transmitter/receiver circuit of claim 12, wherein said first circuit, said second circuit, said image reject mixer circuit, said IF amplifier, and said frequency multiplier circuit are disposed within a PCMCIA form factor.

20. A method, comprising the steps of:

generating an IF signal having a frequency $f_{IF}$ from an incoming signal of frequency $f_S$ received on an antenna by controlling a prescaler of a phase locked loop so that said phase locked loop supplies two third order mixers coupled in an image reject configuration with a local oscillator signal of a first frequency $f_{LO1}$; and

generating a outgoing signal onto said antenna, said outgoing signal having a frequency $f_S$, by controlling a prescaler of said phase locked loop so that said local oscillator signal has a second frequency $f_{LO2}$.

21. The method of claim 20, further comprising the step of:

generating a reference signal and supplying said reference signal to said phase locked loop having said prescaler; and

periodically varying a frequency of said reference signal.

22. A radio transmitter/receiver circuit for a wireless local area network, comprising:

first means, for synthesizing a first signal, said first signal having a first frequency $f_1$ which is digitally switchable between a plurality of frequencies;

second means, for receiving said first signal and for generating a second signal on an output lead, said second signal having a second frequency $f_2$ which is digitally switchable between a first multiple of said first frequency $Af_1$ and a second multiple of said first frequency $Bf_1$;

third means, for performing image rejection comprising a first third order mixer and a second third order mixer, said third means having a first input lead, a second input lead and an output lead, said first input lead of said third means being coupled to an antenna, a signal of frequency $f_S$ being present on said antenna, said second lead of said third means being coupled to said output lead of said second means;

fourth means, for amplifying a signal, an input lead of said fourth means being coupled to said output lead of said third means, an IF signal of frequency $f_{IF}$ being present on said input lead of said fourth means; and

fifth means, for frequency multiplying a signal by a factor C, an input lead of said fifth means being coupled to said output lead of said second means, an output lead of said fifth means being at least selectively coupled to said antenna, wherein $f_S=Af_1*C$ during a transmit mode and wherein $f_{IF}=f_S-2Bf_1$ during a transmit mode.

23. The radio transmitter/receiver circuit of claim 22, wherein C is 2, wherein A is 513, wherein B is 512, and wherein $f_S$ is a frequency which is varied within a range by digitally switching $f_1$ between said plurality of frequencies, said range of $f_S$ is approximately 2.4 GHz to 2.5 GHz, said first, second, third, fourth, and fifth means being disposed in a PCMCIA form factor.

24. A method of operating a local area network including at least one mobile unit and a plurality of access point units connected to each other, the mobile unit being associated with a first one of the access point units and all the access point units capable of being associated by wireless communication with the mobile unit, comprising the steps of:

the mobile unit receiving transmissions from the access point units including data identifying the available access point units and for each available access point unit an indication of at what frequencies it is transmitting;

storing the included data at the mobile unit;

the mobile unit determining that it is to be associated with a second of the access point units which is available;

the mobile unit establishing communication with the second of the access point units in accordance with the stored data; and

the second of the access point units communicating with the first of the access point units and receiving from the first of the access point units a context of the association of the mobile unit to the first access point unit, thereby associating the mobile unit with the second of the access point units.

25. A network adapter for a host, the adapter receiving data from the host for wireless communication to a plurality of connected network access point units, the adapter comprising:

a transceiver for transmitting and receiving the data to the access point units;

means for determining that the host is associated with a first of the access point units;

means for determining from transmissions received from the access point units, at what frequency each access point unit is transmitting, the transmissions including data identifying the available access point units and for each available access point unit, an indication at what frequency the access point unit is transmitting;

a memory for storing the included data; and

means for determining from the stored data that the host is to be associated with a second access point unit.

26. A network access point unit for connection to a plurality of other access point units by a wired connection and for communicating to at least one mobile unit by wireless communication, comprising:

a transceiver for sending to and receiving data from the mobile unit; and

a memory for storing a list of the other access point units, the list including for each other access point unit an identification number associated with the other access point unit, an indication of at what frequency the other access point unit is transmitting at any given time, and an indication of proximity to the access point unit;

whereby the mobile unit being associated with a first of the access point units can change its association to a second of the access point units upon receipt of the list from the access point unit.

27. A method for transmitting time bounded information over a local area network including a plurality of connected network access point units and at least one mobile unit capable of wireless communication with the access point units, comprising the steps of:

providing a local memory in both the mobile unit and in each access point unit, each local memory storing an indication of a future busy state of the network;

setting up communications between the mobile unit and one of the access point units;

transmitting a first frame from the one access point unit to the mobile unit, the first frame including an indication of a time gap between successive frames, and a number of successive frames to be transmitted;

transmitting a second frame in response from the mobile unit to the one access point unit, the second frame also including the indication of the time gap and the number of successive frames;

including in one of the first or second frames time bounded message data; and

in the local memories in the mobile unit and each of the access point units, storing the indication of the time gap and the number of successive frames, thereby inhibiting any of the access point units other than the one access point unit from contending with the communication between the one access point unit and the mobile unit.

28. An access point unit for a local area network including a plurality of connected access point units and at least one mobile unit capable of wireless communication with the access point units, the access point unit comprising:

means for connecting to the network;

a transceiver for the wireless communication with the mobile unit;

a local memory for storing an indication of a future busy state of the network;

means for setting up time bounded service communications to the mobile unit using the transceiver;

means for assembling a first frame to be transmitted to the mobile station, the first frame including an indication of a time gap between successive frames, and a number of successive frames to be reserved and including time bounded message data received from the network; and

means for receiving a second frame from the one mobile station, the second frame including time bounded message data, an indication of a time gap between successive frames, and a number of successive frames to be transmitted, and altering the stored indication in the local memory accordingly.

29. A method of initializing a wireless local area network, comprising the step of:

using a hop number of a current hop period of a first unit to determine whether or not said first unit will scan during a scan portion of said current hop period, said current hop period being one of a sequence of hop periods, each of said hop periods of said sequence of hop periods having a different hop number.

30. The method of claim 29, further comprising the step of:

scanning one frequency during said scan portion for an amount of time X, and transmitting at least once every amount of time X when in said current period but not in said scan portion of said current period.

31. The method of claim 29, further comprising the step of:

scanning multiple frequencies during said scan portion of said current hop, each frequency being scanned for a time X.

32. The method of claim 29, further comprising the step of:

before said using step, scanning for a transmission sent by a unit hardwired to a local area network.

33. The method of claim 29, further comprising the step of:

after said using step, receiving a transmission having a field indicative of a sequence of hop periods used by a second unit and having a field indicative of an amount of time remaining in a current hop period of said second unit.

34. The method of claim 33, said current hop period of said second unit a corresponding current hop number, further comprising the step of:

after said receiving step, determining whether to adopt said current hop number of said second unit or to retain said current hop number of said first unit by comparing a hop number of said current hop period of said first unit with said current hop number of said current hop period of said second unit.

35. The method of claim 34, wherein:

the hop number adopted or retained by said first unit is the hop number having the greatest hop number when the absolute value of said hop number of said current hop period of said second unit minus said hop number of said current hop period of said first unit is less than the total number of hops in said sequence of hops minus the absolute value of said hop number of the current hop period of said second unit minus said hop number of said

current hop period of said first unit, otherwise the hop number adopted or retained is the smallest hop number.

36. The method of claim 34, wherein:

the hop number adopted or retained by said first unit is the hop number having the smallest hop number when the absolute value of said hop number of said current hop period of said second unit minus said hop number of said current hop period of said first unit is less than the total number of hops in said sequence of hops minus the absolute value of said hop number of the current hop period of said second unit minus said hop number of said current hop period of said first unit, otherwise the hop number adopted or retained is the greatest hop number.

37. A wireless mobile unit comprising a connector for coupling to a computer bus, a digital portion, and a radio transmitter/receiver circuit portion, said wireless mobile unit not being in communication with any local area network, a plurality of hop periods comprising a sequence of hop periods, said wireless mobile unit scanning during scanning portions of selected ones of said sequence of hop periods and not scanning during others of said sequence of hop periods, said wireless mobile unit periodically transmitting during each of said plurality of hop periods of said sequence when said wireless mobile unit is not scanning.

38. The wireless mobile unit of claim 37, wherein said digital portion comprises means for determining whether or not to scan during each hop period of said sequence of hop periods.

39. The wireless mobile unit of claim 38, wherein said digital portion comprises means for determining

whether or not to adopt a hop number of another wireless mobile unit.

40. The wireless mobile unit of claim 39, wherein said digital portion comprises means for determining whether or not to synchronize in time to a hop tick of another wireless mobile unit.

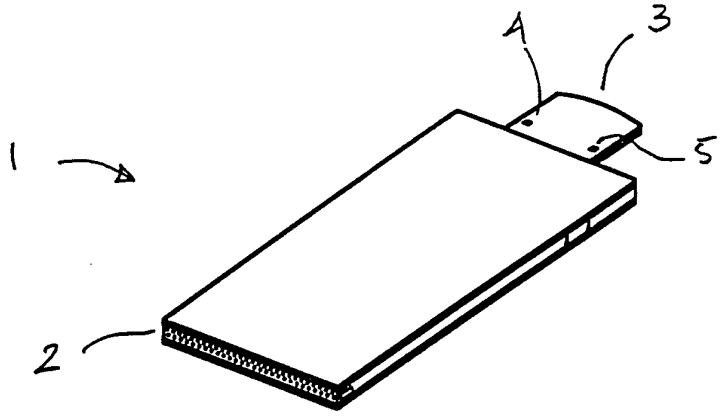



Fig. 1

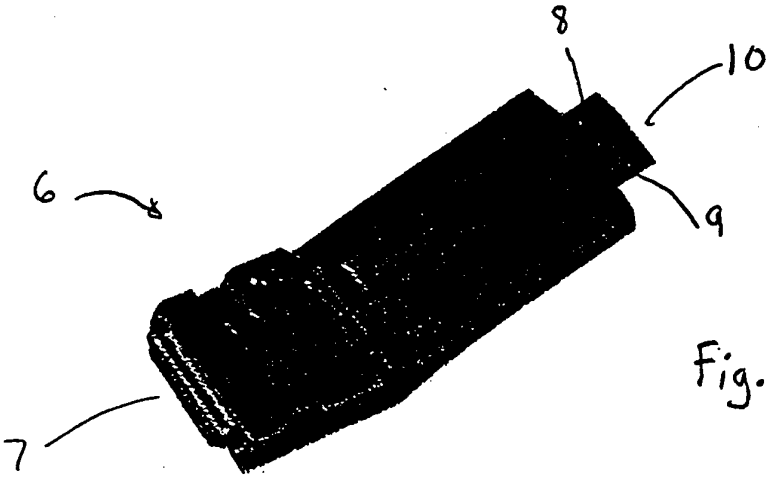



Fig. 2

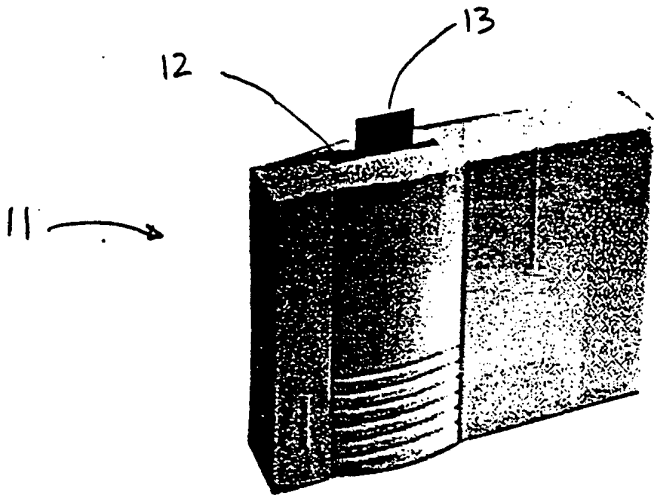



Fig. 3

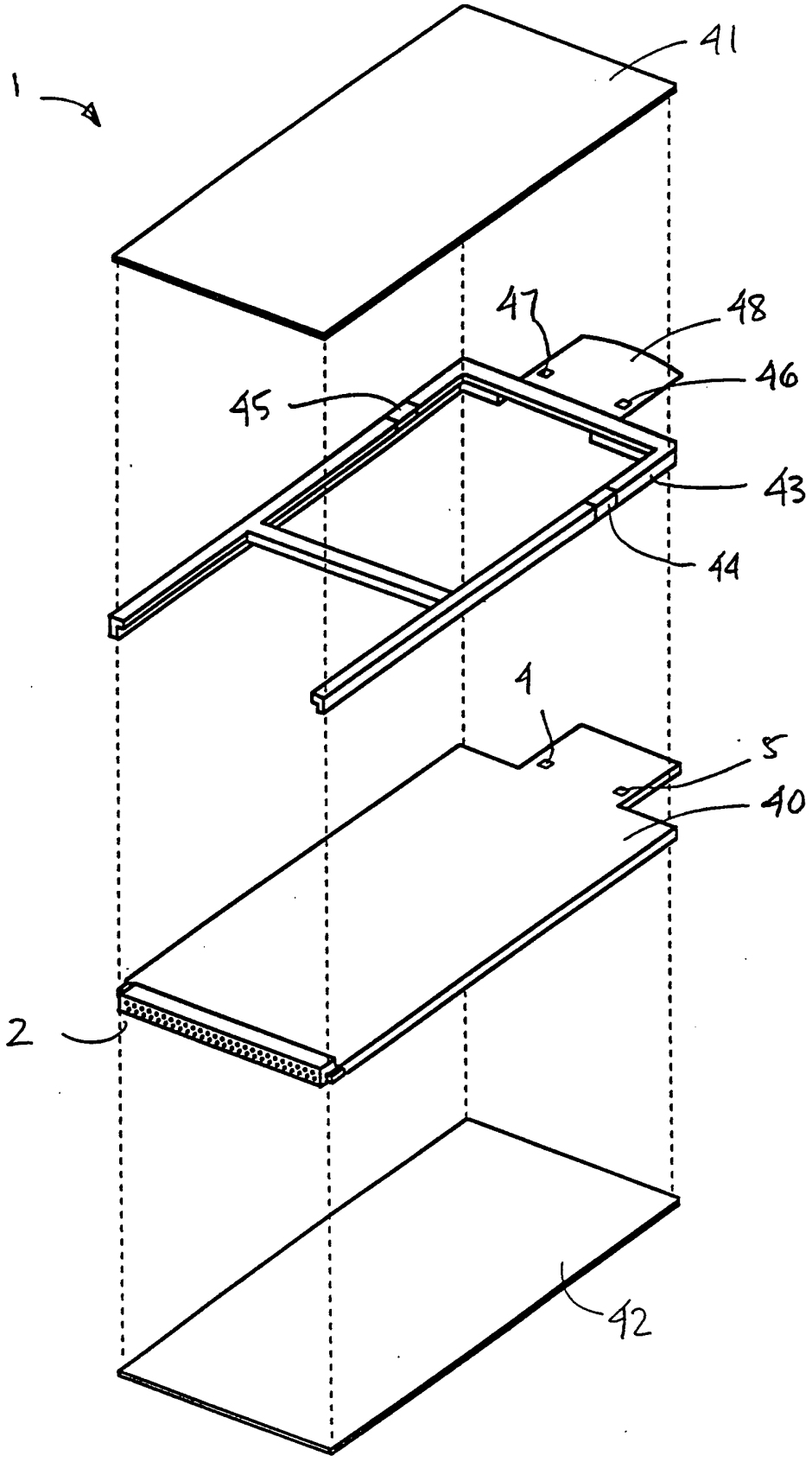



Fig. 4

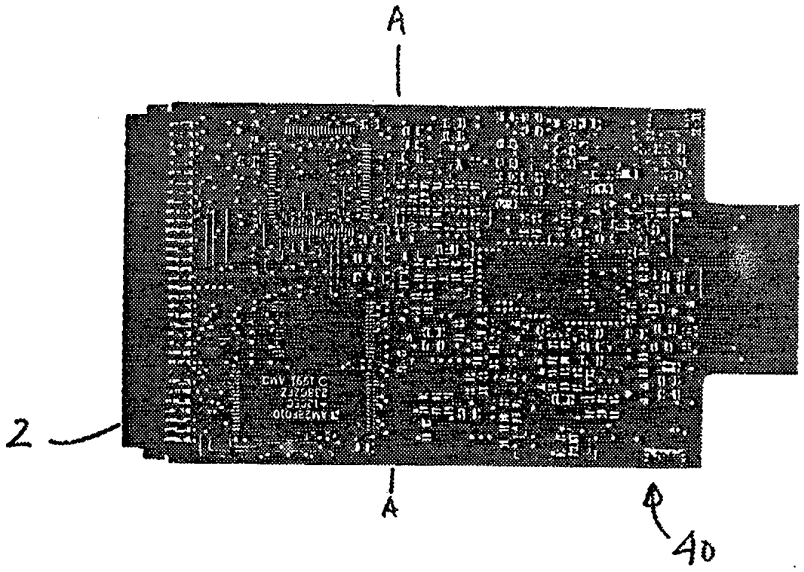



Fig. 6

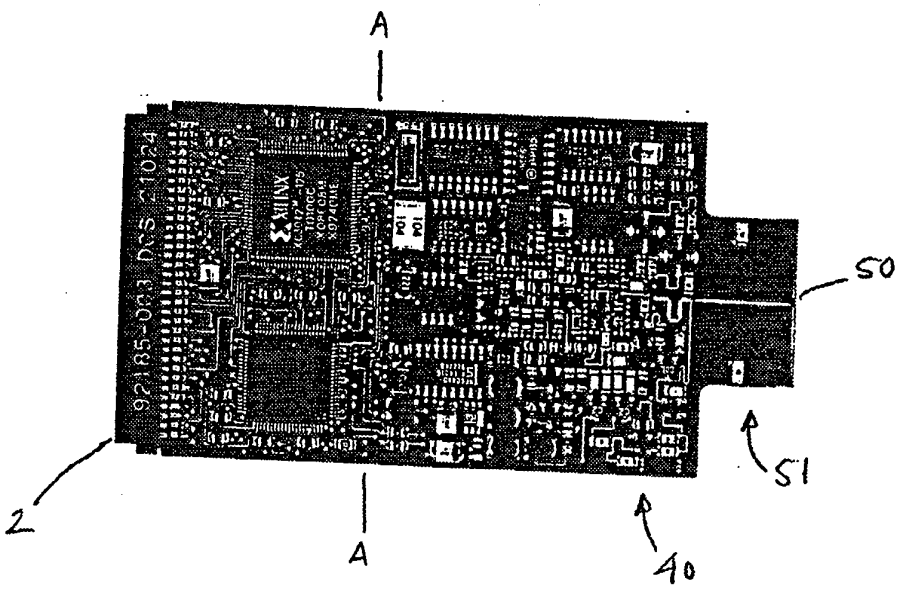



Fig. 5

FIG. 7A

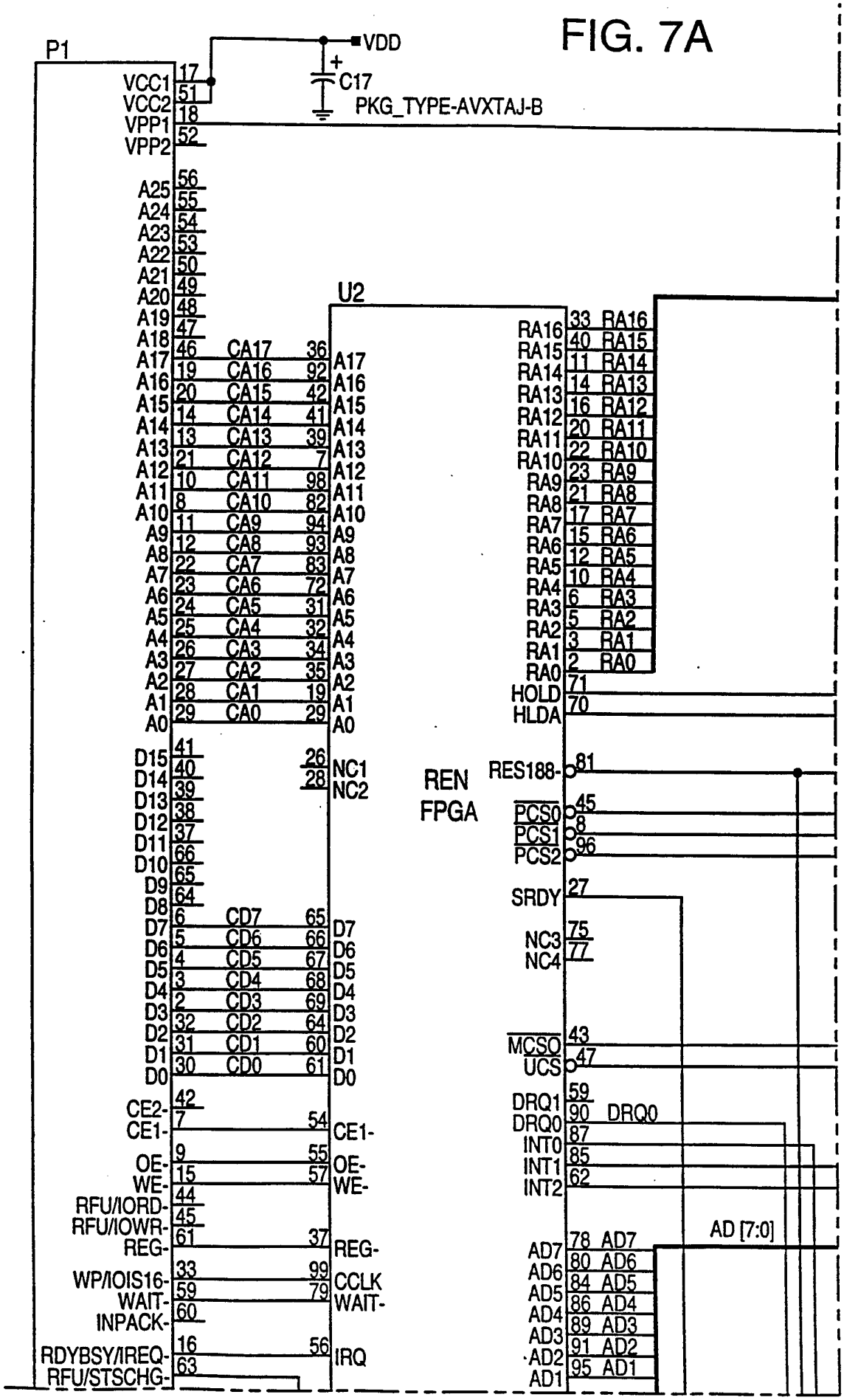

P1
VCC1 17
VCC2 51
VPP1 18
VPP2 52
VDD
C17
PKG_TYPE-AVXTAJ-B
A25 56
A24 55
A23 54
A22 53
A21 50
A20 49
A19 48
A18 47
A17 46
A16 19
A15 20
A14 14
A13 13
A12 21
A11 10
A10 8
A9 11
A8 12
A7 22
A6 23
A5 24
A4 25
A3 26
A2 27
A1 28
A0 29
U2
CA17 36 A17
CA16 92 A16
CA15 42 A15
CA14 41 A14
CA13 39 A13
CA12 7 A12
CA11 98 A11
CA10 82 A10
CA9 94 A9
CA8 93 A8
CA7 83 A7
CA6 72 A6
CA5 31 A5
CA4 32 A4
CA3 34 A3
CA2 35 A2
CA1 19 A1
CA0 29 A0
D15 41
D14 40
D13 39
D12 38
D11 37
D10 66
D9 65
D8 64
D7 6
D6 5
D5 4
D4 3
D3 2
D2 32
D1 31
D0 30
NC1 26
NC2 28
CD7 65 D7
CD6 66 D6
CD5 67 D5
CD4 68 D4
CD3 69 D3
CD2 64 D2
CD1 60 D1
CD0 61 D0
CE2- 42
CE1- 7
CE1- 54
OE- 9
OE- 55
WE- 15
WE- 57
RFU/IORD- 44
RFU/IOWR- 45
REG- 61
REG- 37
WP/IOIS16- 33
CCLK 99
WAIT- 59
WAIT- 79
INPACK- 60
RDYBSY/IREQ- 16
IRQ 56
RFU/STSCHG- 63
RA16 33 RA16
RA15 40 RA15
RA14 11 RA14
RA13 14 RA13
RA12 16 RA12
RA11 20 RA11
RA10 22 RA10
RA9 23 RA9
RA8 21 RA8
RA7 17 RA7
RA6 15 RA6
RA5 12 RA5
RA4 10 RA4
RA3 6 RA3
RA2 5 RA2
RA1 3 RA1
RA0 2 RA0
HOLD 71
HLDA 70
REN
FPGA
RES188- 81
PCS0 45
PCS1 8
PCS2 96
SRDY 27
NC3 75
NC4 77
MCSO 43
UCS 47
DRQ1 59
DRQ0 90 DRQ0
INT0 87
INT1 85
INT2 62
AD7 78 AD7
AD6 80 AD6
AD5 84 AD5
AD4 86 AD4
AD3 89 AD3
AD2 91 AD2
AD1 95 AD1
AD [7:0]

FIG. 7B

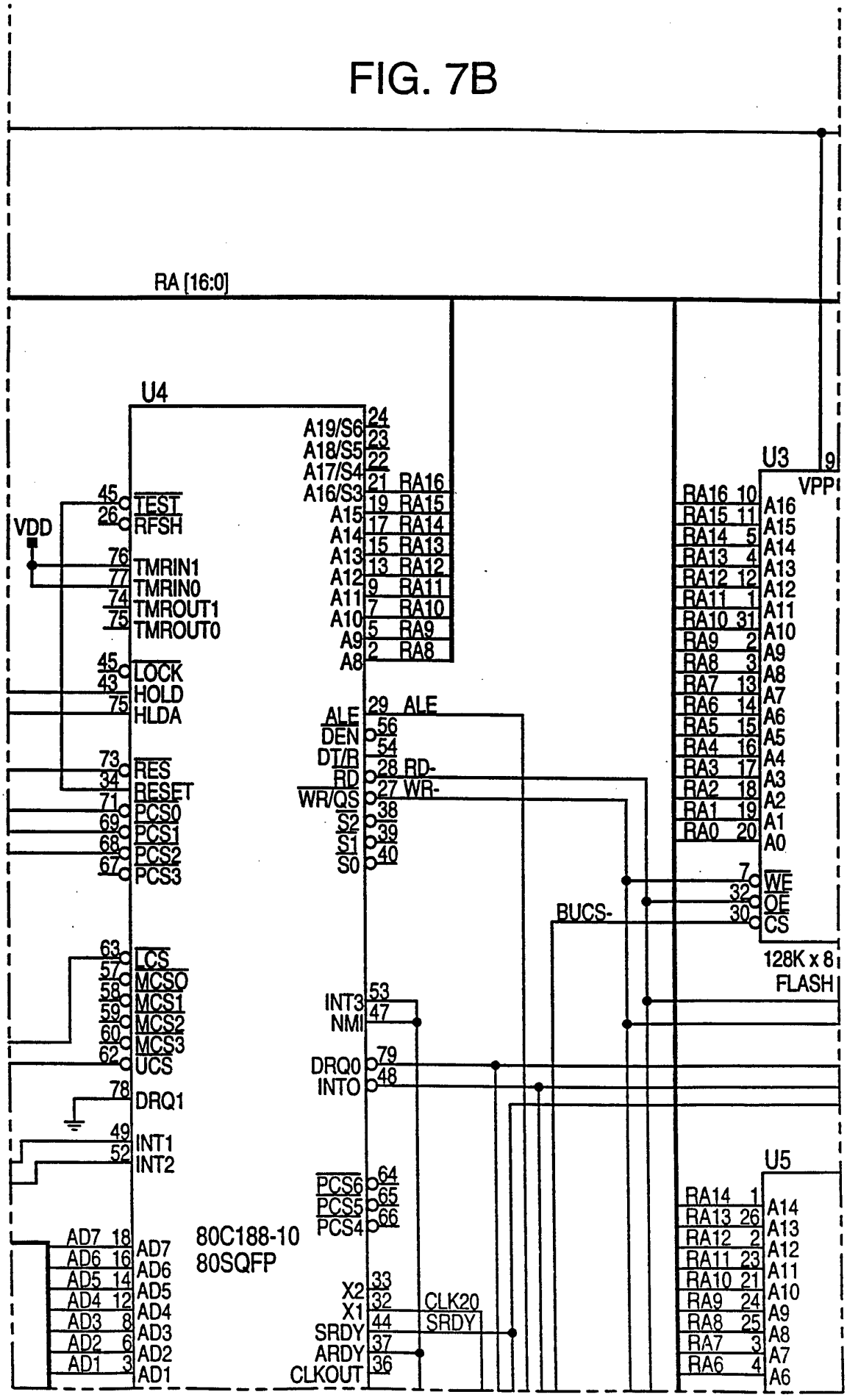
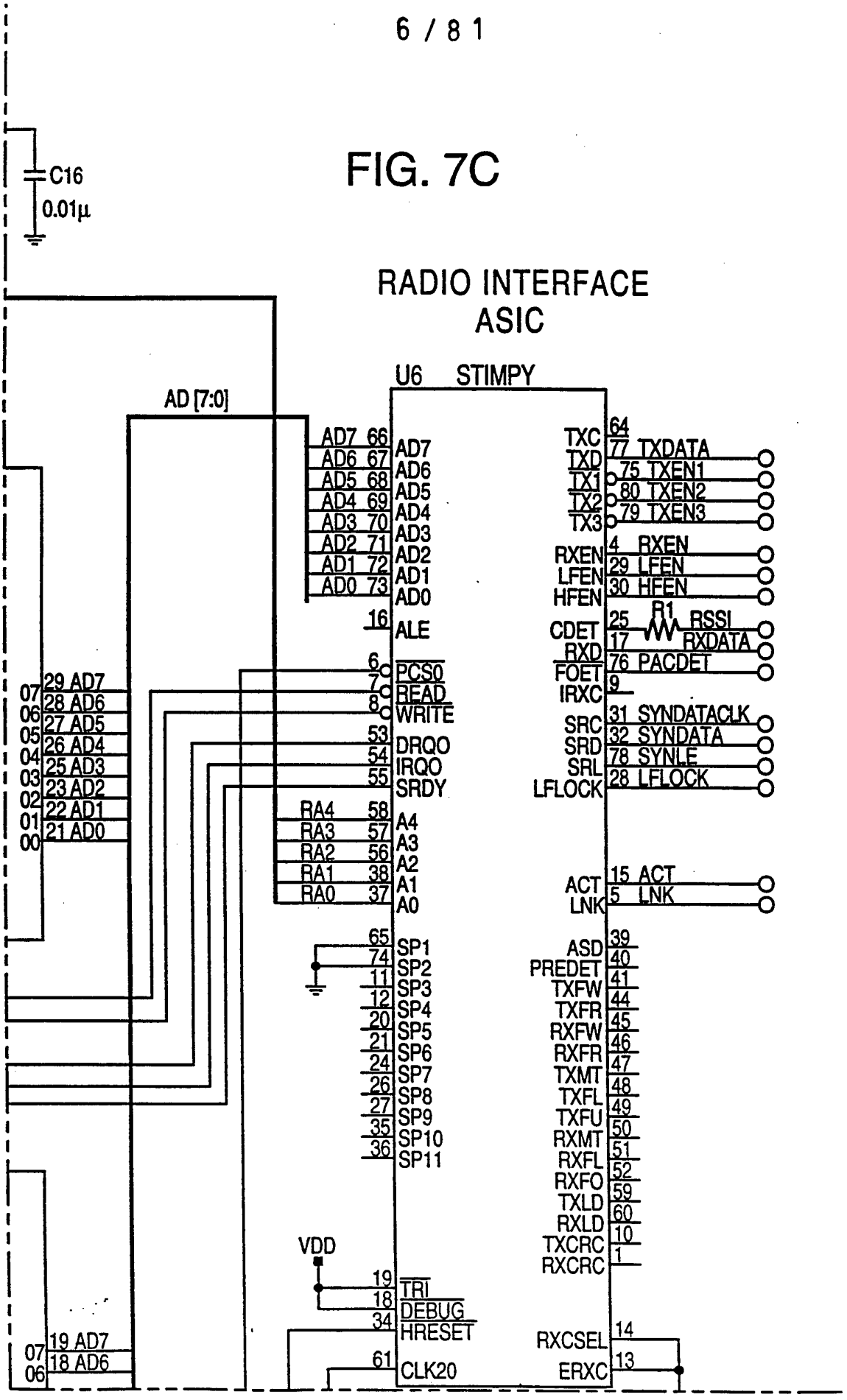

RA [16:0]
U4
80C188-10
80SQFP
VDD
TEST
RFSH
TMRIN1
TMRIN0
TMROUT1
TMROUT0
LOCK
HOLD
HLDA
RES
RESET
PCS0
PCS1
PCS2
PCS3
LCS
MCS0
MCS1
MCS2
MCS3
UCS
DRQ1
INT1
INT2
AD7
AD6
AD5
AD4
AD3
AD2
AD1
A19/S6
A18/S5
A17/S4
A16/S3
A15
A14
A13
A12
A11
A10
A9
A8
RA16
RA15
RA14
RA13
RA12
RA11
RA10
RA9
RA8
ALE
DEN
DT/R
RD
WR/QS
S2
S1
S0
RD-
WR-
INT3
NMI
DRQ0
INTO
PCS6
PCS5
PCS4
X2
X1
SRDY
ARDY
CLKOUT
CLK20
SRDY
U3
VPP
A16
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
WE
OE
CS
BUCS-
128K x 8
FLASH
U5
A14
A13
A12
A11
A10
A9
A8
A7
A6

FIG. 7C


C16
0.01µ
RADIO INTERFACE
ASIC
U6 STIMPY
AD [7:0]
AD7 66 AD7
AD6 67 AD6
AD5 68 AD5
AD4 69 AD4
AD3 70 AD3
AD2 71 AD2
AD1 72 AD1
AD0 73 AD0
16 ALE
6 PCS0
7 READ
8 WRITE
53 DRQO
54 IRQO
55 SRDY
RA4 58 A4
RA3 57 A3
RA2 56 A2
RA1 38 A1
RA0 37 A0
65 SP1
74 SP2
11 SP3
12 SP4
20 SP5
21 SP6
24 SP7
26 SP8
27 SP9
35 SP10
36 SP11
VDD
19 TRI
18 DEBUG
34 HRESET
61 CLK20
07 29 AD7
06 28 AD6
05 27 AD5
04 26 AD4
03 23 AD2
02 22 AD1
01 21 AD0
00
07 19 AD7
06 18 AD6
TXC 64
TXD 77 TXDATA
TX1 75 TXEN1
TX2 80 TXEN2
TX3 79 TXEN3
RXEN 4 RXEN
LFEN 29 LFEN
HFEN 30 HFEN
R1
CDET 25 RSSI
RXD 17 RXDATA
FOET 76 PACDET
IRXC 9
SRC 31 SYNDATACLK
SRD 32 SYNDATA
SRL 78 SYNLE
LFLOCK 28 LFLOCK
ACT 15 ACT
LNK 5 LNK
ASD 39
PREDET 40
TXFW 41
TXFR 44
RXFW 45
RXFR 46
TXMT 47
TXFL 48
TXFU 49
RXMT 50
RXFL 51
RXFO 52
TXLD 59
RXLD 60
TXCRC 10
RXCRC 1
RXCSEL 14
ERXC 13

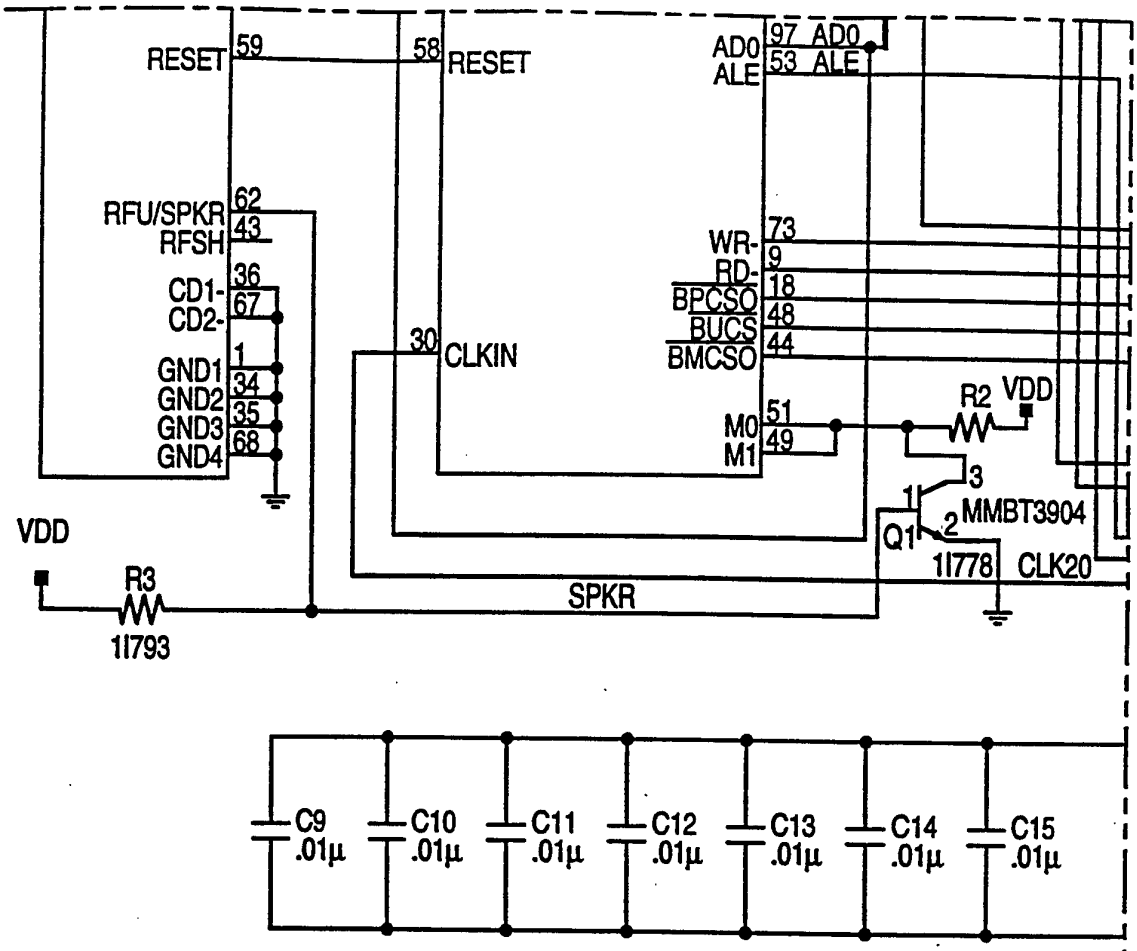

RESET
59
58
RESET
AD0
97 AD0
ALE
53 ALE
RFU/SPKR
62
RFSH
43
WR-
73
RD-
9
BPCSO
18
BUCS
48
BMCSO
44
CD1-
36
CD2-
67
30
CLKIN
GND1
1
GND2
34
GND3
35
GND4
68
M0
51
M1
49
R2
VDD
3
1
2
MMBT3904
Q1
1I778
CLK20
VDD
R3
1I793
SPKR
C9
.01µ
C10
.01µ
C11
.01µ
C12
.01µ
C13
.01µ
C14
.01µ
C15
.01µ

FIG. 7D

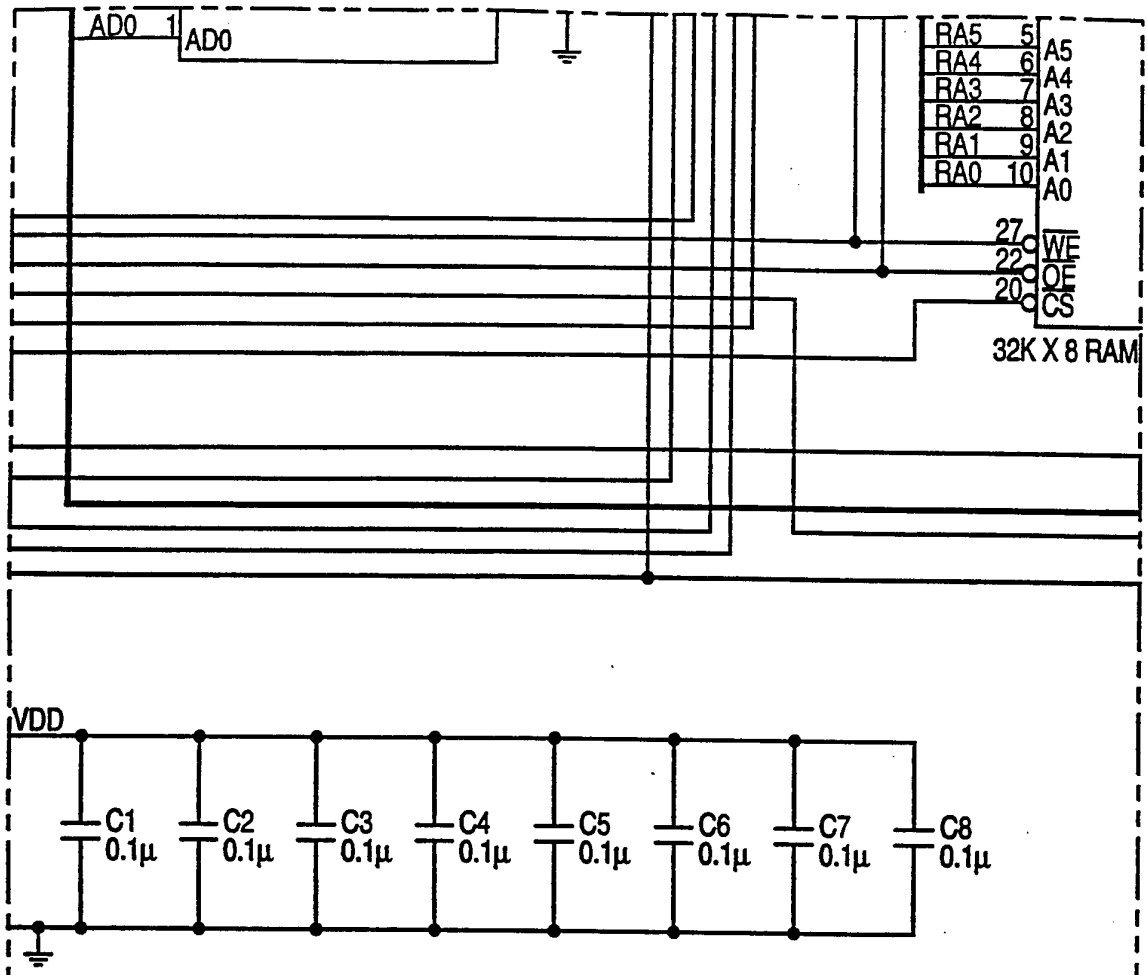

AD0 1
AD0
RA5 5
RA4 6
RA3 7
RA2 8
RA1 9
RA0 10
A5
A4
A3
A2
A1
A0
27 WE
22 OE
20 CS
32K X 8 RAM
VDD
C1 0.1µ
C2 0.1µ
C3 0.1µ
C4 0.1µ
C5 0.1µ
C6 0.1µ
C7 0.1µ
C8 0.1µ

FIG. 7E

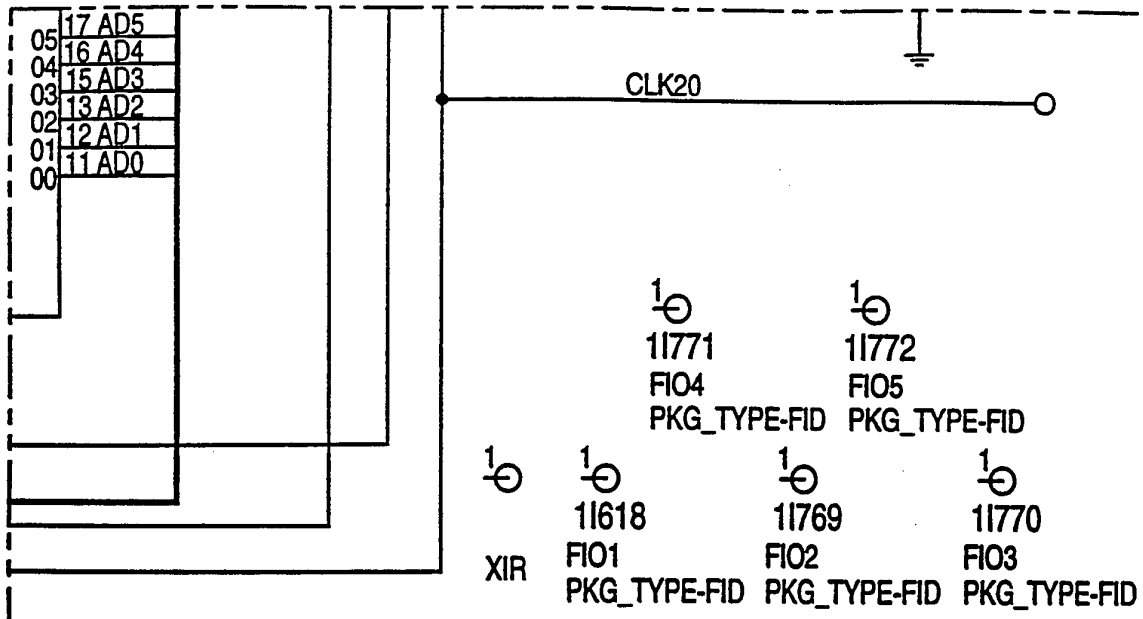



FIG. 7F

KEY TO
FIG. 7

| | | |
|---|---|---|
| FIG. 7A | FIG. 7B | FIG. 7C |
| FIG. 7D | FIG. 7E | FIG. 7F |

FIG. 8A

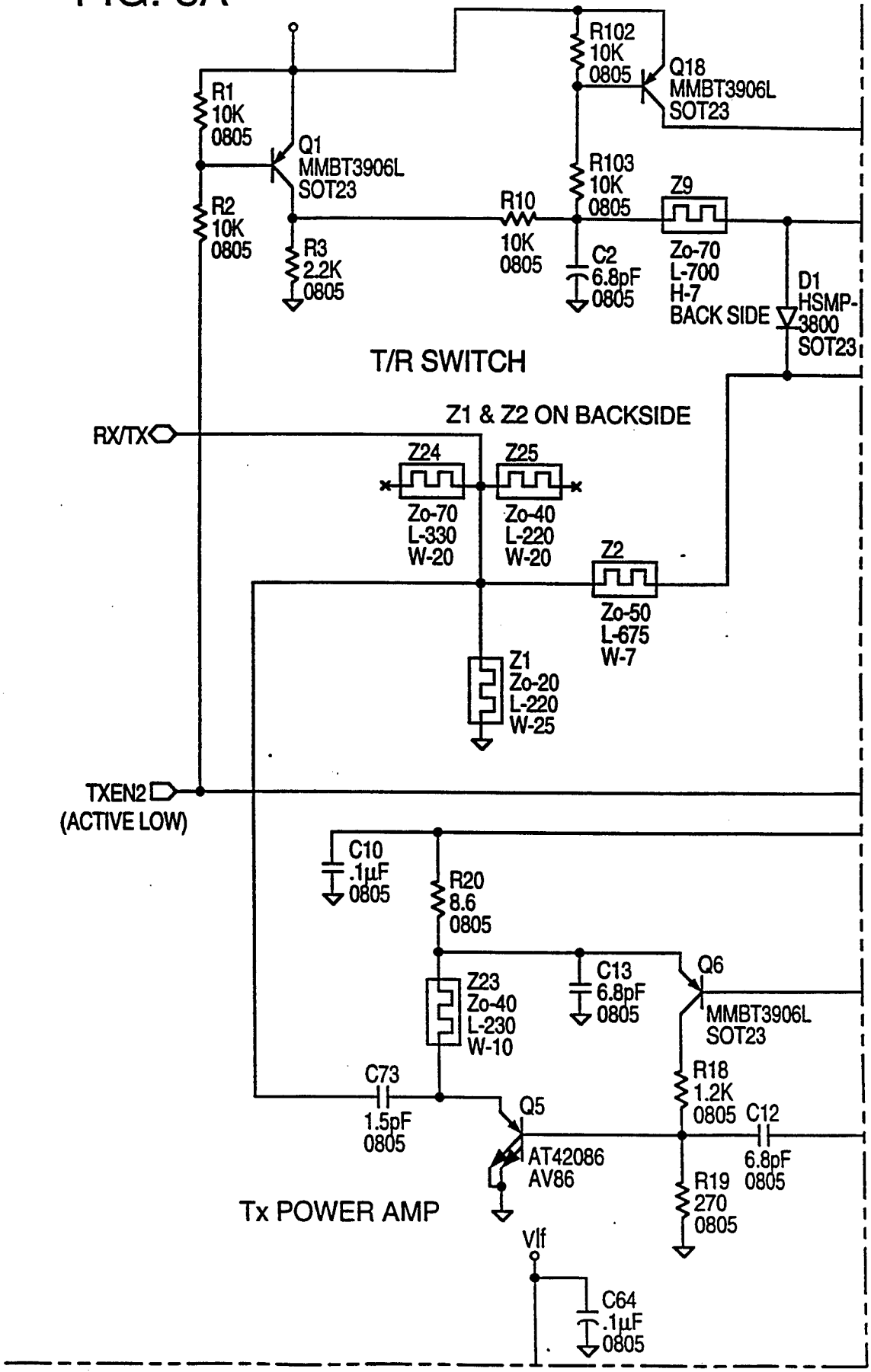

R102
10K
0805
Q18
MMBT3906L
SOT23
R1
10K
0805
Q1
MMBT3906L
SOT23
R103
10K
0805
Z9
R2
10K
0805
R10
10K
0805
R3
2.2K
0805
C2
6.8pF
0805
Zo-70
L-700
H-7
BACK SIDE
D1
HSMP-
3800
SOT23
T/R SWITCH
Z1 & Z2 ON BACKSIDE
RX/TX
Z24
Z25
Zo-70
L-330
W-20
Zo-40
L-220
W-20
Z2
Zo-50
L-675
W-7
Z1
Zo-20
L-220
W-25
TXEN2
(ACTIVE LOW)
C10
.1µF
0805
R20
8.6
0805
Z23
Zo-40
L-230
W-10
C13
6.8pF
0805
Q6
MMBT3906L
SOT23
C73
1.5pF
0805
Q5
R18
1.2K
0805
C12
6.8pF
0805
AT42086
AV86
R19
270
0805
Tx POWER AMP
Vlf
C64
.1µF
0805

FIG. 8B

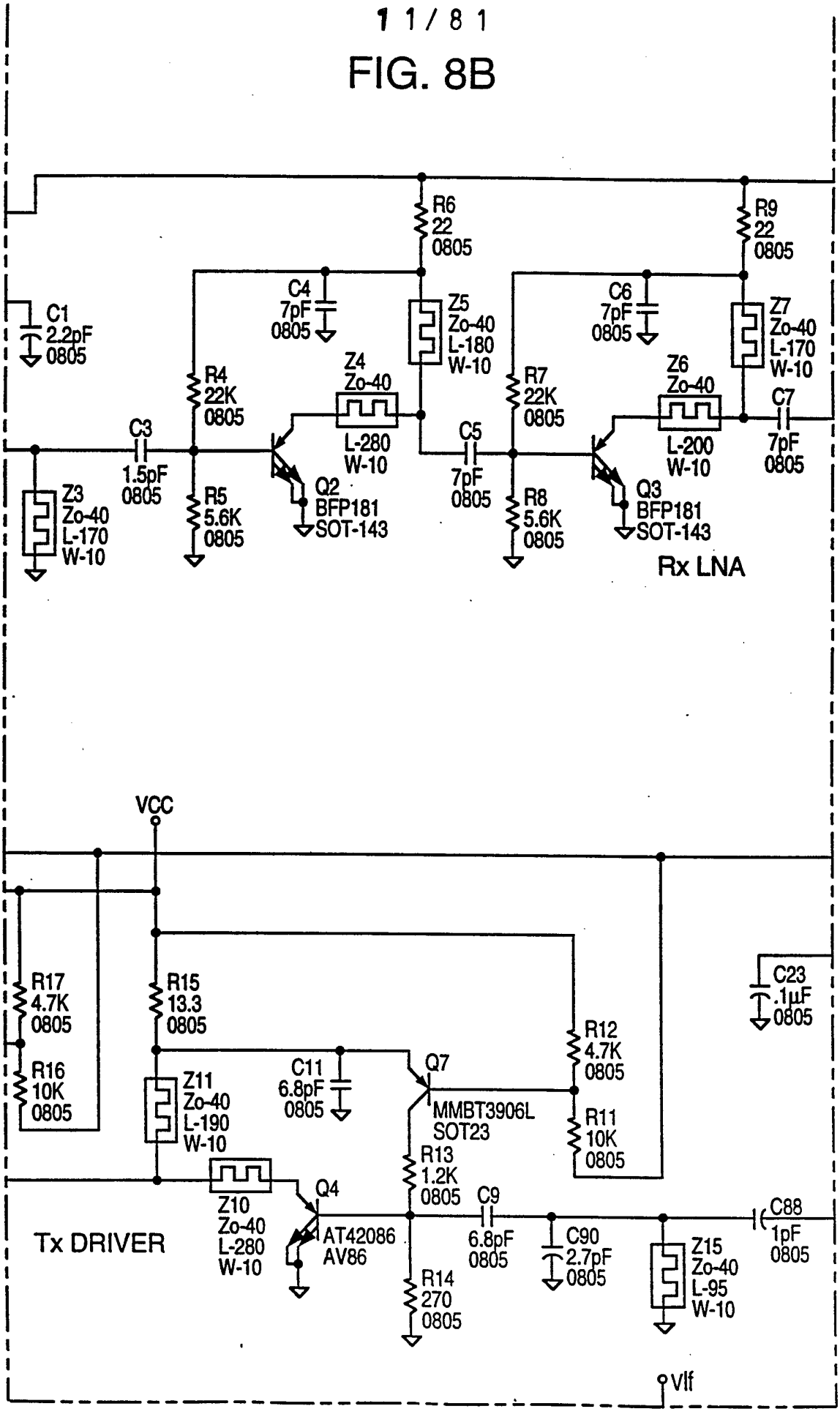

R6
22
0805
R9
22
0805
C1
2.2pF
0805
C4
7pF
0805
Z5
Zo-40
L-180
W-10
C6
7pF
0805
Z7
Zo-40
L-170
W-10
Z4
Zo-40
L-280
W-10
Z6
Zo-40
L-200
W-10
R4
22K
0805
R7
22K
0805
C3
1.5pF
0805
C5
7pF
0805
C7
7pF
0805
Z3
Zo-40
L-170
W-10
R5
5.6K
0805
Q2
BFP181
SOT-143
R8
5.6K
0805
Q3
BFP181
SOT-143
Rx LNA
VCC
R17
4.7K
0805
R15
13.3
0805
C23
.1µF
0805
R12
4.7K
0805
R16
10K
0805
Z11
Zo-40
L-190
W-10
C11
6.8pF
0805
Q7
MMBT3906L
SOT23
R11
10K
0805
R13
1.2K
0805
Z10
Zo-40
L-280
W-10
Q4
AT42086
AV86
C9
6.8pF
0805
C88
1pF
0805
Tx DRIVER
C90
2.7pF
0805
Z15
Zo-40
L-95
W-10
R14
270
0805
Vlf

FIG. 8C

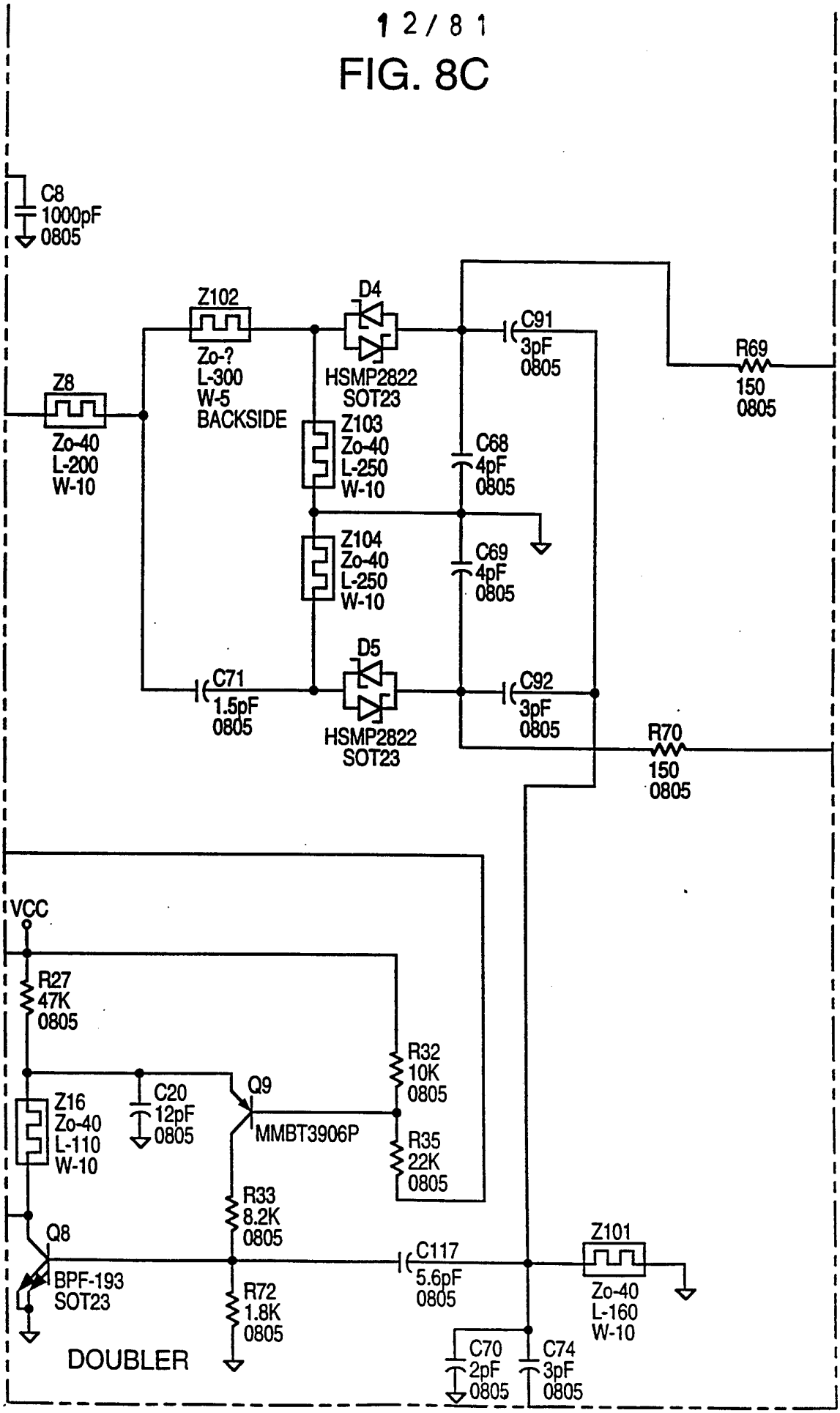
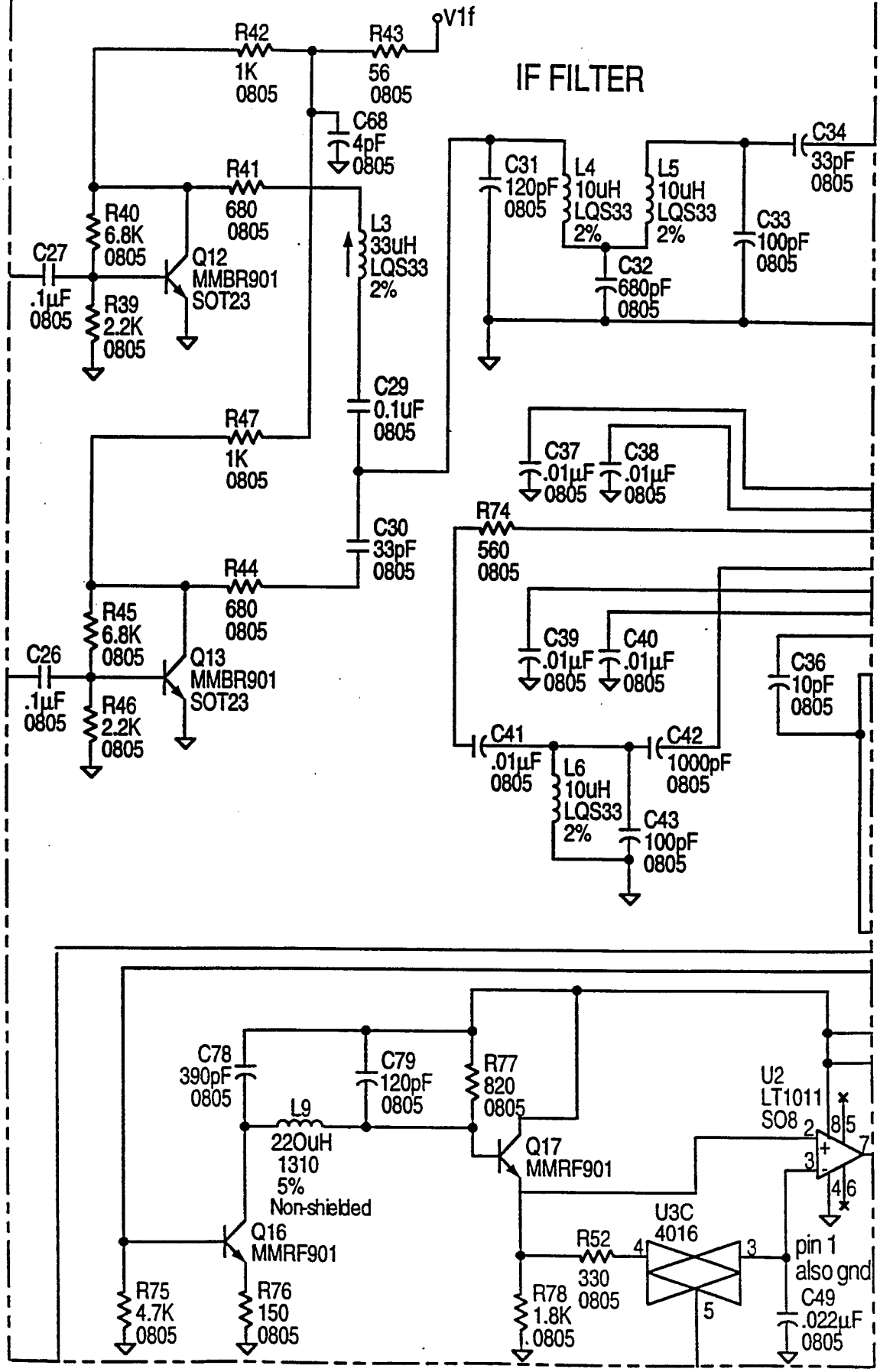

C8
1000pF
0805
Z102
Zo-?
L-300
W-5
BACKSIDE
D4
HSMP2822
SOT23
C91
3pF
0805
R69
150
0805
Z8
Zo-40
L-200
W-10
Z103
Zo-40
L-250
W-10
C68
4pF
0805
Z104
Zo-40
L-250
W-10
C69
4pF
0805
D5
C71
1.5pF
0805
HSMP2822
SOT23
C92
3pF
0805
R70
150
0805
VCC
R27
47K
0805
R32
10K
0805
Z16
Zo-40
L-110
W-10
C20
12pF
0805
Q9
MMBT3906P
R35
22K
0805
R33
8.2K
0805
Q8
BPF-193
SOT23
C117
5.6pF
0805
Z101
Zo-40
L-160
W-10
R72
1.8K
0805
C70
2pF
0805
C74
3pF
0805
DOUBLER

FIG. 8D


V1f
IF FILTER
R42
1K
0805
R43
56
0805
C68
4pF
0805
R41
680
0805
R40
6.8K
0805
C27
.1µF
0805
Q12
MMBR901
SOT23
R39
2.2K
0805
L3
33uH
LQS33
2%
C29
0.1uF
0805
C31
120pF
0805
L4
10uH
LQS33
2%
L5
10uH
LQS33
2%
C34
33pF
0805
C33
100pF
0805
C32
680pF
0805
R47
1K
0805
C37
.01µF
0805
C38
.01µF
0805
R74
560
0805
C30
33pF
0805
R44
680
0805
R45
6.8K
0805
C26
.1µF
0805
Q13
MMBR901
SOT23
R46
2.2K
0805
C39
.01µF
0805
C40
.01µF
0805
C36
10pF
0805
C41
.01µF
0805
L6
10uH
LQS33
2%
C42
1000pF
0805
C43
100pF
0805
C78
390pF
0805
C79
120pF
0805
R77
820
0805
L9
220uH
1310
5%
Non-shielded
Q16
MMRF901
Q17
MMRF901
U2
LT1011
SO8
U3C
4016
R52
330
0805
R78
1.8K
0805
R75
4.7K
0805
R76
150
0805
pin 1
also gnd
C49
.022µF
0805

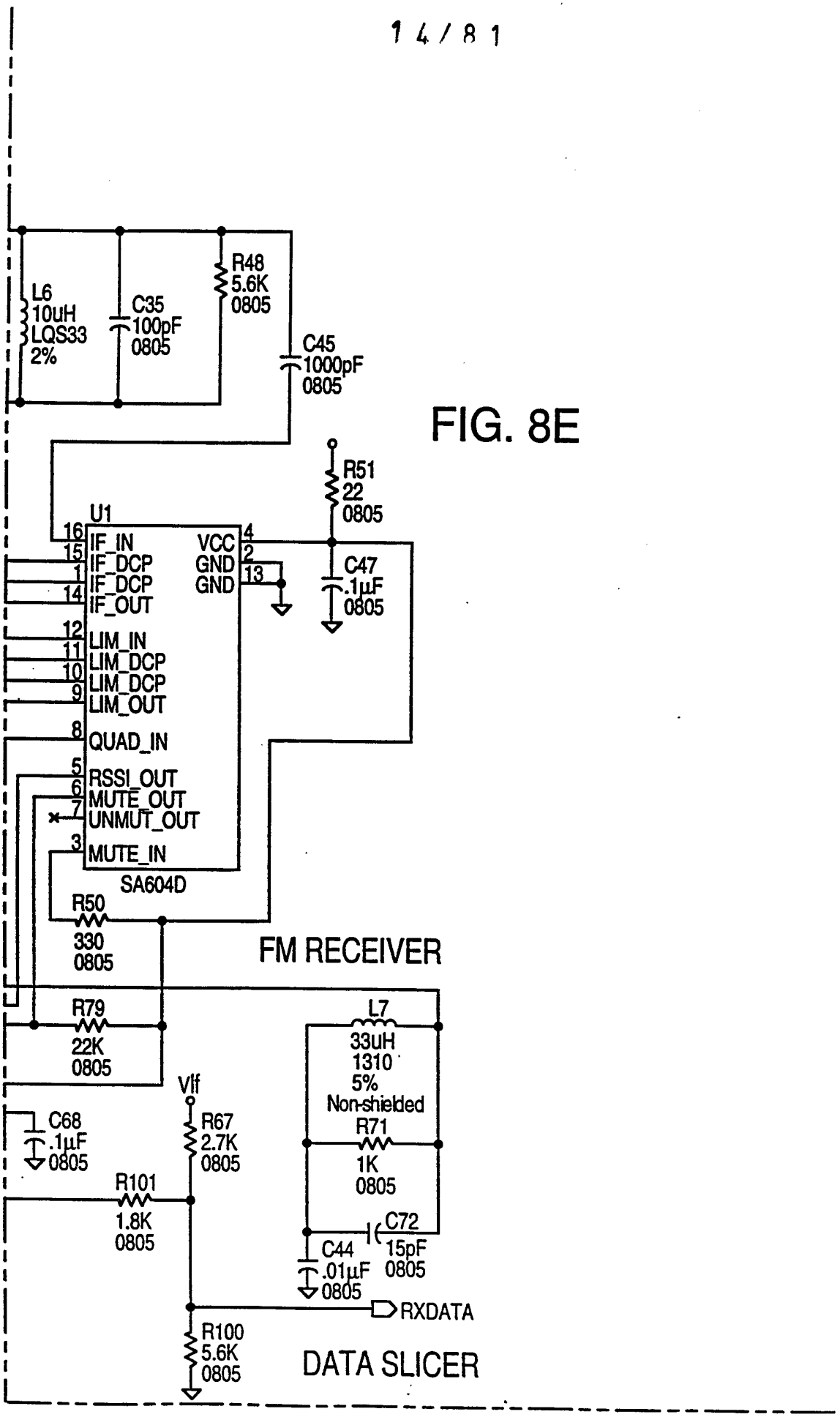

L6
10uH
LQS33
2%
C35
100pF
0805
R48
5.6K
0805
C45
1000pF
0805
R51
22
0805
U1
IF_IN
IF_DCP
IF_DCP
IF_OUT
LIM_IN
LIM_DCP
LIM_DCP
LIM_OUT
QUAD_IN
RSSI_OUT
MUTE_OUT
UNMUT_OUT
MUTE_IN
VCC
GND
GND
SA604D
C47
.1µF
0805
R50
330
0805
FM RECEIVER
R79
22K
0805
L7
33uH
1310
5%
Non-shielded
R71
1K
0805
C72
15pF
0805
Vlf
C68
.1µF
0805
R67
2.7K
0805
R101
1.8K
0805
C44
.01µF
0805
RXDATA
R100
5.6K
0805
DATA SLICER

FIG. 8E

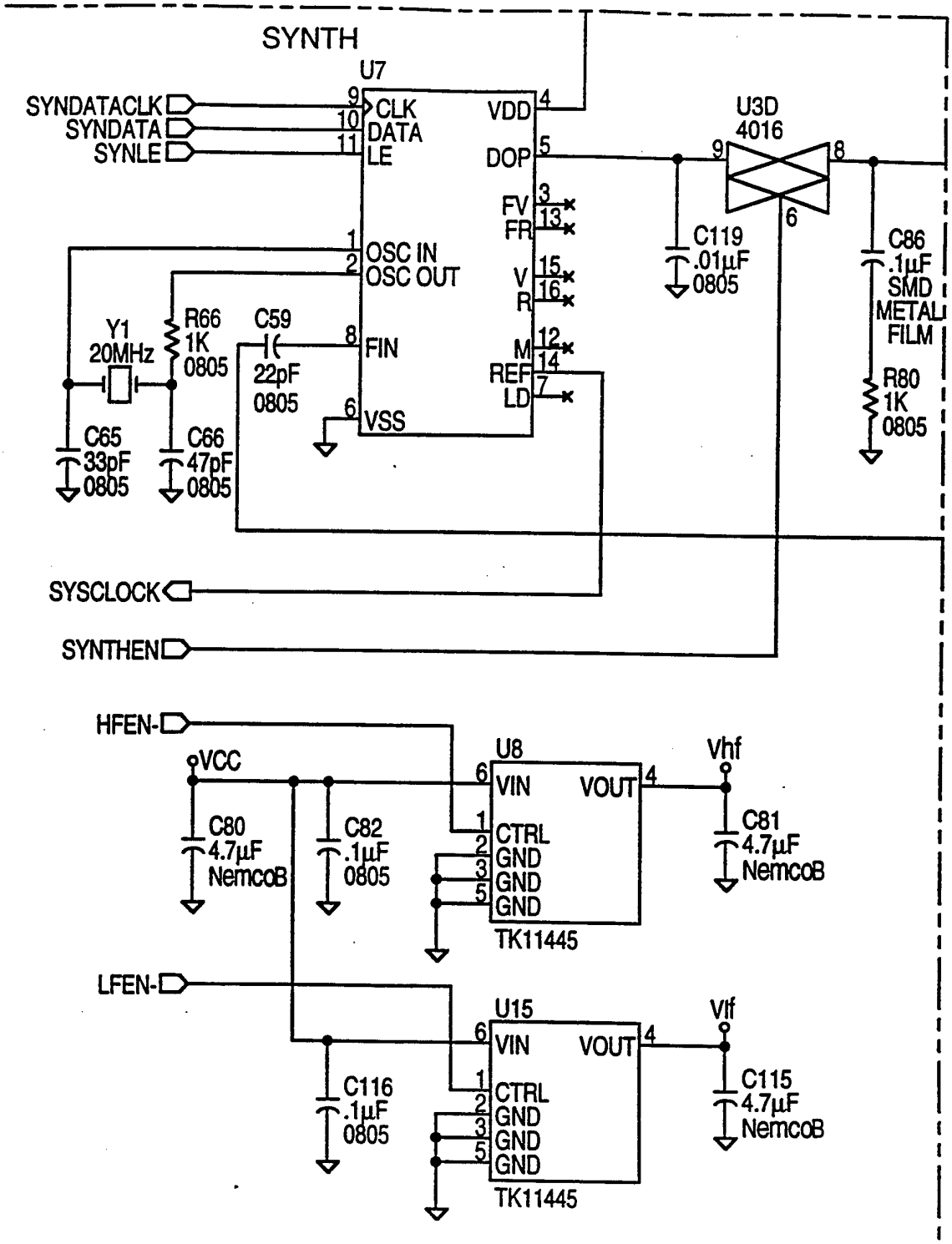

SYNTH
U7
SYNDATACLK
SYNDATA
SYNLE
CLK
DATA
LE
VDD
DOP
FV
FR
V
R
M
REF
LD
OSC IN
OSC OUT
FIN
VSS
Y1
20MHz
R66
1K
0805
C59
22pF
0805
C65
33pF
0805
C66
47pF
0805
U3D
4016
C119
.01µF
0805
C86
.1µF
SMD
METAL
FILM
R80
1K
0805
SYSCLOCK
SYNTHEN
HFEN-
VCC
U8
VIN
VOUT
CTRL
GND
GND
GND
TK11445
C80
4.7µF
NemcoB
C82
.1µF
0805
Vhf
C81
4.7µF
NemcoB
LFEN-
U15
VIN
VOUT
CTRL
GND
GND
GND
TK11445
C116
.1µF
0805
Vlf
C115
4.7µF
NemcoB

FIG. 8F

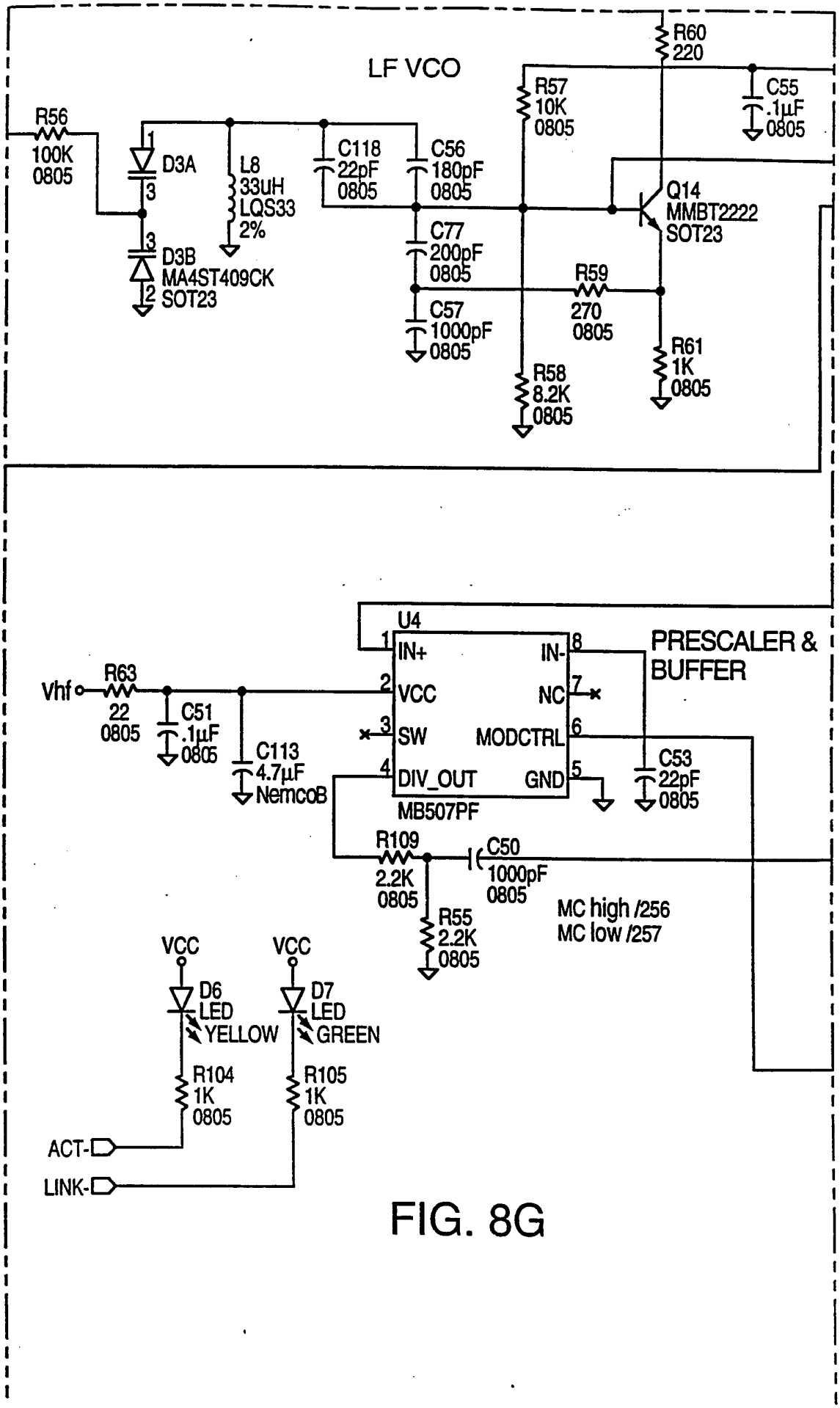

LF VCO
R56
100K
0805
D3A
D3B
MA4ST409CK
SOT23
L8
33uH
LQS33
2%
C118
22pF
0805
C56
180pF
0805
C77
200pF
0805
C57
1000pF
0805
R57
10K
0805
R58
8.2K
0805
R59
270
0805
R60
220
R61
1K
0805
C55
.1µF
0805
Q14
MMBT2222
SOT23
U4
IN+
VCC
SW
DIV_OUT
IN-
NC
MODCTRL
GND
MB507PF
PRESCALER &
BUFFER
Vhf
R63
22
0805
C51
.1µF
0805
C113
4.7µF
NemcoB
C53
22pF
0805
R109
2.2K
0805
C50
1000pF
0805
R55
2.2K
0805
MC high /256
MC low /257
VCC
VCC
D6
LED
YELLOW
D7
LED
GREEN
R104
1K
0805
R105
1K
0805
ACT-
LINK-

FIG. 8G

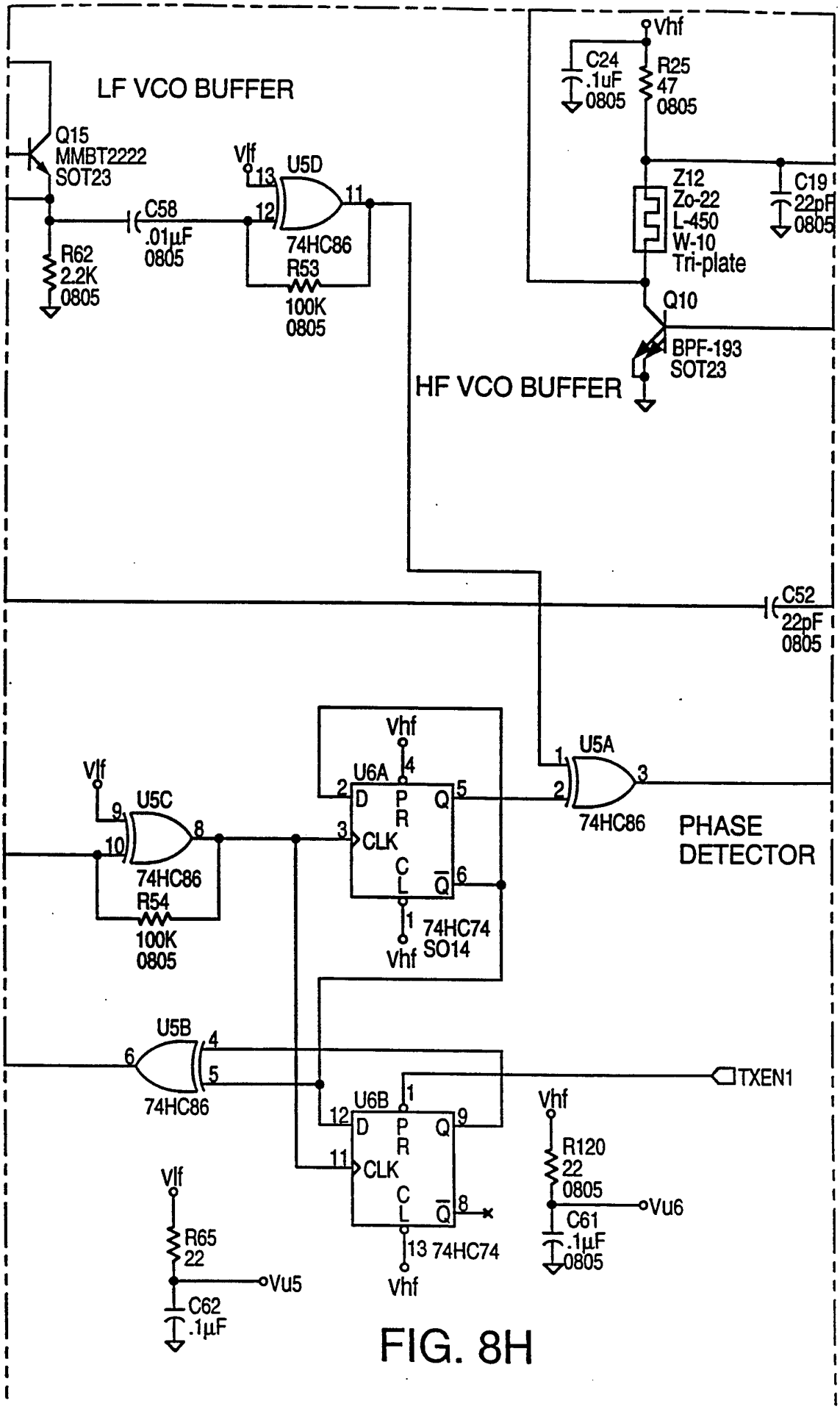

LF VCO BUFFER
Q15
MMBT2222
SOT23
R62
2.2K
0805
C58
.01µF
0805
Vlf
U5D
74HC86
R53
100K
0805
Vhf
C24
.1uF
0805
R25
47
0805
Z12
Zo-22
L-450
W-10
Tri-plate
C19
22pF
0805
Q10
BPF-193
SOT23
HF VCO BUFFER
C52
22pF
0805
U5A
74HC86
PHASE
DETECTOR
U6A
74HC74
SO14
U5C
74HC86
R54
100K
0805
U5B
74HC86
U6B
74HC74
TXEN1
R120
22
0805
C61
.1µF
0805
Vu6
R65
22
C62
.1µF
Vu5

FIG. 8H

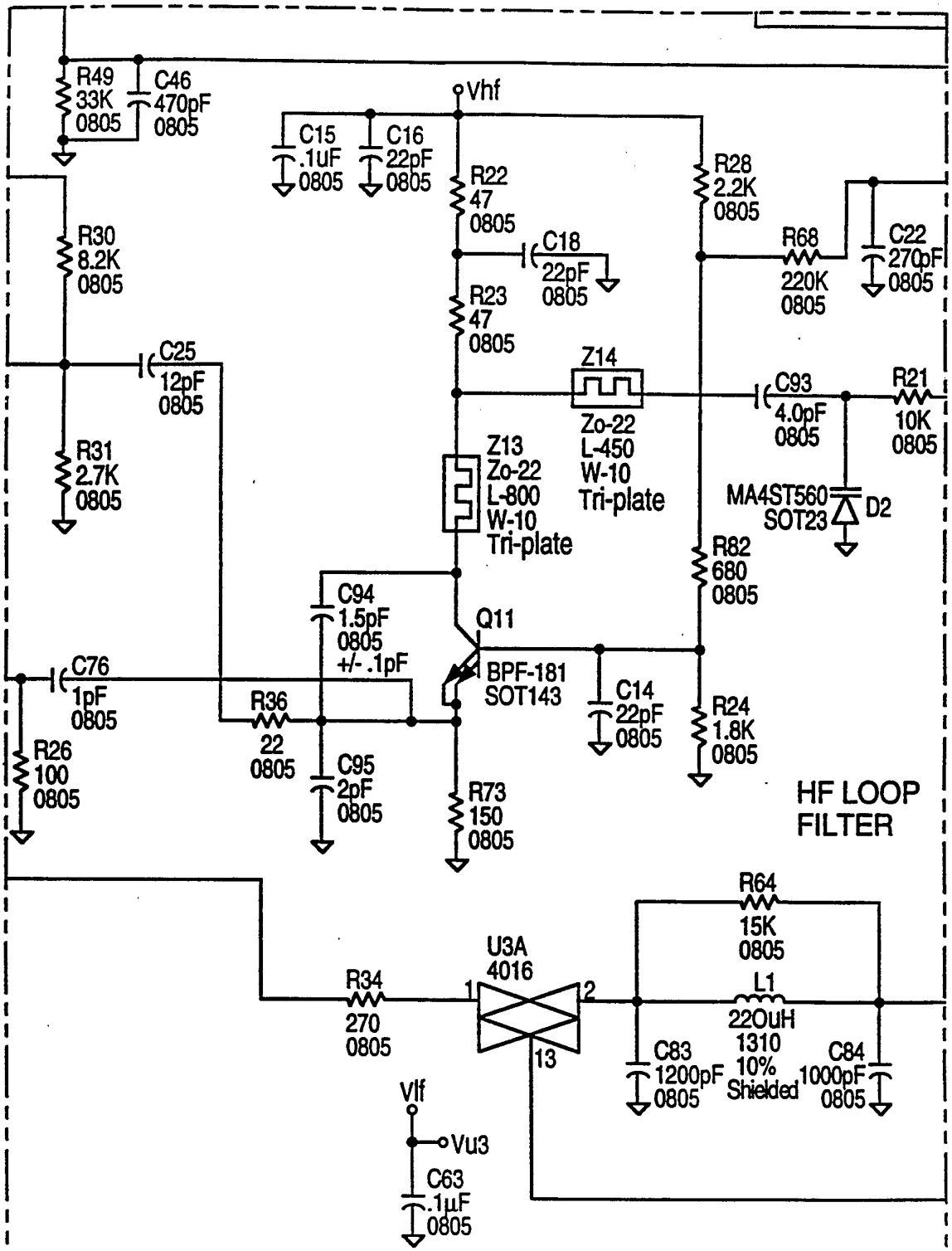

R49
33K
0805
C46
470pF
0805
Vhf
C15
.1uF
0805
C16
22pF
0805
R22
47
0805
R28
2.2K
0805
C18
22pF
0805
R68
220K
0805
C22
270pF
0805
R30
8.2K
0805
R23
47
0805
C25
12pF
0805
Z14
Zo-22
L-450
W-10
Tri-plate
C93
4.0pF
0805
R21
10K
0805
R31
2.7K
0805
Z13
Zo-22
L-800
W-10
Tri-plate
MA4ST560
SOT23
D2
R82
680
0805
C94
1.5pF
0805
+/- .1pF
Q11
C76
1pF
0805
BPF-181
SOT143
C14
22pF
0805
R24
1.8K
0805
R36
22
0805
R26
100
0805
C95
2pF
0805
R73
150
0805
HF LOOP
FILTER
R64
15K
0805
U3A
4016
R34
270
0805
1
2
13
L1
22OuH
1310
10%
Shielded
C83
1200pF
0805
C84
1000pF
0805
Vlf
Vu3
C63
.1µF
0805

FIG. 8I

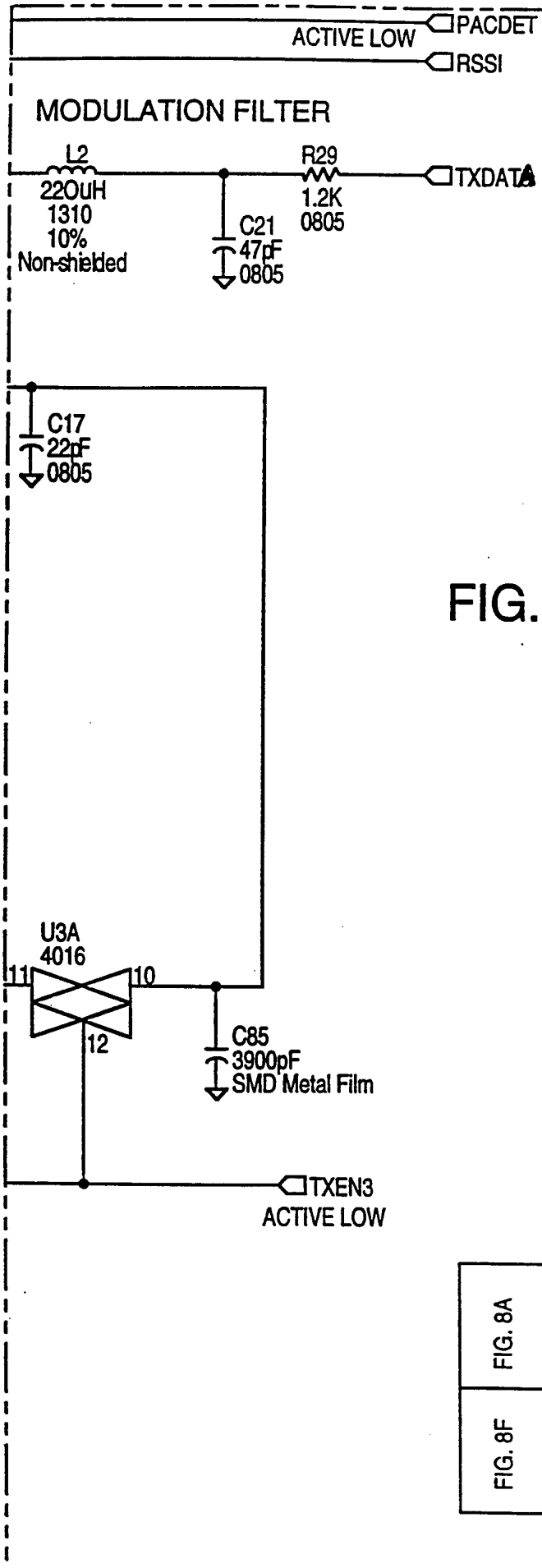



FIG. 8J

KEY TO FIG. 8

| | | | | |
|---|---|---|---|---|
| FIG. 8A | FIG. 8B | FIG. 8C | FIG. 8D | FIG. 8E |
| FIG. 8F | FIG. 8G | FIG. 8H | FIG. 8I | FIG. 8J |

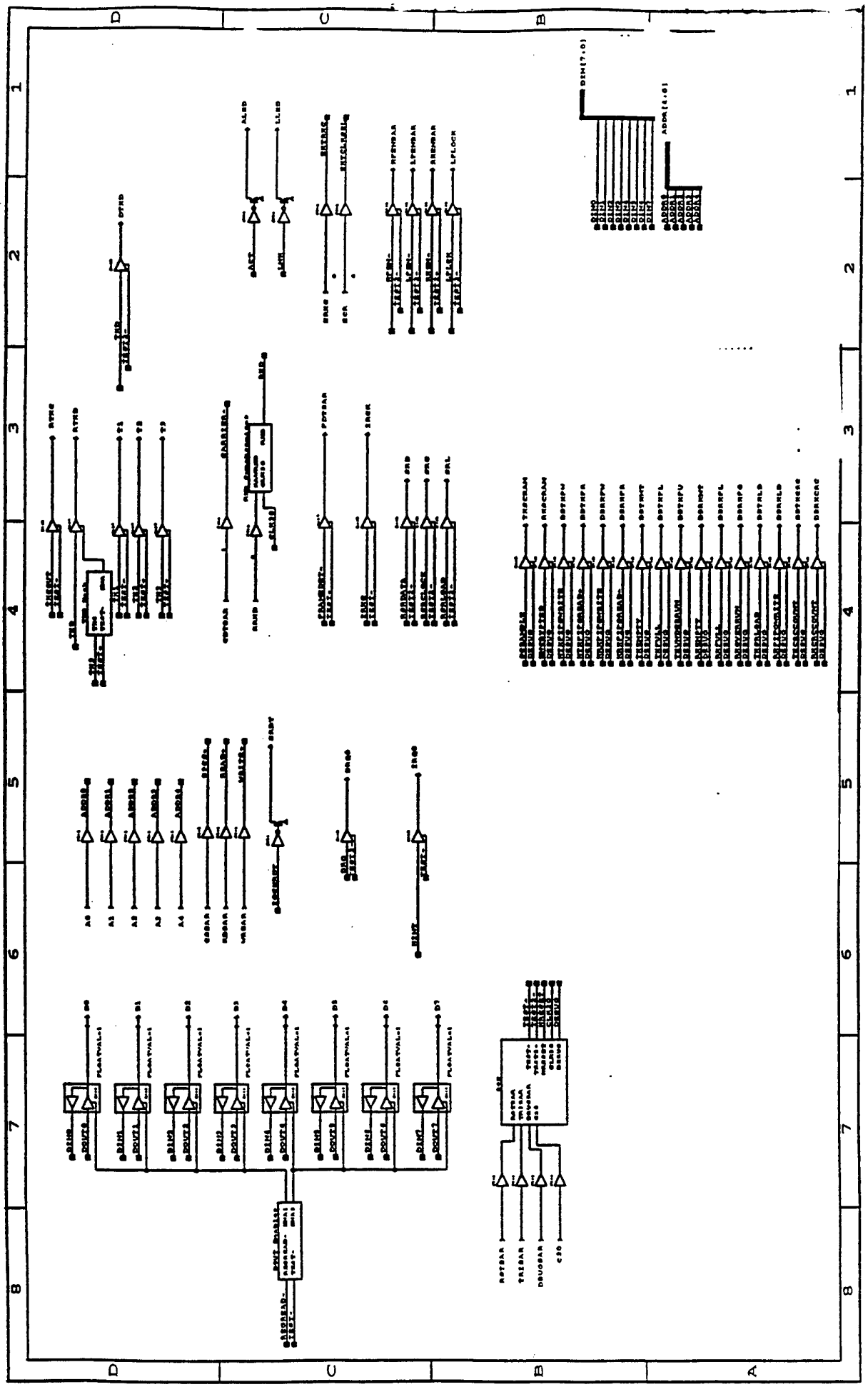
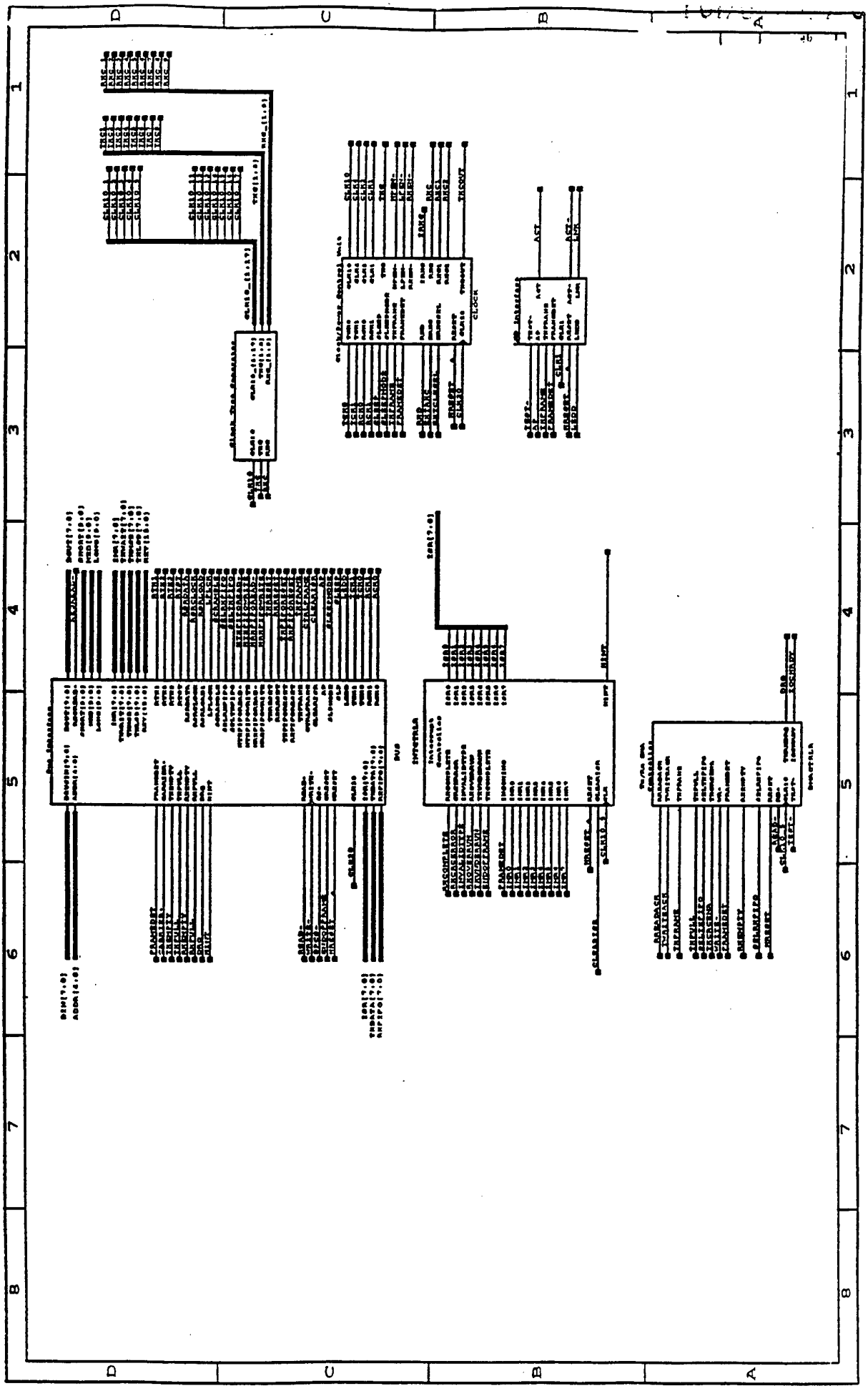
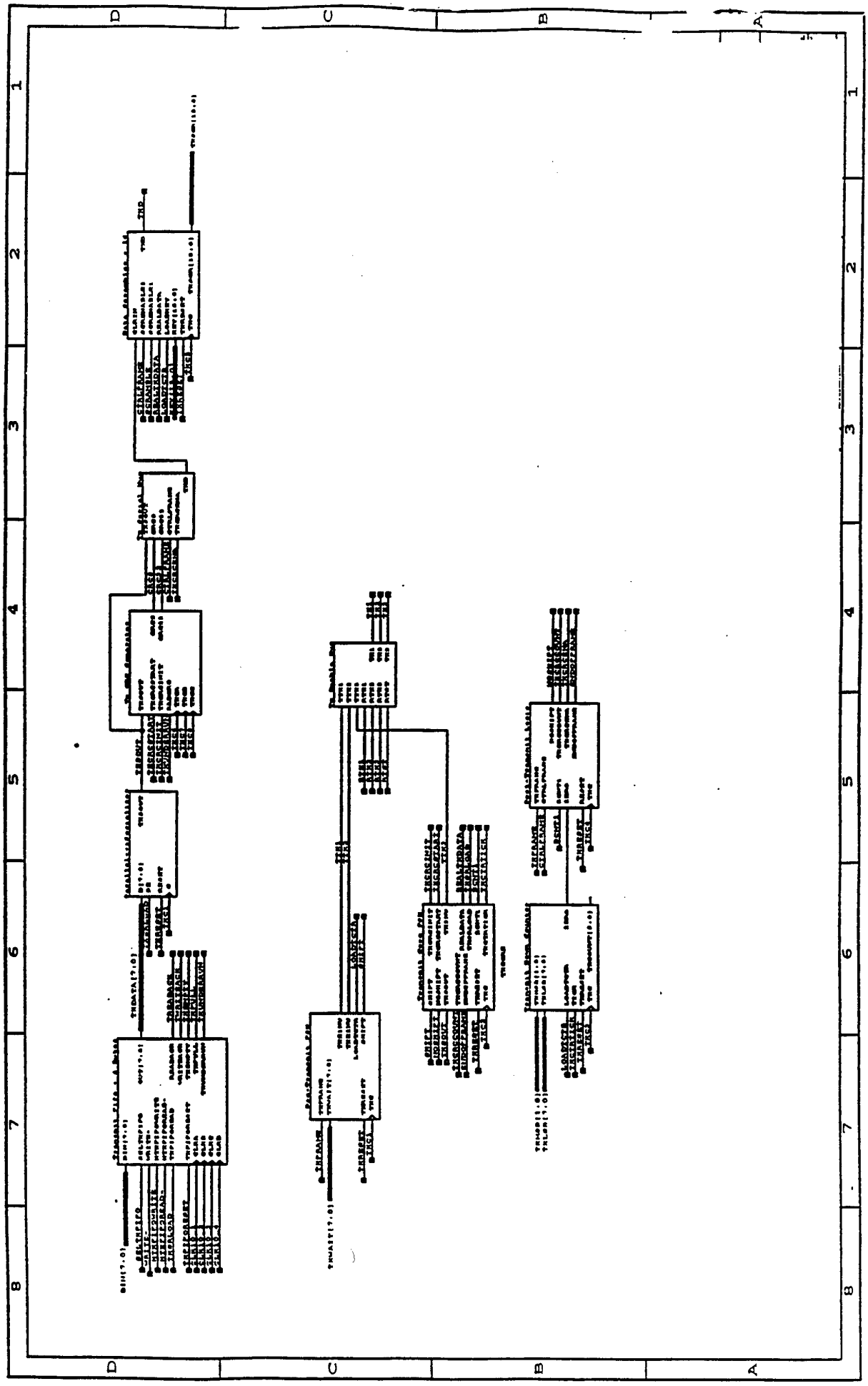


Fig. 9A





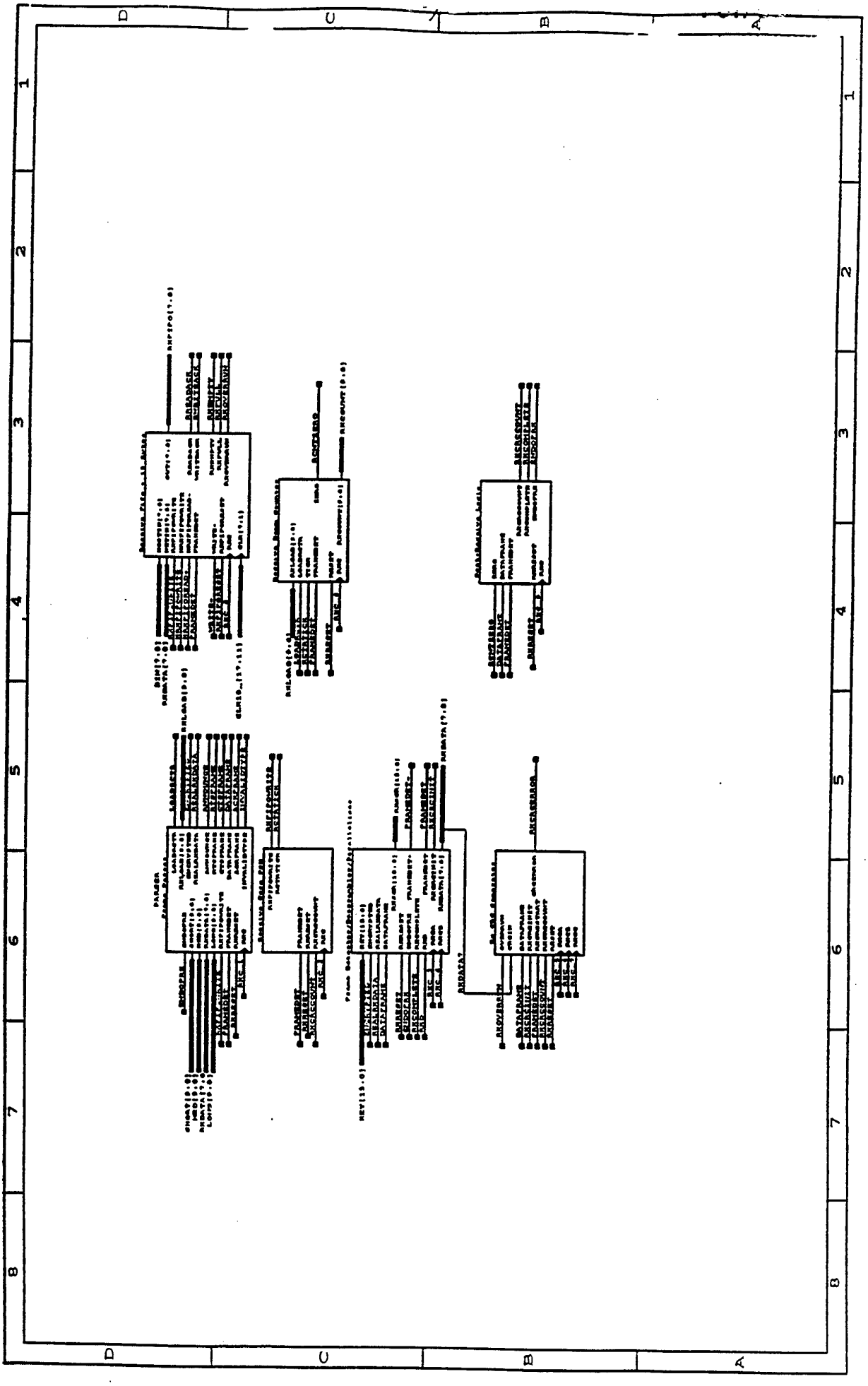


Fig 9D

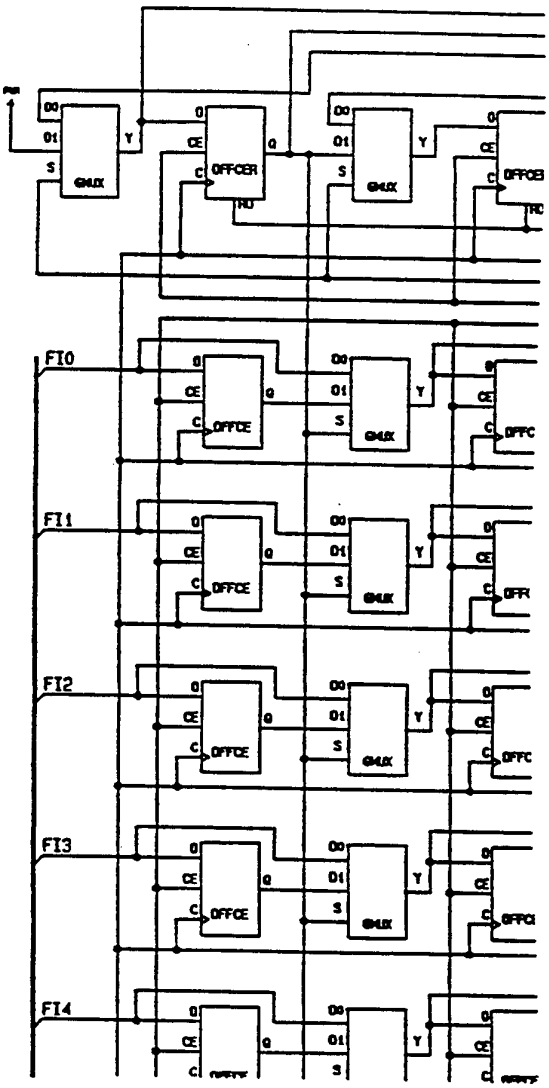



FIG 10A

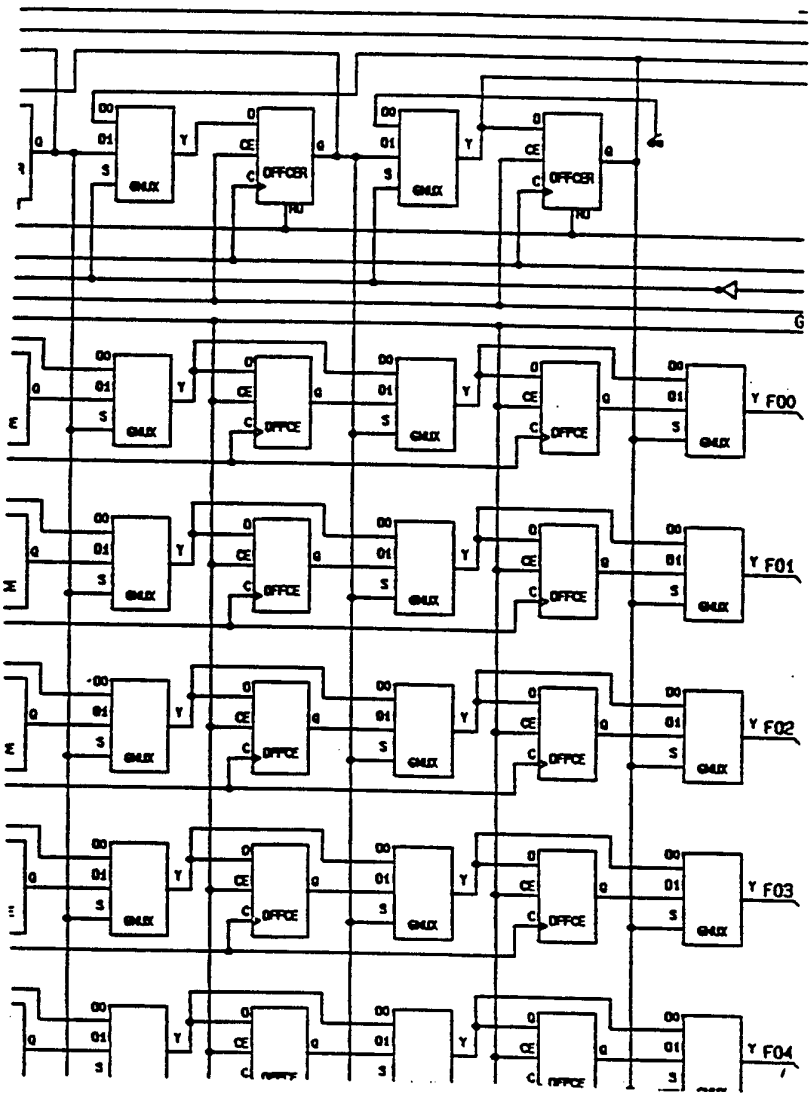



FIG 10 B

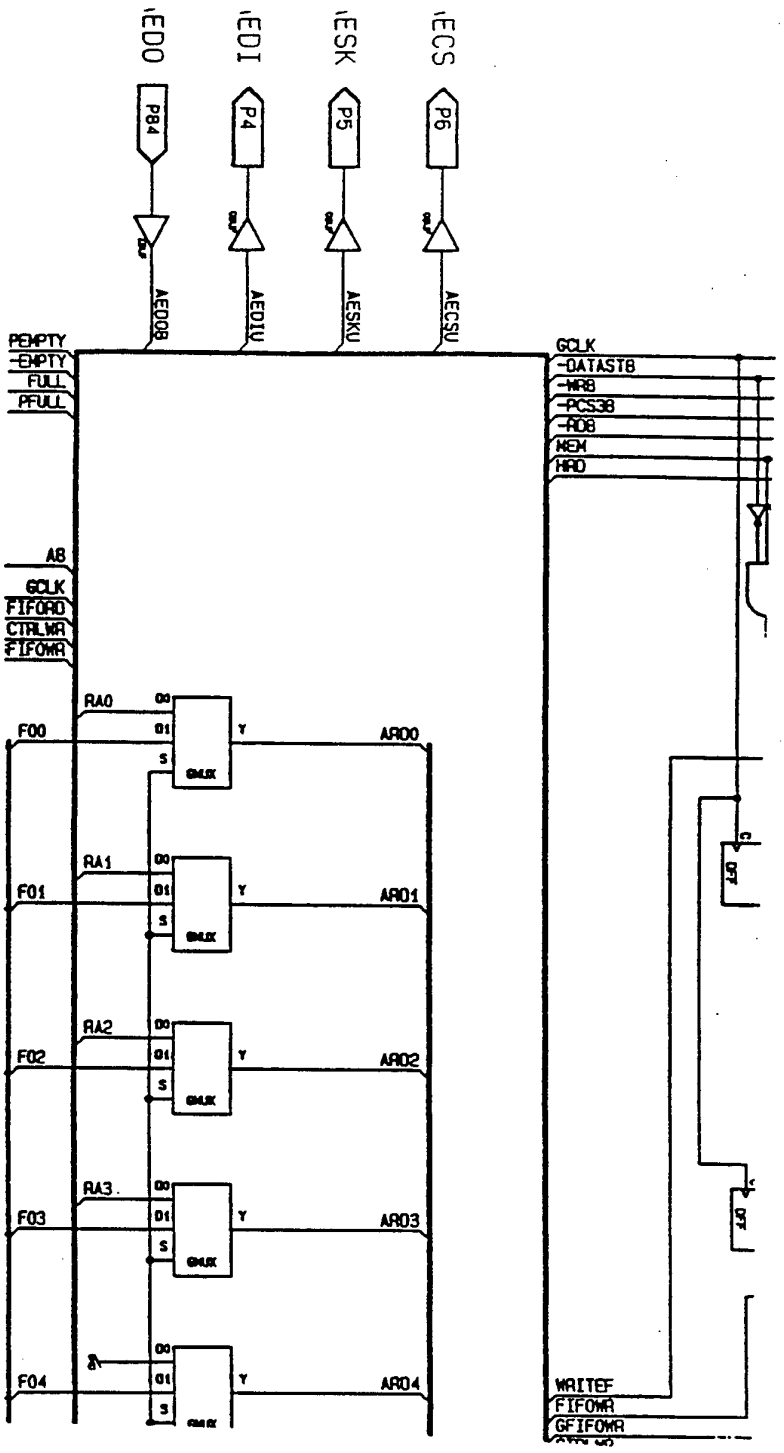

EDO
EDI
ESK
ECS
P84
P4
P5
P6
AEDOB
AEDIU
AESKU
AECSU
PEMPTY
-EMPTY
FULL
PFULL
A8
GCLK
FIFORD
CTRLWR
FIFOWR
GCLK
-DATAST8
-WR8
-PCS38
-RD8
MEM
HRD
RA0
RA1
RA2
RA3
F00
F01
F02
F03
F04
AR00
AR01
AR02
AR03
AR04
WAITEF
FIFOWR
GFIFOWR

FIG 10C

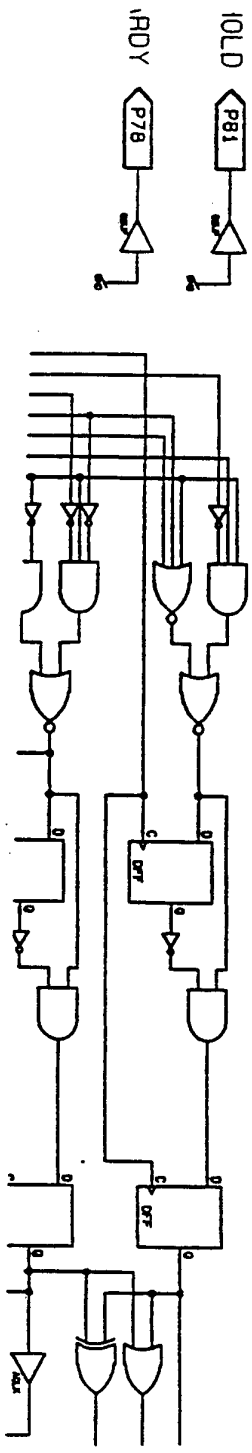



FIG. 10D

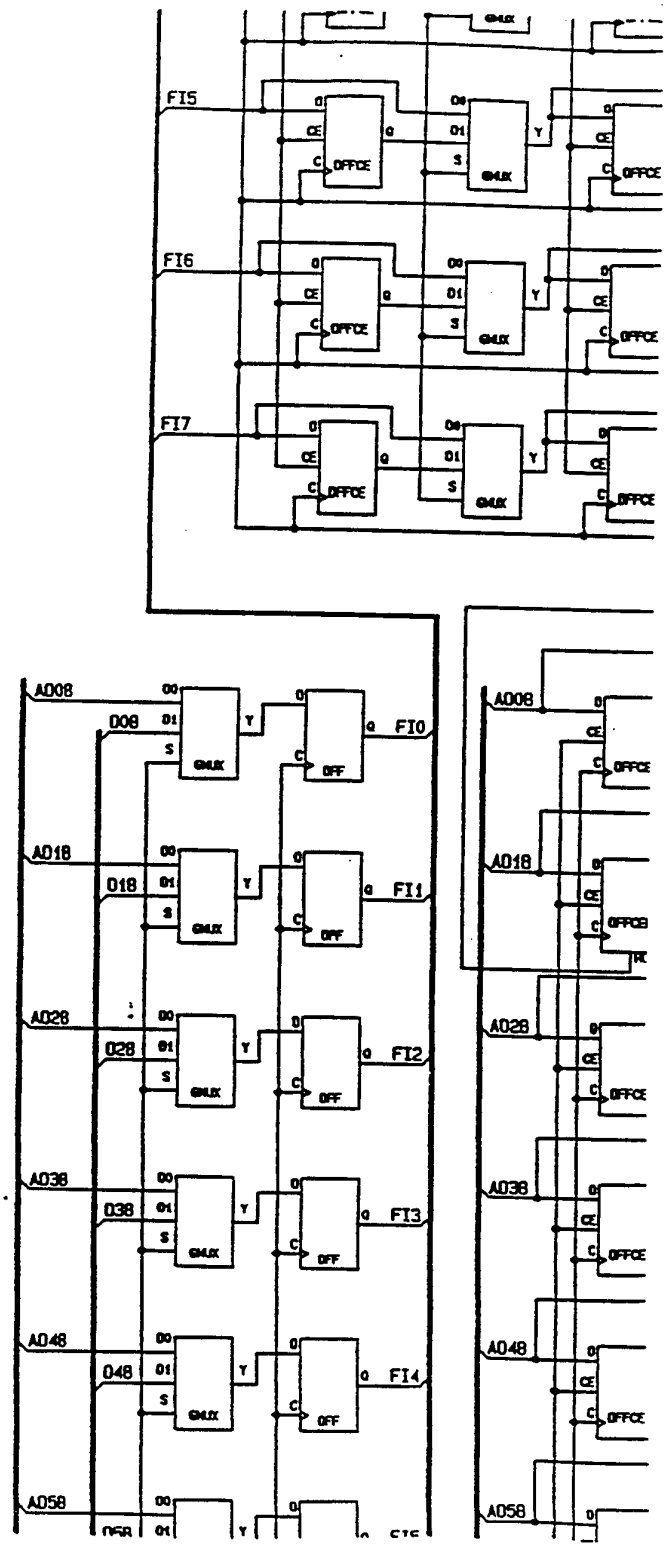

FI5
FI6
FI7
A008
D08
FI0
A018
D18
FI1
A028
D28
FI2
A038
D38
FI3
A048
D48
FI4
A058
A008
A018
A028
A038
A048
A058

FIG 10E

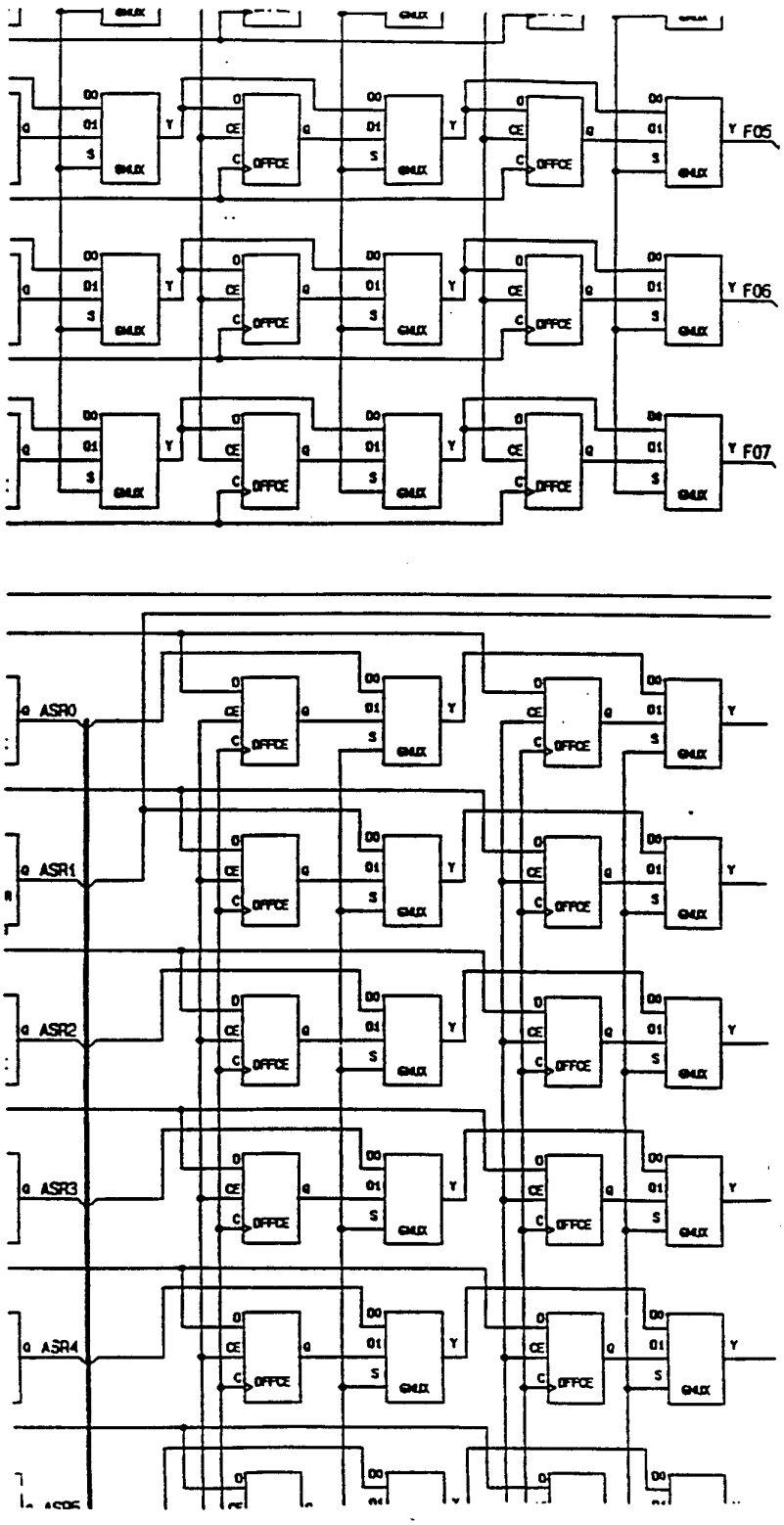



FIG 10F

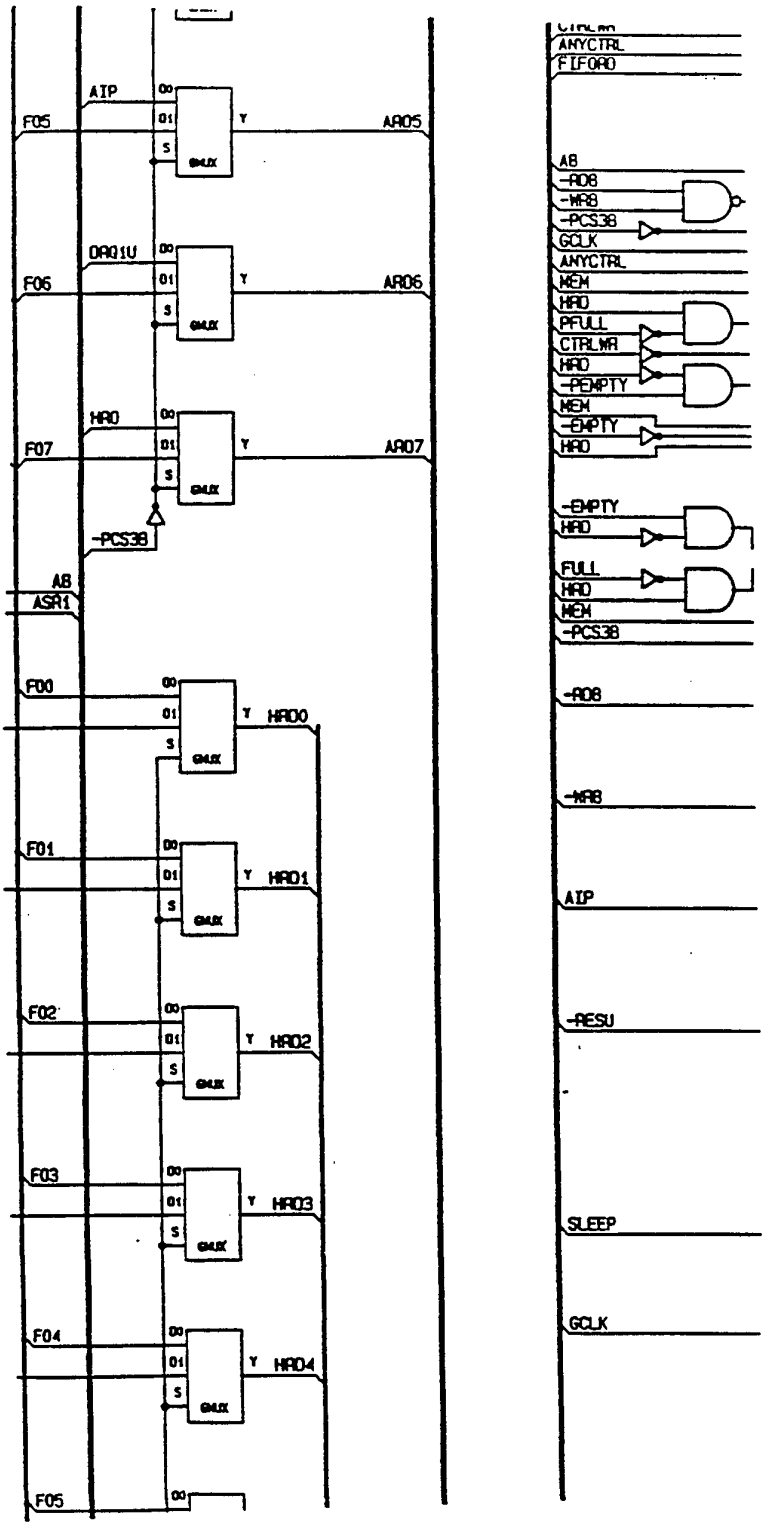

AIP
F05
AR05
DAQ1U
F06
AR06
HR0
F07
AR07
-PCS3B
AB
ASR1
F00
HRD0
F01
HRD1
F02
HRD2
F03
HRD3
F04
HRD4
F05
ANYCTRL
FIFORD
AB
-RDB
-WRB
-PCS3B
GCLK
ANYCTRL
MEM
HRD
PFULL
CTRLWR
HRD
-PEMPTY
MEM
-EMPTY
HRD
-EMPTY
HRD
FULL
HRD
MEM
-PCS3B
-RDB
-WRB
AIP
-RESU
SLEEP
GCLK

FIG 10G

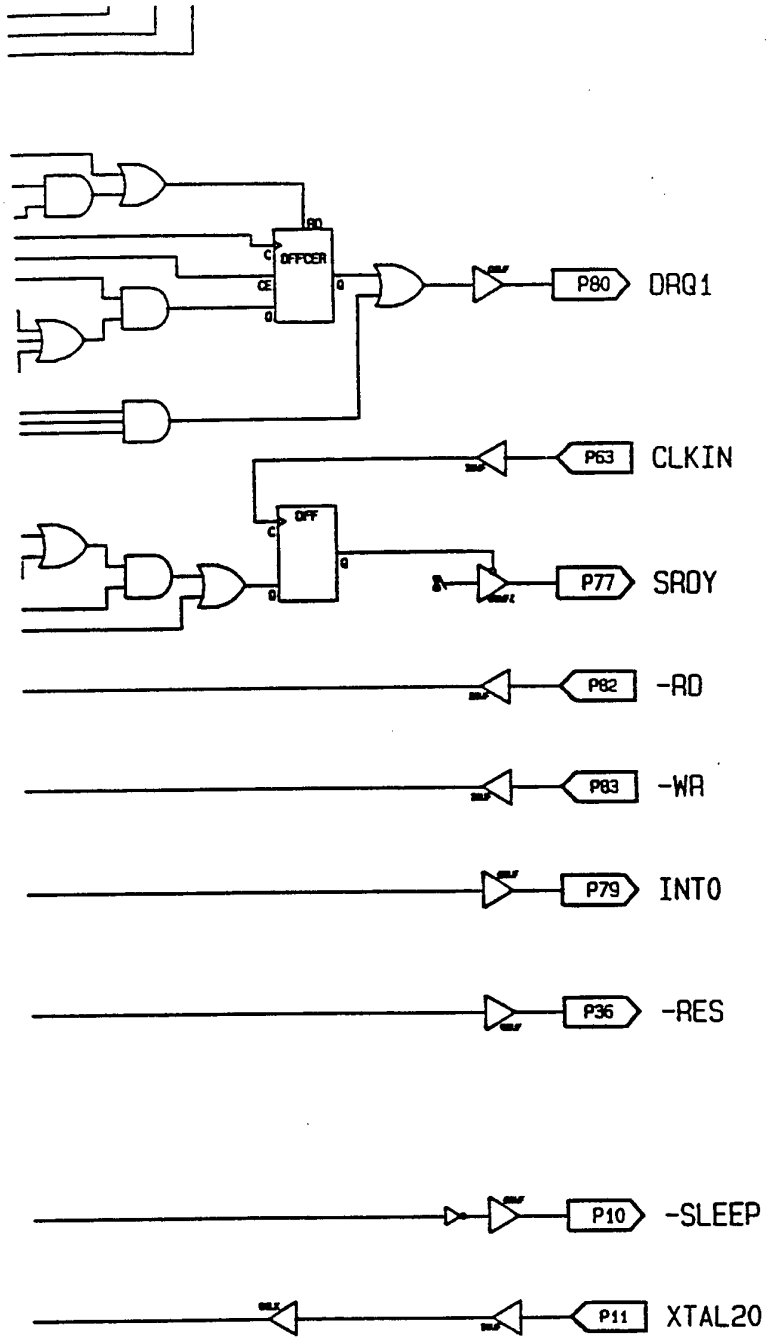

DFFCER
C
CE
D
Q
P80
DRQ1
P63
CLKIN
DFF
P77
SRDY
P82
-RD
P83
-WR
P79
INT0
P36
-RES
P10
-SLEEP
P11
XTAL20

Fig. 10H

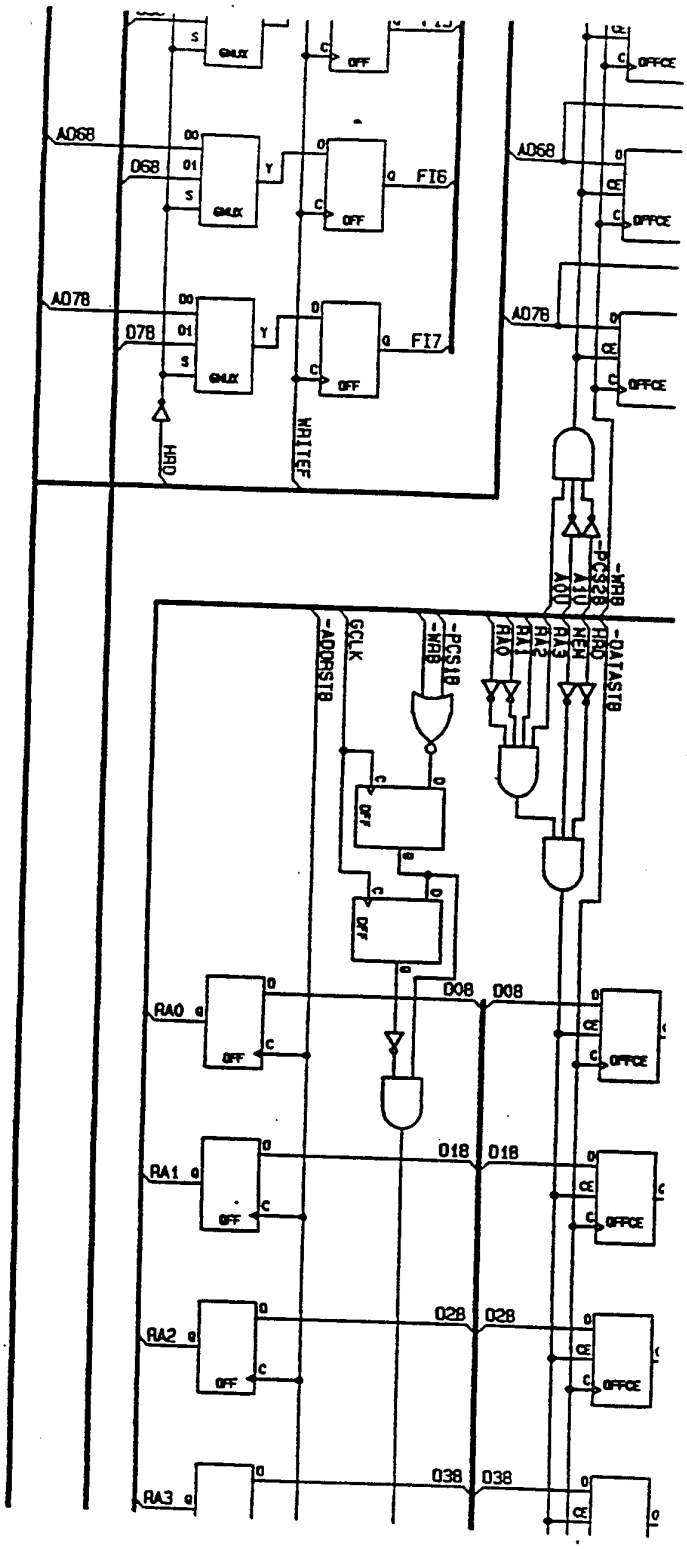

AD6B
D6B
AD7B
D7B
FI6
FI7
HRD
WRITEF
OFFCE
-WRB
-PCS2B
A0U
A1U
-DATASTB
RA0
RA1
RA2
RA3
NEW
-PCS1B
-ADDRSTB
GCLK
D0B
D1B
D2B
D3B

FIG 10I

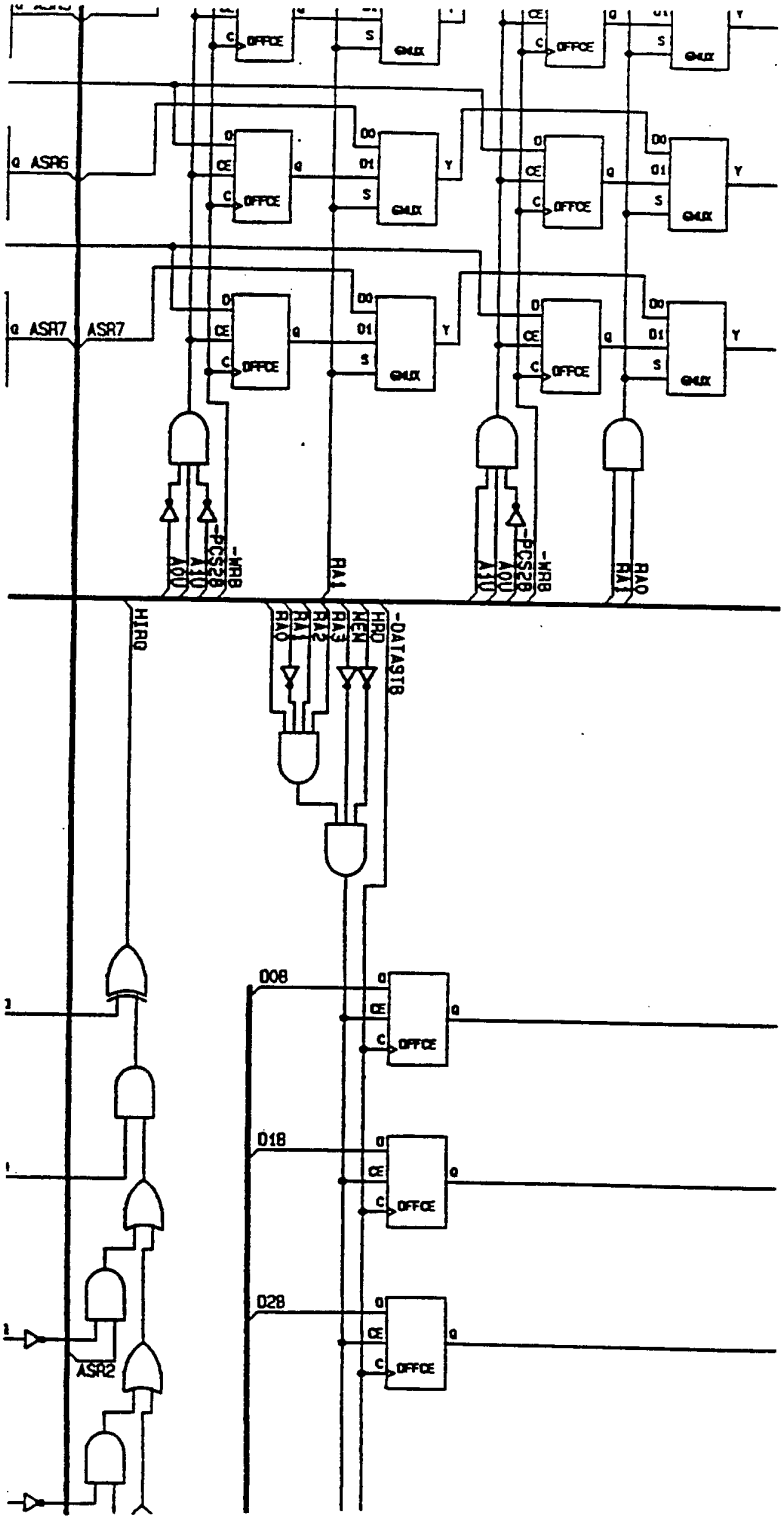



FIG. 10J

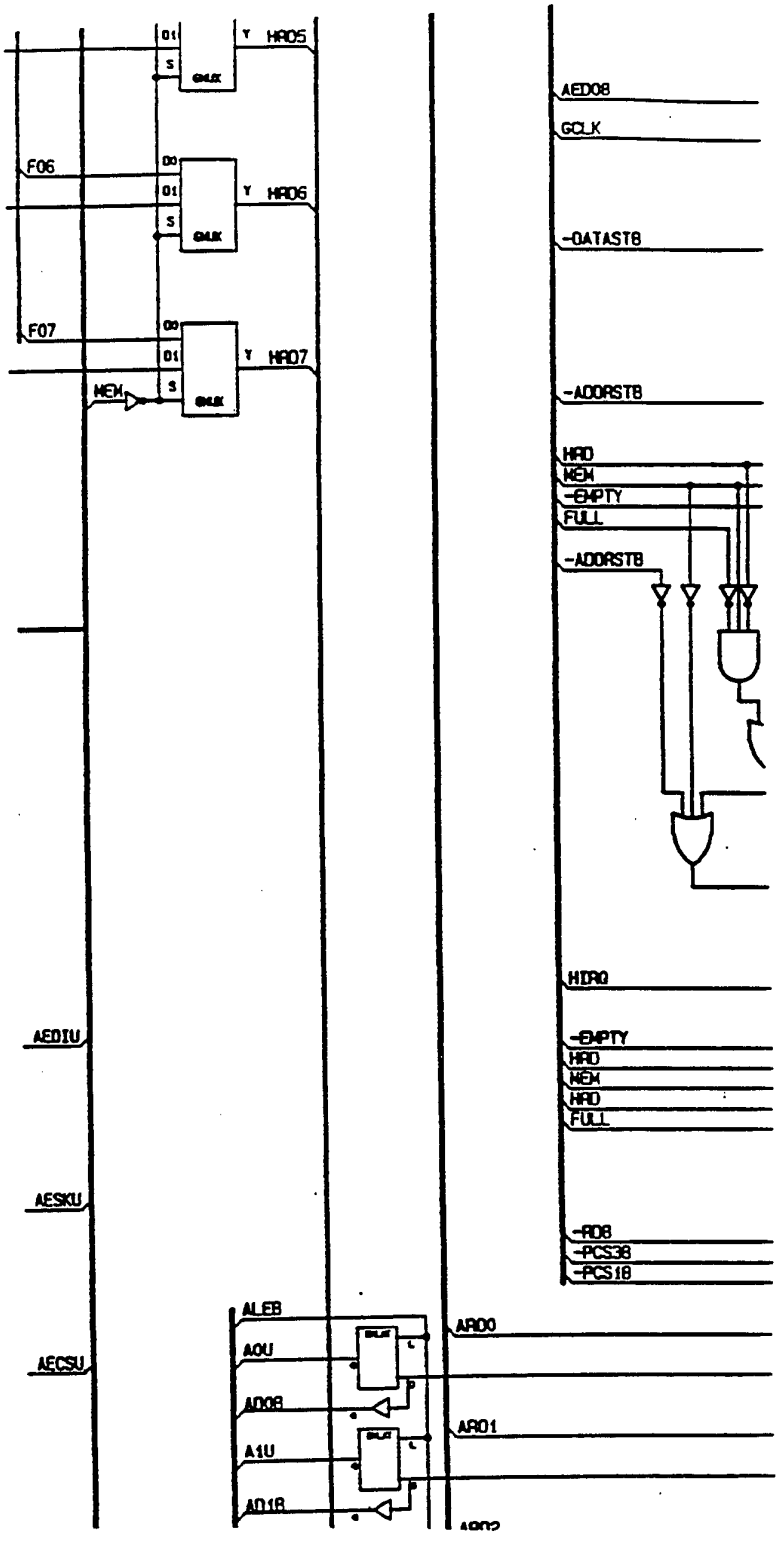

HRD5
HRD6
HRD7
F06
F07
MEM
AED08
GCLK
-DATAST8
-ADDRST8
HRD
MEM
-EMPTY
FULL
-ADDRST8
HIRQ
-EMPTY
HRD
MEM
HRD
FULL
-RD8
-PCS38
-PCS18
AEDIU
AESKU
AECSU
ALEB
A0U
AD08
A1U
AD18
AR00
AR01

FIG. 10K

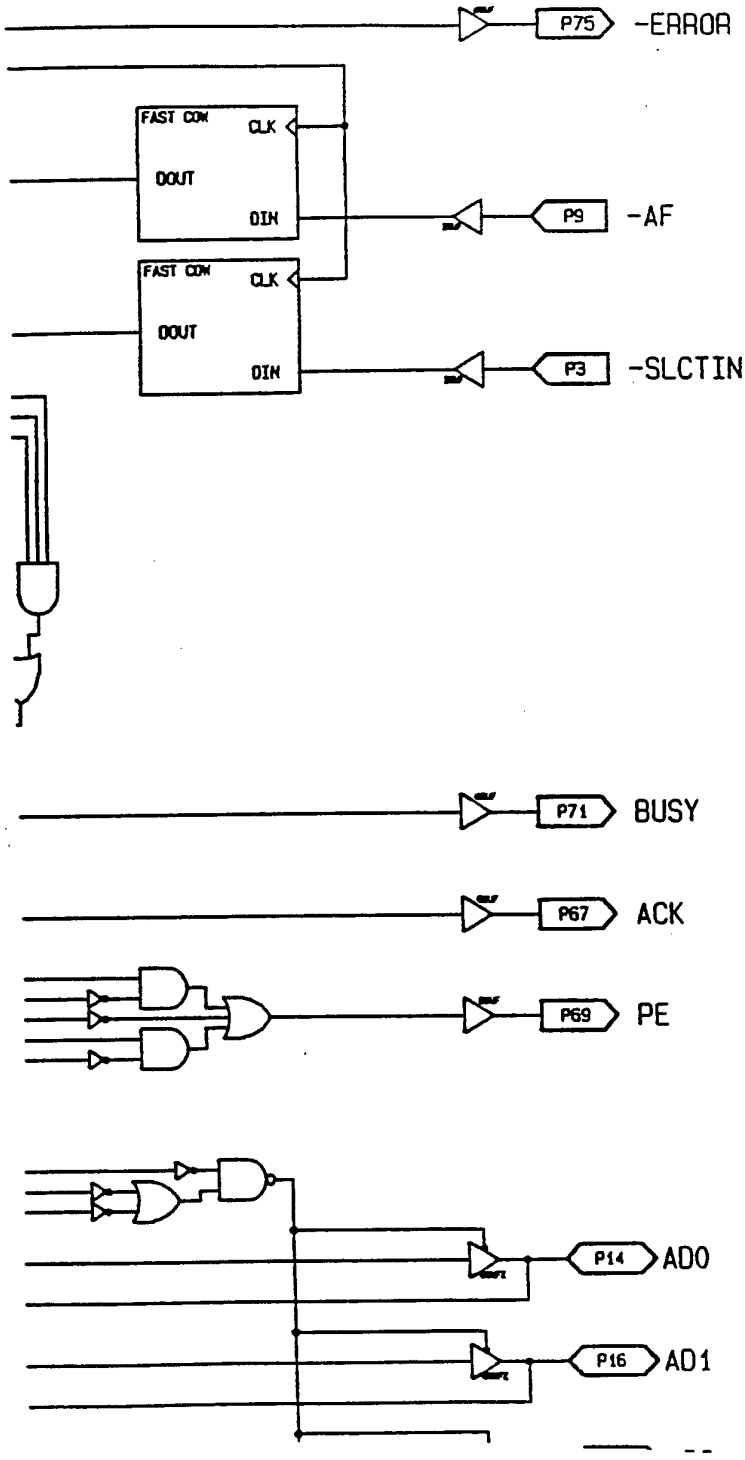

P75
-ERROR
FAST CON
CLK
DOUT
DIN
P9
-AF
FAST CON
CLK
DOUT
DIN
P3
-SLCTIN
P71
BUSY
P67
ACK
P69
PE
P14
AD0
P16
AD1

Fig. 10L

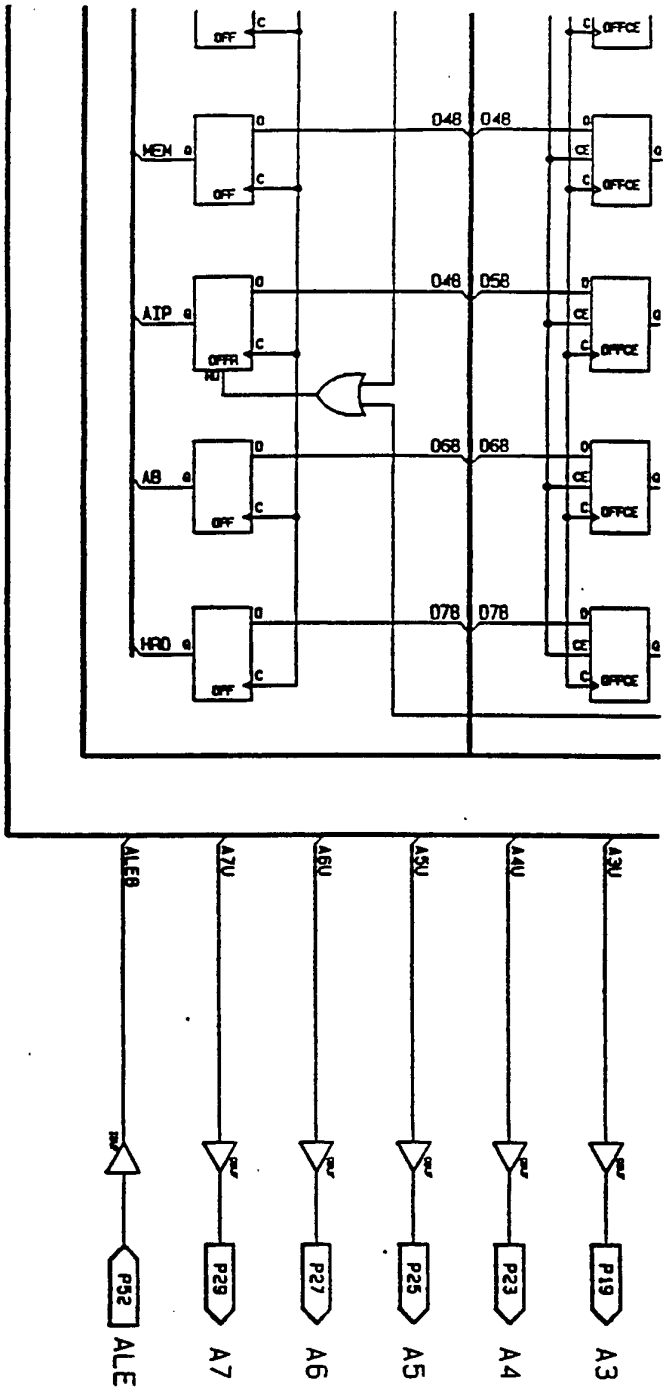

MEM
AIP
AB
HRD
048
058
068
078
ALE0
A7U
A6U
A5U
A4U
A3U
P32
P29
P27
P25
P23
P19
ALE
A7
A6
A5
A4
A3

FIG 10M

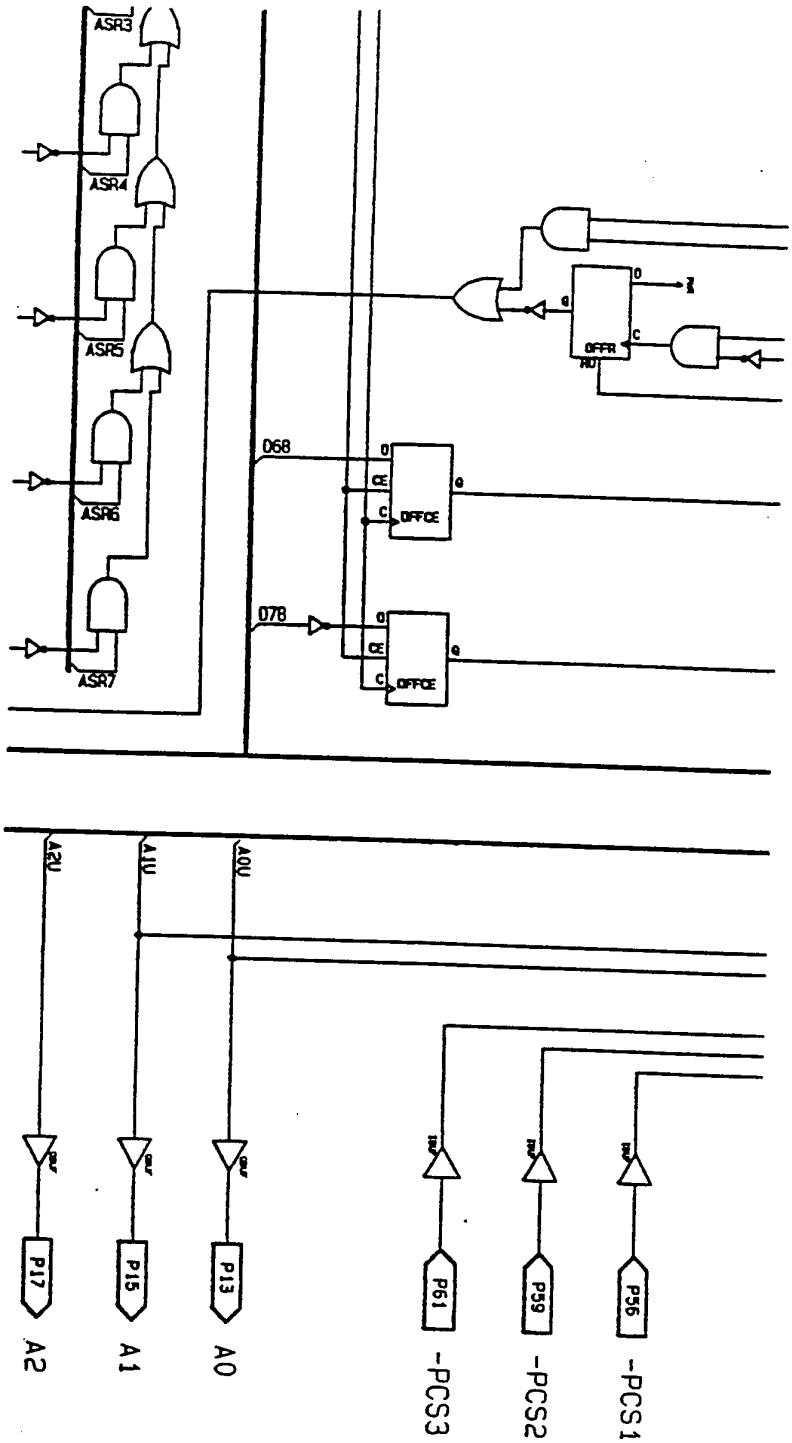

ASR3
ASR4
ASR5
ASR6
ASR7
068
078
OFFCE
OFFCE
OFFR
A2U
A1U
A0U
P17
P15
P13
P61
P59
P56
A2
A1
A0
-PCS3
-PCS2
-PCS1

Fig. 10 N

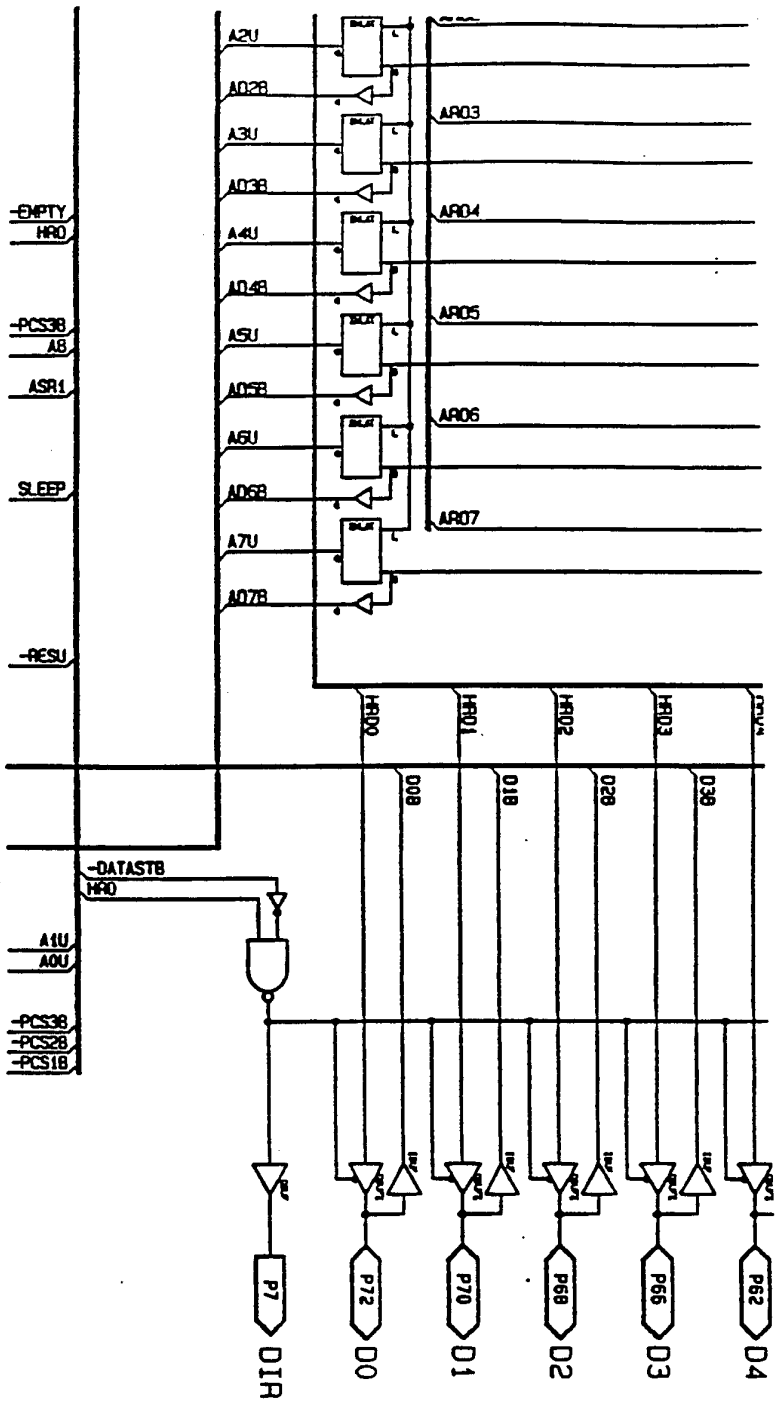
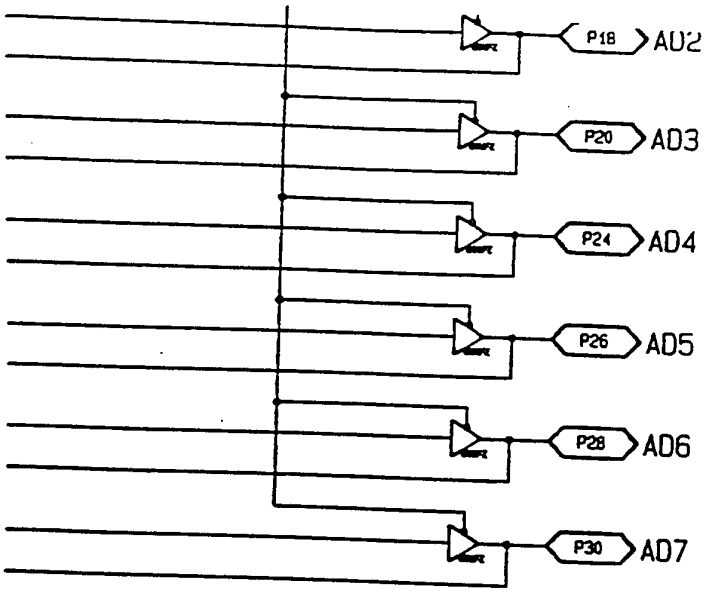
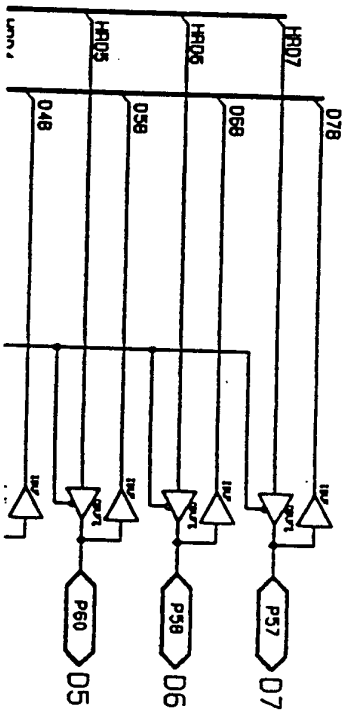

A2U
A02B
A3U
A03B
A4U
A04B
A5U
A05B
A6U
A06B
A7U
A07B
AR03
AR04
AR05
AR06
AR07
-EMPTY
HR0
-PCS3B
A8
ASR1
SLEEP
-RESU
HR00
HR01
HR02
HR03
D00
D10
D20
D30
-DATASTB
HR0
A1U
A0U
-PCS3B
-PCS2B
-PCS1B
P7
P72
P70
P68
P66
P62
DIR
D0
D1
D2
D3
D4

FIG. 100




Key to Fig. 10

| | | | |
|---|---|---|---|
| Fig. 10A | Fig. 10B | Fig. 10C | Fig. 10D |
| Fig. 10E | Fig. 10F | Fig. 10G | Fig. 10H |
| Fig. 10I | Fig. 10J | Fig. 10K | Fig. 10L |
| Fig. 10M | Fig. 10N | Fig. 10O | Fig. 10P |

Fig. 10P

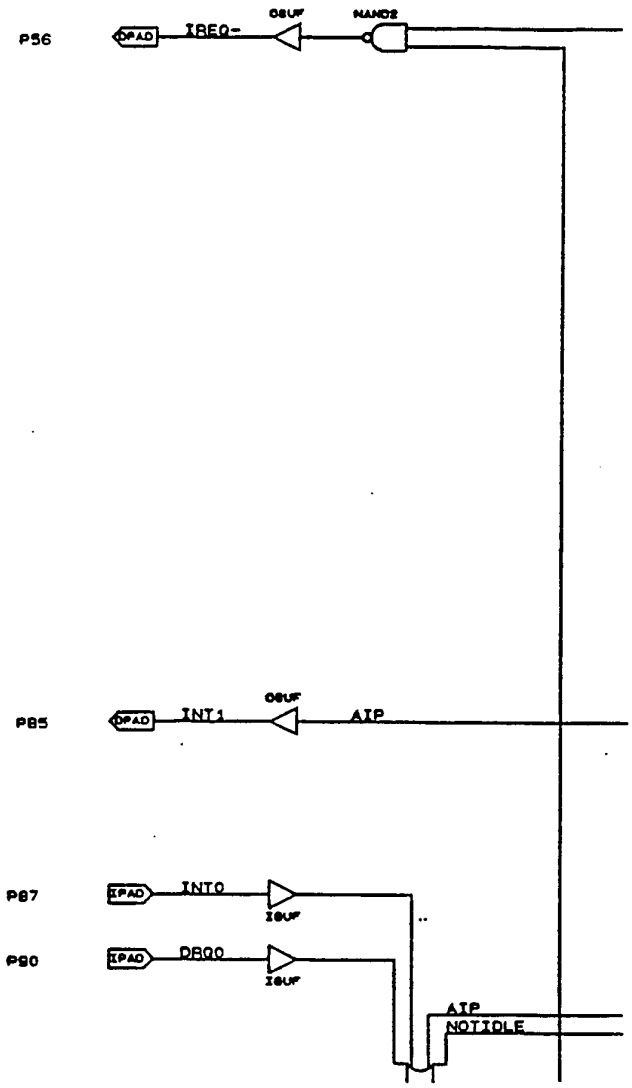

P56
OPAD
IREQ-
OBUF
NAND2
P85
OPAD
INT1
OBUF
AIP
P87
IPAD
INT0
IBUF
P90
IPAD
DRQ0
IBUF
AIP
NOTIDLE

Fig. 11 A

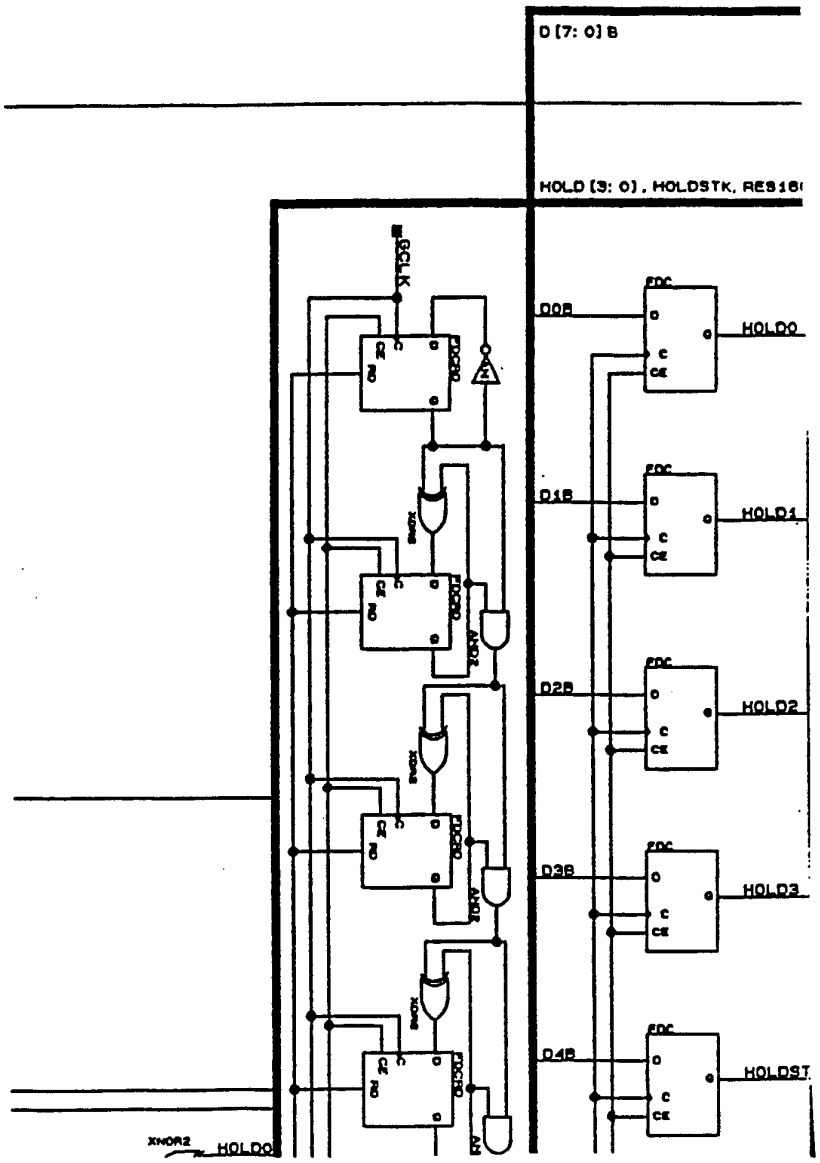

D[7:0]B
HOLD[3:0], HOLDSTK, RES1&
BCLK
D0B
D1B
D2B
D3B
D4B
HOLD0
HOLD1
HOLD2
HOLD3
HOLDST
FDC
FDCRD
XNOR2
AND2
XNOR2
HOLD0

Fig. 11B

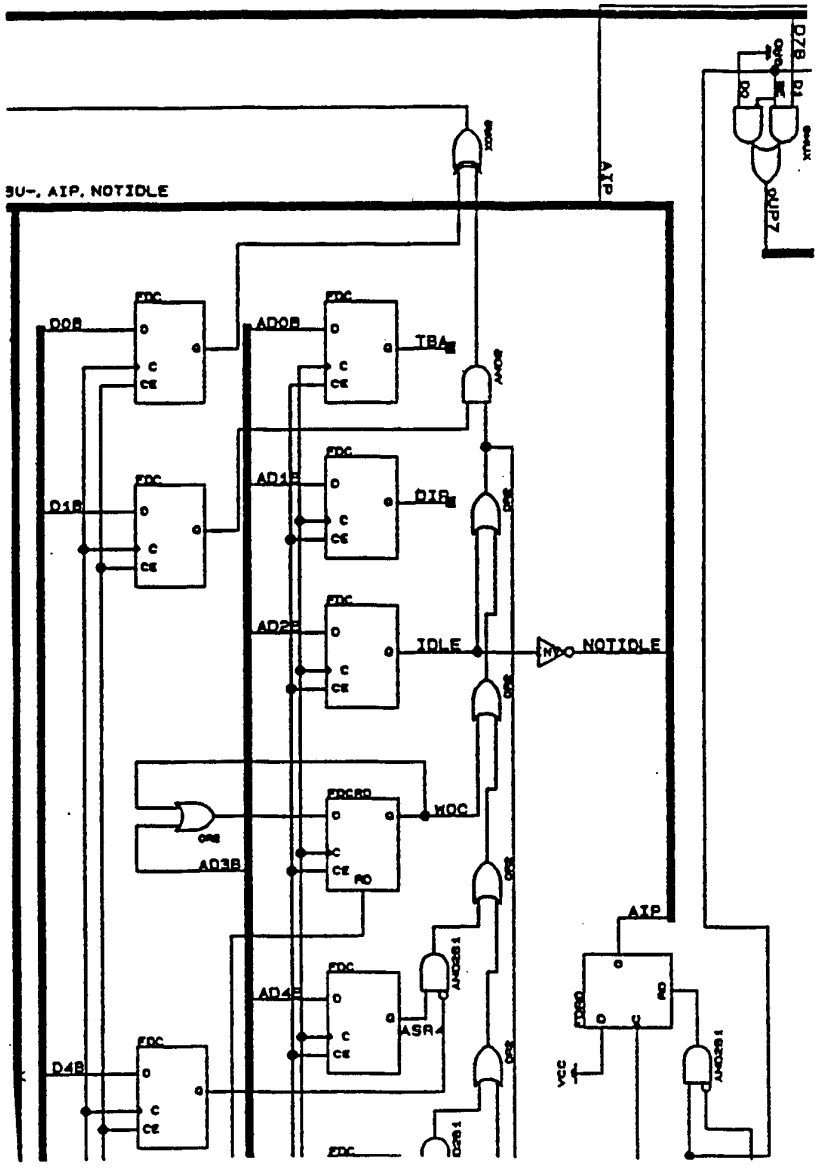

AIP
NOTIDLE
D0B
AD0B
TBA
D1B
AD1B
DIB
AD2B
IDLE
NOTIDLE
WOC
AD3B
AIP
AD4B
ASR
D4B
FDC

Fig. 11 C

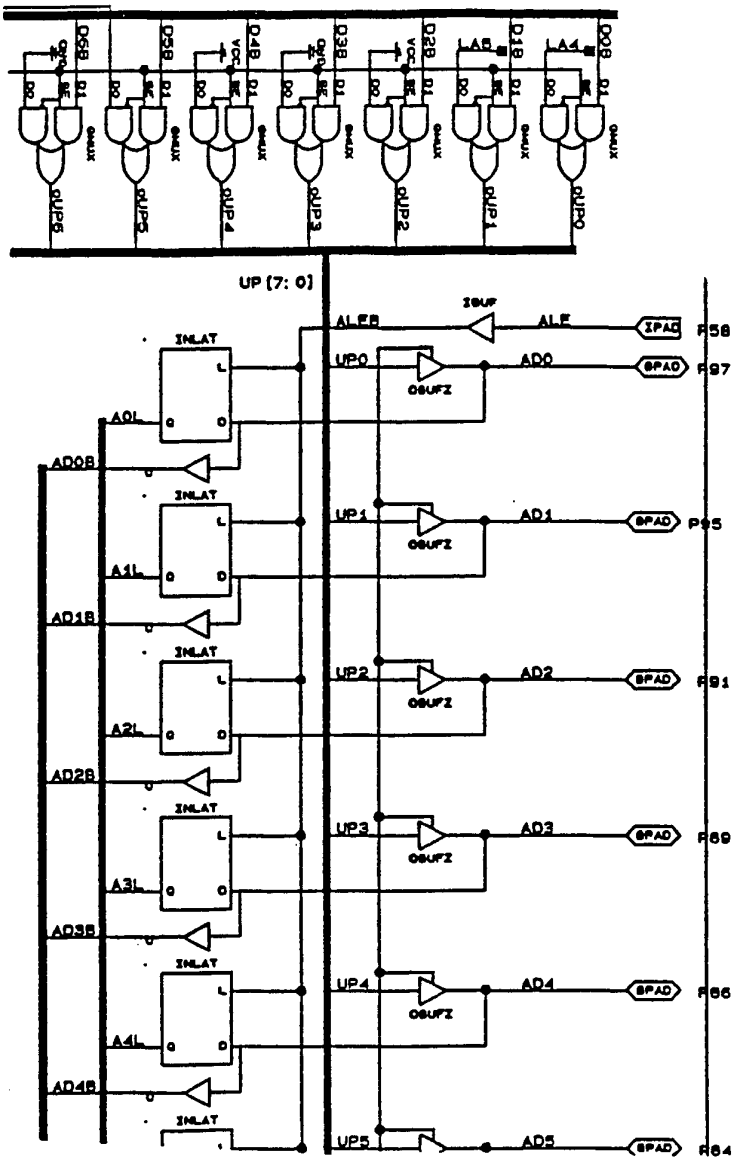

UP [7: 0]
ALEB
IBUF
ALE
IPAD
INLAT
UP0
OBUFZ
AD0
OPAD
A0L
AD08
UP1
AD1
A1L
AD18
UP2
AD2
A2L
AD28
UP3
AD3
A3L
AD38
UP4
AD4
A4L
AD48
UP5
AD5

Fig. 11D

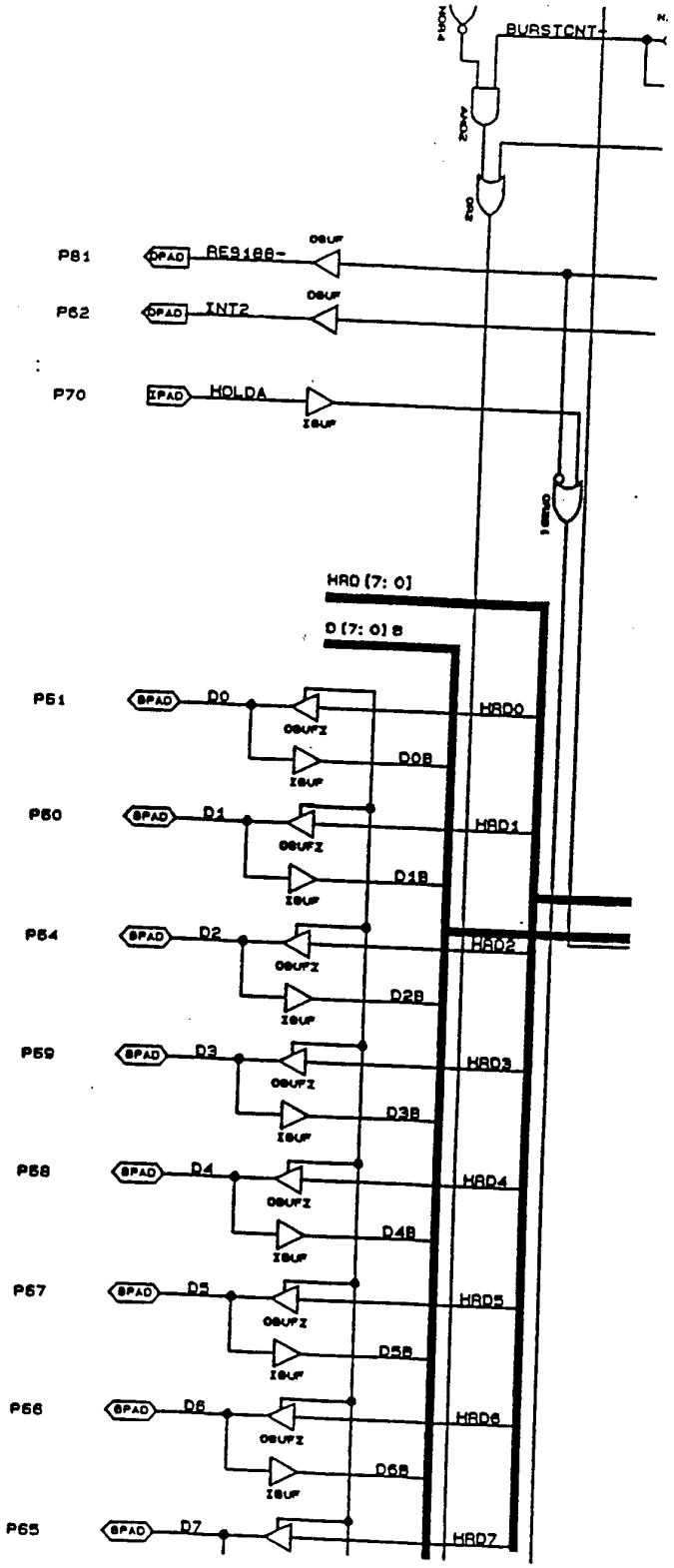

BURSTCNT
P81
RES188-
P62
INT2
P70
HOLDA
HRD [7: 0]
D [7: 0] B
P51
D0
HRD0
D0B
P50
D1
HRD1
D1B
P54
D2
HRD2
D2B
P59
D3
HRD3
D3B
P58
D4
HRD4
D4B
P57
D5
HRD5
D5B
P56
D6
HRD6
D6B
P65
D7
HRD7

Fig. 11 E

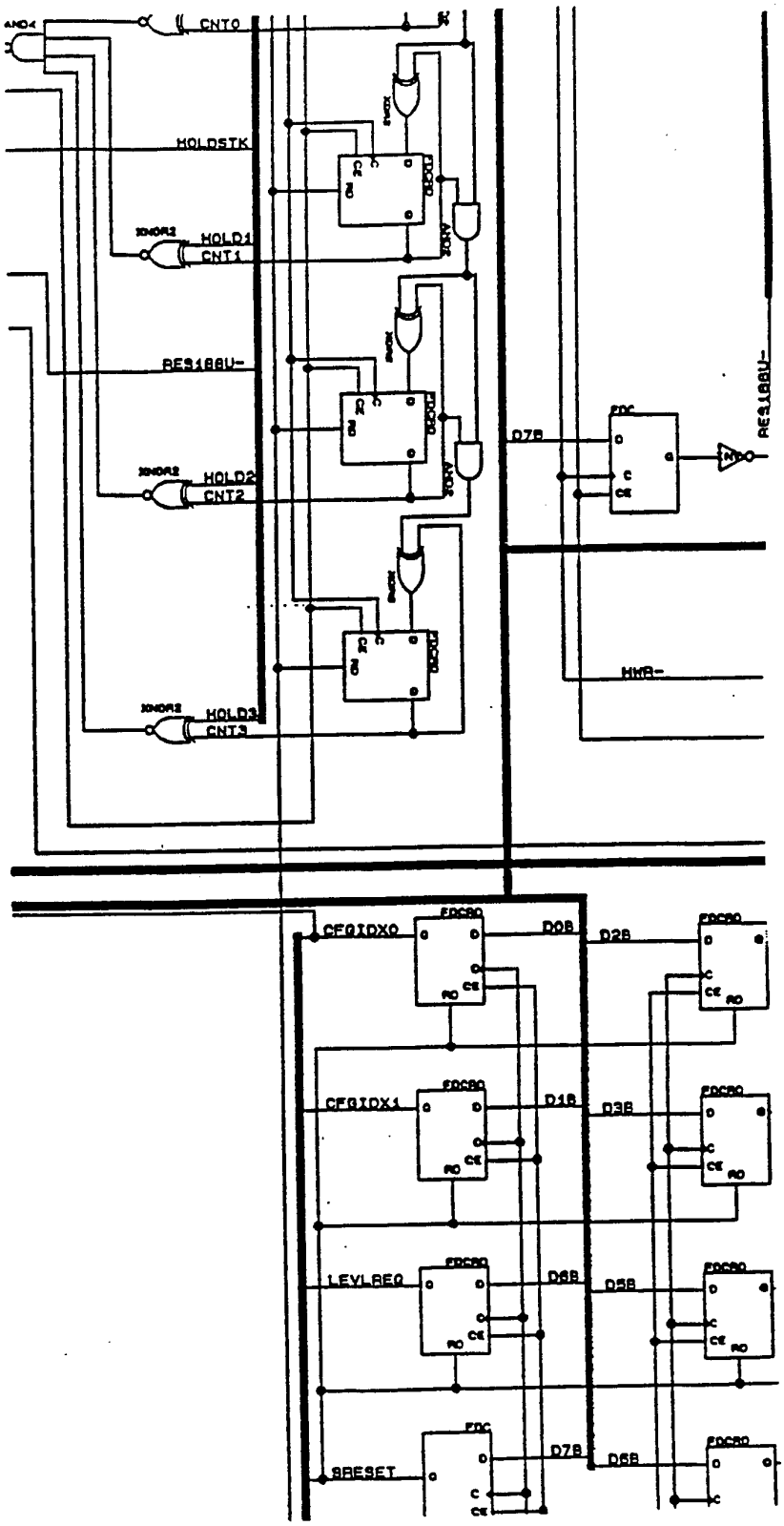

CNT0
HOLDSTK
HOLD1
CNT1
RES188U-
HOLD2
CNT2
HOLD3
CNT3
D7B
RES188U-
HWB-
CFGIDX0
D0B
D2B
CFGIDX1
D1B
D3B
LEVLREQ
D6B
D5B
SRESET
D7B
D6B

Fig. 11F

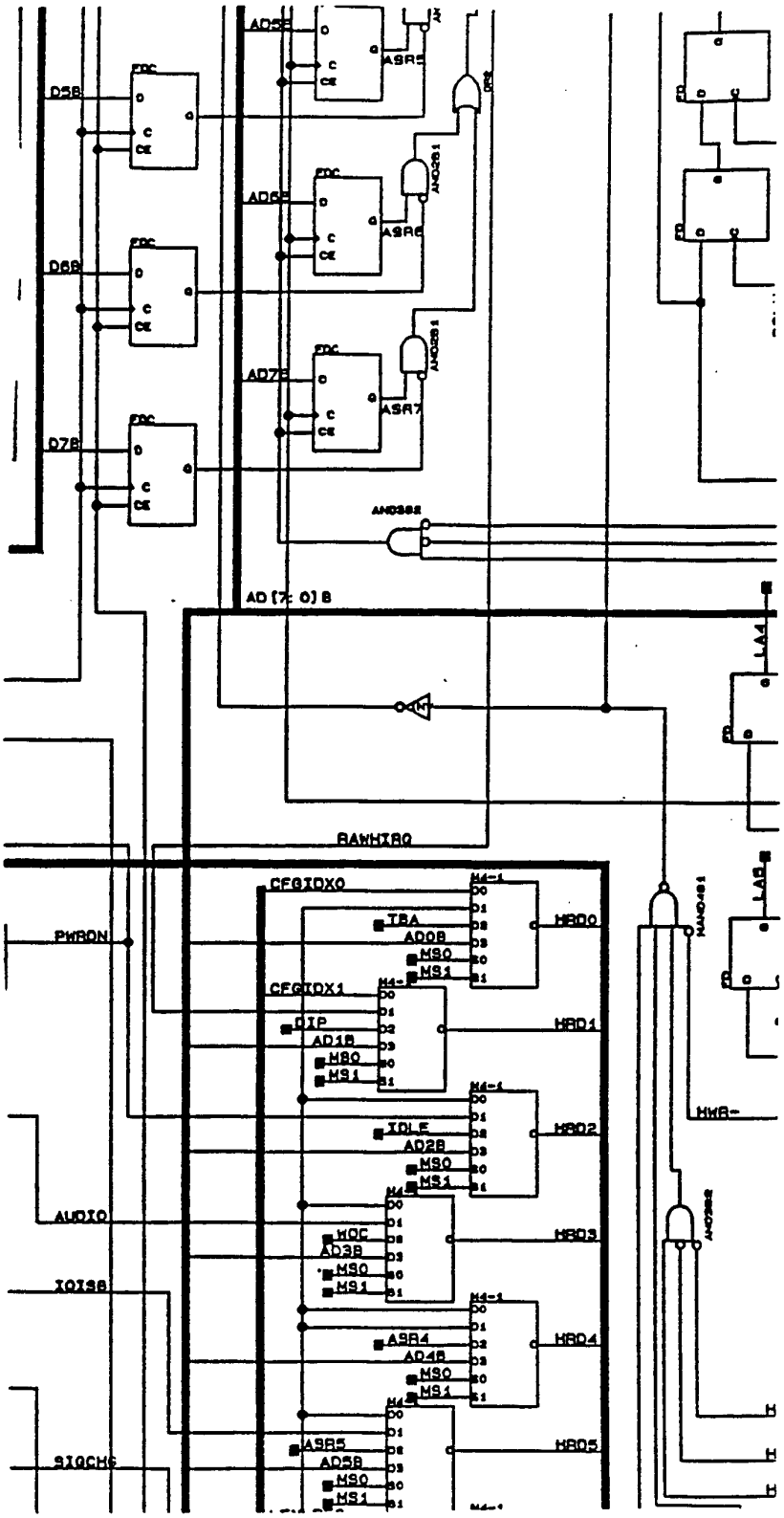

AD5B
ASR5
D5B
AD6B
ASR6
D6B
AD7B
ASR7
D7B
AD[7:0]B
RAWHIRQ
CFGIDX0
TRA
AD0B
MS0
MS1
HRD0
CFGIDX1
DIP
AD1B
HRD1
IDLE
AD2B
HRD2
AUDIO
HDC
AD3B
HRD3
IOIRB
ASR4
AD4B
HRD4
ASR5
AD5B
HRD5
PWRON
HWR-

Fig. 11G

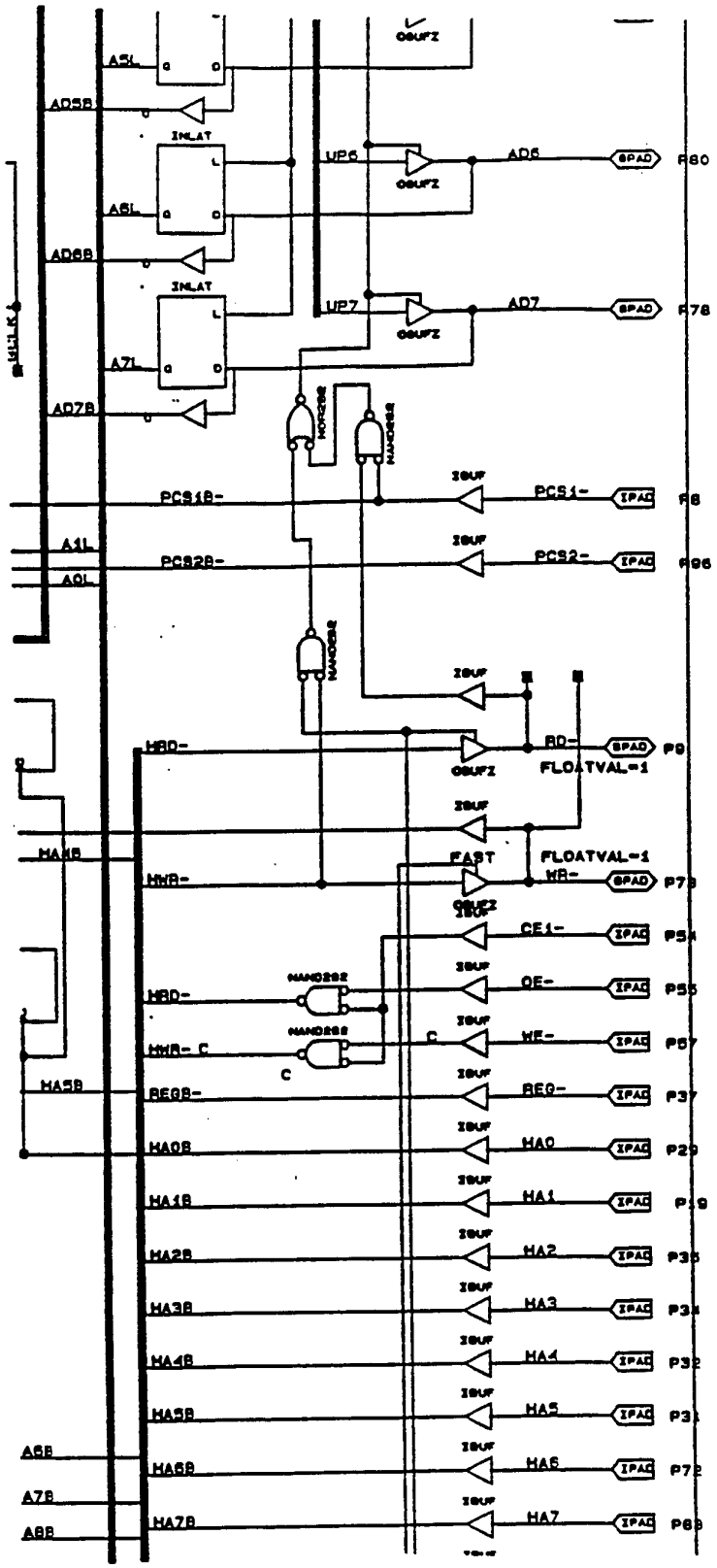

A5L
AD5B
INLAT
UP6
AD6
A6L
AD6B
UP7
AD7
A7L
AD7B
PCS1B-
PCS1-
A1L
PCS2B-
PCS2-
A0L
HRD-
RD-
FLOATVAL=1
HA1B
FAST
HWR-
WR-
CE1-
OE-
WE-
REQB-
REQ-
HA0B
HA0
HA1B
HA1
HA2B
HA2
HA3B
HA3
HA4B
HA4
HA5B
HA5
A6B
HA6B
HA6
A7B
HA7B
HA7
A8B

Fig. 11H

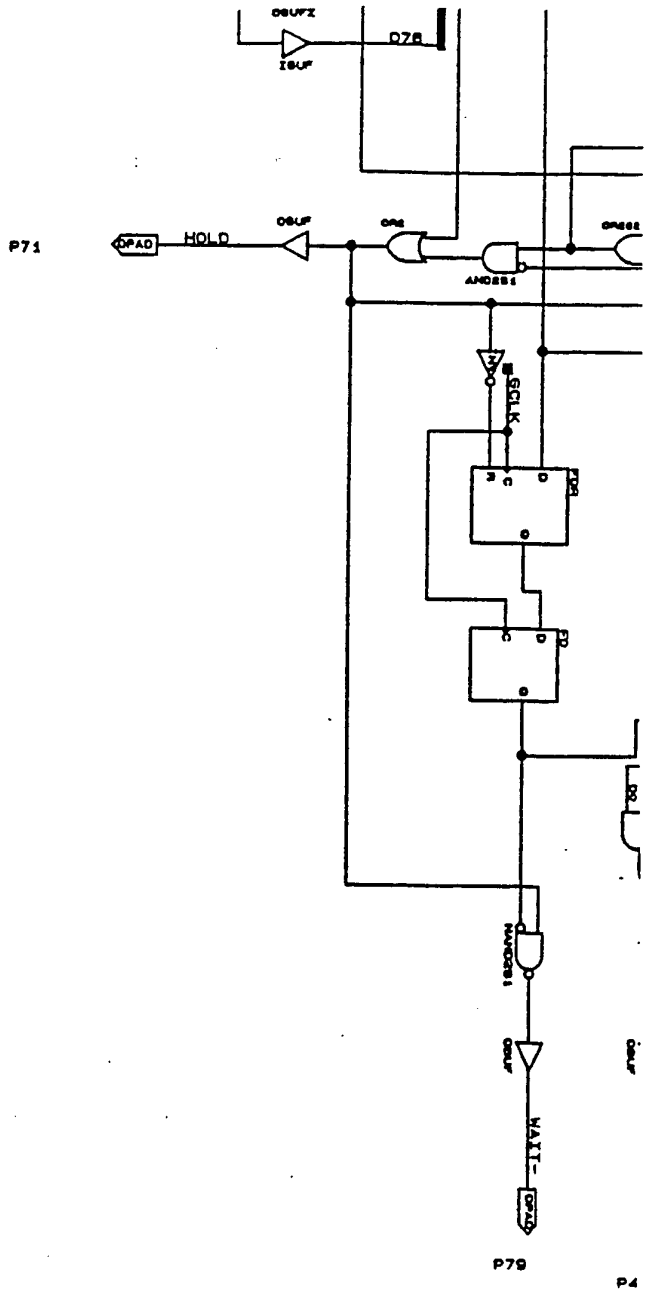



Fig. 11I

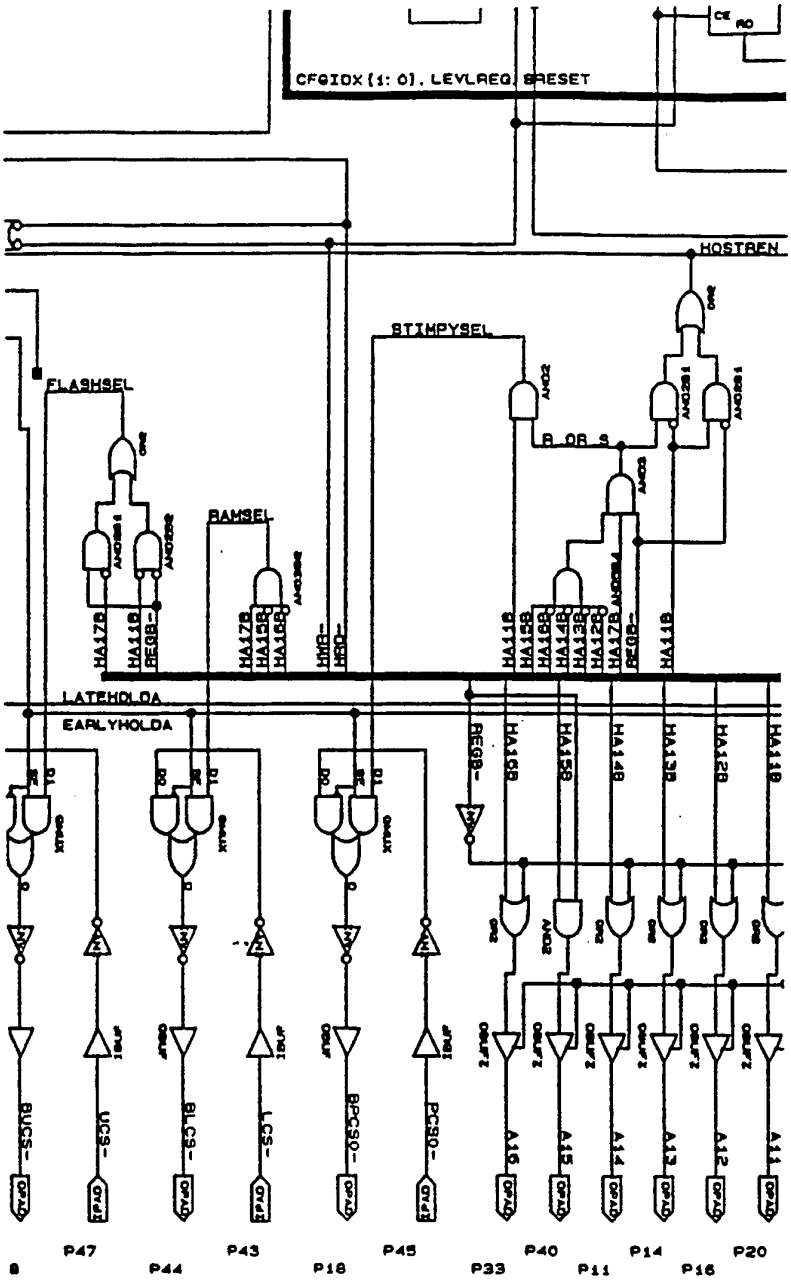

CFGIOX(1:0), LEVLAEQ, SRESET
HOSTAEN
STIMPYSEL
FLASHSEL
RAMSEL
R_OR_S
LATEHOLDA
EARLYHOLDA
BUCS-
UCS-
BLCS-
LCS-
BPCS80-
PCS80-
A16
A15
A14
A13
A12
A11
P47
P44
P43
P18
P45
P33
P40
P11
P14
P16
P20

Fig. 11 J

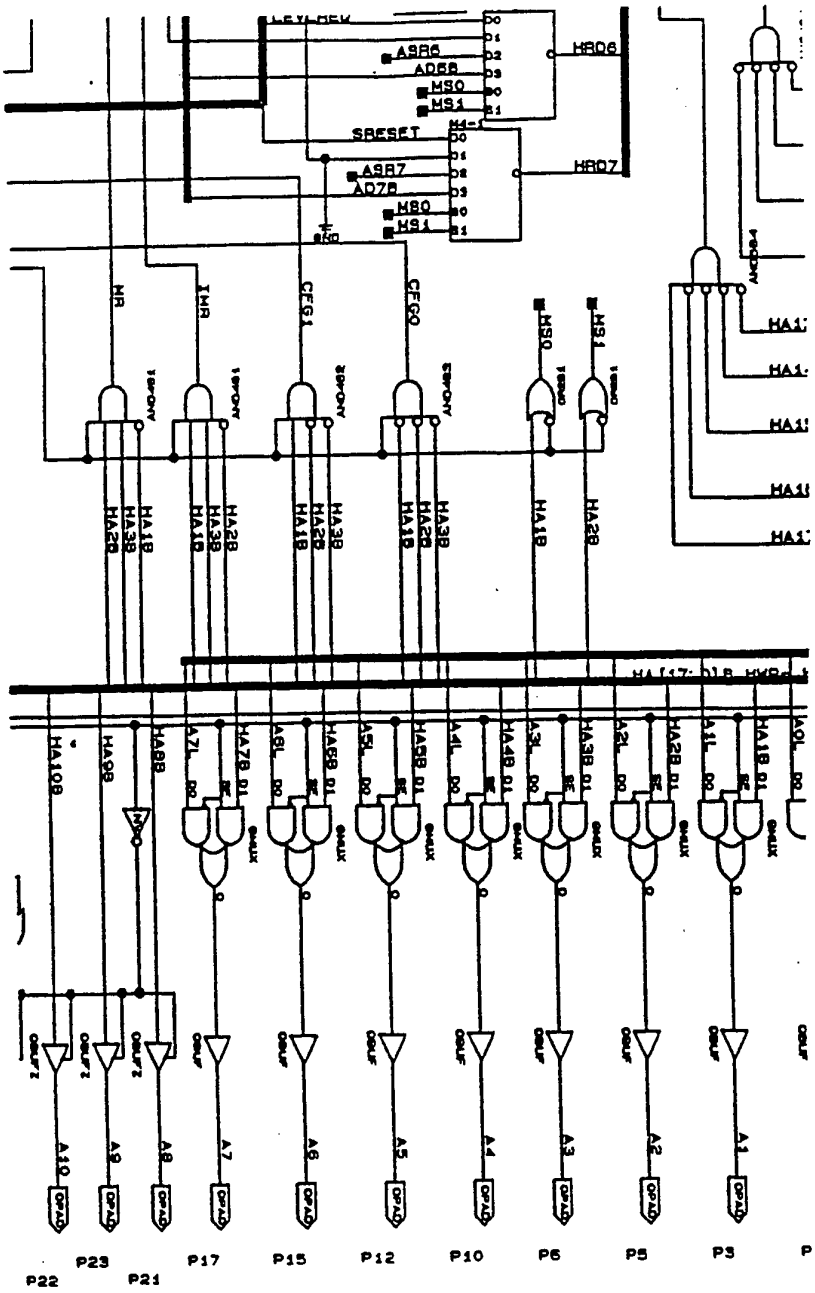

A9R6
AD6R
MS0
MS1
SRESET
A9R7
AD7R
MS0
MS1
HR06
HR07
HR
IHR
CFG1
CFG0
MS0
MS1
HA10
HA9R
HA8R
HA7L
HA7R
HA6L
HA6R
HA5L
HA5R
HA4L
HA4R
HA3L
HA3R
HA2L
HA2R
HA1L
HA1R
HA0L
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
P22
P23
P21
P17
P15
P12
P10
P6
P5
P3

Fig. 11 K

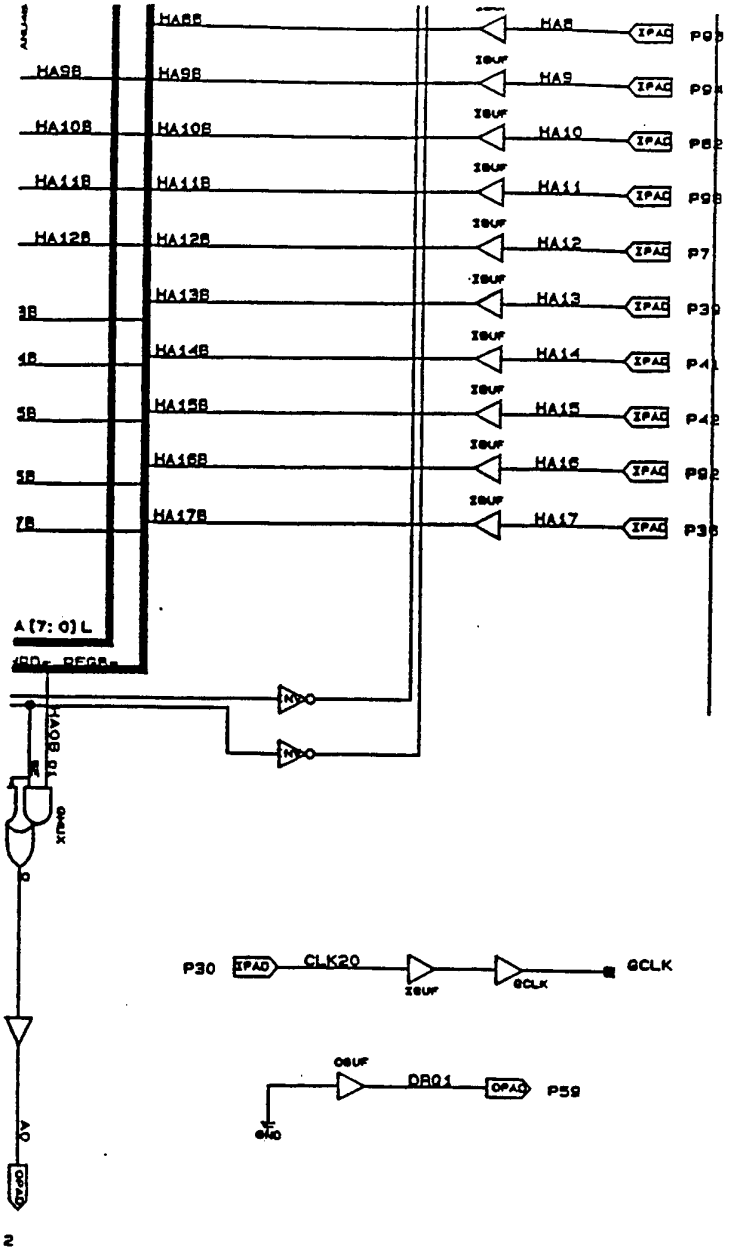



Fig. 11L

Key to Fig. 11

| | | | |
|---|---|---|---|
| Fig. 11A | Fig. 11B | Fig. 11C | Fig. 11D |
| Fig. 11E | Fig. 11F | Fig. 11G | Fig. 11H |
| Fig. 11I | Fig. 11J | Fig. 11K | Fig. 11L |

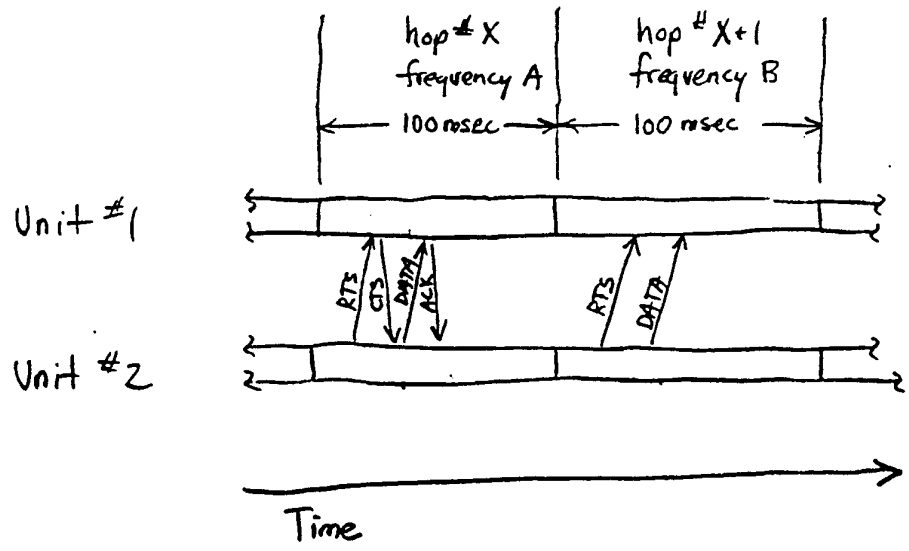

hop # X
frequency A
hop # X+1
frequency B
100 msec
100 msec
Unit #1
Unit #2
RTS
CTS
DATA
ACK
RTS
DATA
Time

Fig. 12

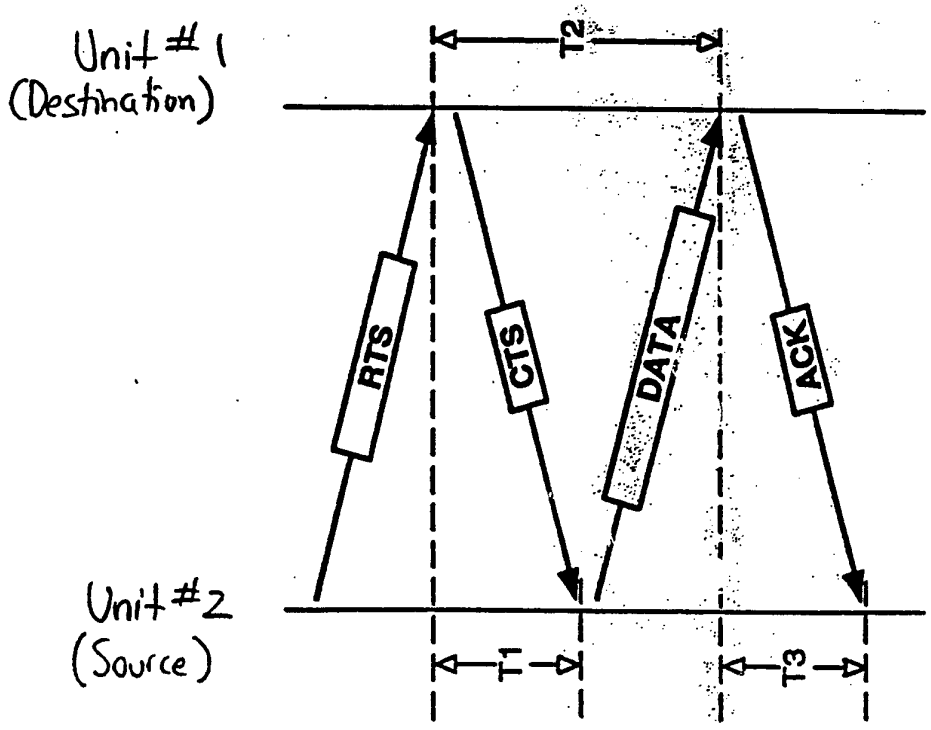

Unit #1
(Destination)
T2
RTS
CTS
DATA
ACK
Unit #2
(Source)
T1
T3

Fig. 13

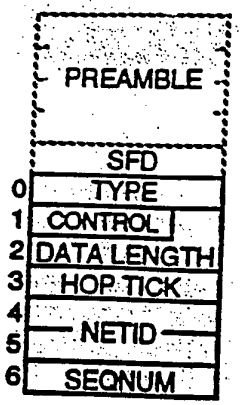



Fig. 14

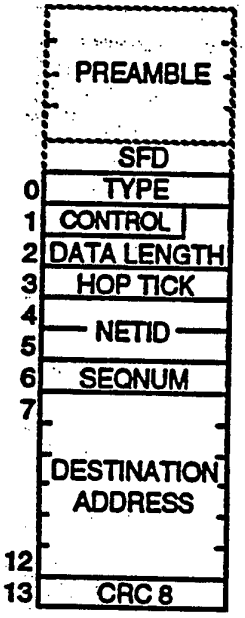



Fig. 15

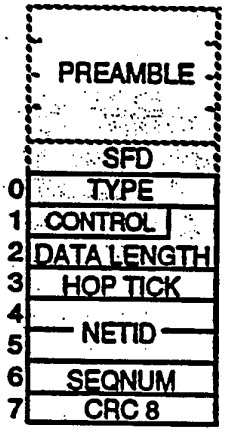



Fig. 16

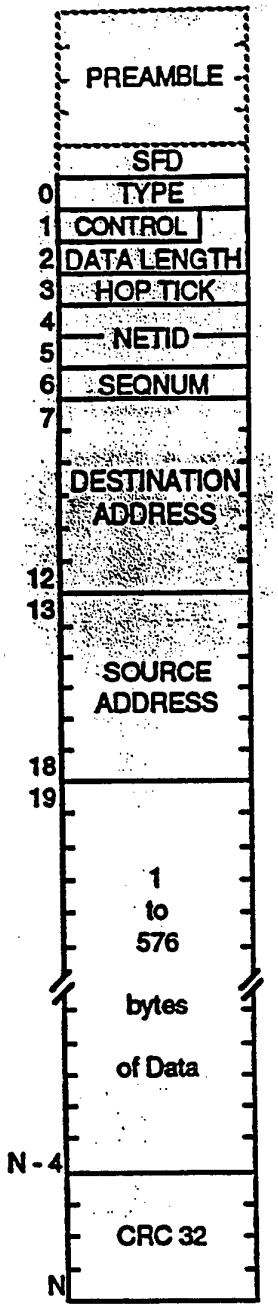



Fig. 17

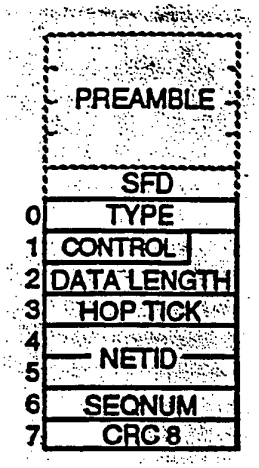



Fig. 18

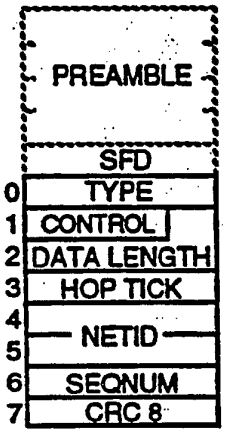



Fig. 19

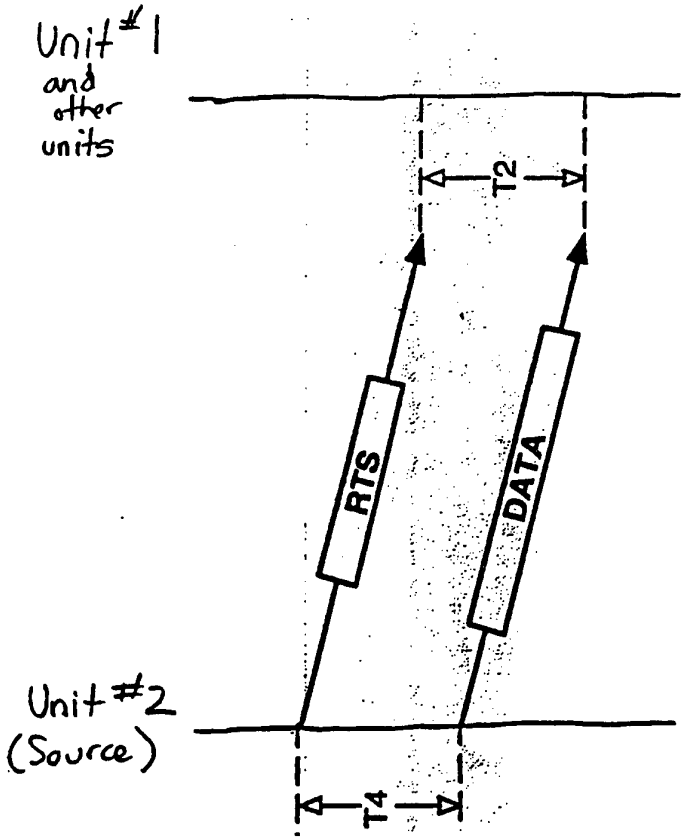



Fig. 20

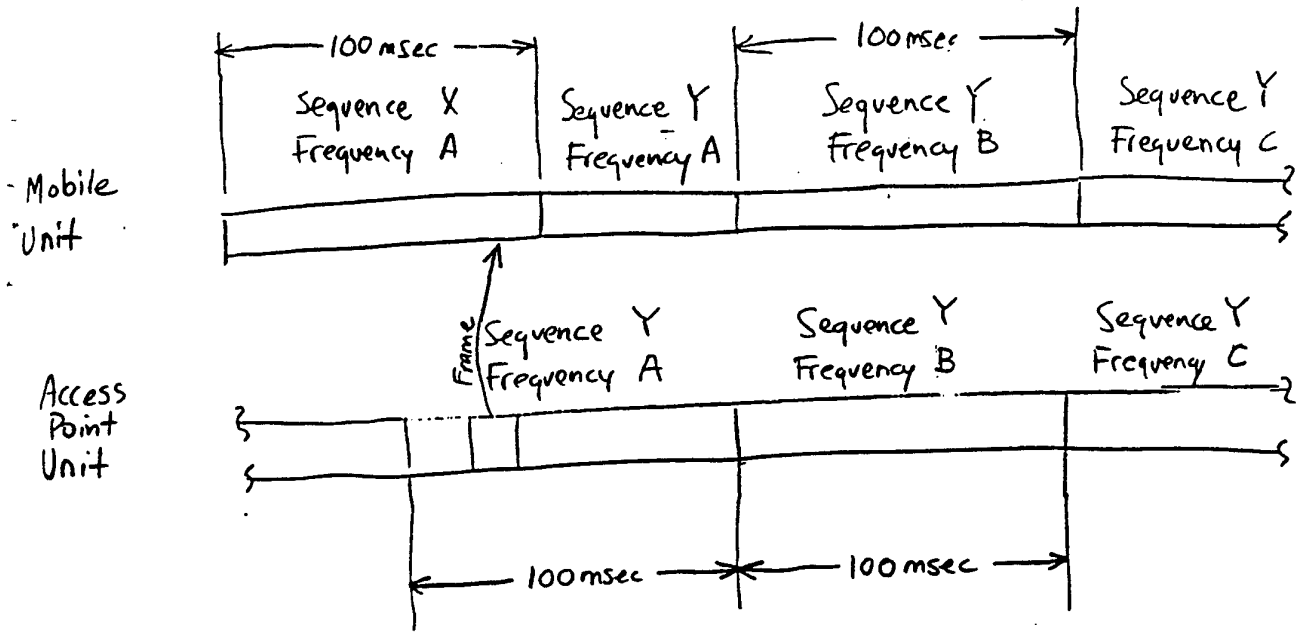

100 msec
100 msec
Sequence X
Frequency A
Sequence Y
Frequency A
Sequence Y
Frequency B
Sequence Y
Frequency C
Mobile Unit
Frame
Sequence Y
Frequency A
Sequence Y
Frequency B
Sequence Y
Frequency C
Access Point Unit
100 msec
100 msec

Fig. 21

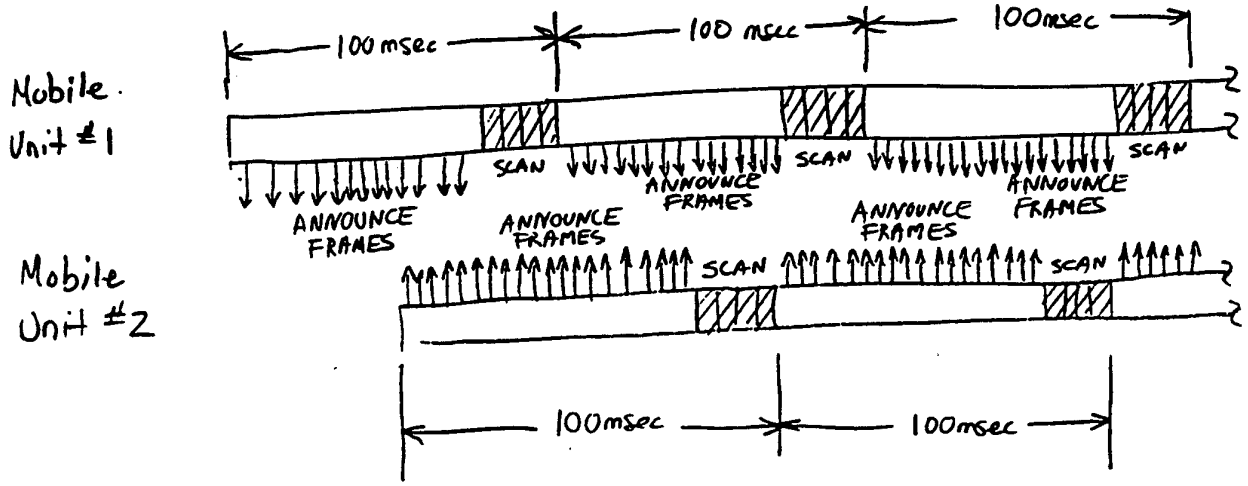

100 msec
100 msec
100 msec
Mobile Unit #1
SCAN
SCAN
SCAN
ANNOUNCE FRAMES
ANNOUNCE FRAMES
ANNOUNCE FRAMES
ANNOUNCE FRAMES
Mobile Unit #2
SCAN
SCAN
100 msec
100 msec

Fig. 22

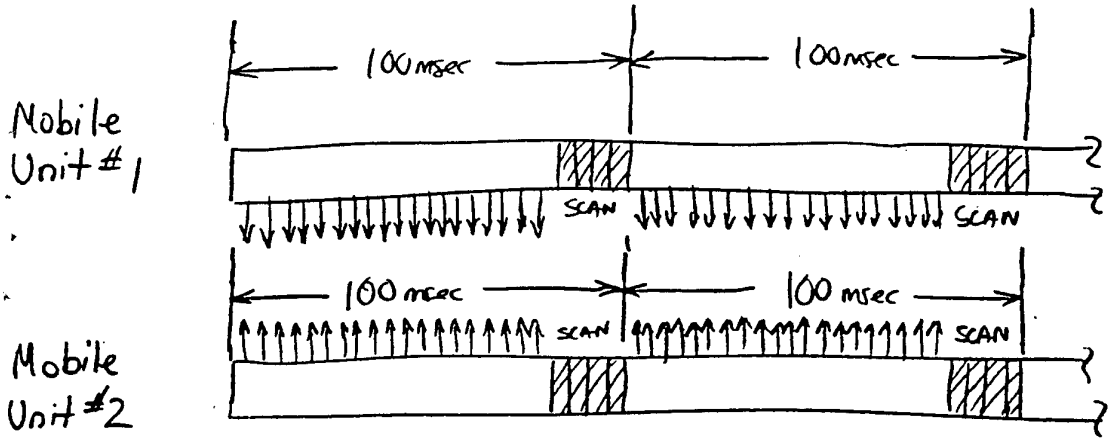

Mobile Unit #1
100 msec
100 msec
SCAN
SCAN
Mobile Unit #2
100 msec
100 msec
SCAN
SCAN

Fig. 23

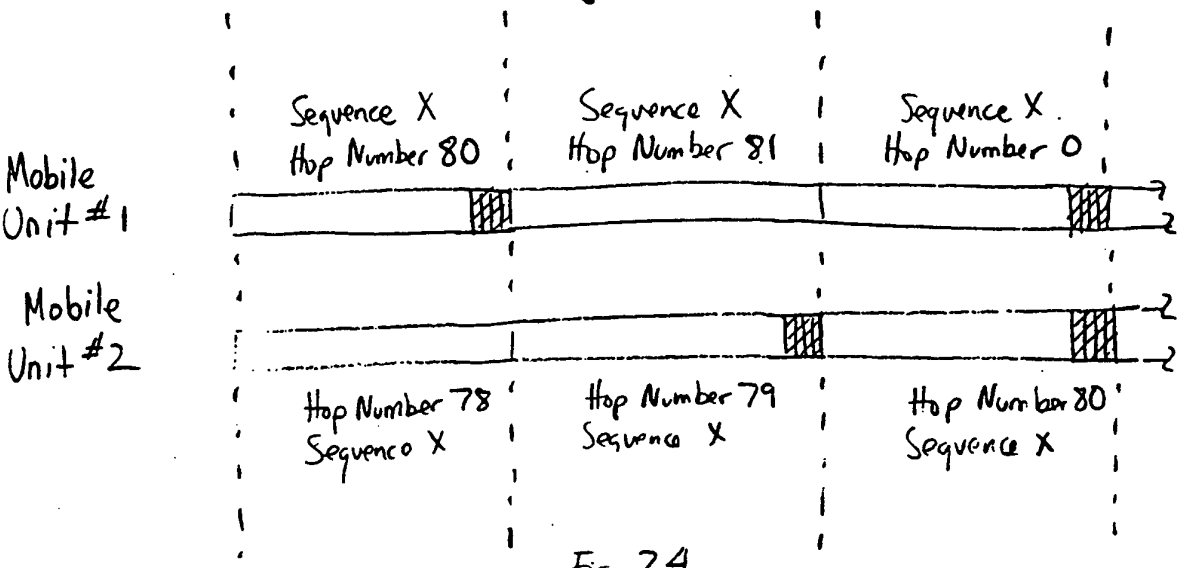

Mobile Unit #1
Mobile Unit #2
Sequence X Hop Number 80
Sequence X Hop Number 81
Sequence X Hop Number 0
Hop Number 78 Sequence X
Hop Number 79 Sequence X
Hop Number 80 Sequence X

Fig. 24

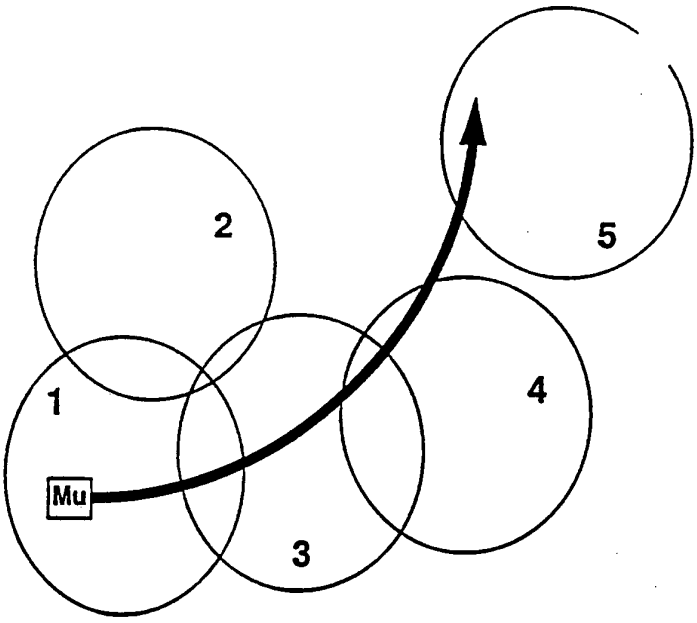

1
2
3
4
5
Mu

Fig. 25

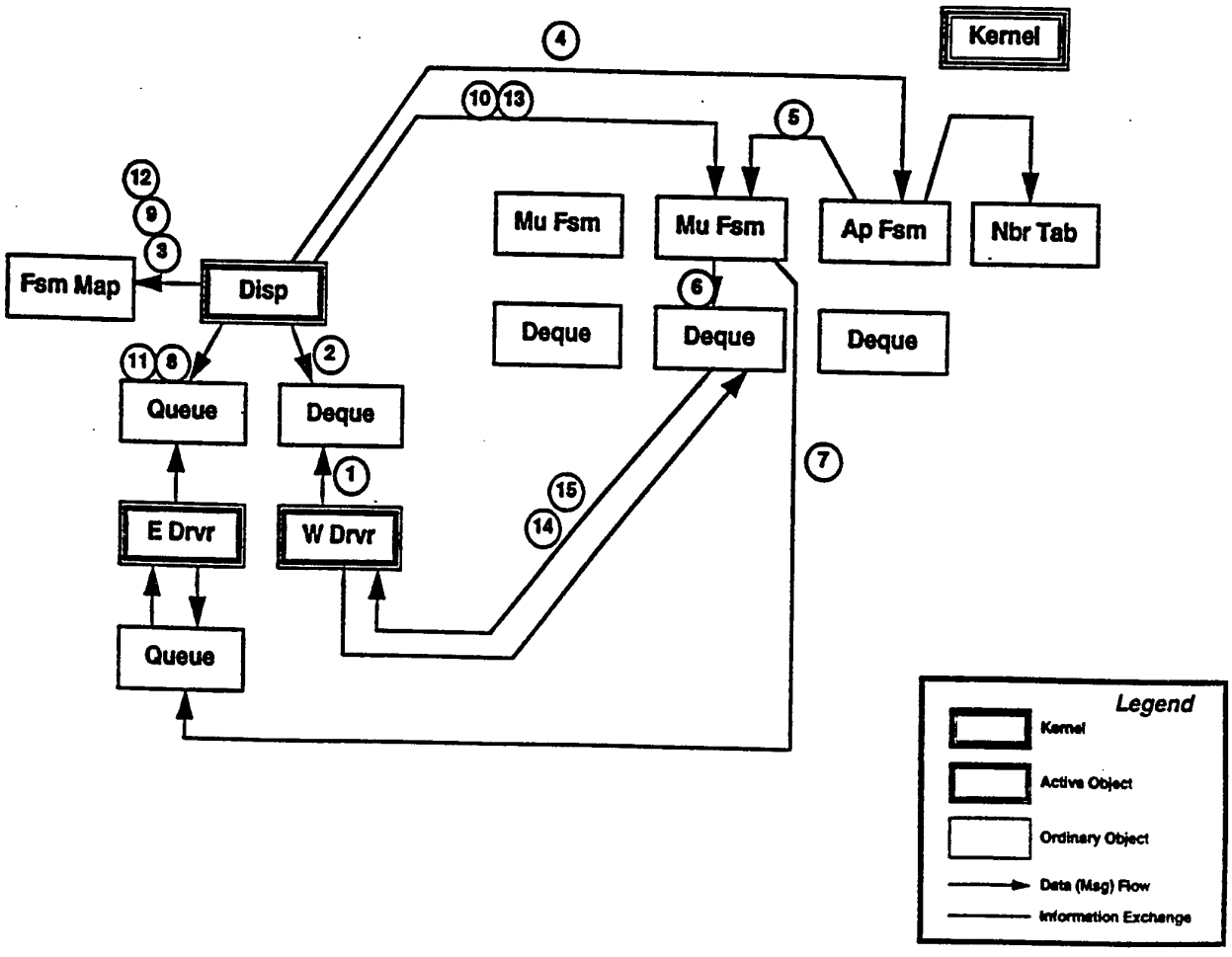

Kernel
Mu Fsm
Mu Fsm
Ap Fsm
Nbr Tab
Fsm Map
Disp
Deque
Deque
Deque
Queue
Deque
E Drvr
W Drvr
Queue
Legend
Kernel
Active Object
Ordinary Object
Data (Msg) Flow
Information Exchange

Fig. 26

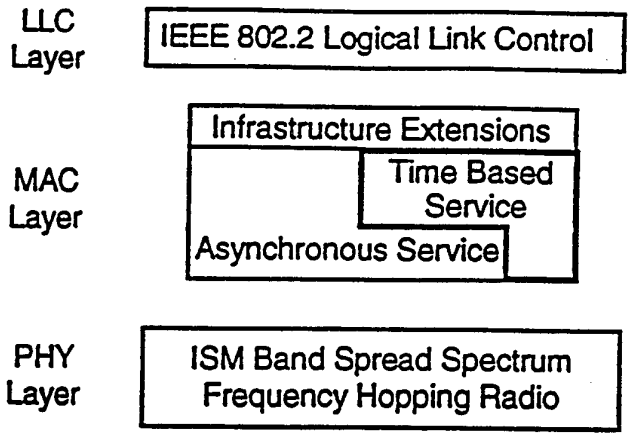

LLC Layer
IEEE 802.2 Logical Link Control
MAC Layer
Infrastructure Extensions
Time Based Service
Asynchronous Service
PHY Layer
ISM Band Spread Spectrum Frequency Hopping Radio

Fig. 27

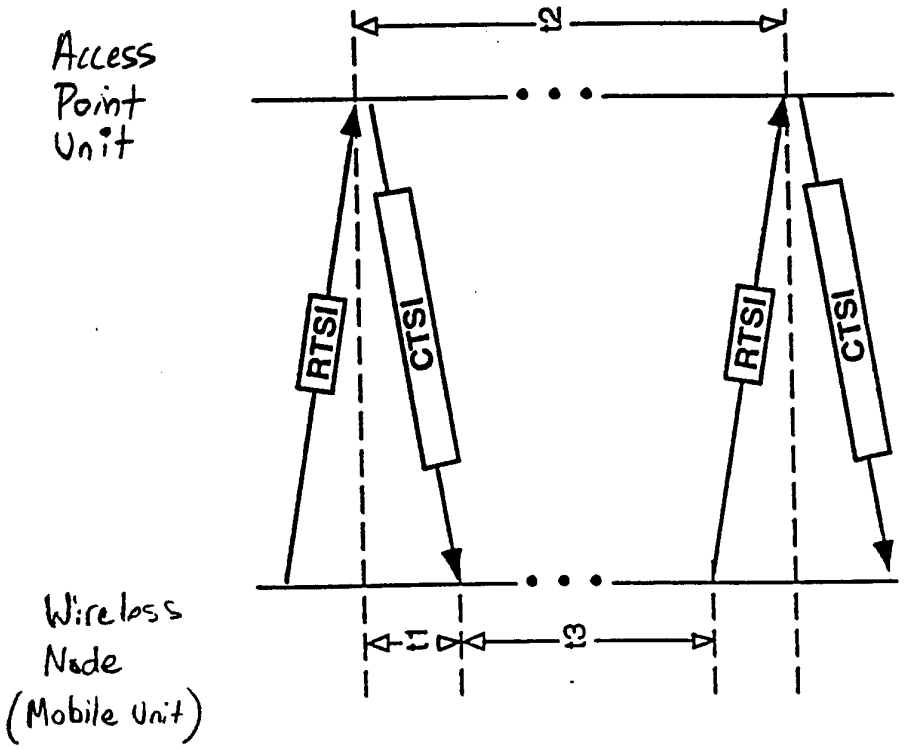

Access Point Unit
RTSI
CTSI
RTSI
CTSI
Wireless Node (Mobile Unit)

Fig. 28

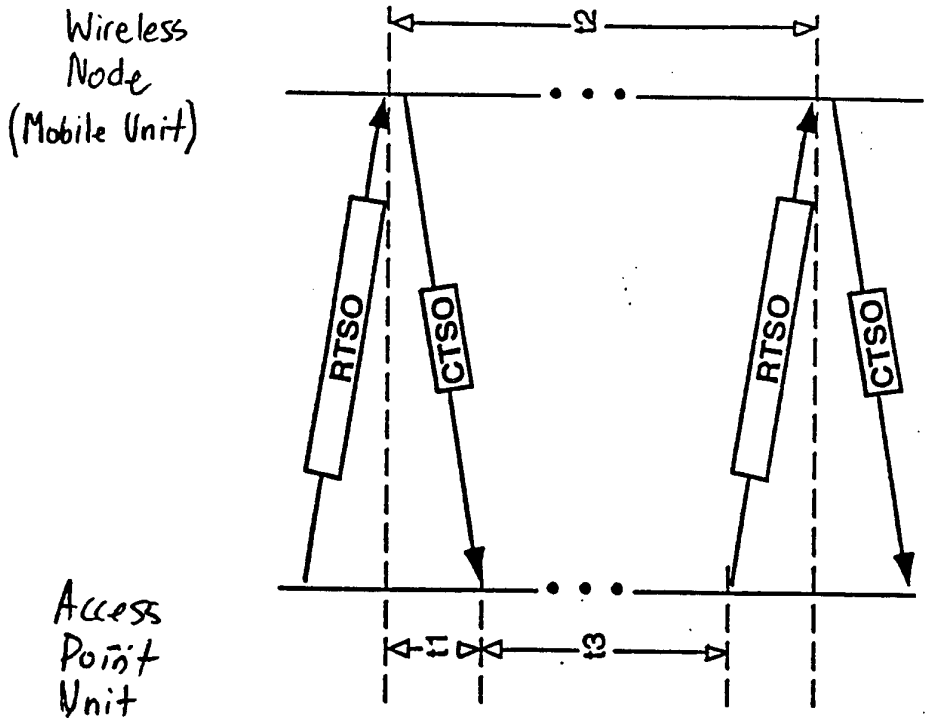

Wireless
Node
(Mobile Unit)
RTSO
CTSO
RTSO
CTSO
Access
Point
Unit

Fig. 29

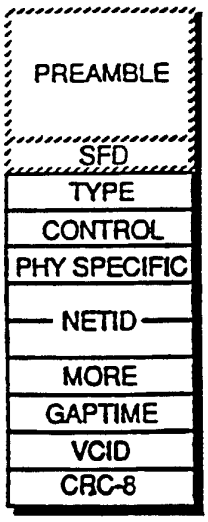



Fig. 30

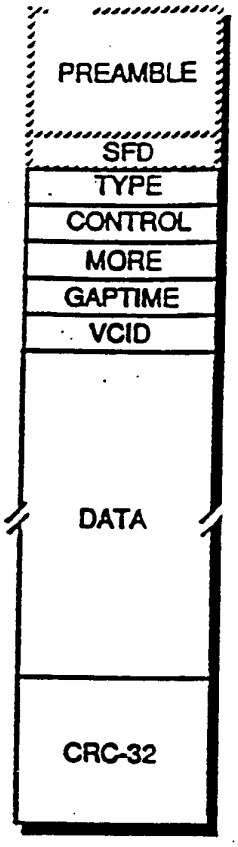



Fig. 31

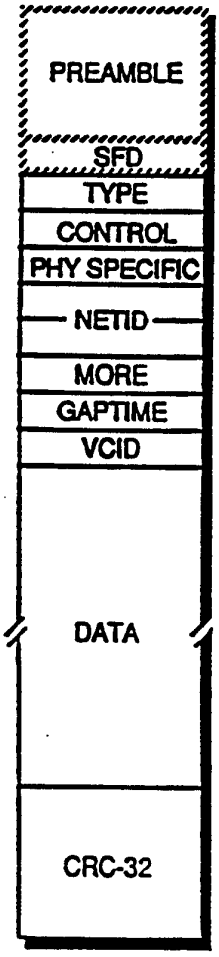



Fig. 32

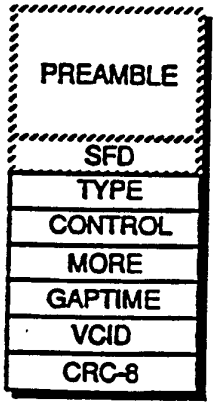



Fig. 33

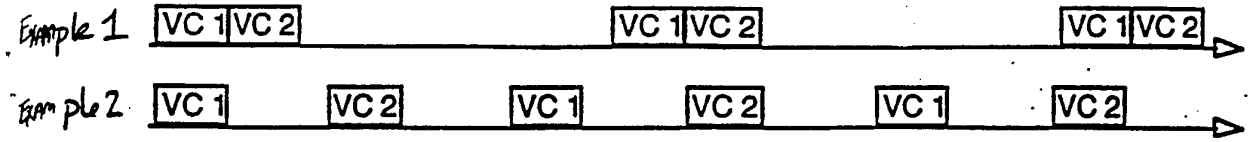



Fig. 34

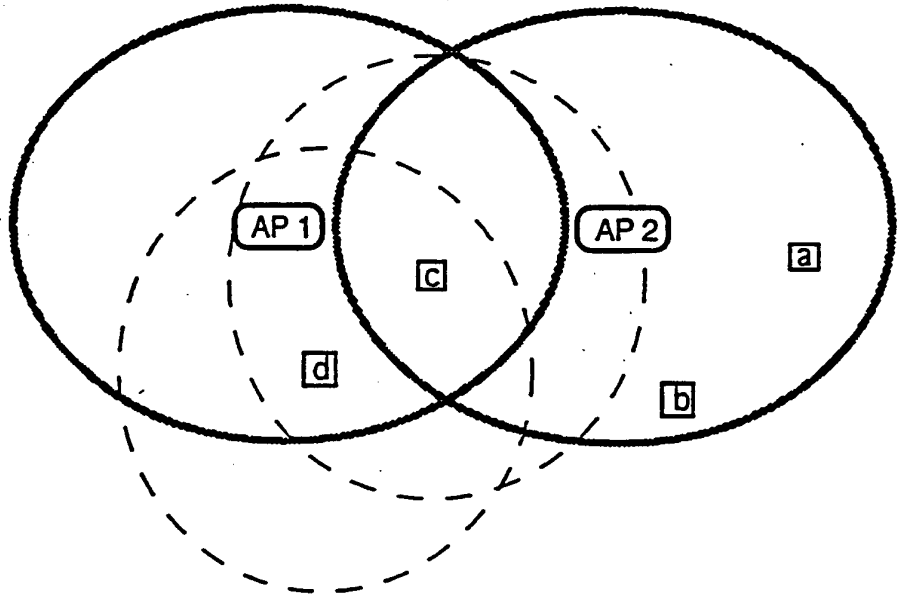
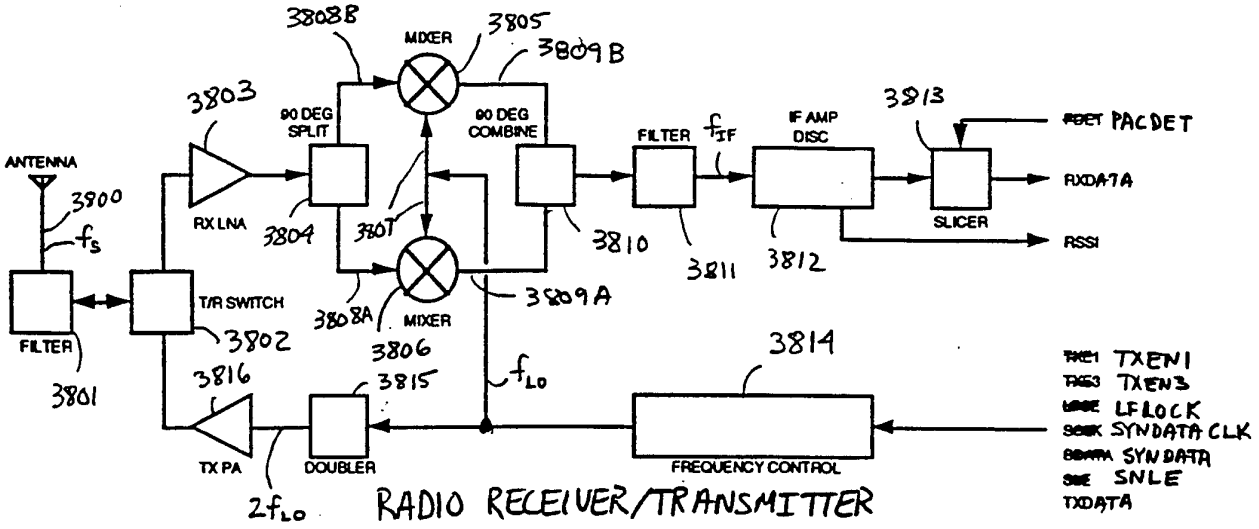
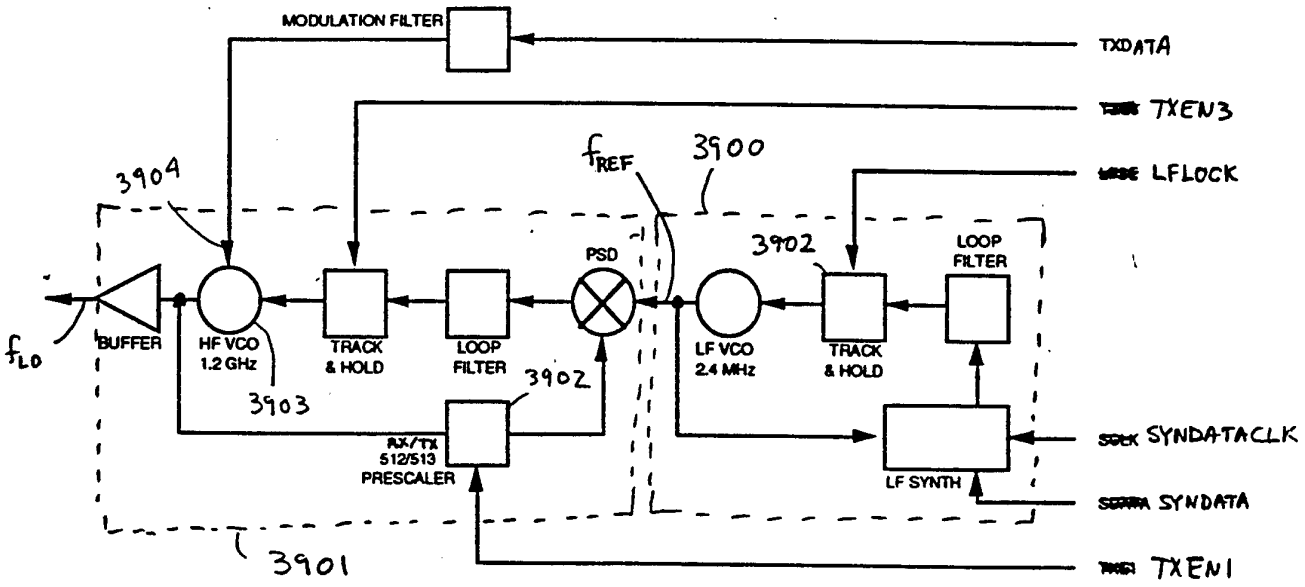



Fig. 37
Overlapping BBS with Time Bounded Traffic




RADIO RECEIVER/TRANSMITTER

Fig. 38




FREQUENCY CONTROL BLOCK

Fig. 39

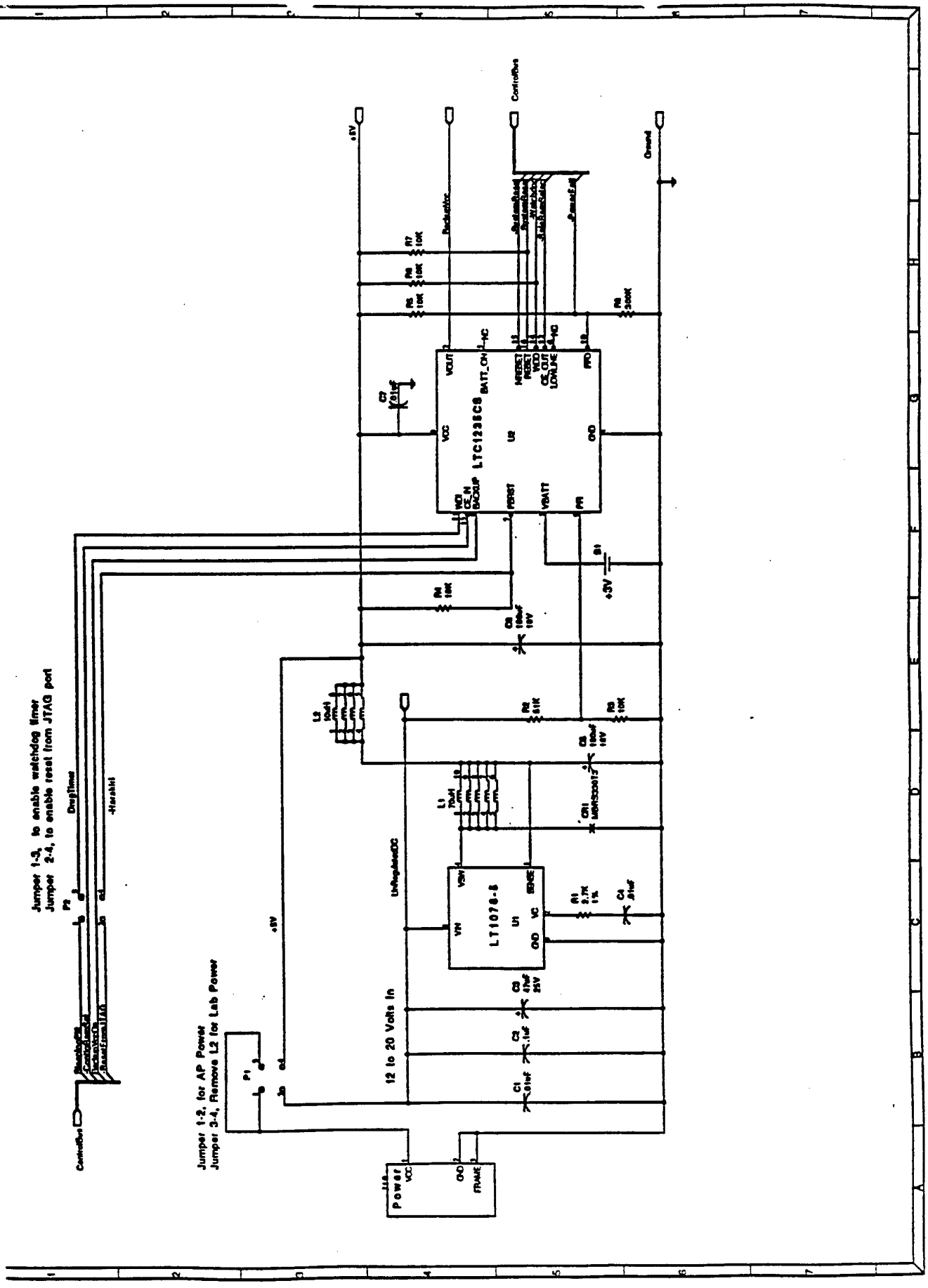

Jumper 1-3, to enable watchdog timer
Jumper 2-4, to enable reset from JTAG port
Jumper 1-2, for AP Power
Jumper 3-4, Remove L2 for Lab Power
12 to 20 Volts In
LTC1235CS
LT1076-5
Power
Controlbus
Ground
+5V
+3V

Fig. 40A

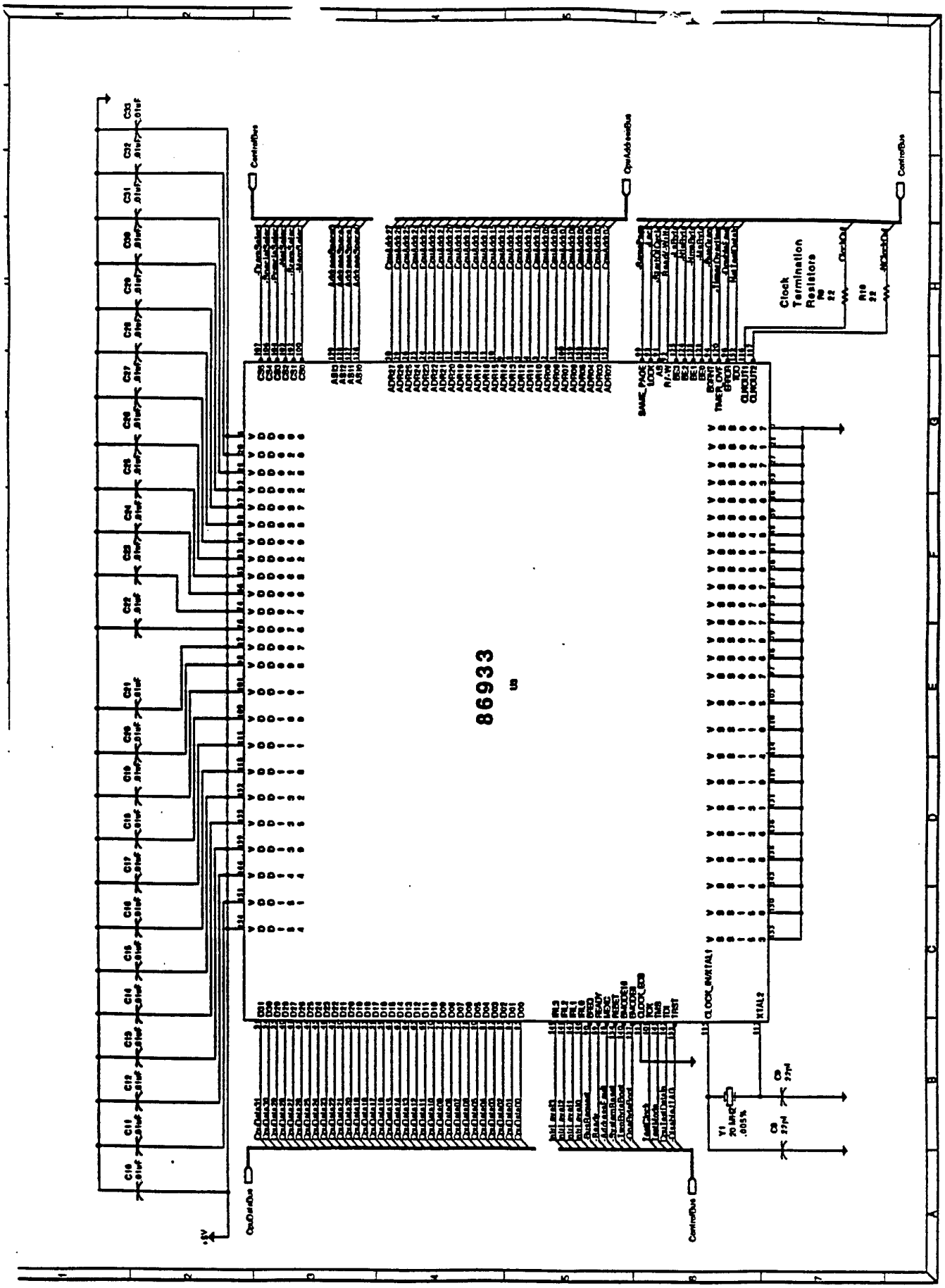



Fig 40B

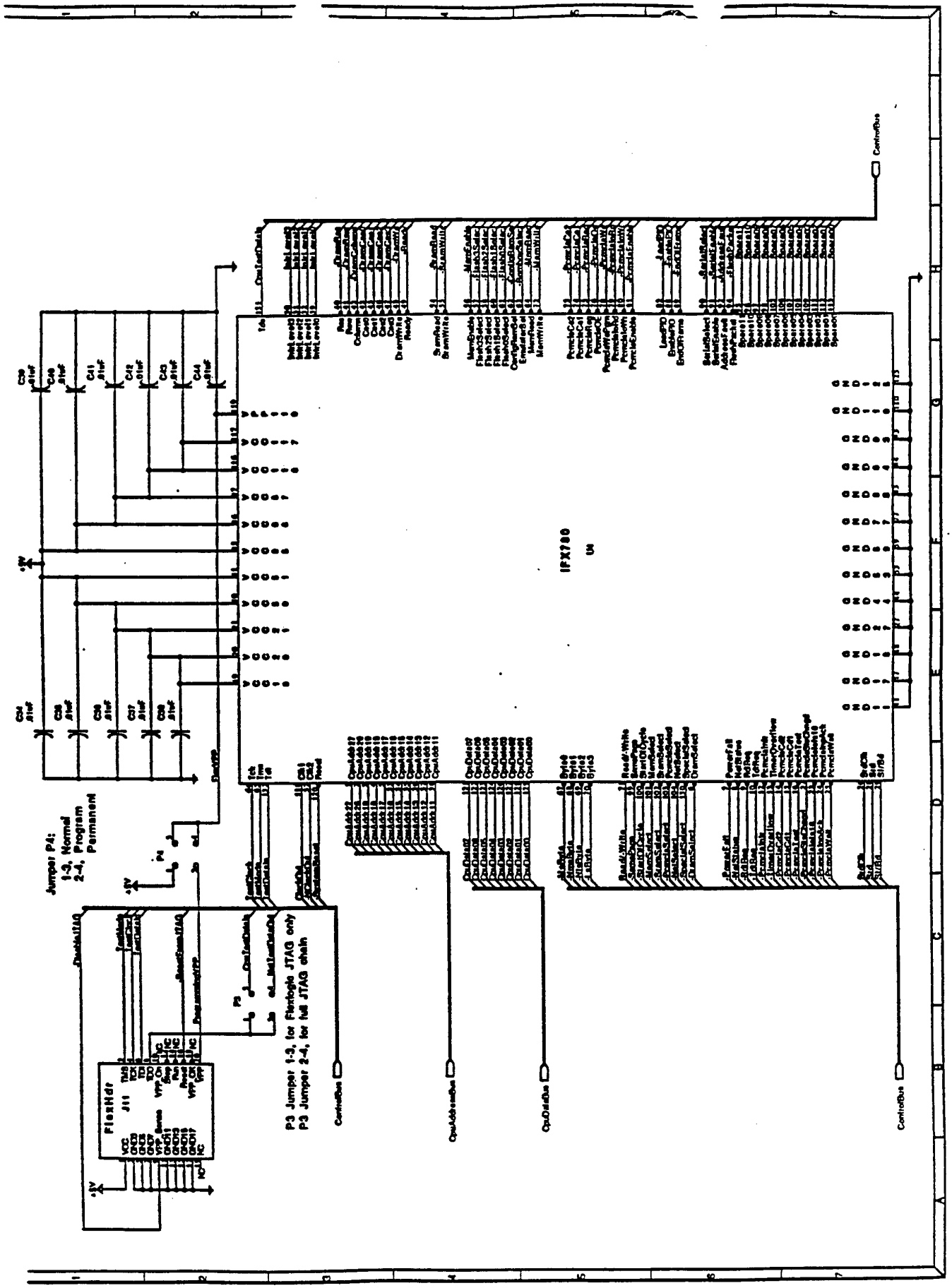

IFX780
U4
Jumper P4:
1-3, Normal
2-4, Program
Permanent
P3 Jumper 1-3, for Flextogic JTAG only
P3 Jumper 2-4, for full JTAG chain
FlexHdr
J11
ControlBus
CpuAddressBus
CpuDataBus

Fig. 40C

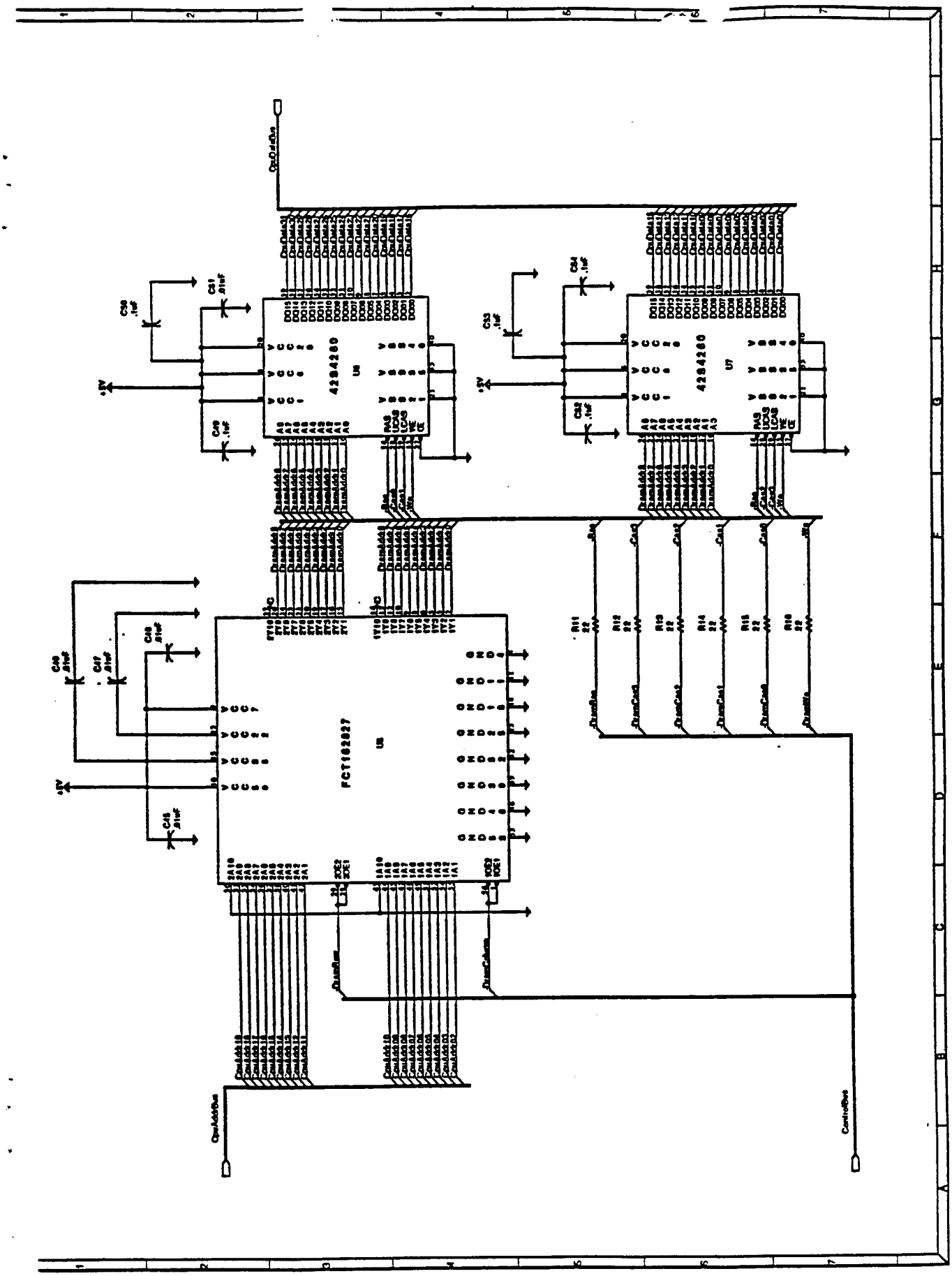


Fig. 40D

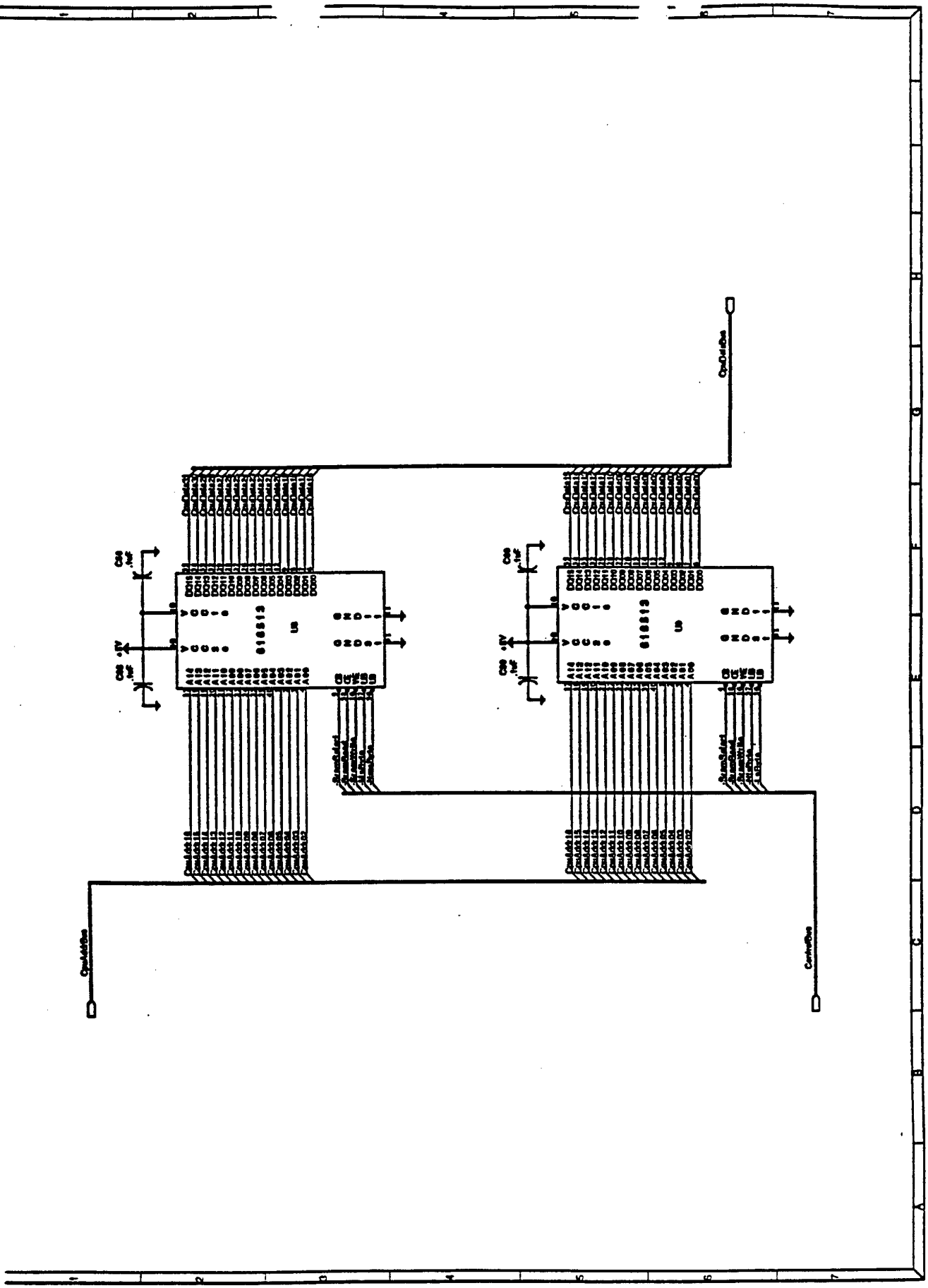


Fig. 40E

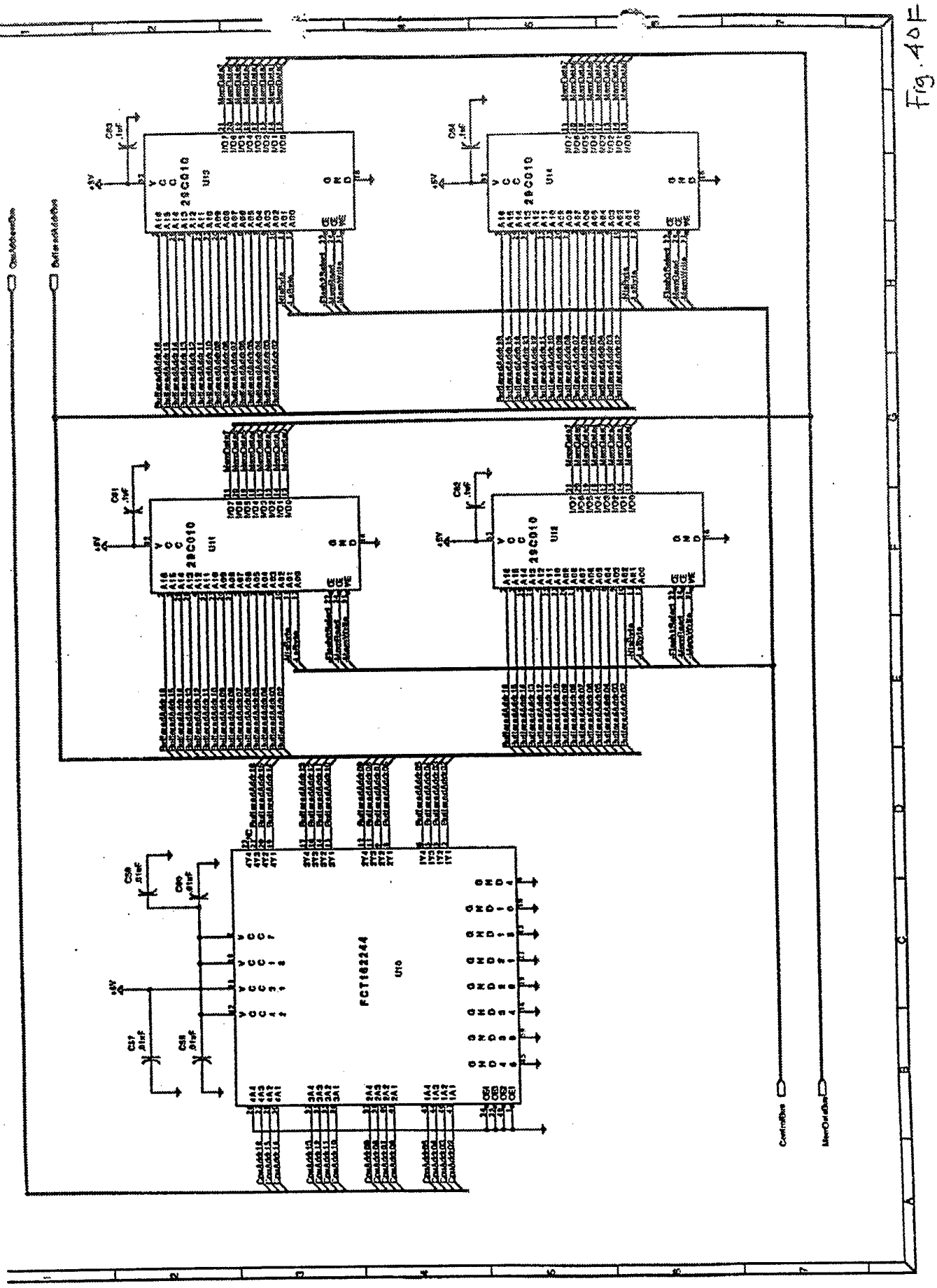



Fig. 40E

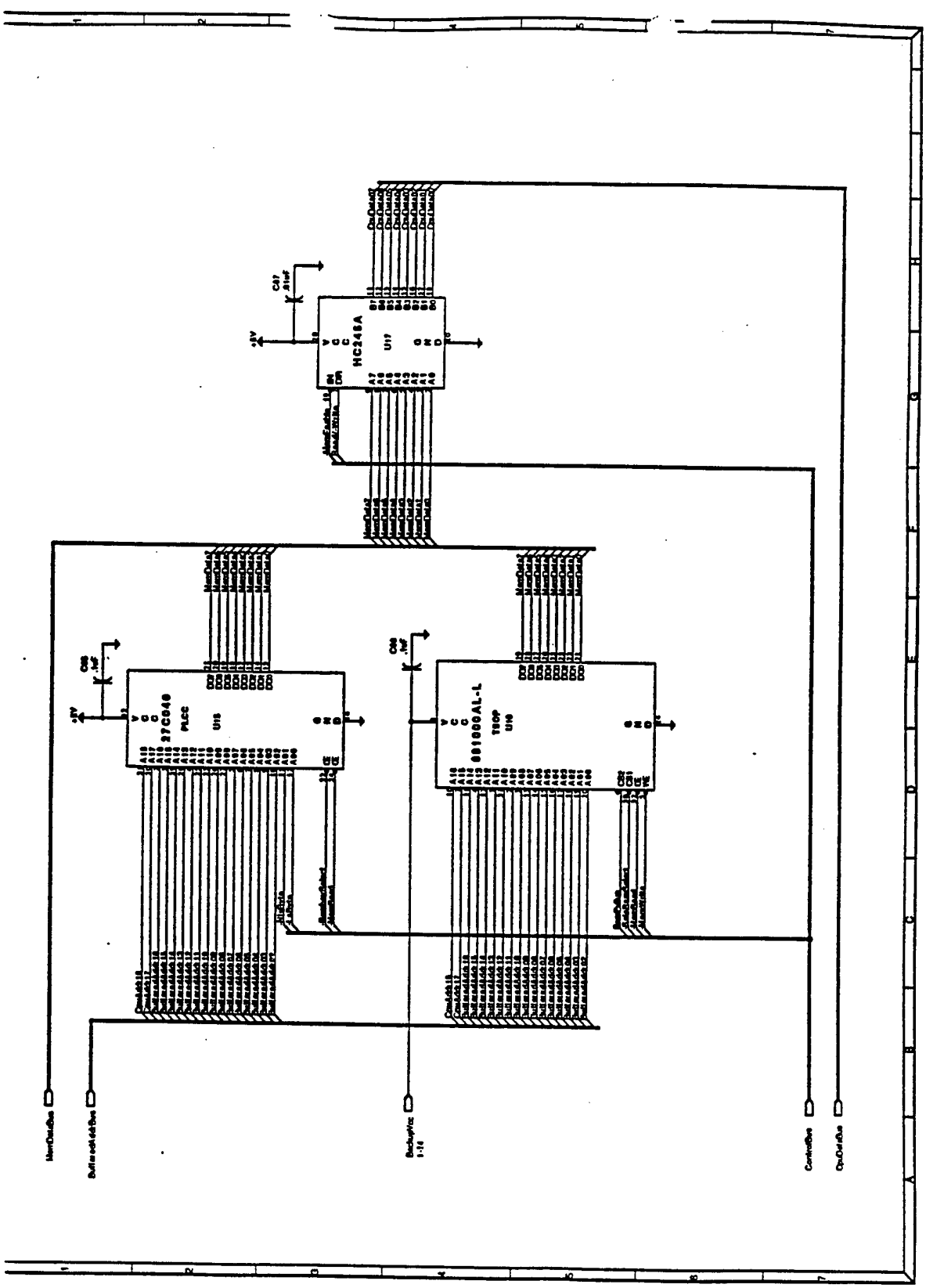



Fig. 40G

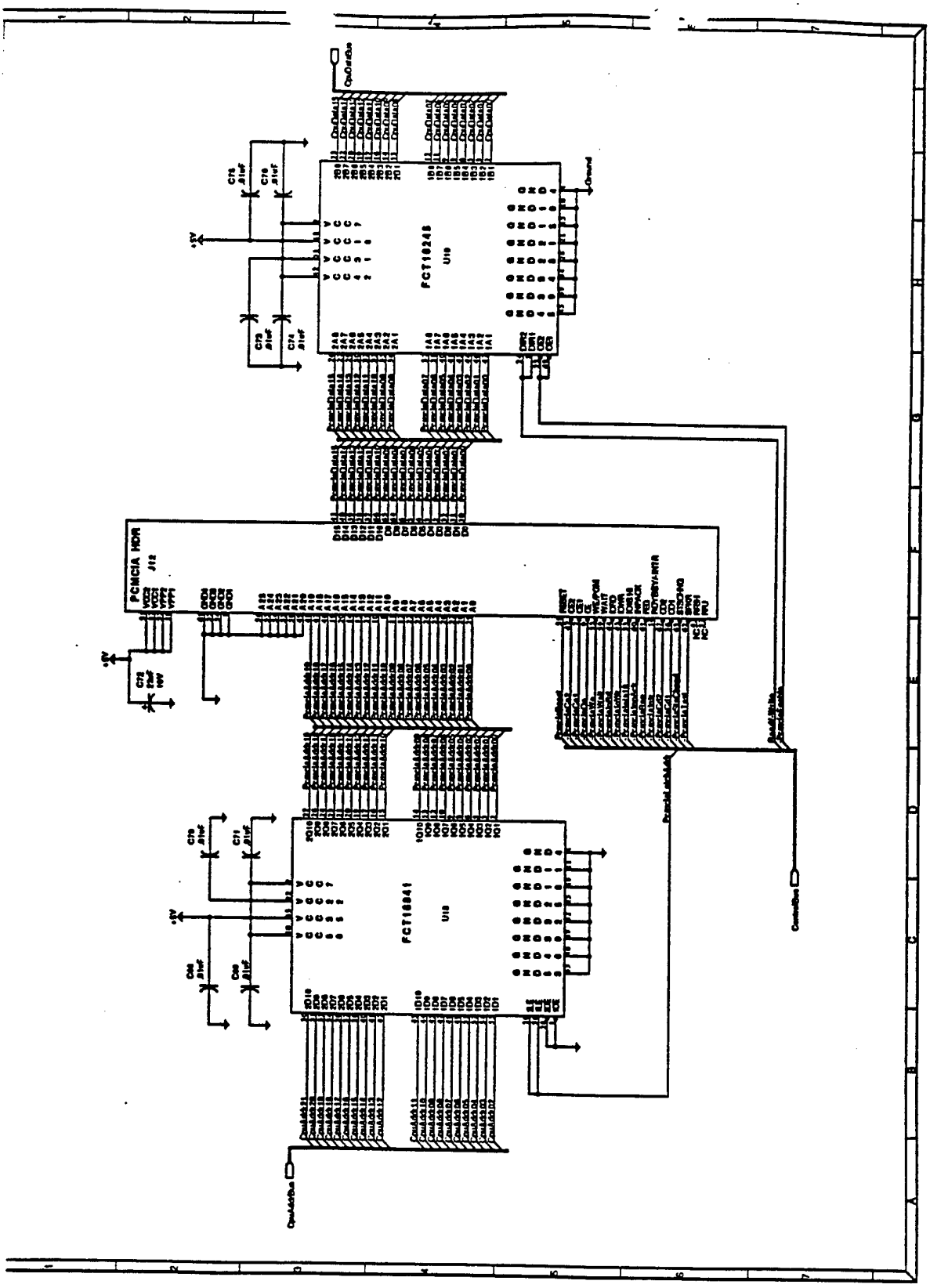



Fig. 40H

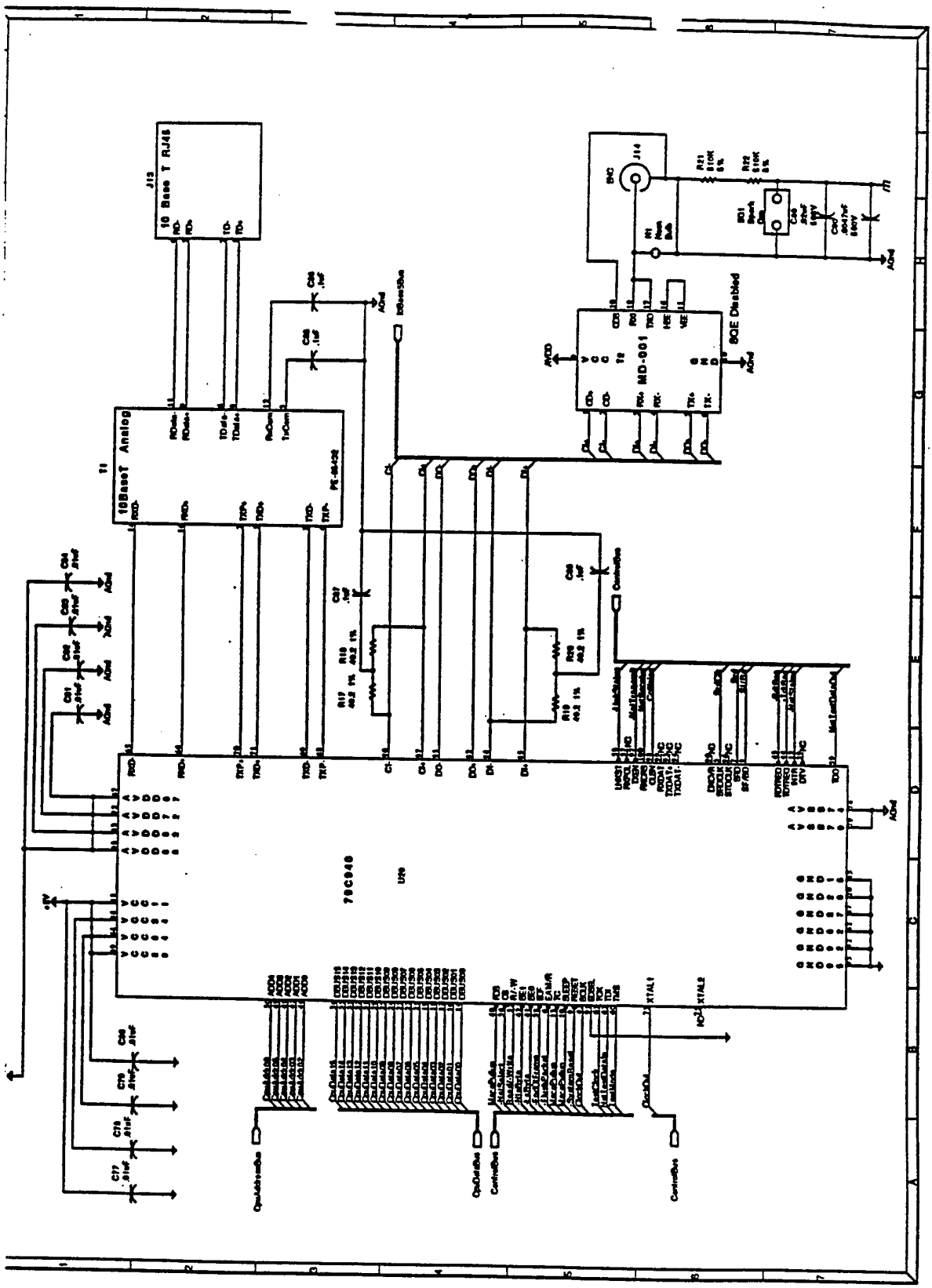



Fig. 40I

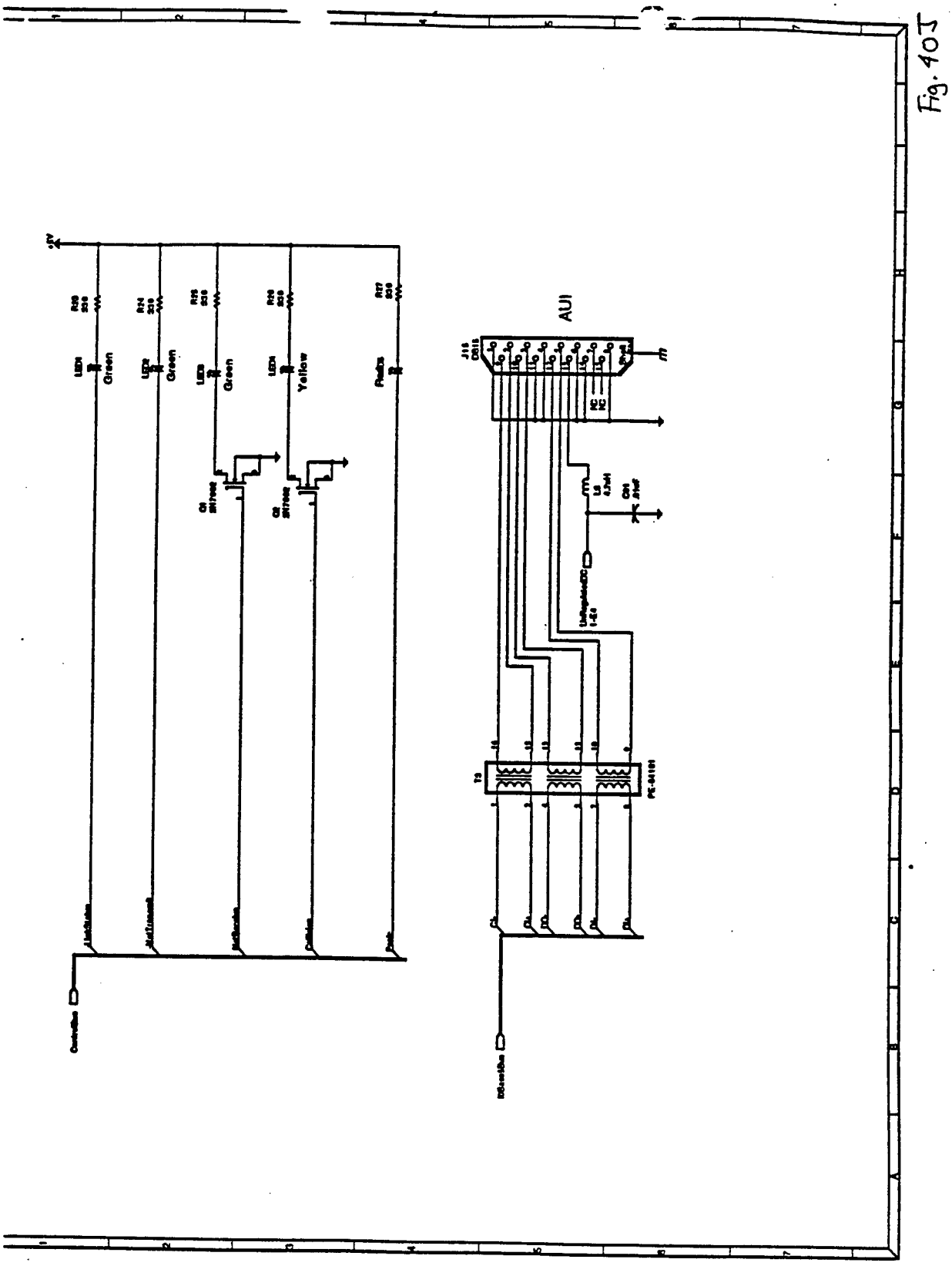



Fig. 40J

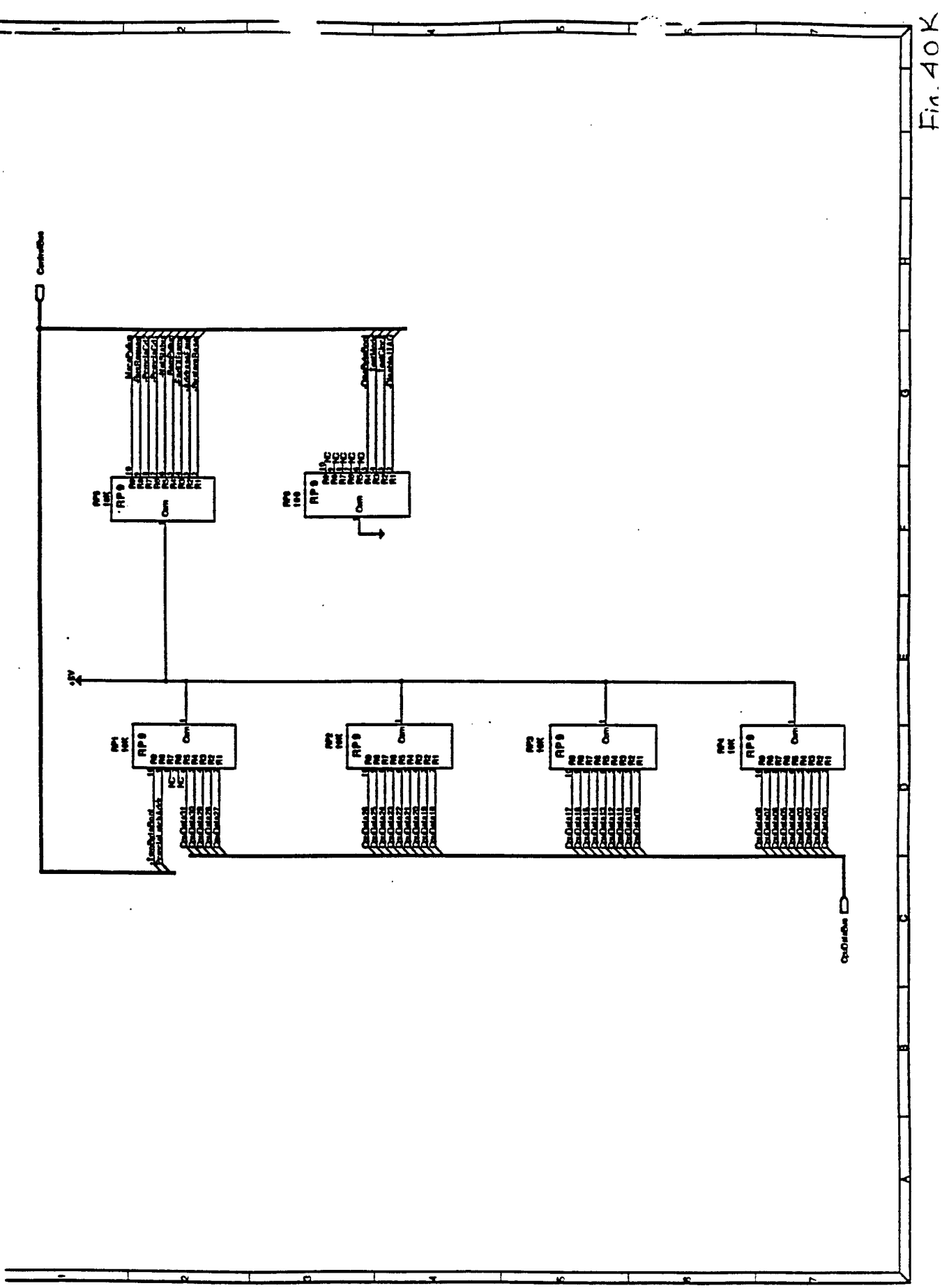


Fig. 40K

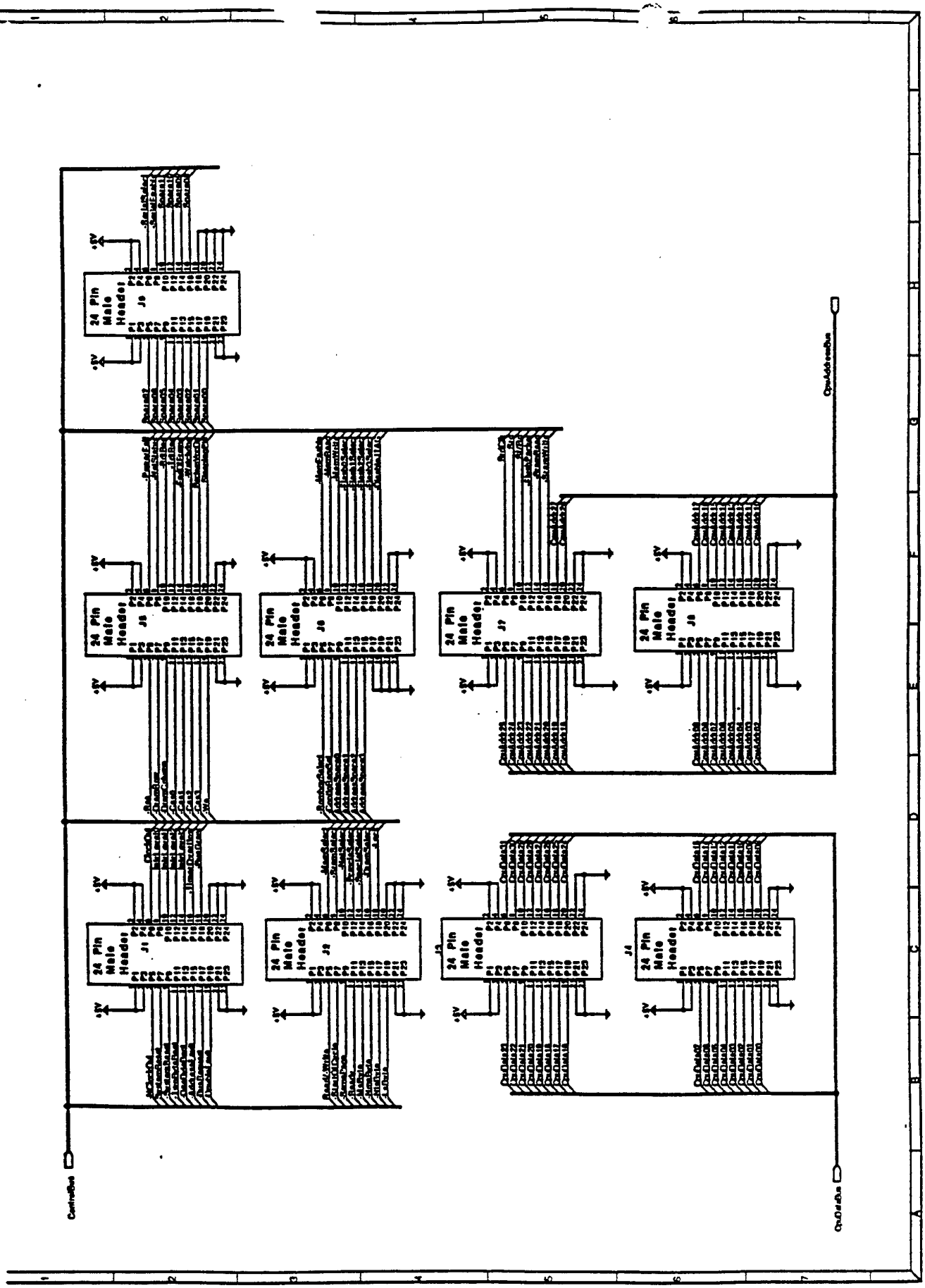
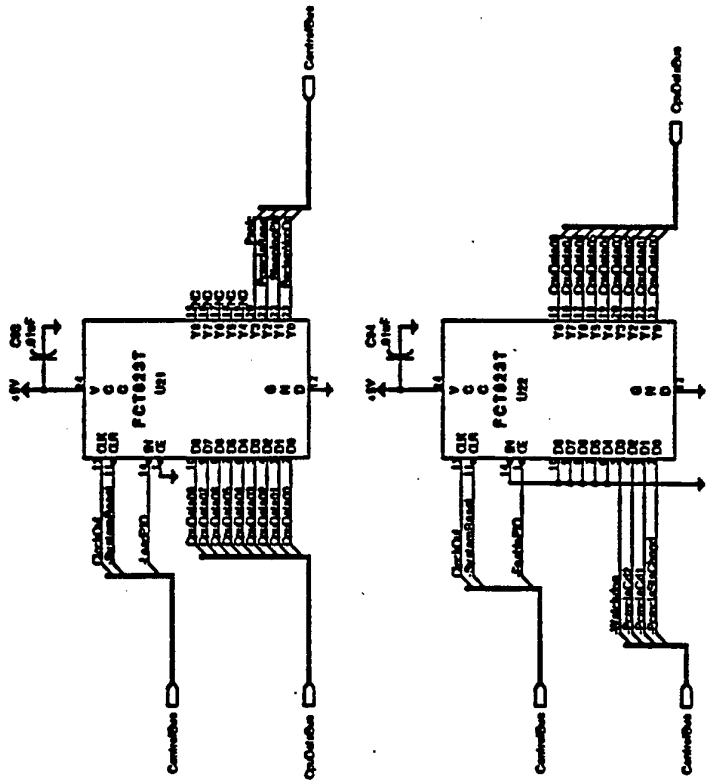



Fig. 40L



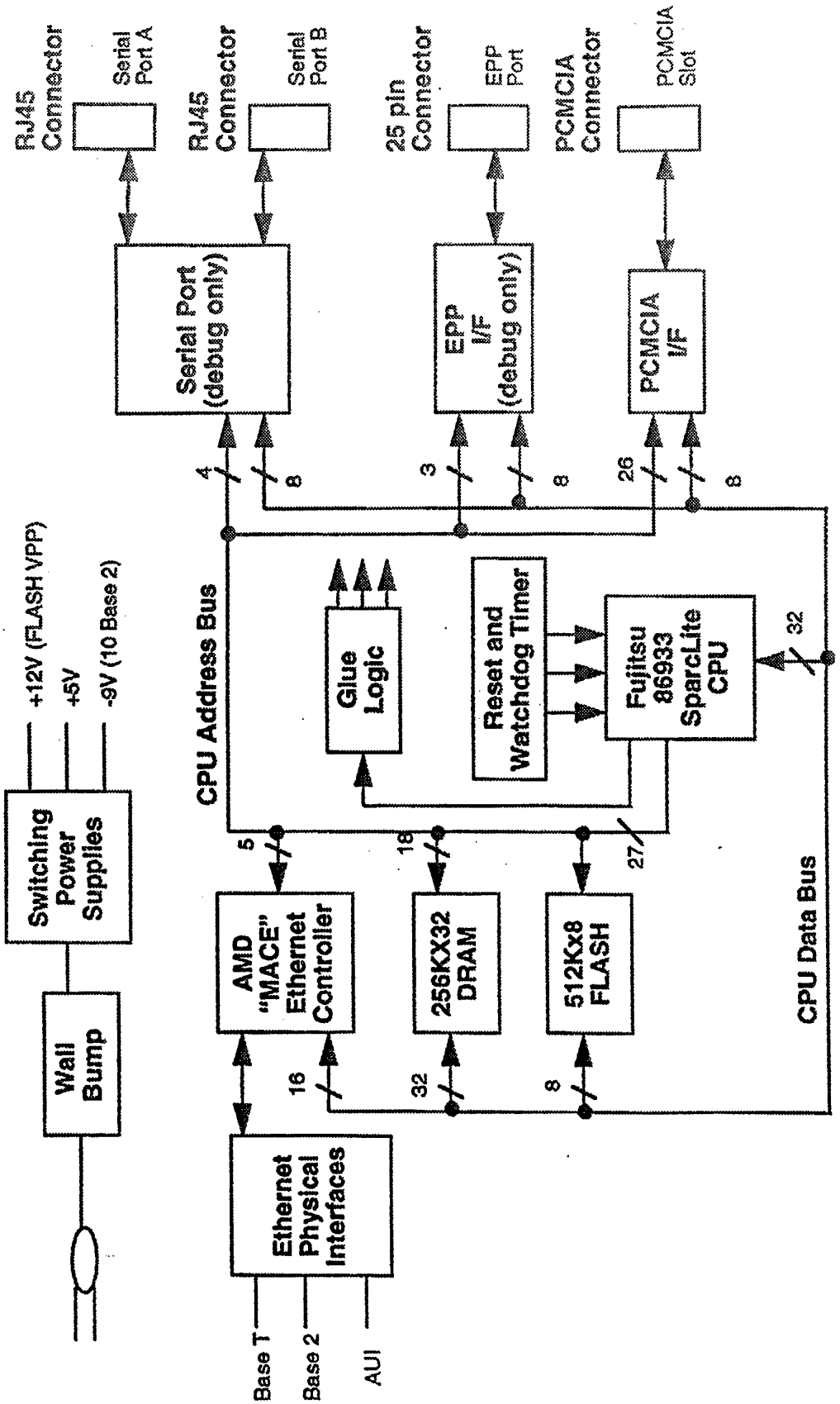

RJ45 Connector
Serial Port A
RJ45 Connector
Serial Port B
25 pin Connector
EPP Port
PCMCIA Connector
PCMCIA Slot
Serial Port (debug only)
EPP I/F (debug only)
PCMCIA I/F
4
8
3
8
26
8
CPU Address Bus
Glue Logic
Reset and Watchdog Timer
Fujitsu 86933 SparcLite CPU
32
+12V (FLASH VPP)
+5V
-9V (10 Base 2)
Switching Power Supplies
5
18
27
AMD "MACE" Ethernet Controller
256KX32 DRAM
512Kx8 FLASH
CPU Data Bus
Wall Bump
16
32
8
Ethernet Physical Interfaces
Base T
Base 2
AUI

Fig. 41

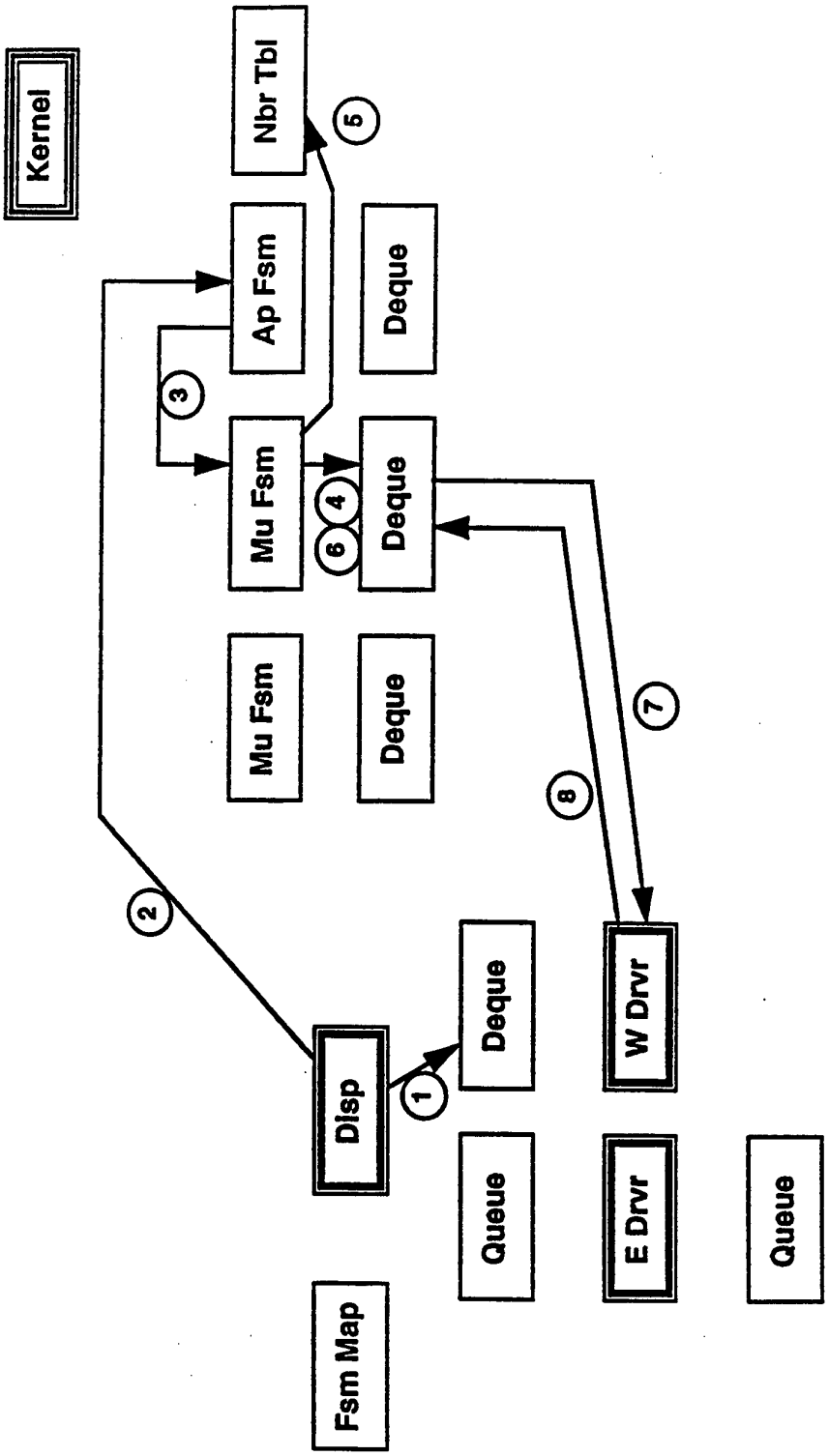

Kernel
Nbr Tbl
Ap Fsm
Deque
Mu Fsm
Deque
Mu Fsm
Deque
Disp
Deque
W Drvr
Queue
E Drvr
Queue
Fsm Map
1
2
3
4
5
6
7
8

Fig. 42

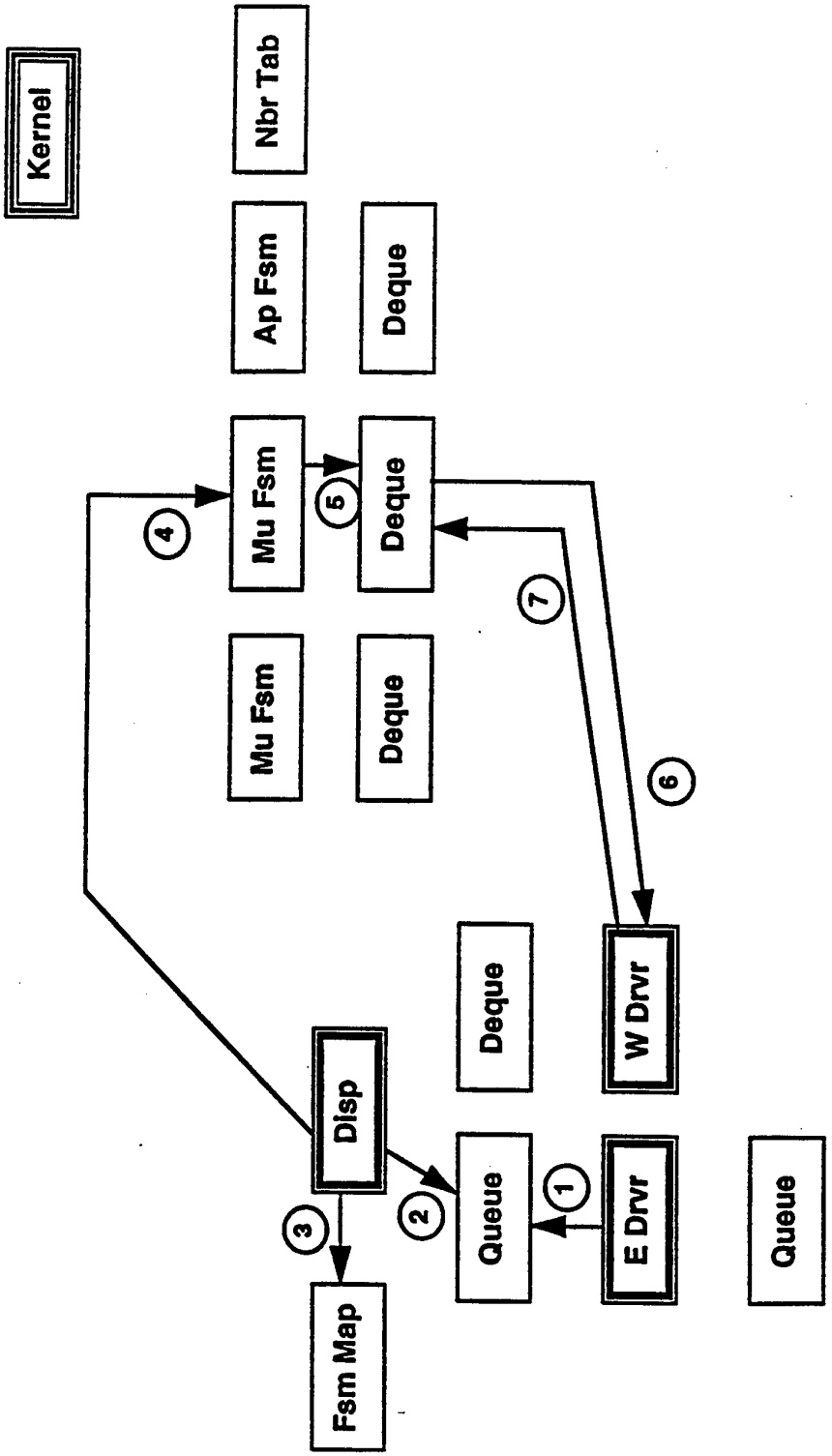

Kernel
Nbr Tab
Ap Fsm
Deque
Mu Fsm
Deque
Mu Fsm
Deque
Deque
W Drvr
Disp
Queue
E Drvr
Queue
Fsm Map
1
2
3
4
5
6
7

Fig. 43A

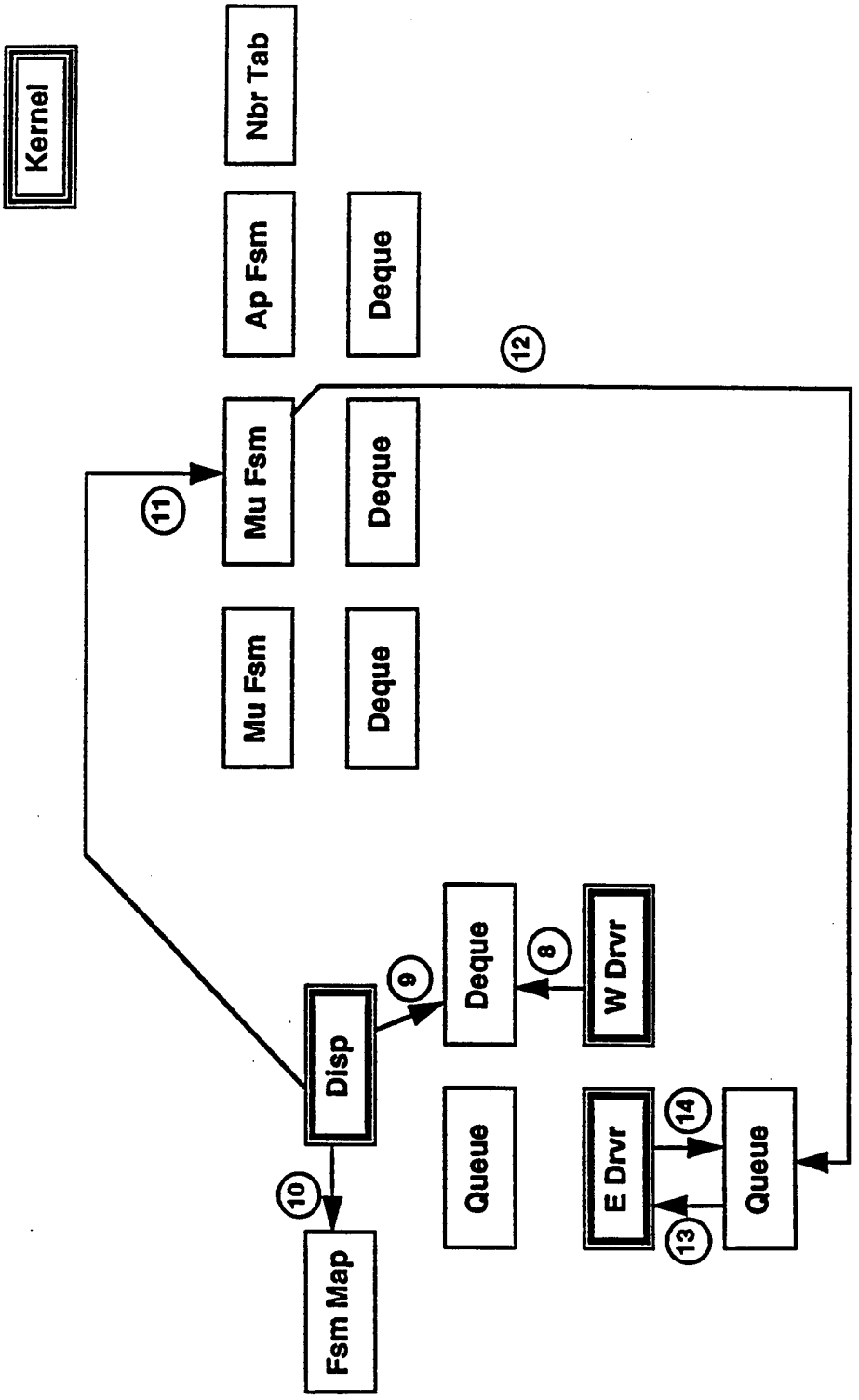

Kernel
Nbr Tab
Ap Fsm
Deque
12
Mu Fsm
Deque
11
Mu Fsm
Deque
Deque
8
W Drvr
9
Disp
Queue
E Drvr
14
Queue
13
10
Fsm Map

Fig. 43B

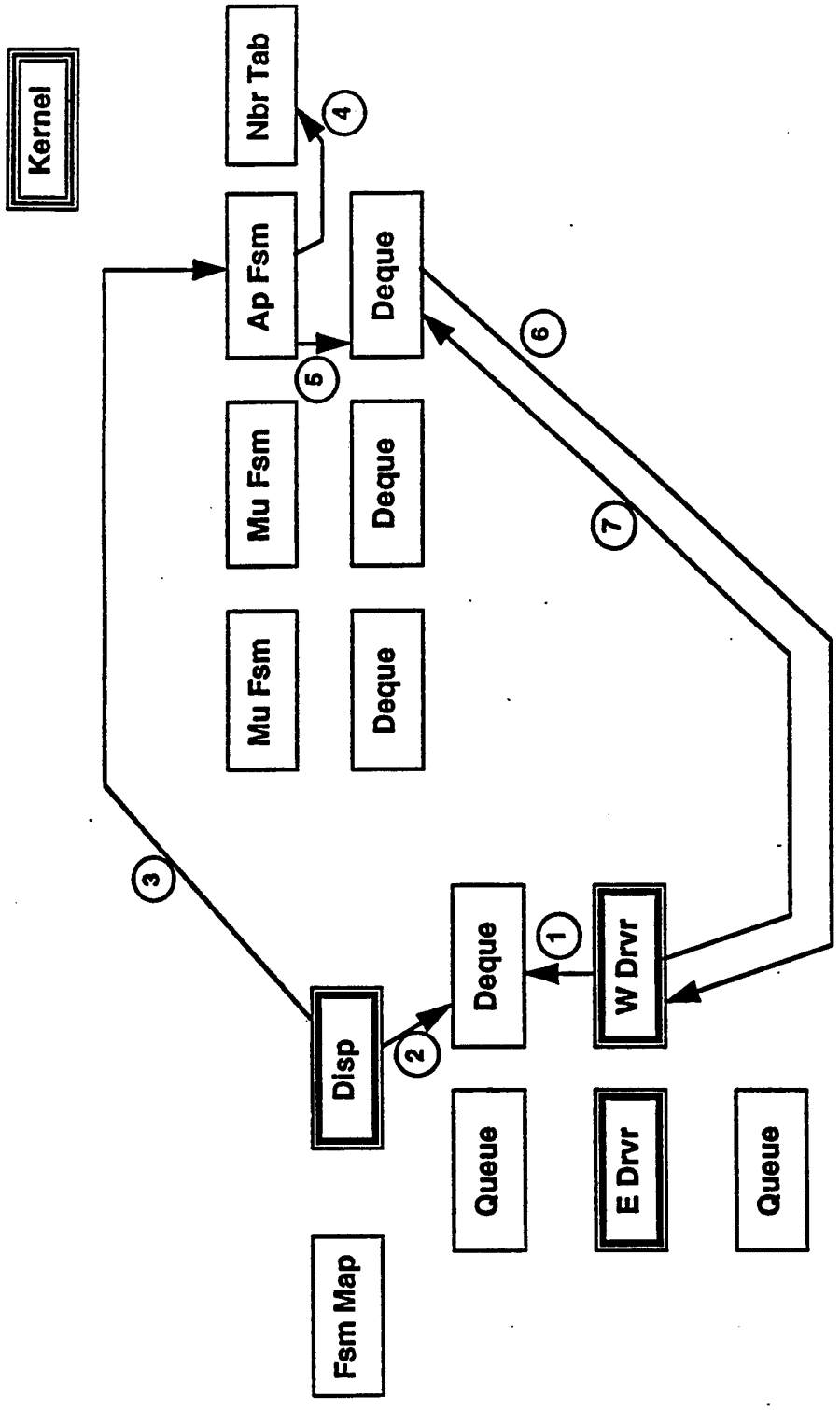

Kernel
Nbr Tab
Ap Fsm
Deque
Mu Fsm
Deque
Mu Fsm
Deque
Deque
W Drvr
Disp
Queue
E Drvr
Queue
Fsm Map
1
2
3
4
5
6
7

Fig. 44

# INTERNATIONAL SEARCH REPORT

International application No.
PCT/US94/07004

## A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :H04L 7/00, 9/00
US CL :375/1, 68, 107; 395/200
According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 375/1, 62, 68, 107; 395/200

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Extra Sheet.

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| P,A | US, A, 5,287,384 (Avery et al.) 15 Febuary 1994, abstract, col.7 line 35-44, col.11 line 55 - col.12 line 2. | 1-11, 24-40 |
| A | US, A, 5,048,057 (Saleh et al.) 10 September 1991, Abtract. | 1-40 |
| Y,A | US, A, 5,123,029 (Bantz et al.) 16 June 1992, col. 5 - col. 6. | 1-11, 24-40 |
| Y | US, A, 5,029,180 (Cowart) 02 July 1991, abstract. | 12-23 |
| P,A | US, A, 5,323,419 (Mori et al.) 21 June 1994, abstract. | 12-23 |
| E,A | US, A, 5,335,276 (Thompson et al) 02 August 1994, abstract. | 12-23 |

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

| | | | |
|---|---|---|---|
| * | Special categories of cited documents: | "T" | later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention |
| "A" | document defining the general state of the art which is not considered to be part of particular relevance | "X" | document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone |
| "E" | earlier document published on or after the international filing date | "Y" | document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art |
| "L" | document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) | "&" | document member of the same patent family |
| "O" | document referring to an oral disclosure, use, exhibition or other means | | |
| "P" | document published prior to the international filing date but later than the priority date claimed | | |

Date of the actual completion of the international search
28 OCTOBER 1994

Date of mailing of the international search report
NOV 17 1994

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer
THOMAS LEE
Telephone No. (703) 305-9600

Form PCT/ISA/210 (second sheet)(July 1992)*

# INTERNATIONAL SEARCH REPORT

International application No.
PCT/US94/07004

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

| Category* | Citation of document, with indication, where appropriate, of the relevant passages | Relevant to claim No. |
|---|---|---|
| Y | US, A, 5,146,471 (Cowart) 08 September 1992, abstract. | 12-23 |

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

APS
L1 389 S IMAGE (2A) REJECT?
L2 136 S 375/68/CCLS
L3 0 S 735/62/CCLS
L4 162 S 375/62/CCLS
L5 1 S (L2 OR L4 ) AND L1

=> s phase locked loop
350562 PHASE
146921 LOCKED

161284 LOOP
L6 8698 PHASE LOCKED LOOP
(PHASE(W)LOCKED(W)LOOP)

=> s spread spectrum
93565 SPREAD
97526 SPECTRUM
L7 1224 SPREAD SPECTRUM
(SPREAD(W)SPECTRUM)

=> s l7 and l6 and l1
L8 0 L7 AND L6 AND L1

=>

=> d his

(FILE 'USPAT' ENTERED AT 07:55:49 ON 27 OCT 94)
L1 547 S FREQUENCY HOPPING
L2 416 S WIRELESS (P) NETWORK#
L3 28 S L2 AND L1
L4 9894 S LENGTH (5A) (TRANSMISSION OR TRANSMIT###)
L5 1 S L4 AND L3
L6 839 S 375/1/CCLS