

[54] **PROGRAMMABLE CONTROLLER**

[75] Inventors: **Lawrence W. Hill**, Arlington;
Thomas J. Stoodley III, Lowell;
Ronald Malcolm, Andover, all of
Mass.

[73] Assignee: **Modicon Div. Gould Inc.**, Andover,
Mass.

[21] Appl. No.: **895,581**

[22] Filed: **Apr. 12, 1978**

[51] Int. Cl.³ **G06F 15/46**

[52] U.S. Cl. **364/104; 364/900**

[58] Field of Search ... 364/104, 107, 120, 200 MS File,
364/900 MS File

[56] **References Cited**

U.S. PATENT DOCUMENTS

3,686,639	8/1972	Fletcher	364/200
3,868,648	2/1975	Levin	364/120 X
3,930,233	12/1975	Morley et al.	364/900
3,944,984	3/1976	Morley et al.	364/900
3,964,026	6/1976	Vamauchi et al.	364/900 X
3,975,622	8/1976	Horn et al.	364/120 X
3,976,981	8/1976	Bowden	364/120 X
4,021,783	5/1977	Higberger	364/900
4,038,533	7/1977	Dummermuth et al.	364/104
4,115,853	9/1978	Dummermuth	364/900

FOREIGN PATENT DOCUMENTS

2744434 2/1979 Fed. Rep. of Germany 364/900

OTHER PUBLICATIONS

Allen-Bradley, "Users Manual for the PLC-2 Programmable Controller," Bulletin 1722, Catalog 1772-UM, Part No. 634873-50, (Fourth Edition Jan. 17, 1978) pp. 3-79, 3-82, 3-83.

Instruments and Control Systems, May 1978, "ICS Guide to Programmable Controllers," pp. 25-27, 31, 38, 39.

Jeffery et al., "Retrofitting with CNC," 15th Numerical

Control Society Annual Meeting & Technical Conference, Apr. 9-12, 1978, Chicago, pp. 194-205, 197-200.

Primary Examiner—Joseph F. Ruggiero
Attorney, Agent, or Firm—Mattern, Ware, Stoltz & Fressola

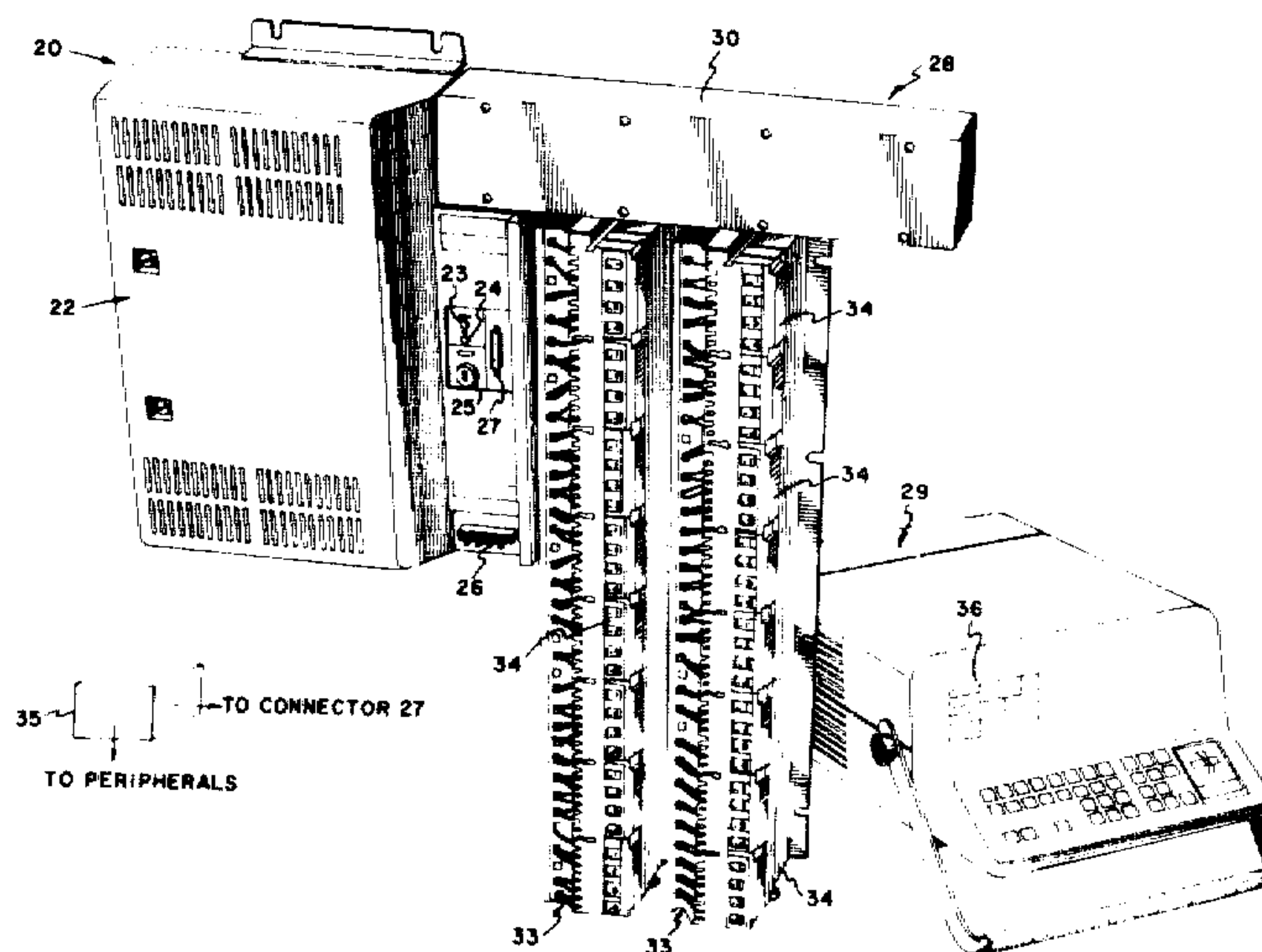
[57] **ABSTRACT**

A small, low cost, programmable controller is described capable of solving user programs represented in networks having up to seven rows and eleven columns. A column solver is utilized to provide efficient and fast solution of the user control network. The programmable controller also solves calculate functions having multiple outputs to facilitate use of the output information in the control program.

A programming panel using a CRT display shows one or more selected control networks and, in conjunction with the central processing unit of the programmable controller, provides for the insertion of networks between two existing networks. Since the networks are solved by the controller mainframe in a sequential fashion, this network insertion allows the user to optimize his or her control program when solution order of the networks is important.

The output coil numbers of network rows may also be assigned by the user independent of their placement in the control program to further facilitate programming the controller. The programming panel includes a movable cursor on the CRT display which, in conjunction with a light-emitting diode (LED), allows the user to monitor the real-time power flow at any particular point in the displayed ladder diagram network. Specialized search features can also be specified by the user to simplify monitoring and de-bugging the control program.

18 Claims, 106 Drawing Figures



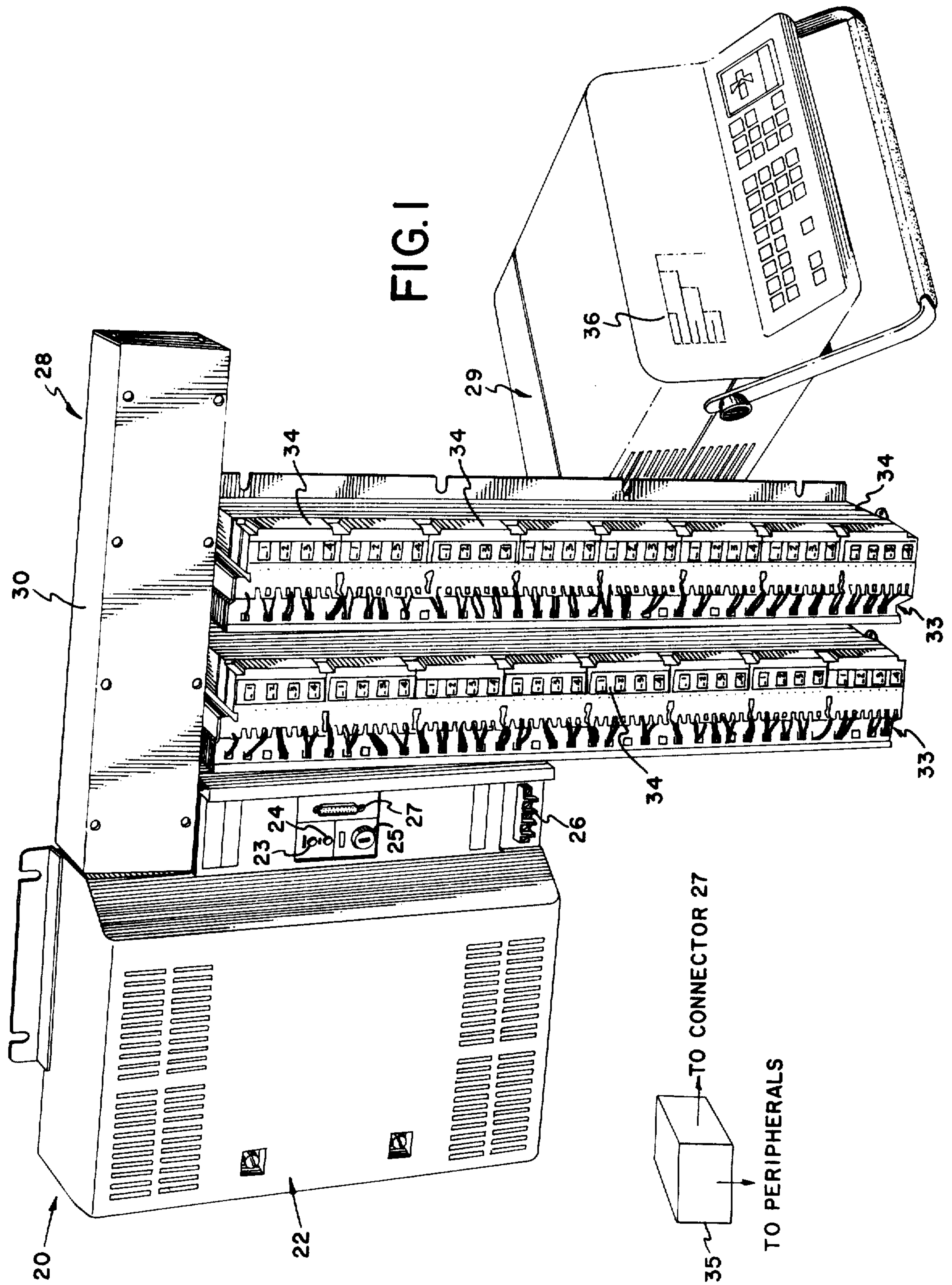


FIG. IA

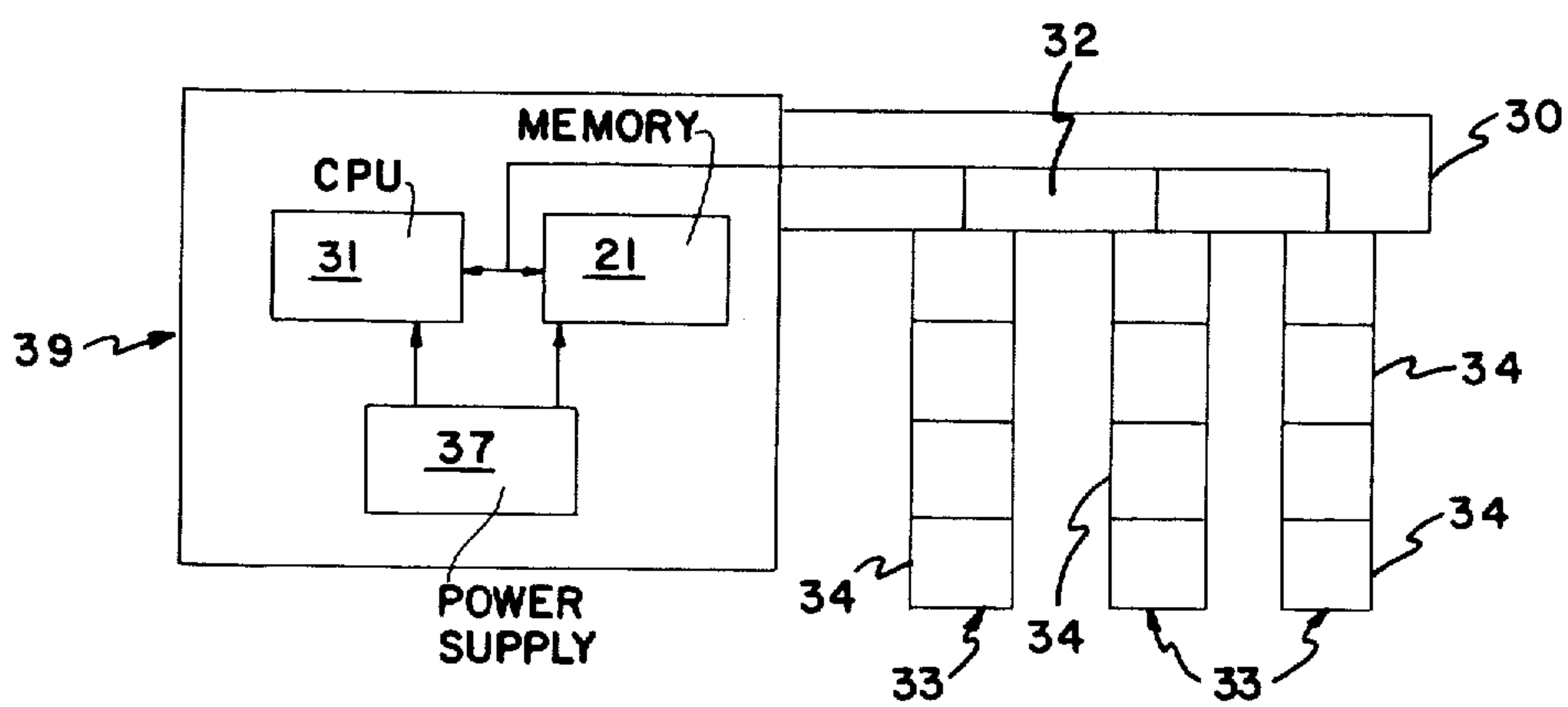
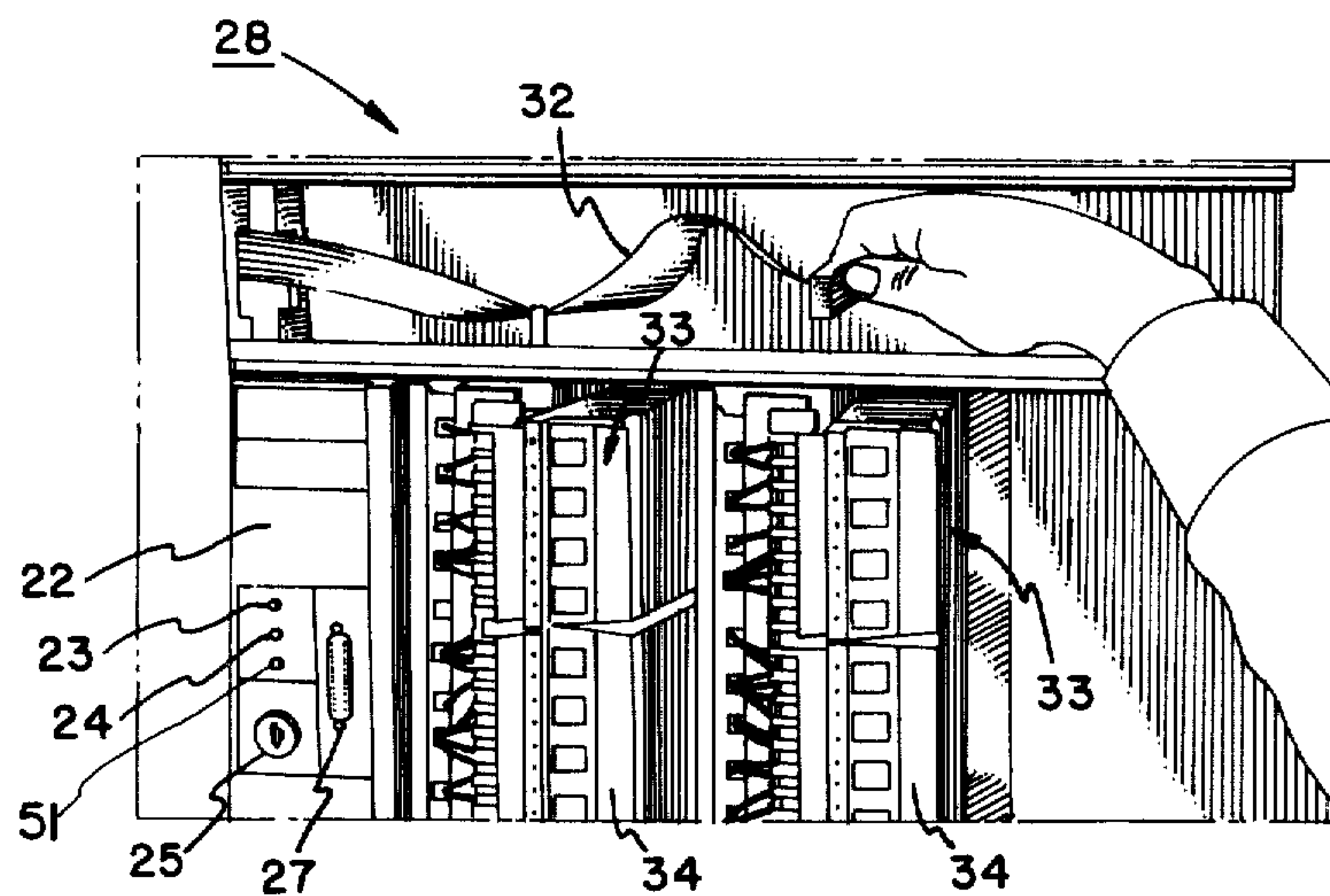


FIG. IB

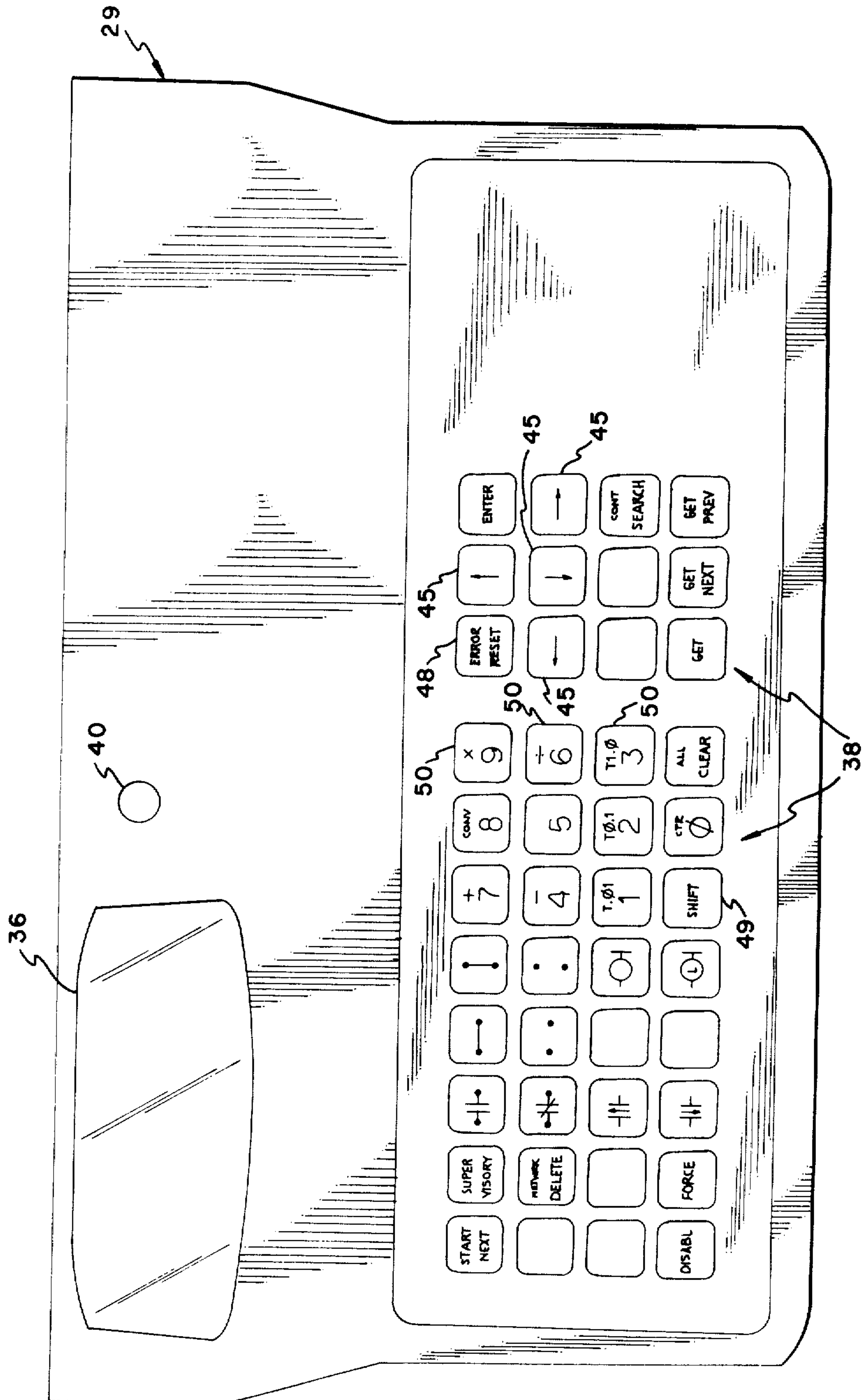


FIG. 2

FIG. 3

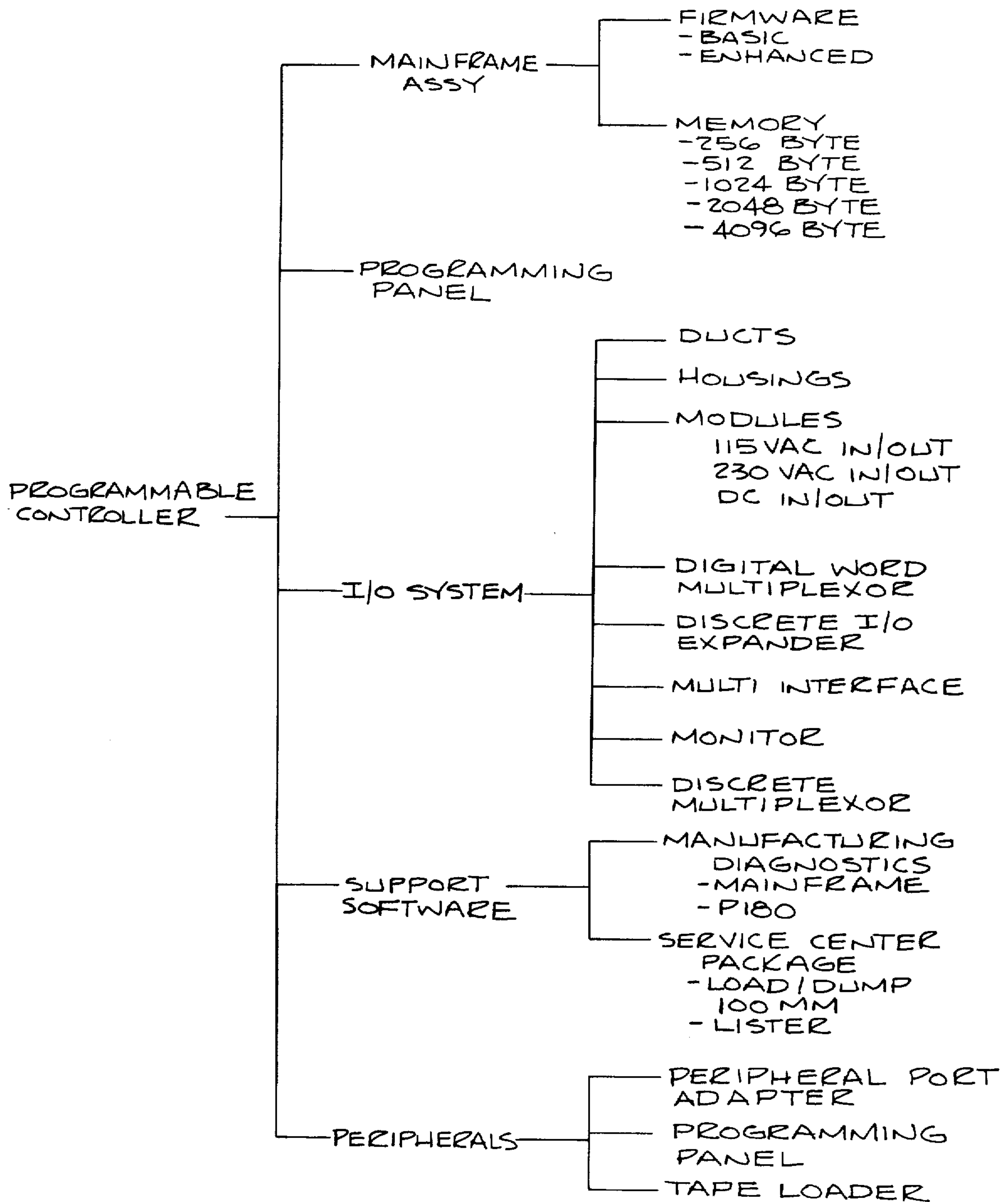


FIG. 4

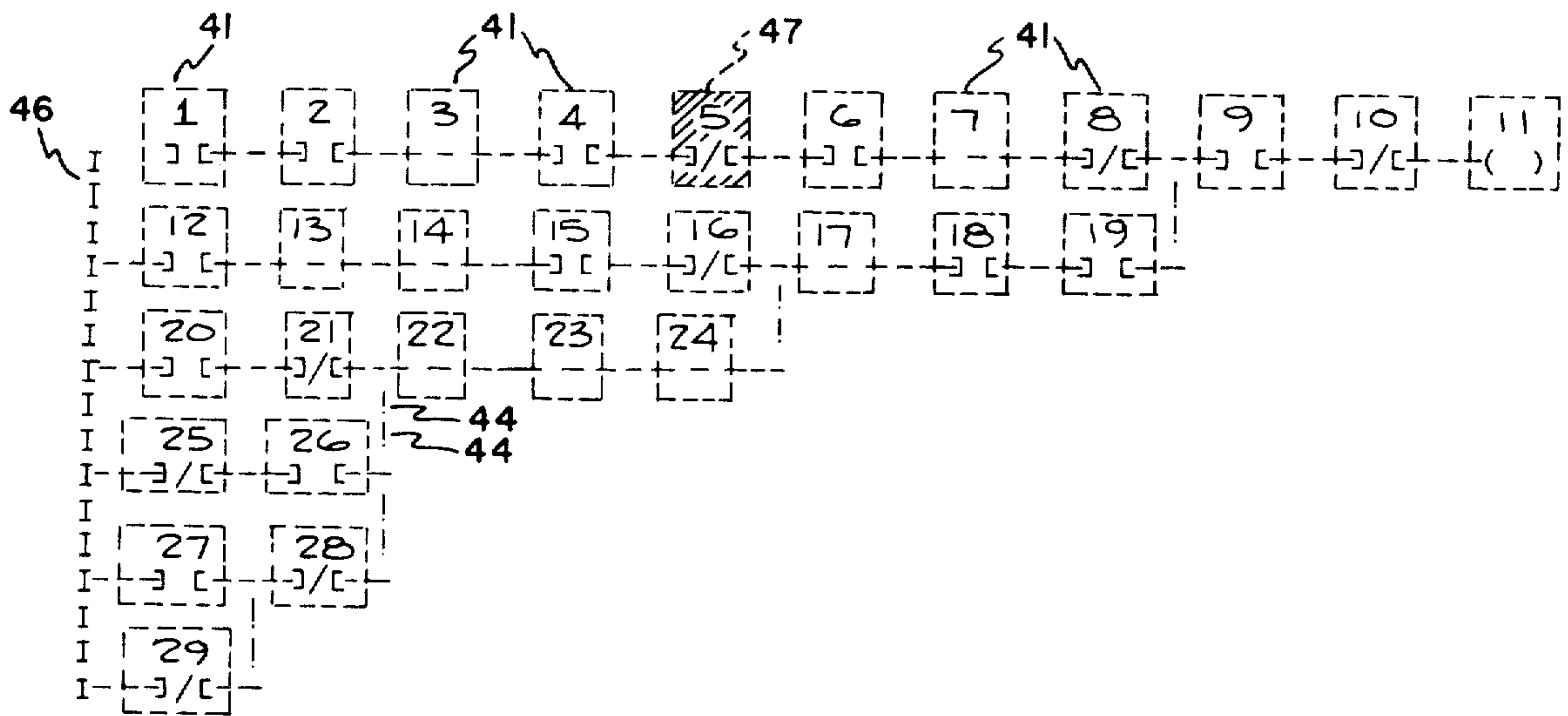
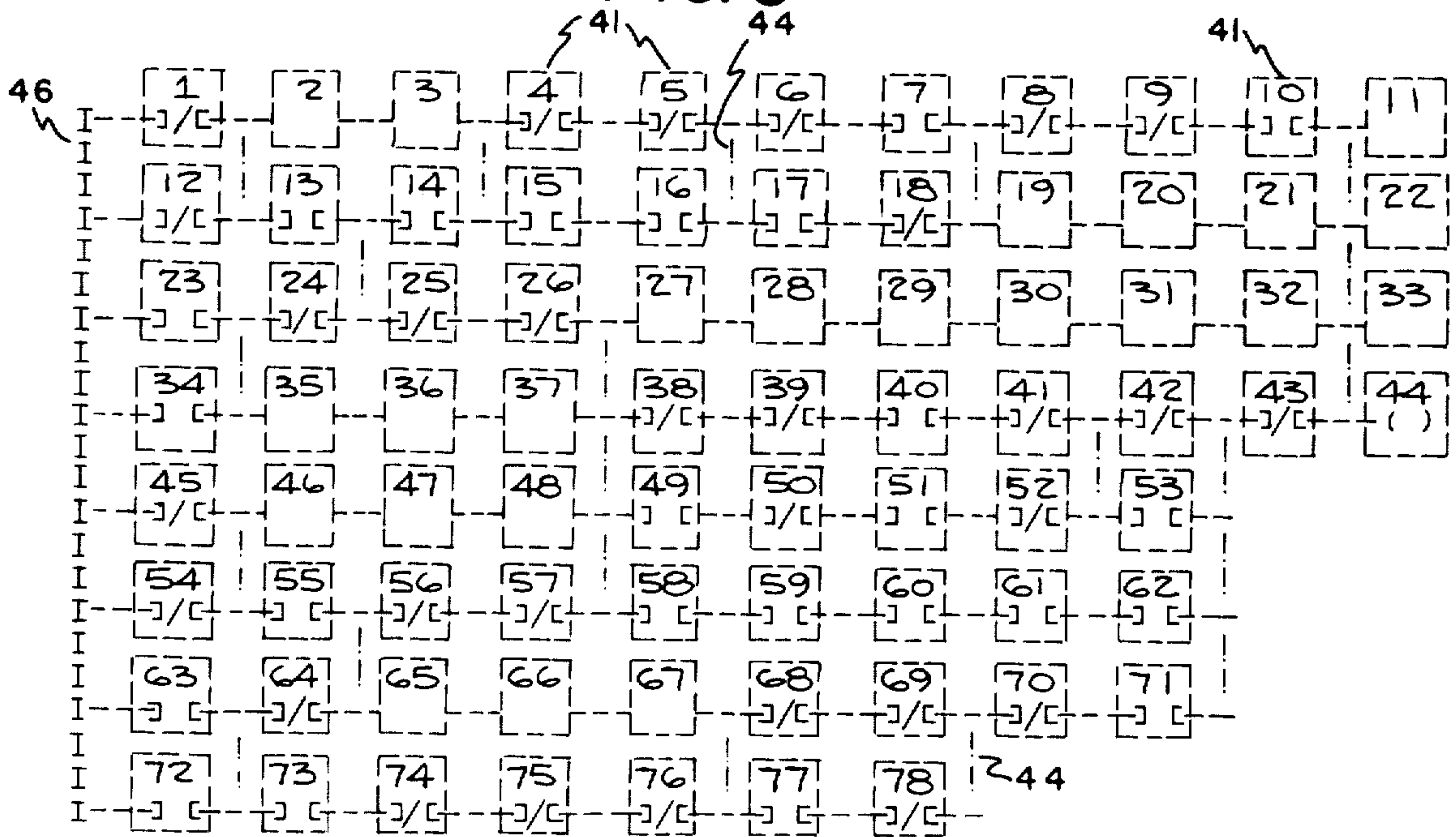


FIG. 5



CCCCV	EEEEEEEEEEEE	STEP#	USED	REF	NNNN	NNNN	NNNN	NNNN	NNNN	NNNN
RRRV	S	AAAAAAA	NNN	XX	VAL					

FIG. 6A

-]	[+		STEP#	USED	REF	4101	4071	4051	0002	1010	2001
1023!			0031	40							

FIG. 6B

-]	[-		STEP#	USED	REF	4001	1010	0001	0002
1001		START	0001	97					

FIG. 6C

-]	[-	MEM PROTECT	STEP#	USED	REF	1008	4025	0100	1100
1123			0342	83	VAL				

FIG. 6D

LEGEND

- CCCC: CONTACT TYPE
- EEEEEEEEEEEEEE: ERROR MESSAGE
- VV: VERTICAL CONNECTOR
- S: SHIFT ON
- AAAAAAA: ADVISORY MESSAGE

DISPLAY FORMAT & ASSEMBLY/STATUS EXAMPLES

FIG. 6E

FIG. 7

	ASSEMBLY AREA	CONTACT AT CURSOR	RESULT AT CURSOR
1.			
2.			
3.			
4.			
5.			
6.			
7.			

- 1 - CHANGE REFERENCE NUMBER, CONTACT TYPE VERTICAL.
- 2 - CHANGE REFERENCE NUMBER
- 3 - CHANGE CONTACT TYPE
- 4 - CHANGE VERTICAL
- 5 - CHANGE REFERENCE NUMBER, CONTACT TYPE
- 6 - CHANGE CONTACT TYPE VERTICAL
- 7 - CHANGE REFERENCE NUMBER, VERTICAL

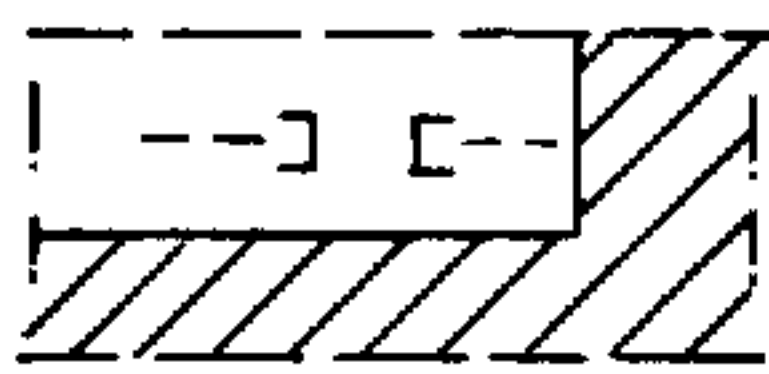
ASSEMBLY AREA

SEARCH



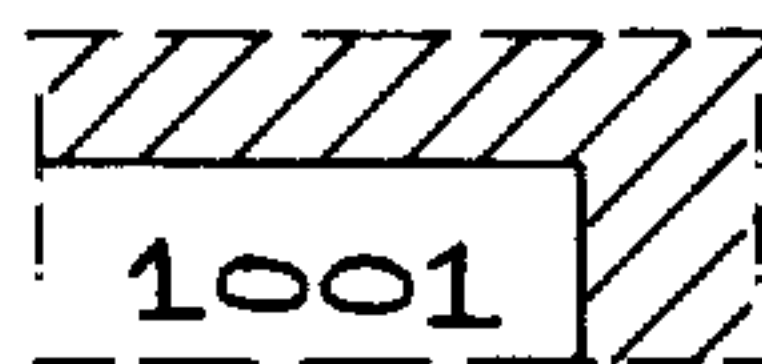
SEARCH FOR FIRST NODE

FIG. 8A



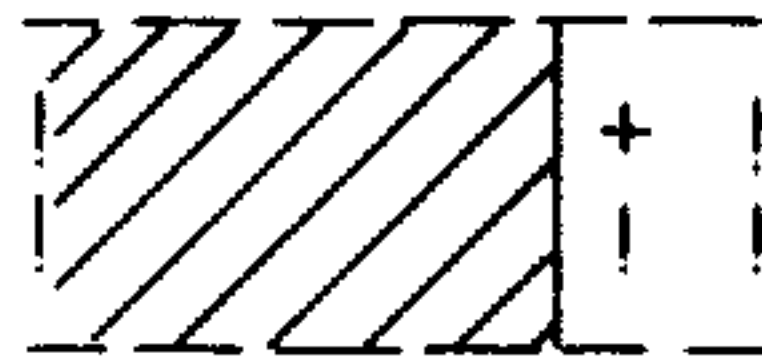
SEARCH FOR FIRST OCCURRENCE OF THAT CONTACT TYPE

FIG. 8B



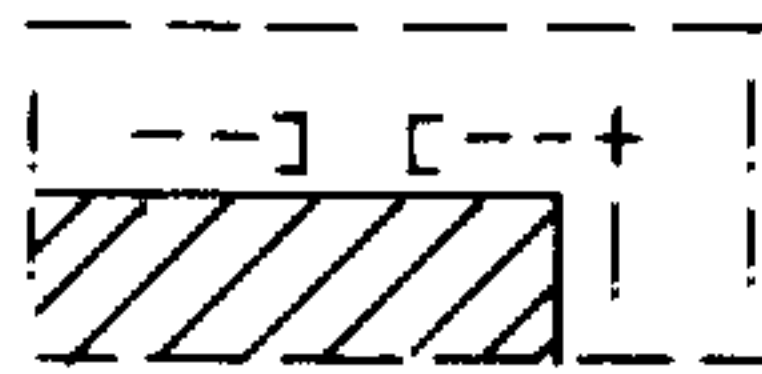
SEARCH FOR FIRST OCCURRENCE OF THAT REFERENCE NUMBER

FIG. 8C



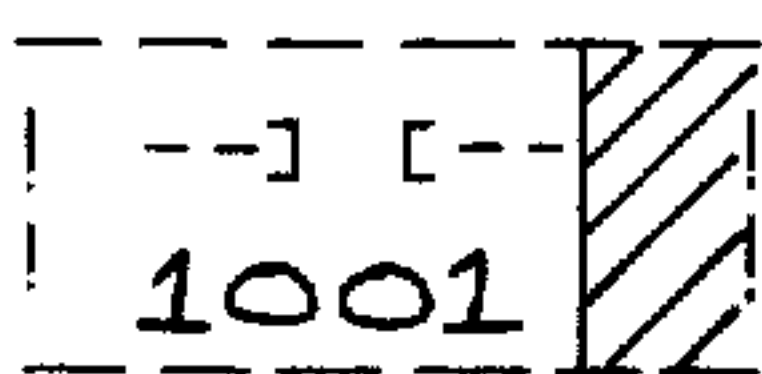
SEARCH FOR FIRST OCCURRENCE OF THAT VERTICAL CONNECTOR

FIG. 8D



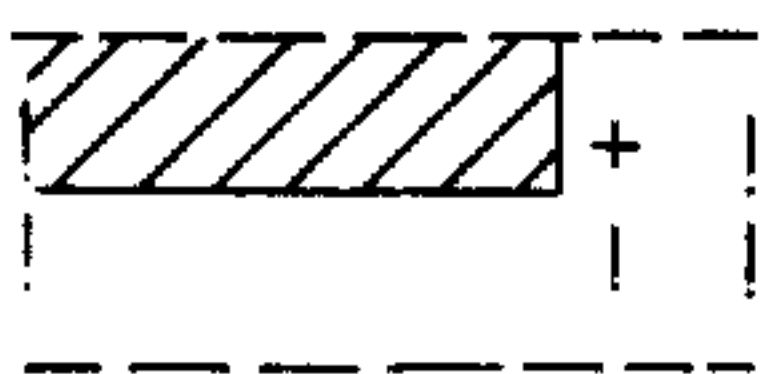
SEARCH FOR FIRST OCCURRENCE OF THAT CONTACT TYPE AND VERTICAL

FIG. 8E



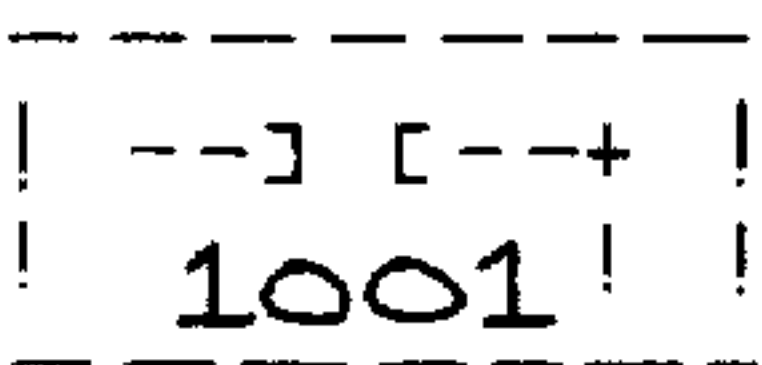
SEARCH FOR FIRST OCCURRENCE OF THAT CONTACT TYPE AND REFERENCE NUMBER

FIG. 8F



SEARCH FOR FIRST OCCURRENCE OF THAT REFERENCE NUMBER AND VERTICAL

FIG. 8G



SEARCH FOR THE FIRST OCCURRENCE OF THAT NODE

FIG. 8H

FIG. 9

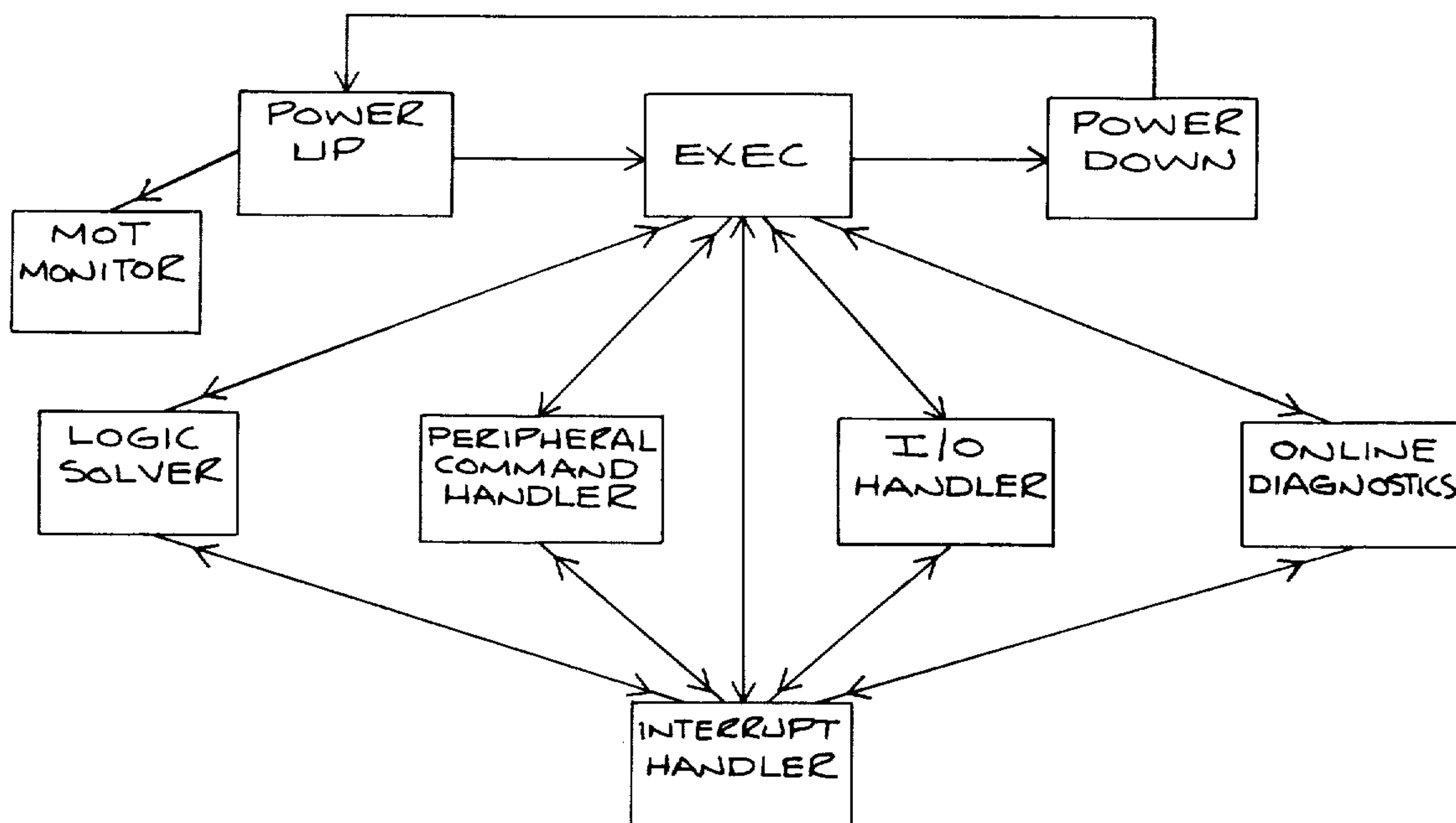
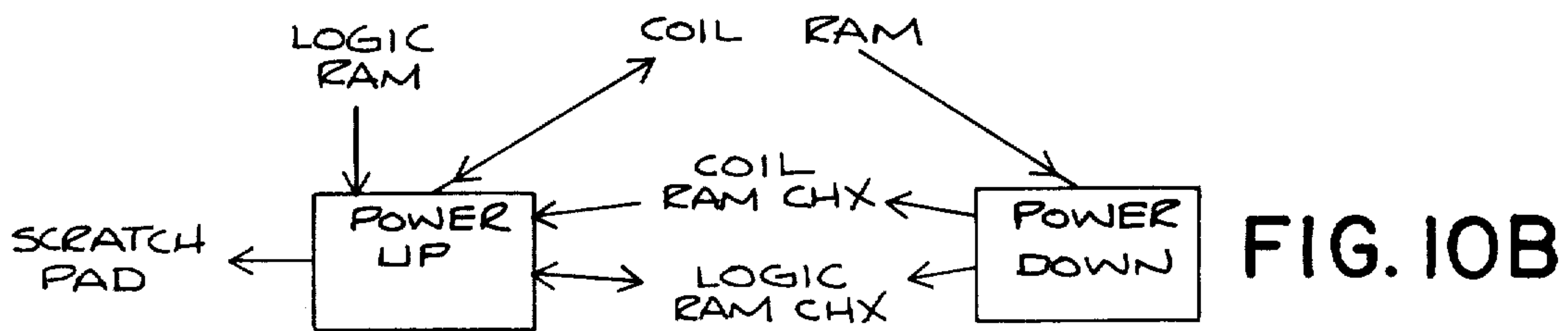
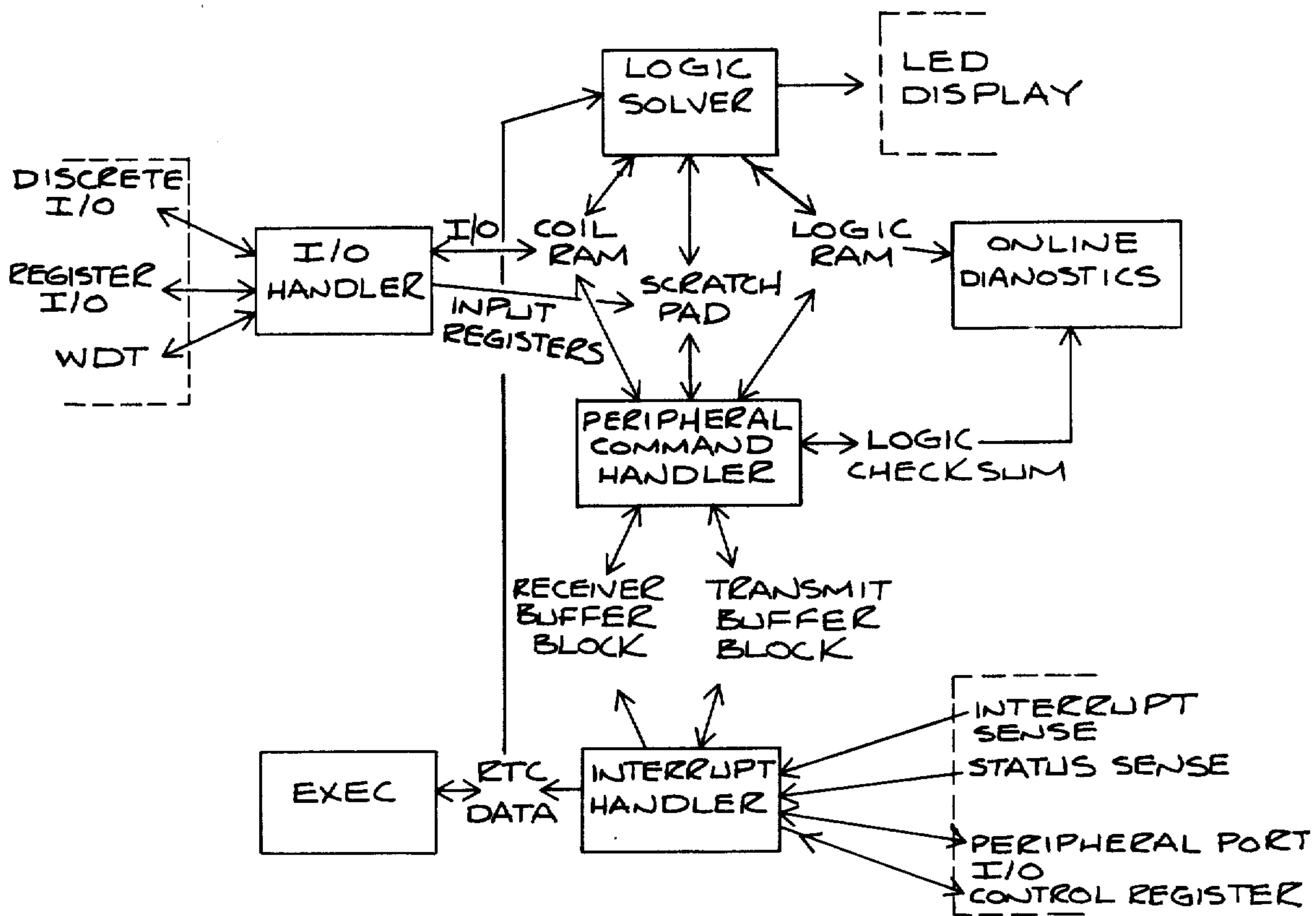


FIG. 10A



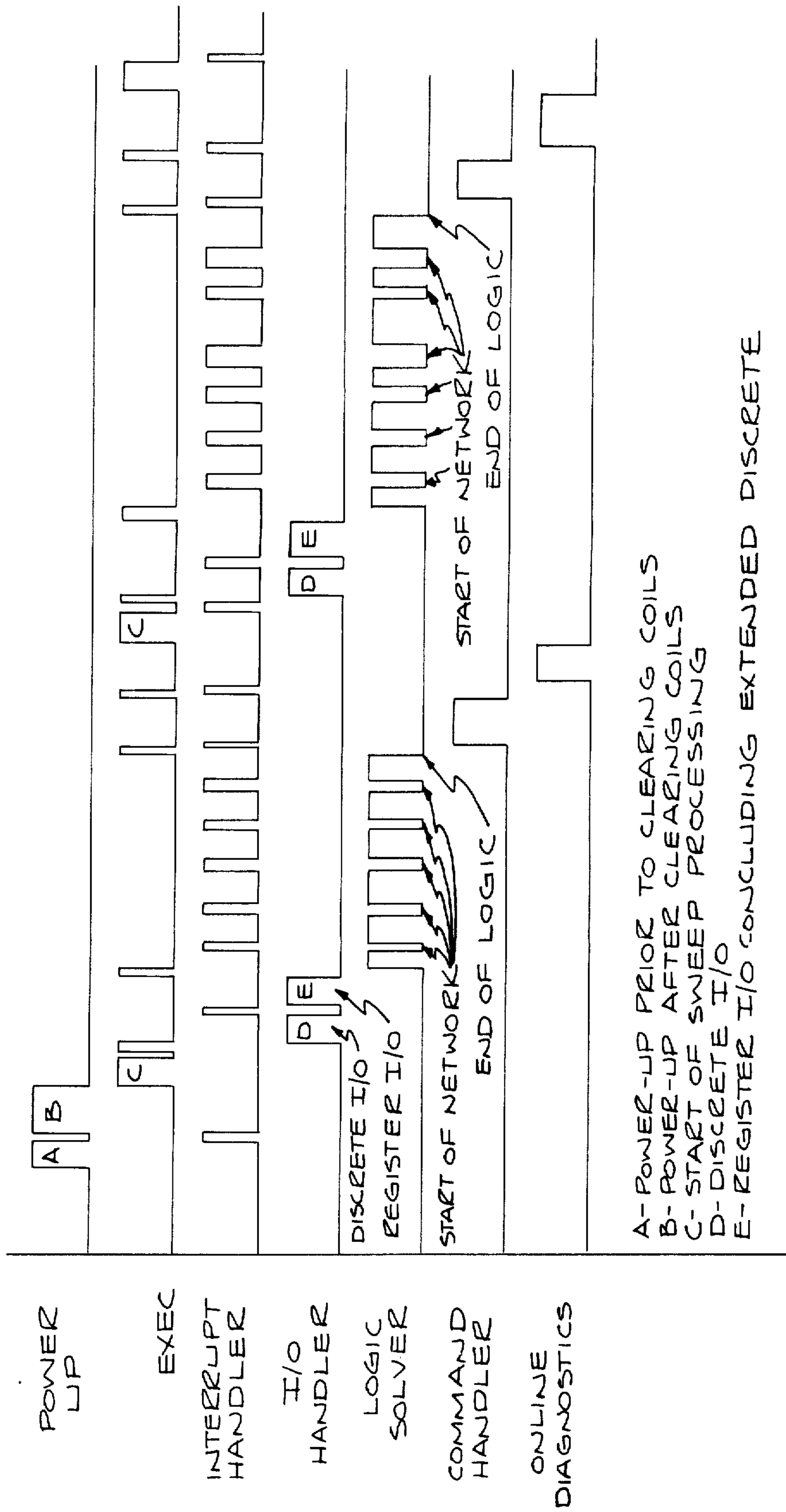
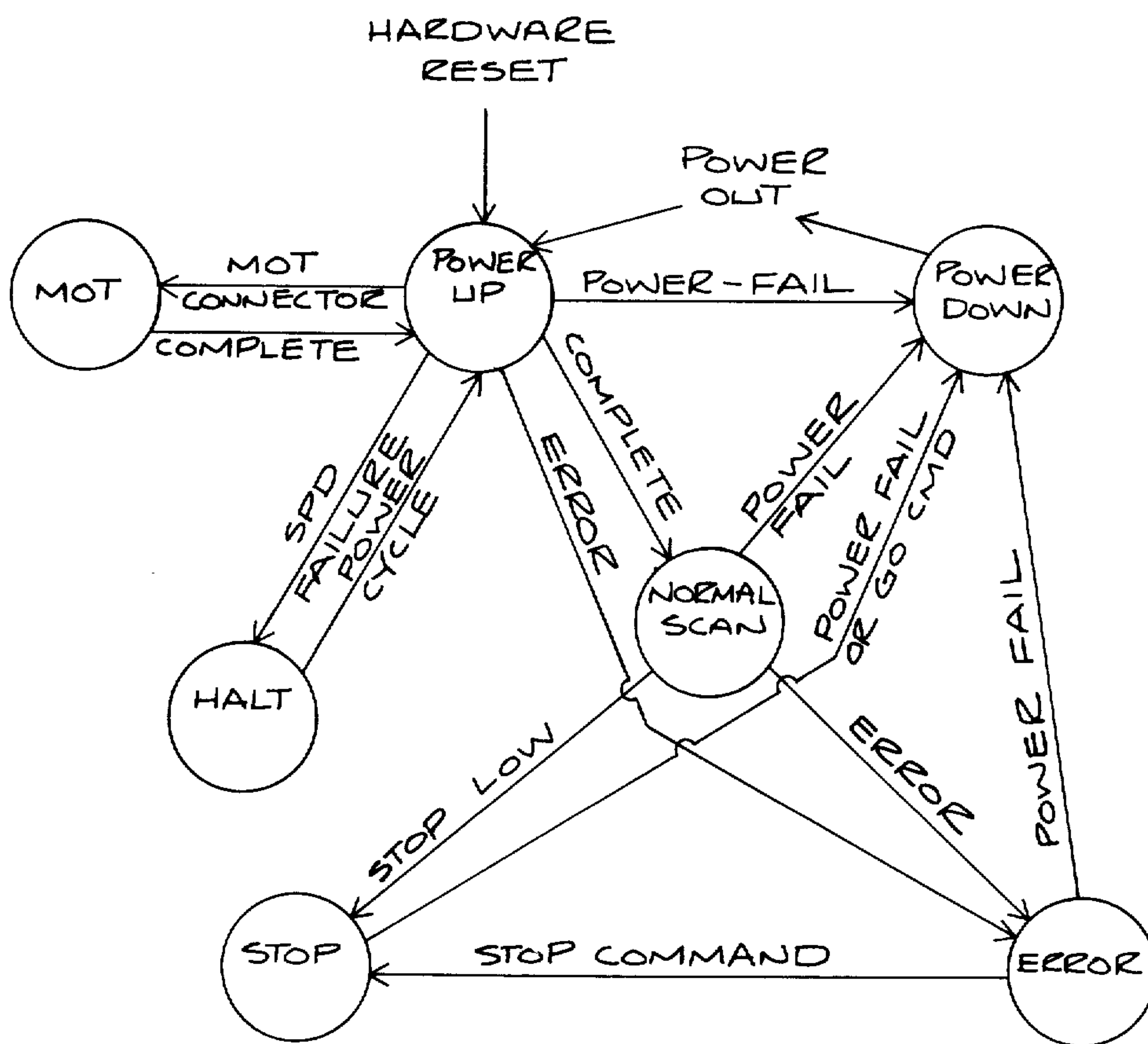


FIG. 11

FIG. 12



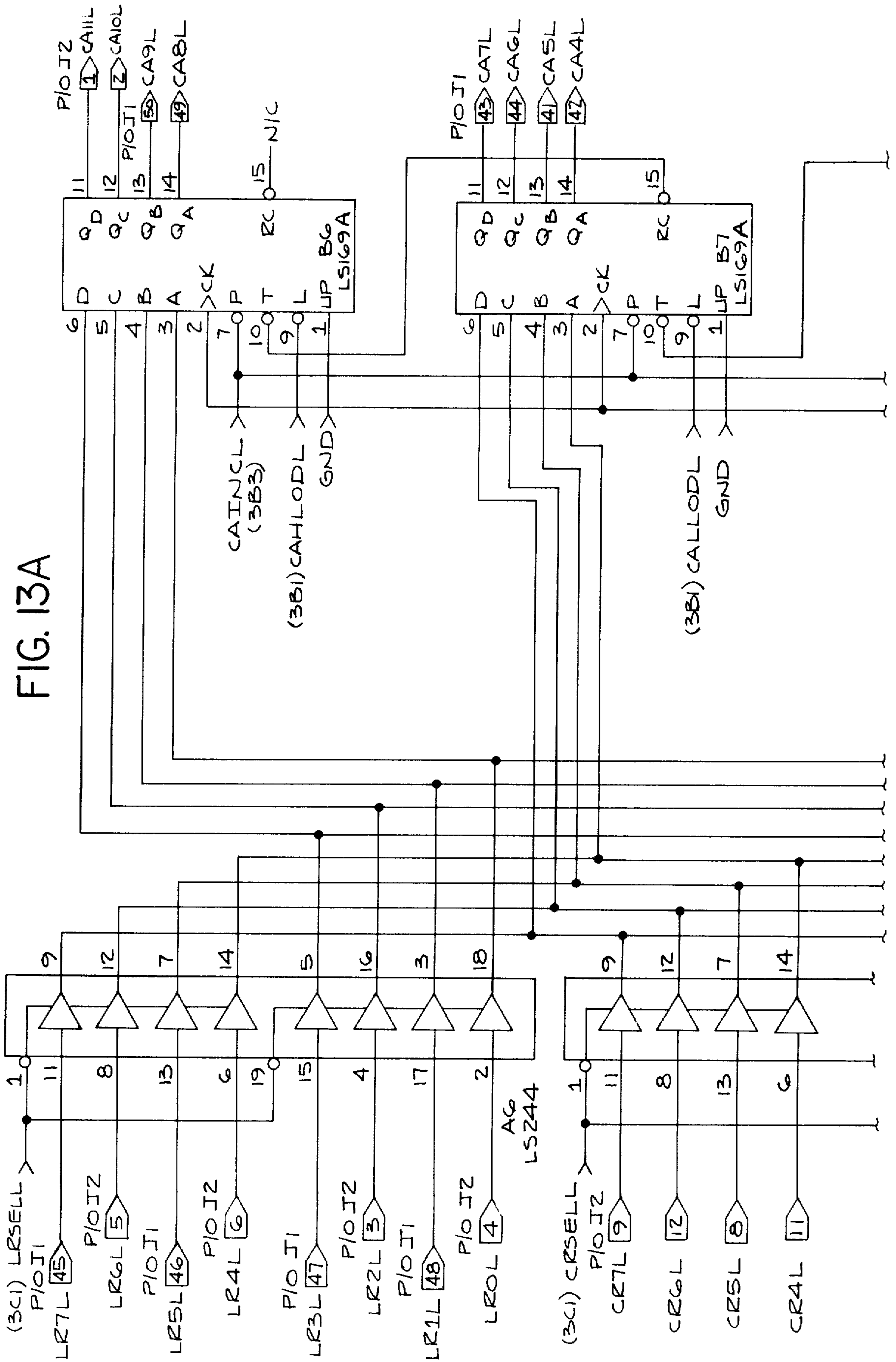
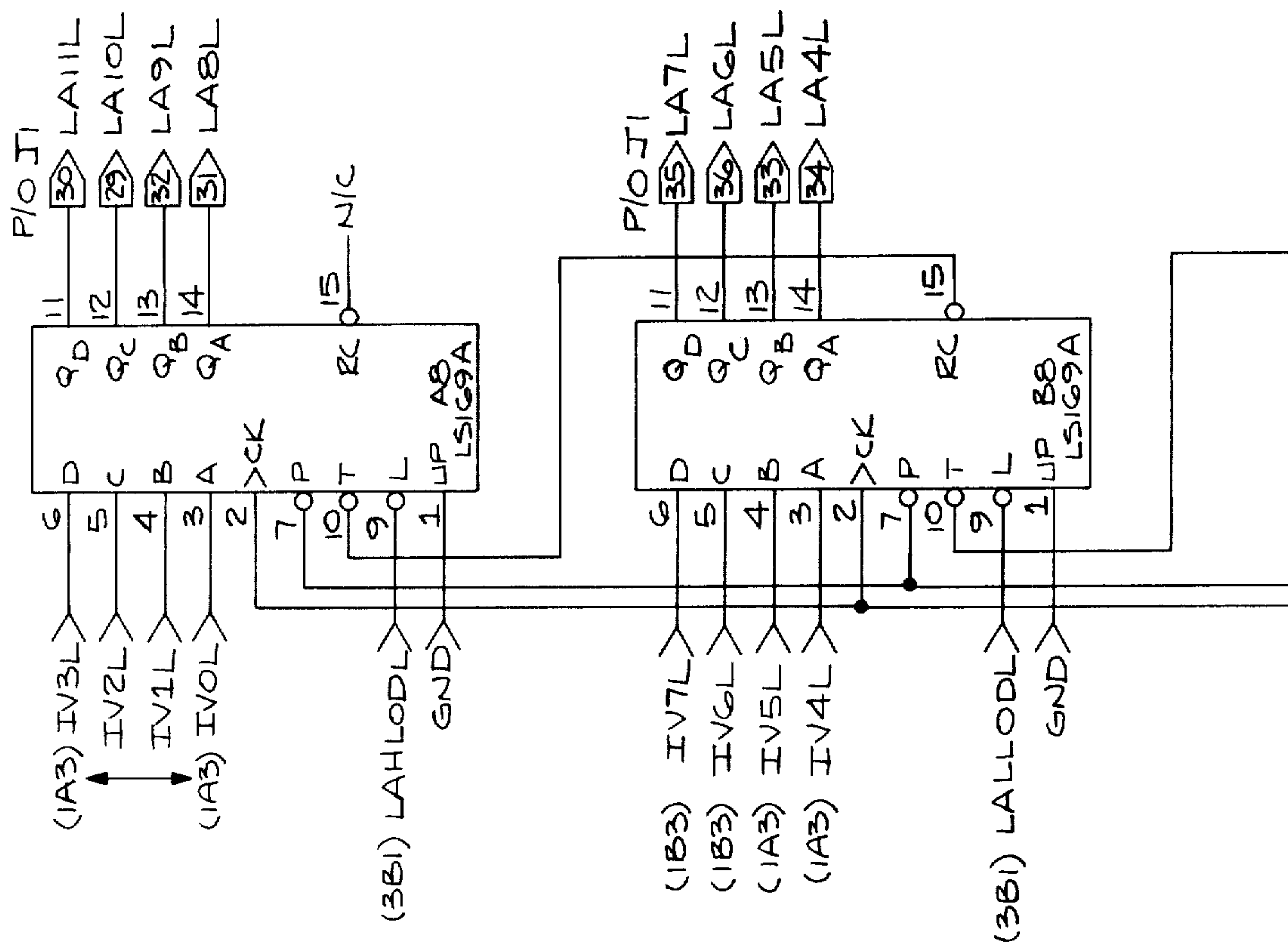
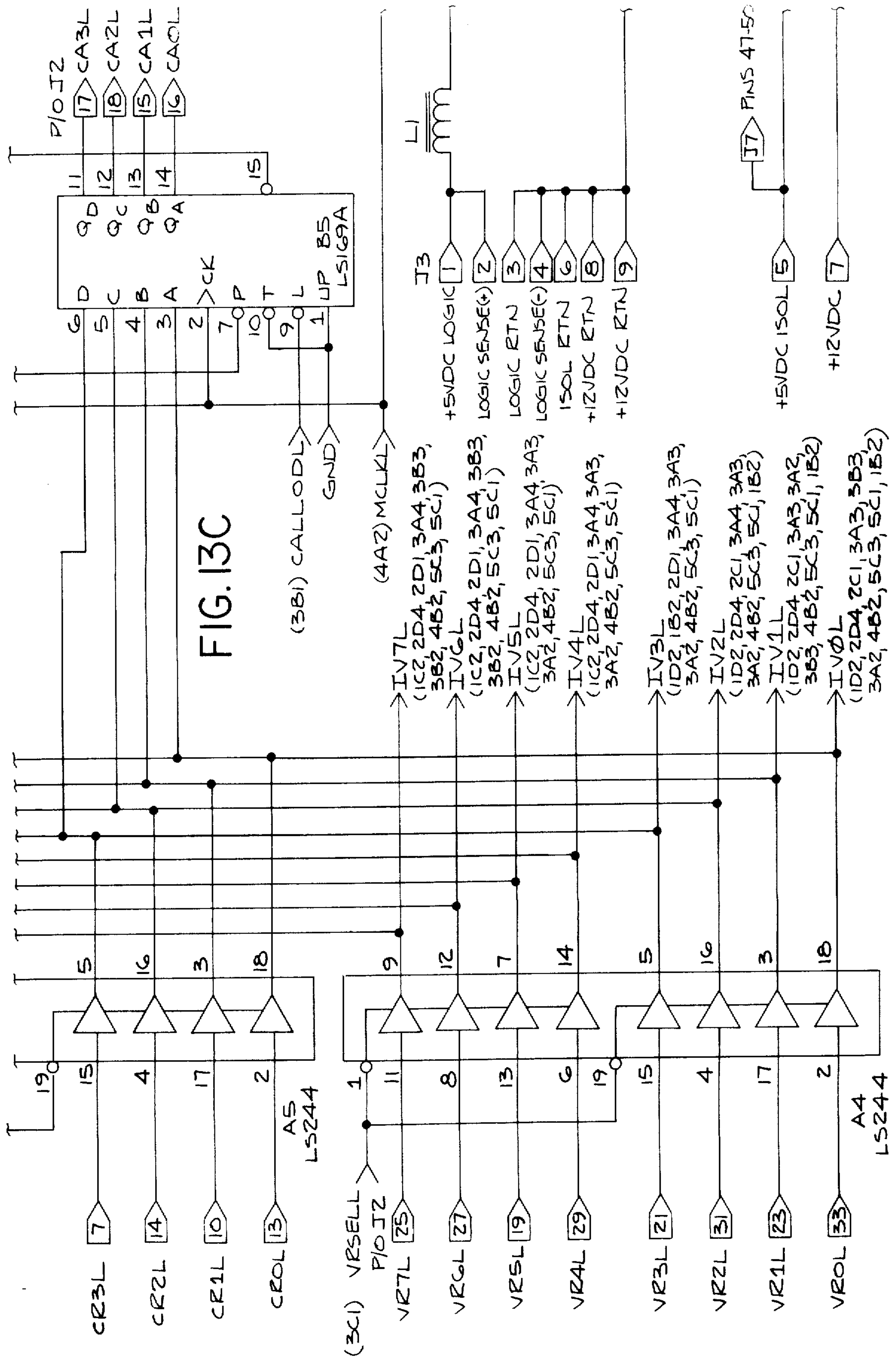


FIG. 13B





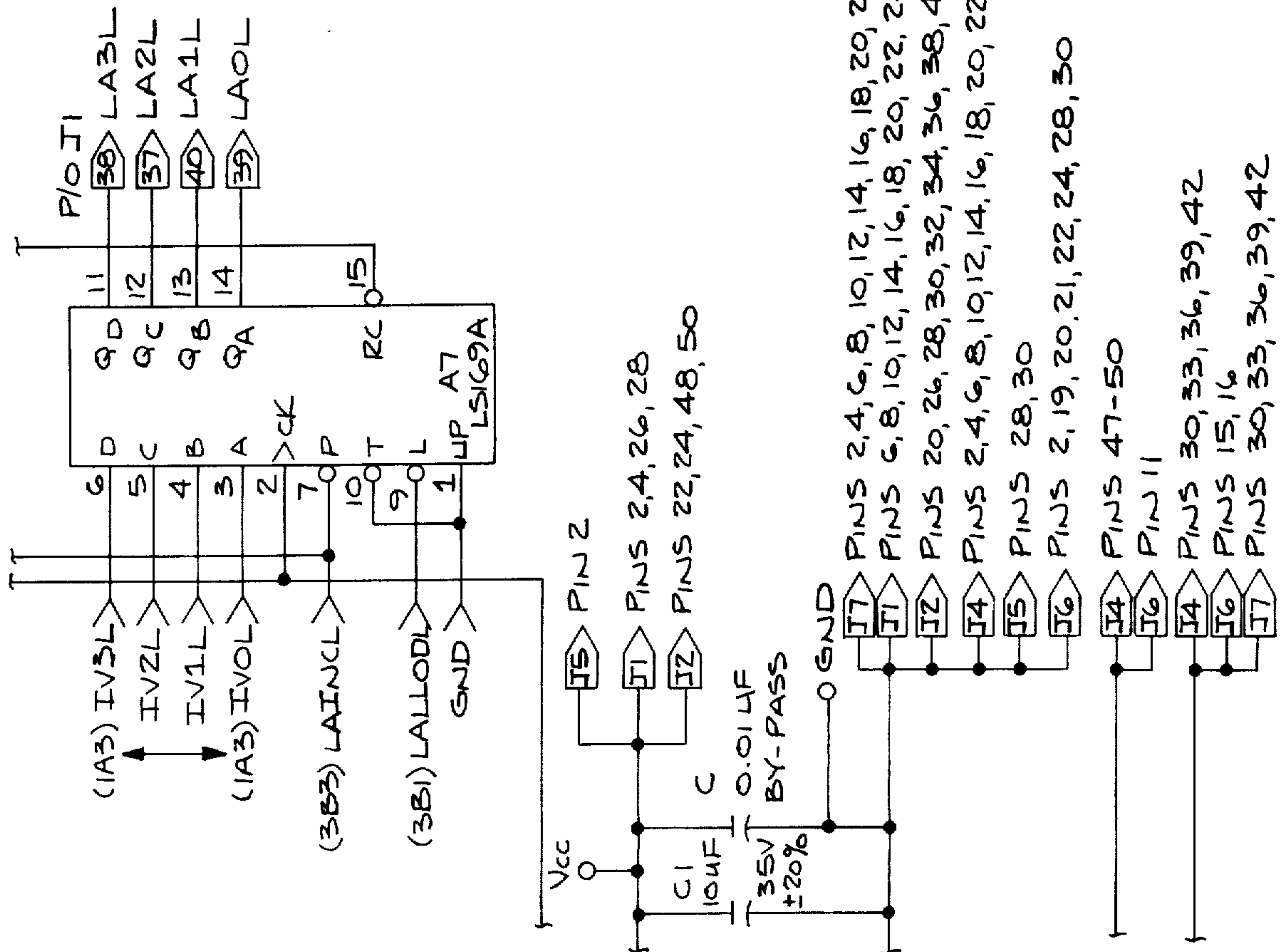


FIG. 13D

FIG. 13A	FIG. 13B
FIG. 13C	FIG. 13D

FIG. 13E

FIG. 14A	FIG. 14B
FIG. 14C	FIG. 14D

FIG. 14E

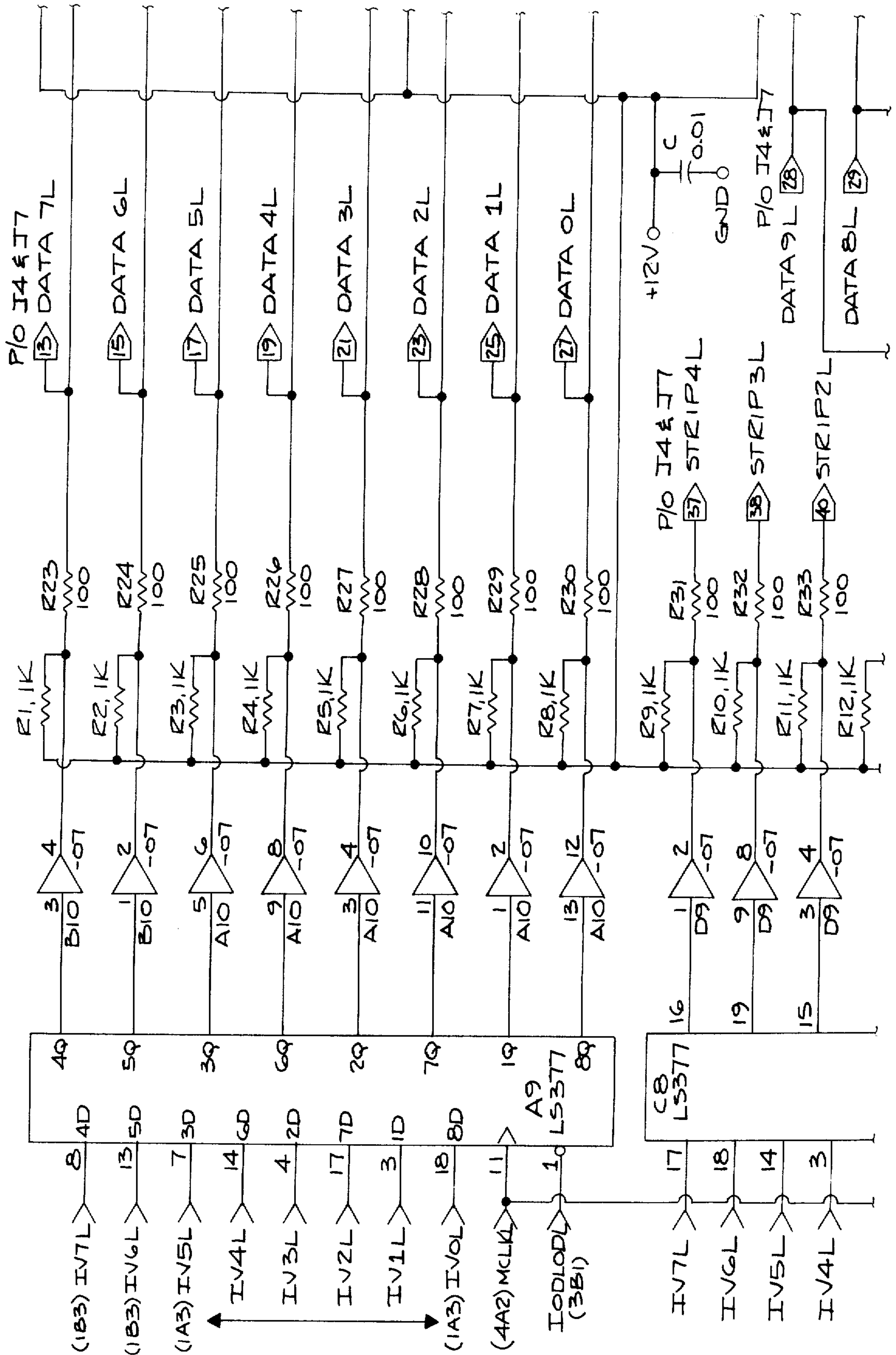
FIG. 15A	FIG. 15B
FIG. 15C	FIG. 15D

FIG. 15E

FIG. 16A	FIG. 16B
FIG. 16C	FIG. 16D

FIG. 16E

FIG. 14A



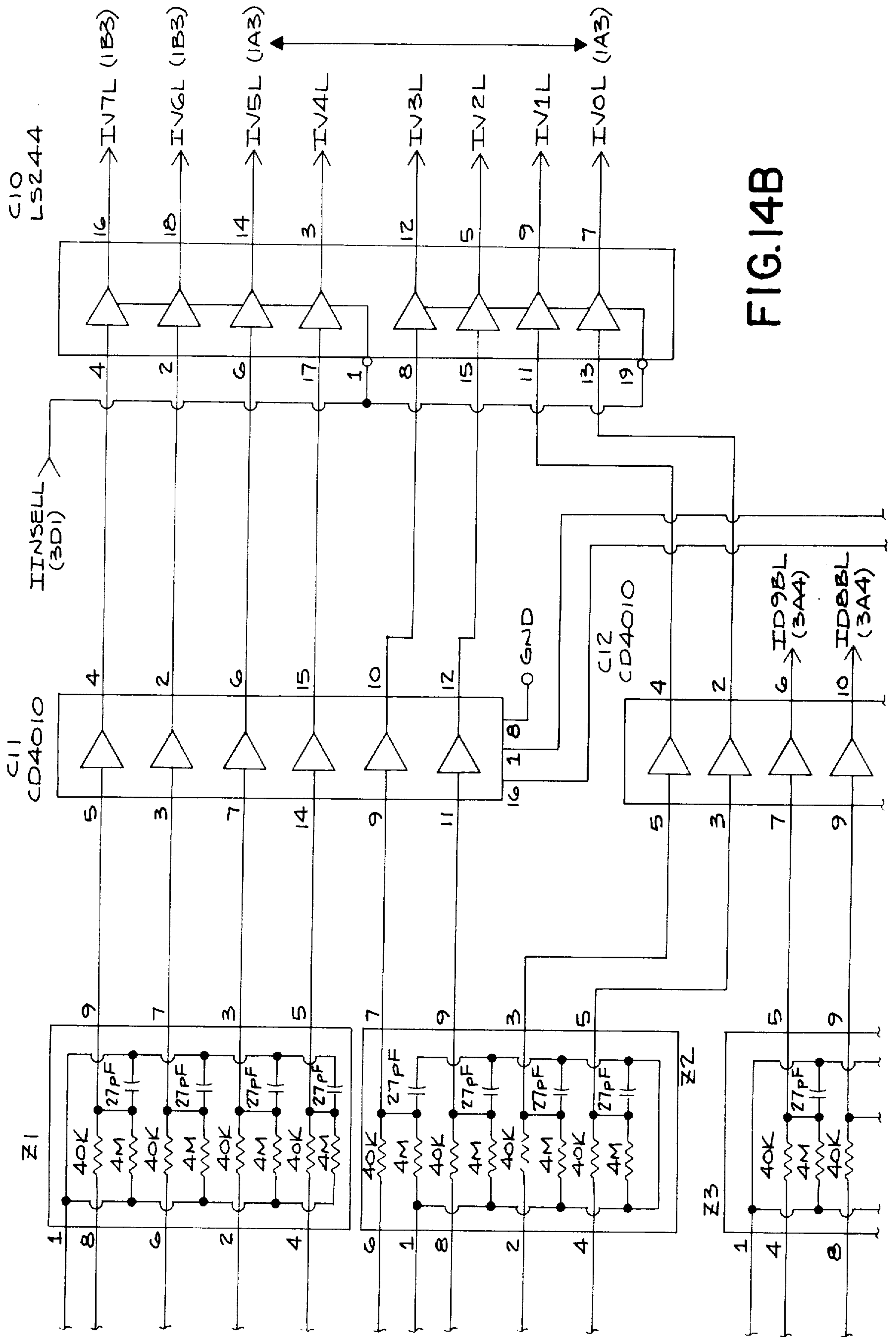


FIG. 14B

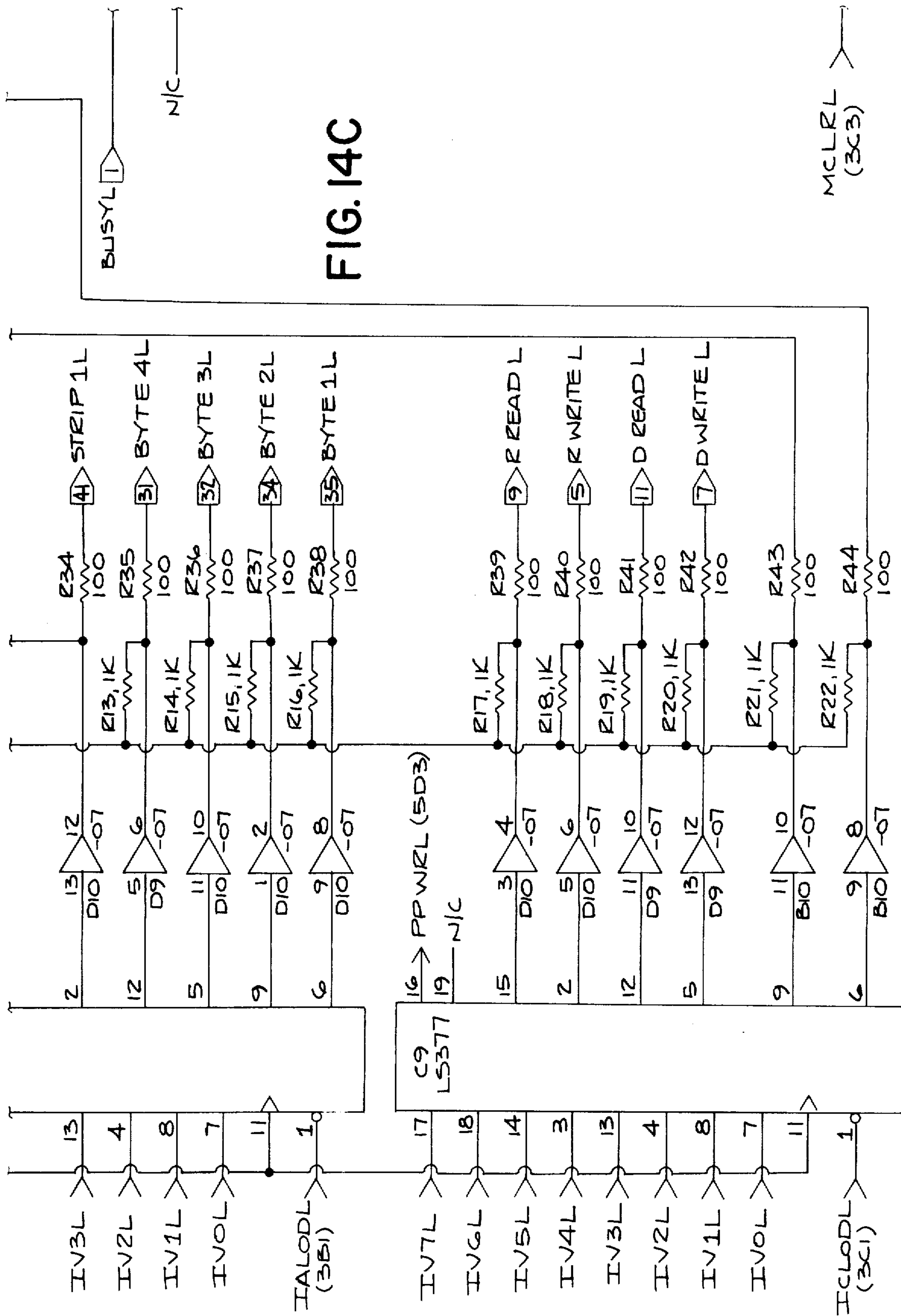


FIG. 14C

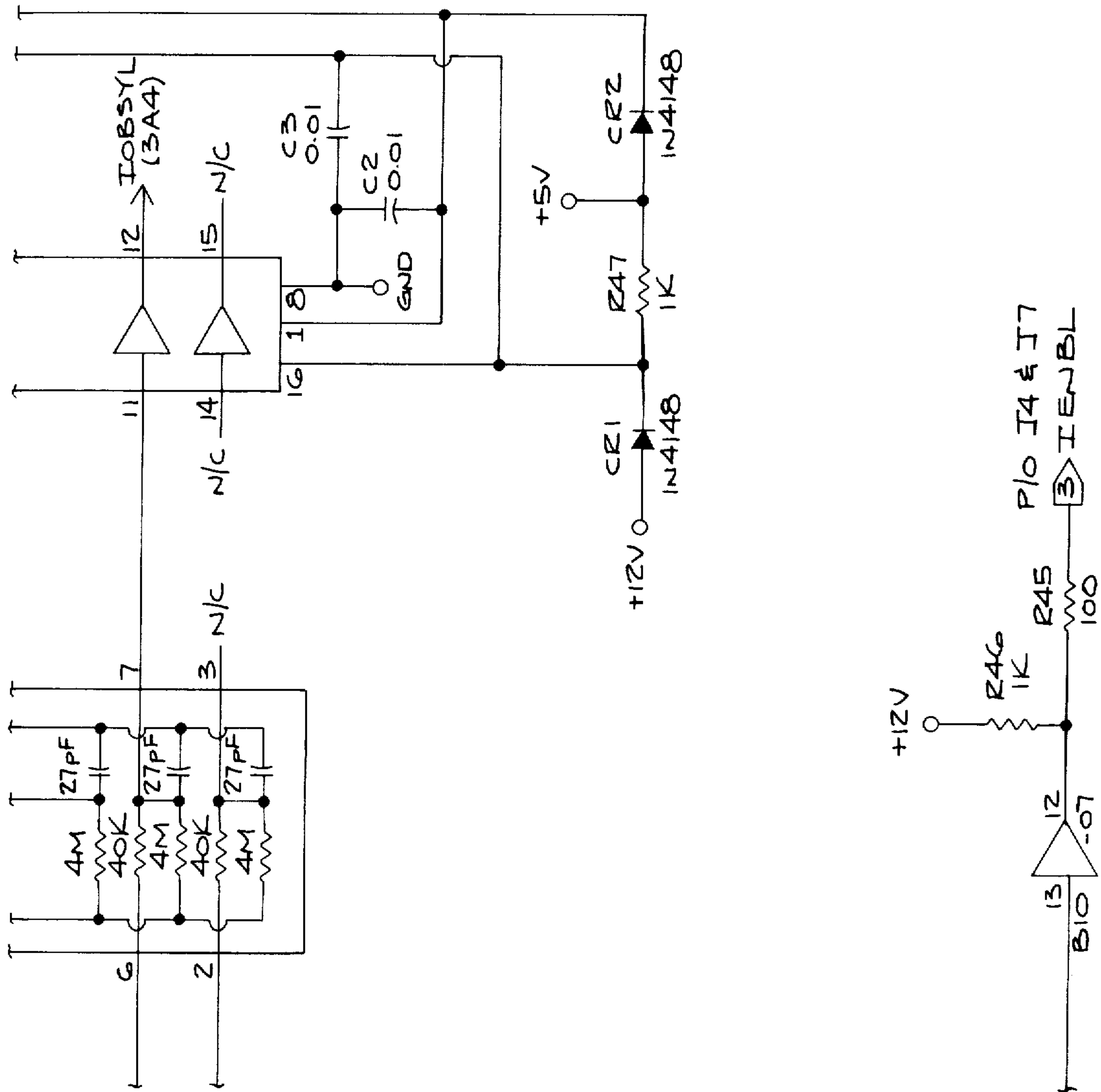


FIG. 14D

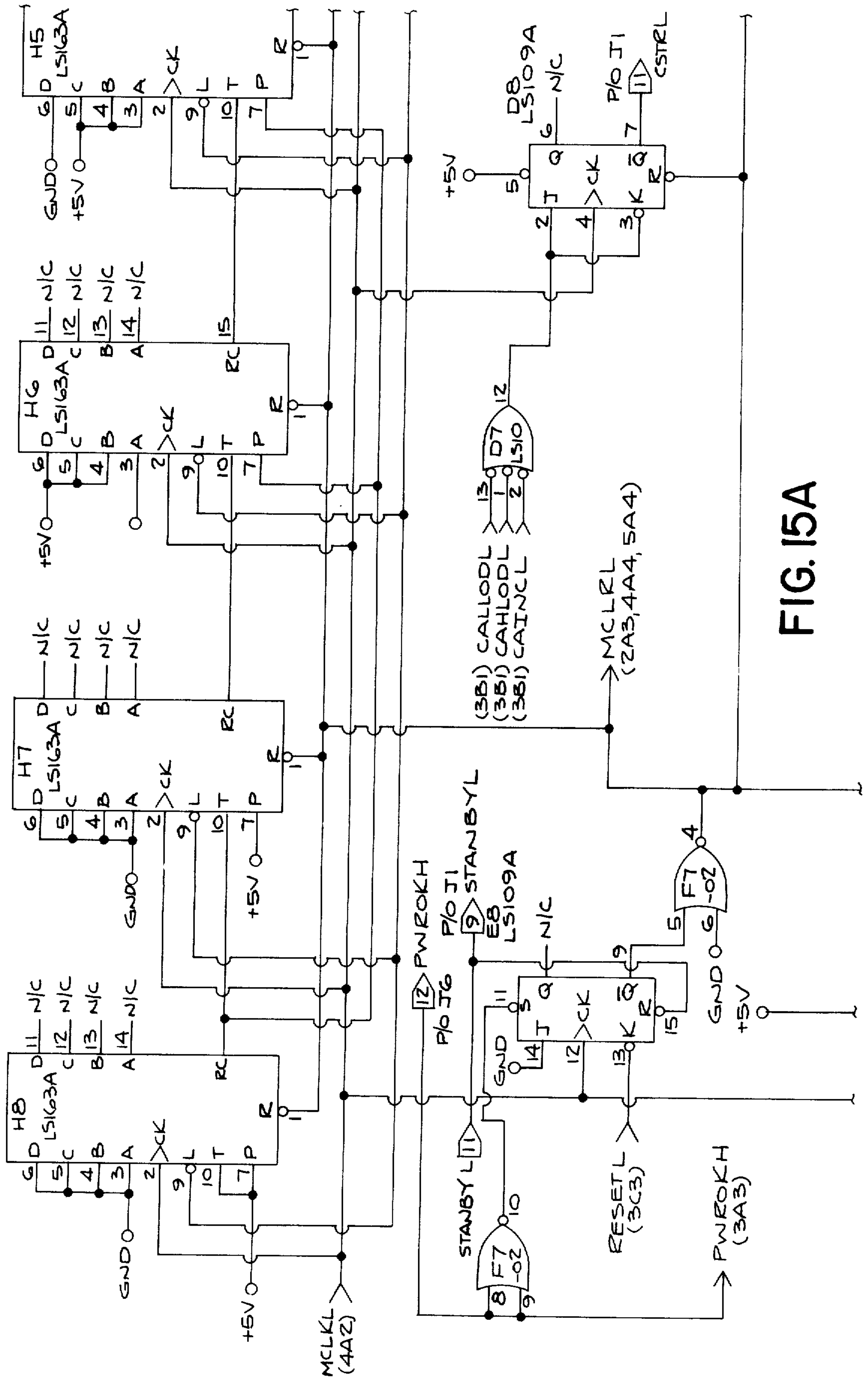
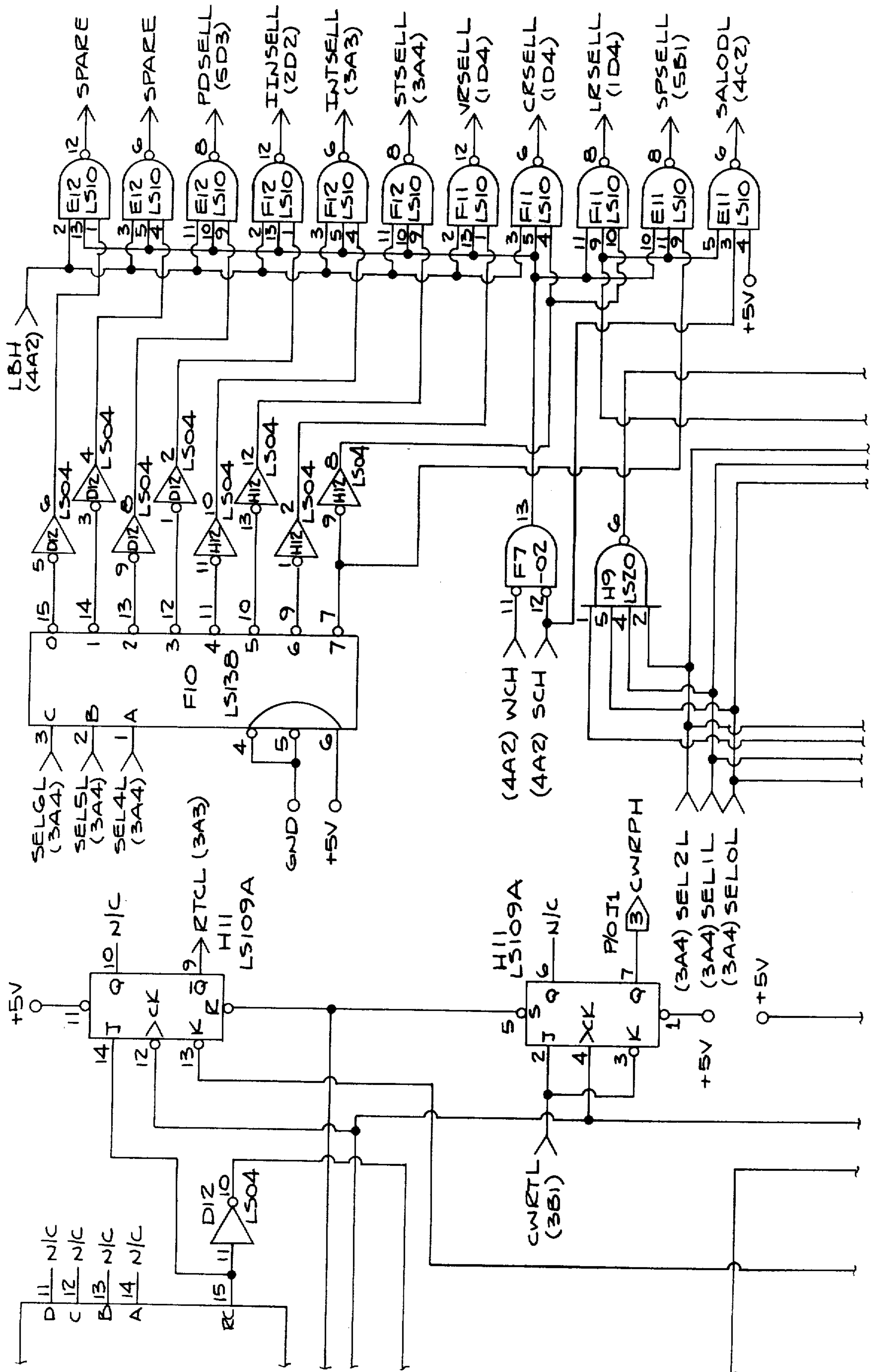
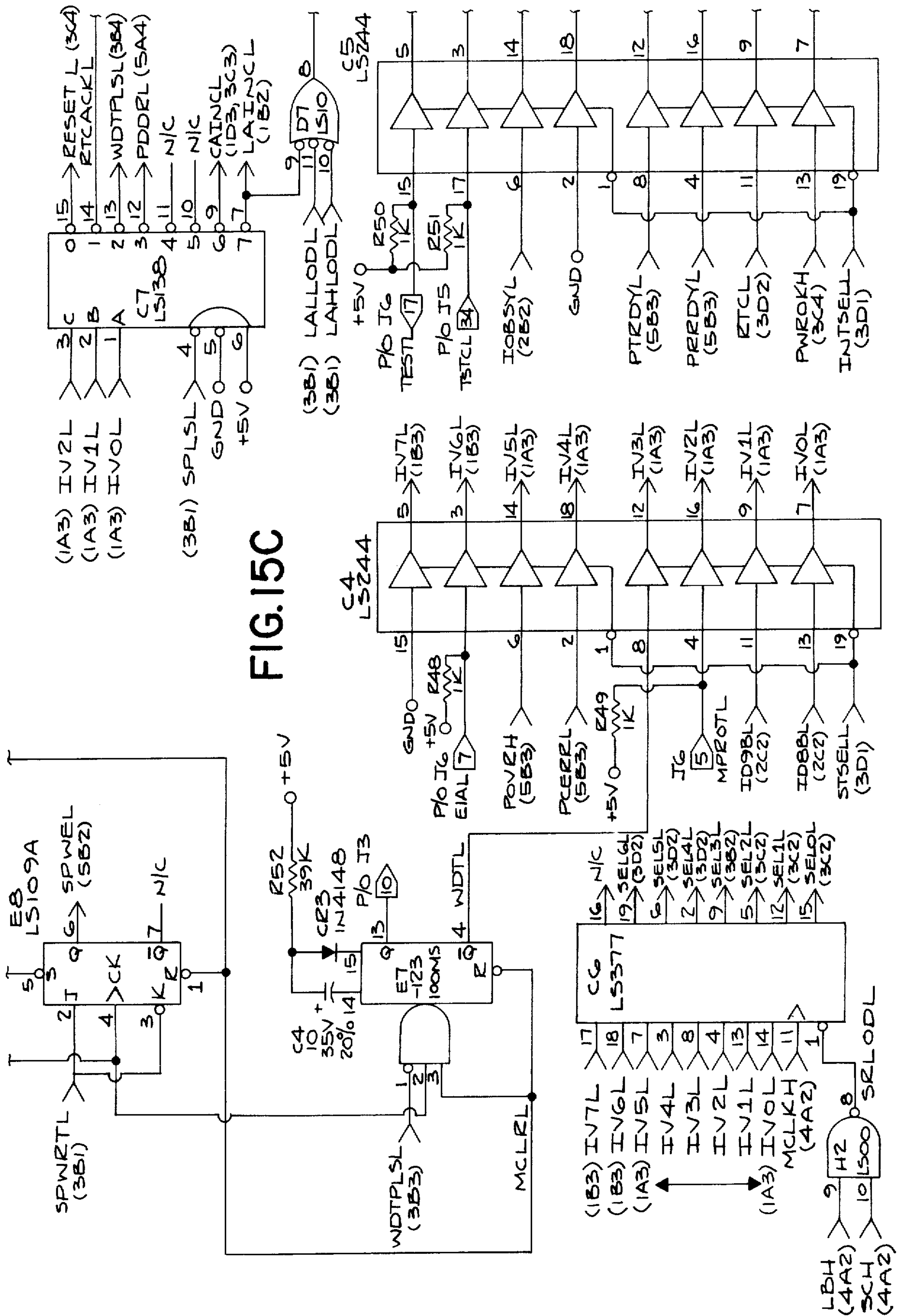


FIG. 15A

FIG. 15B





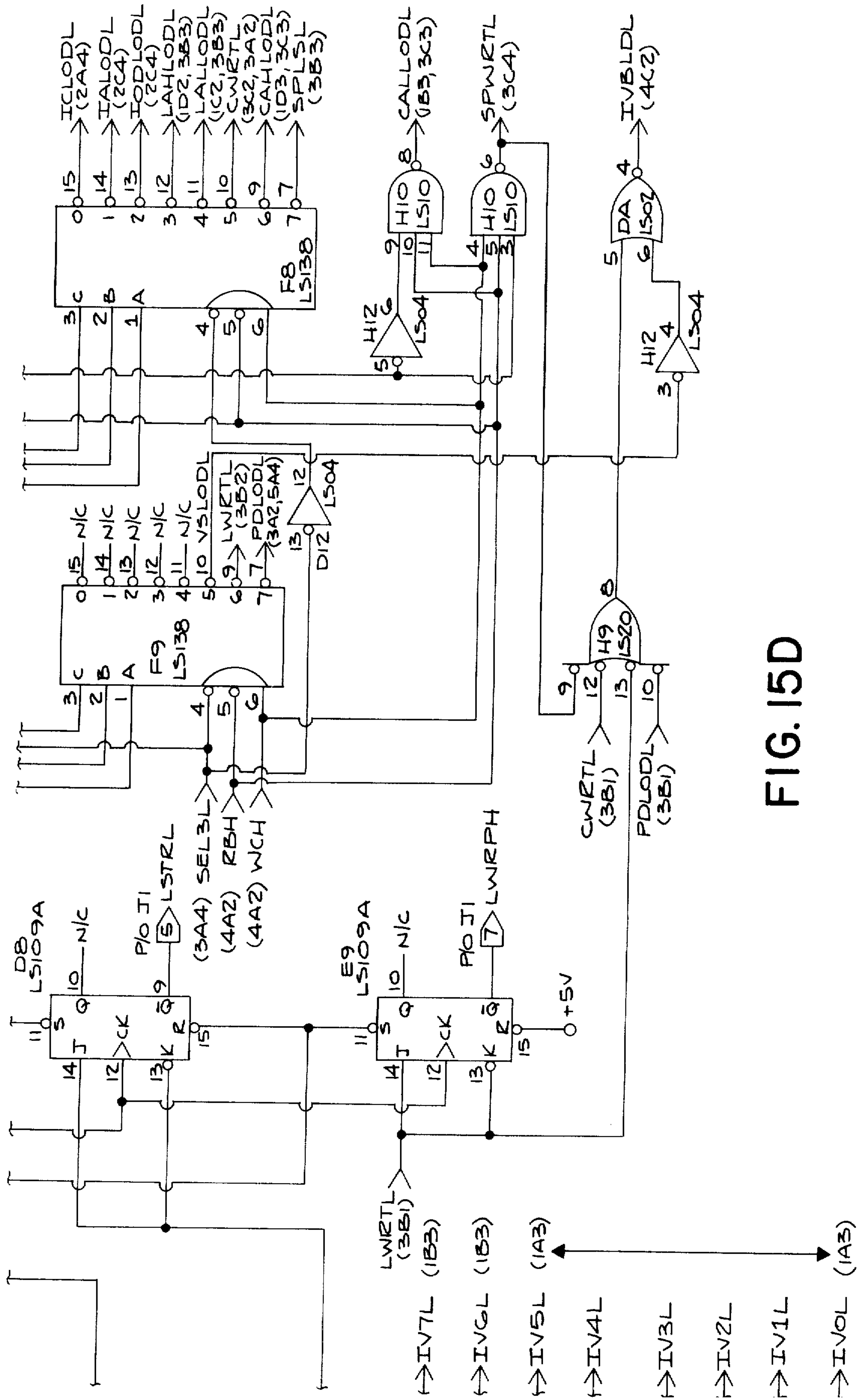


FIG. 15D

FIG. 16A

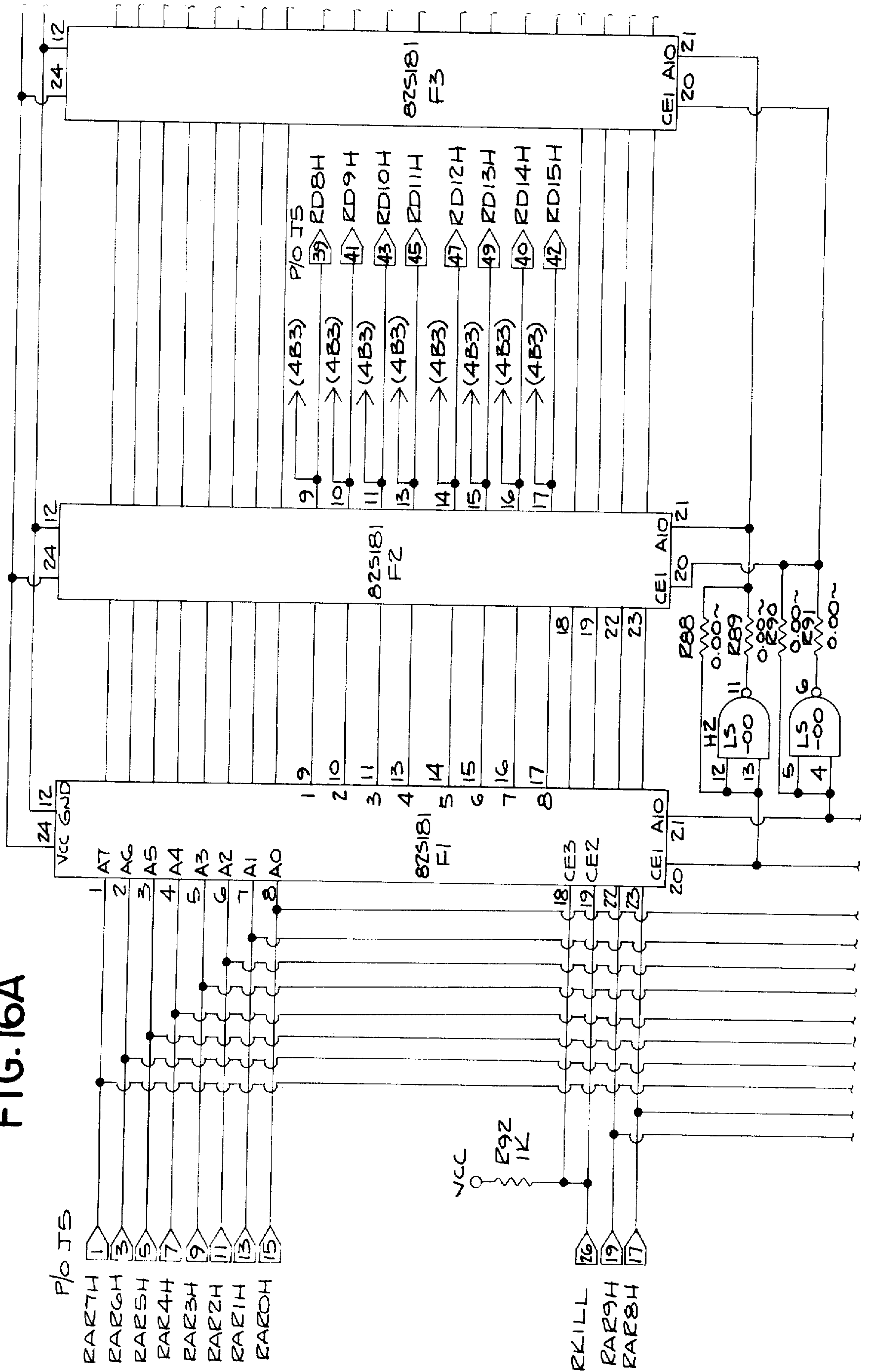
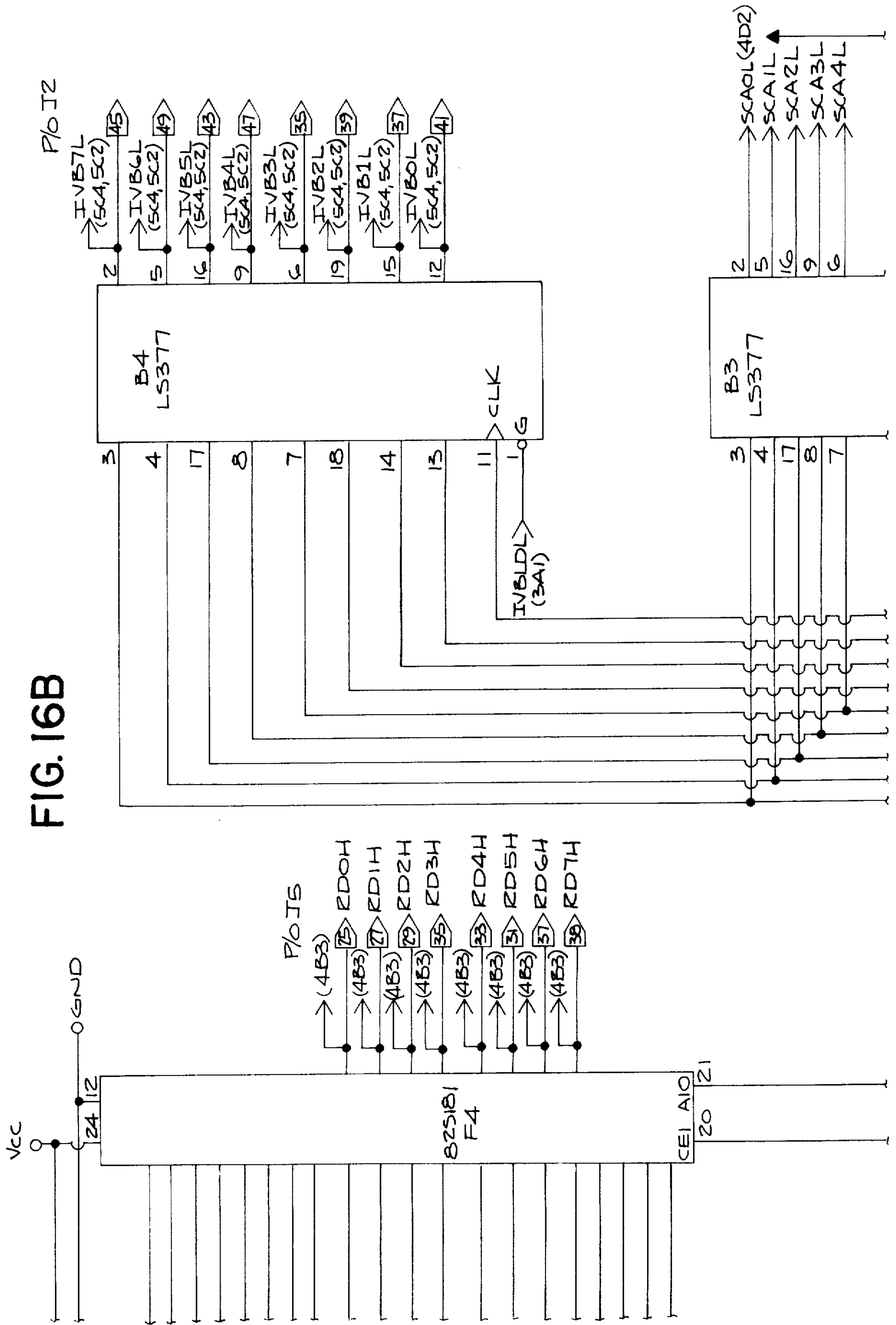
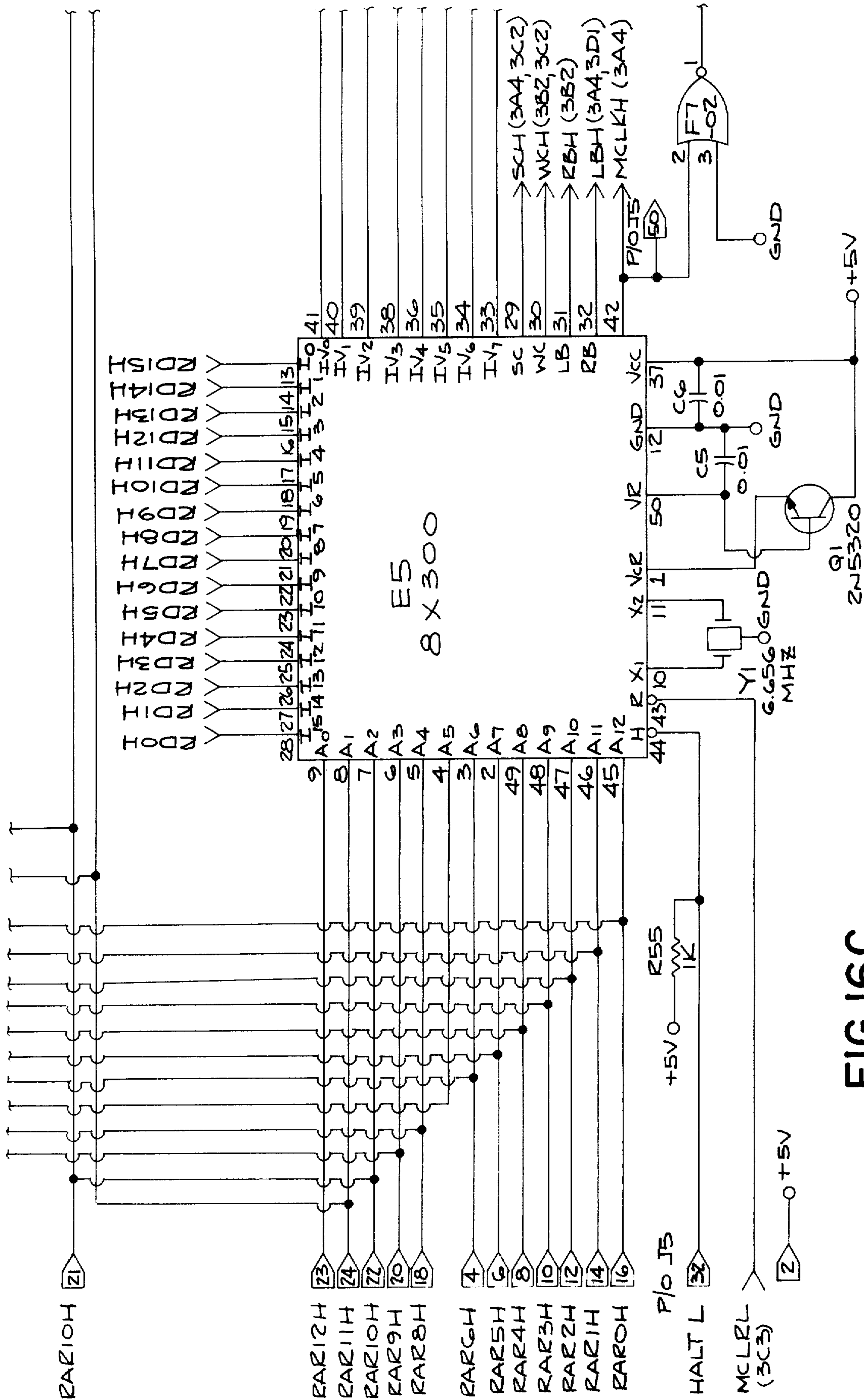


FIG. 16B





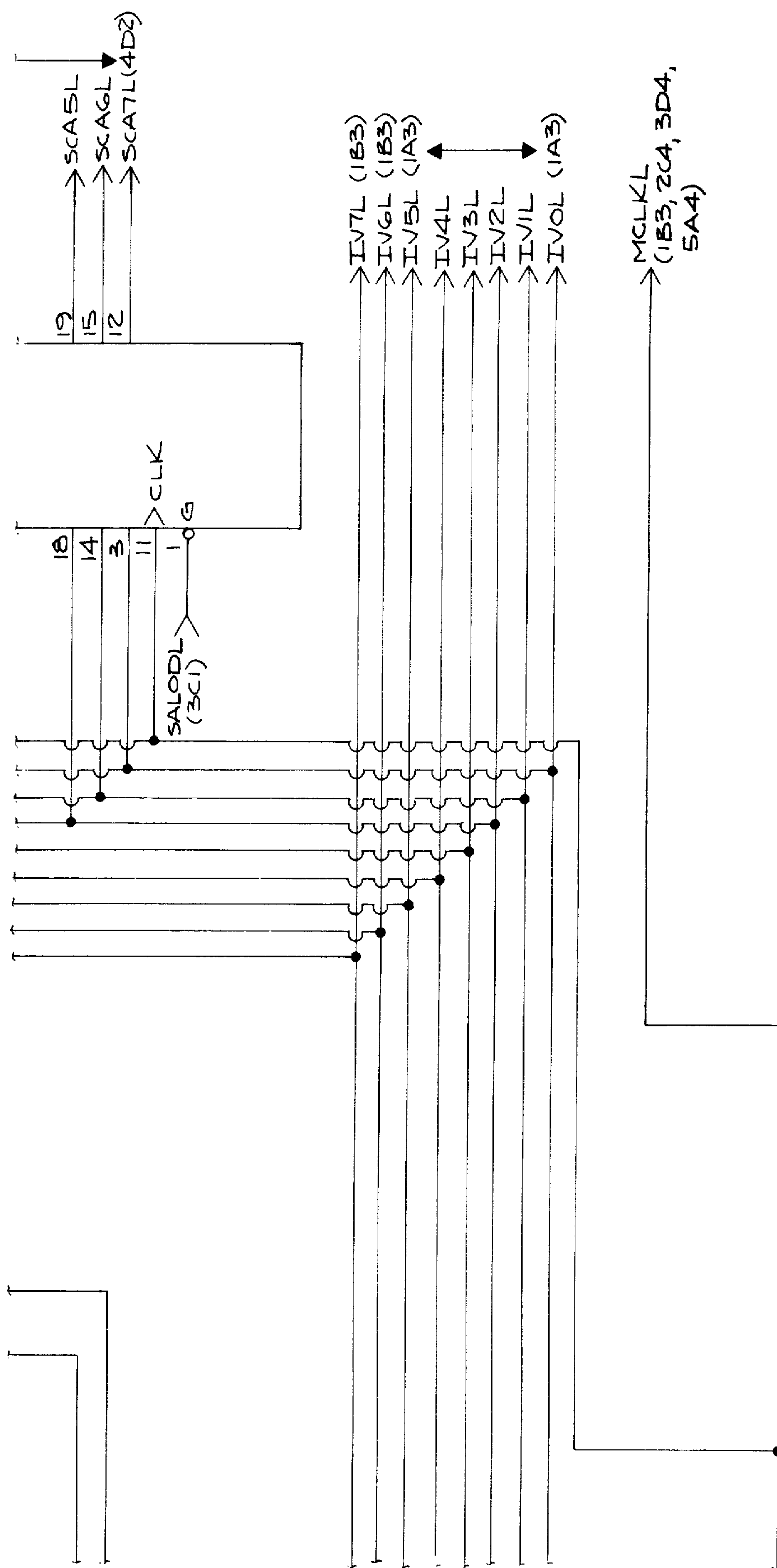


FIG. 16D

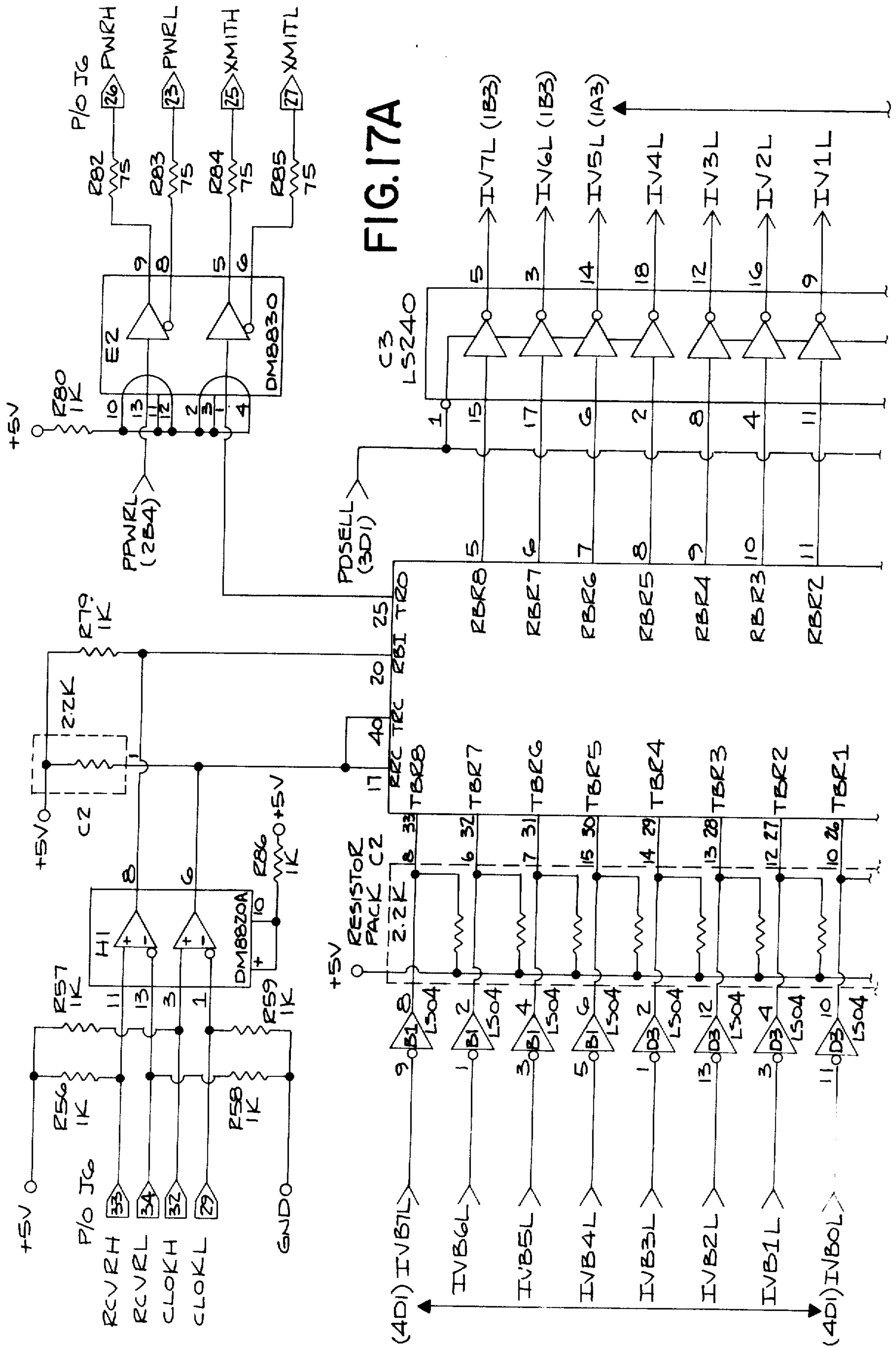
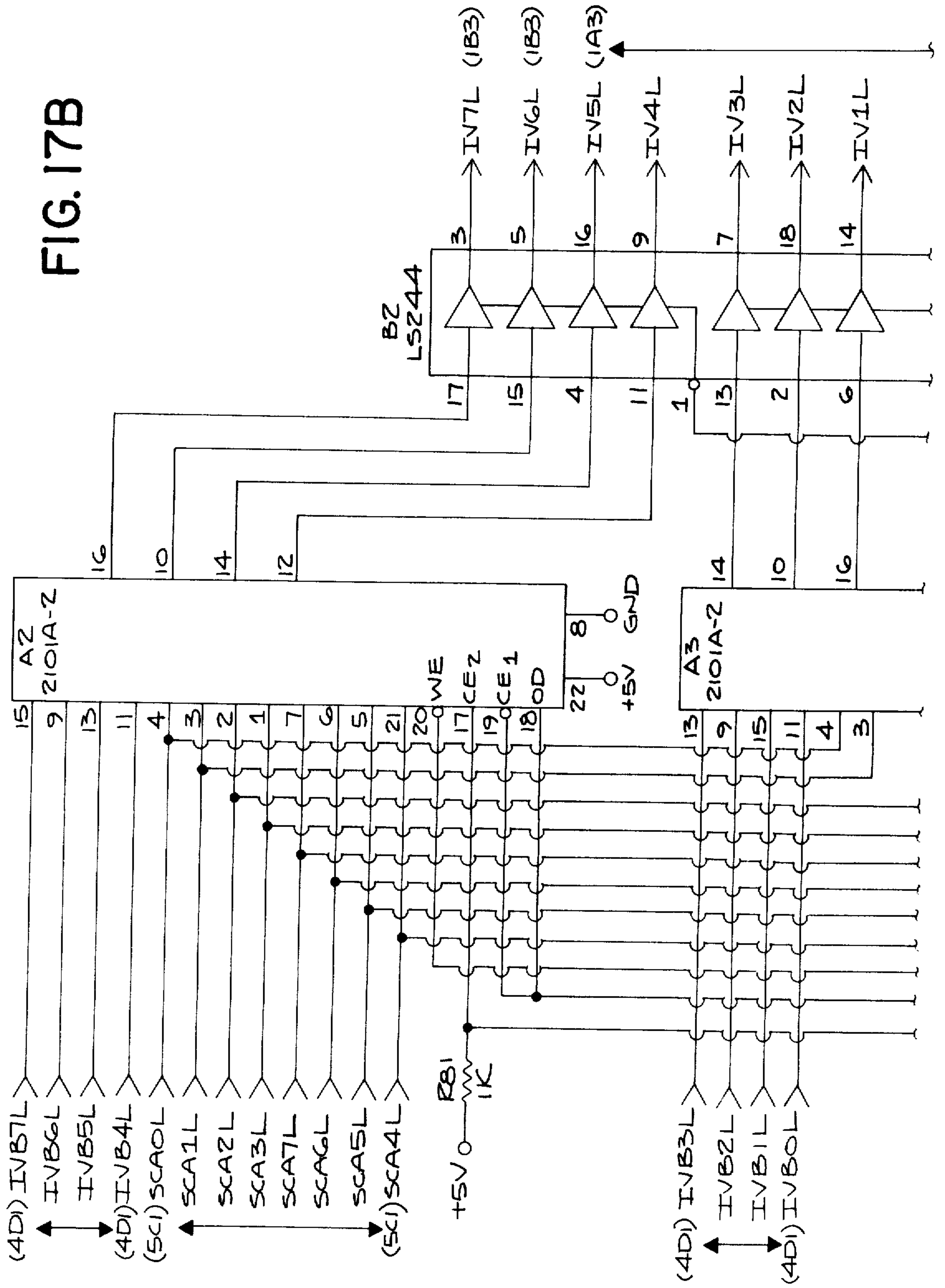


FIG. 17B



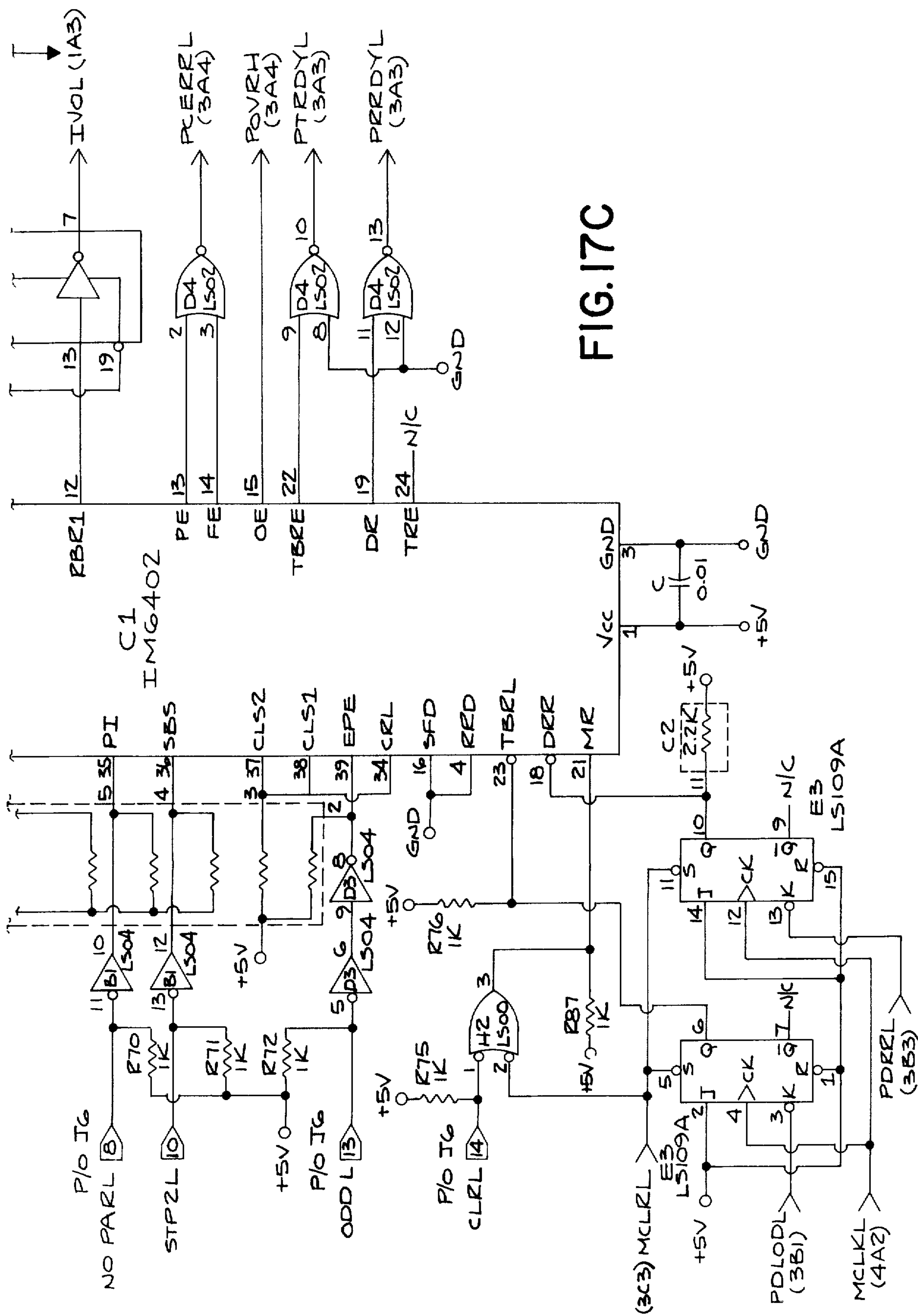


FIG. 17C

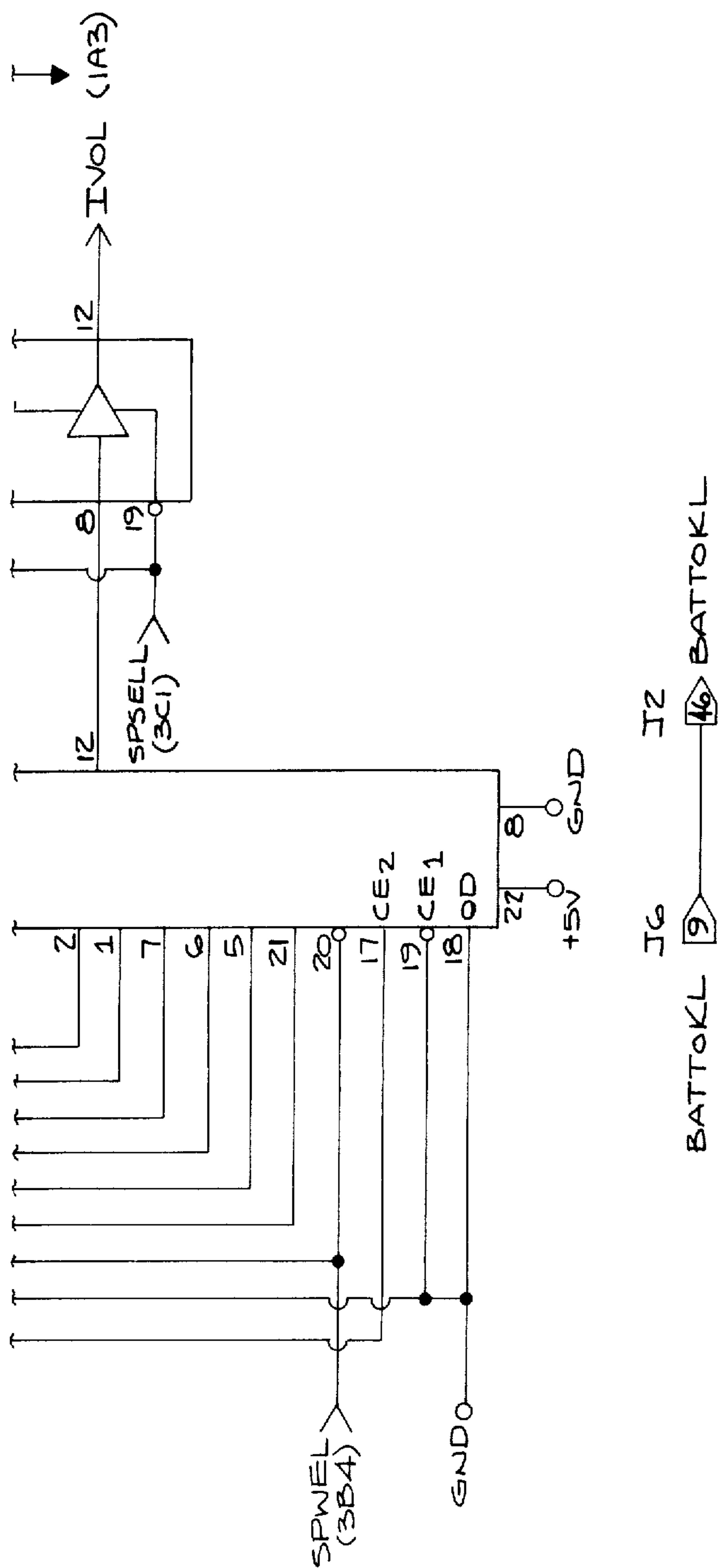


FIG. 17D

FIG. 17A	FIG. 17B
FIG. 17C	FIG. 17D

FIG. 17E

FIG. 18A	FIG. 18B
FIG. 18C	FIG. 18D

FIG. 18E

FIG. 19A	FIG. 19B
FIG. 19C	FIG. 19D

FIG. 19E

FIG. 20A	FIG. 20B
FIG. 20C	FIG. 20D

FIG. 20E

FIG. 18A

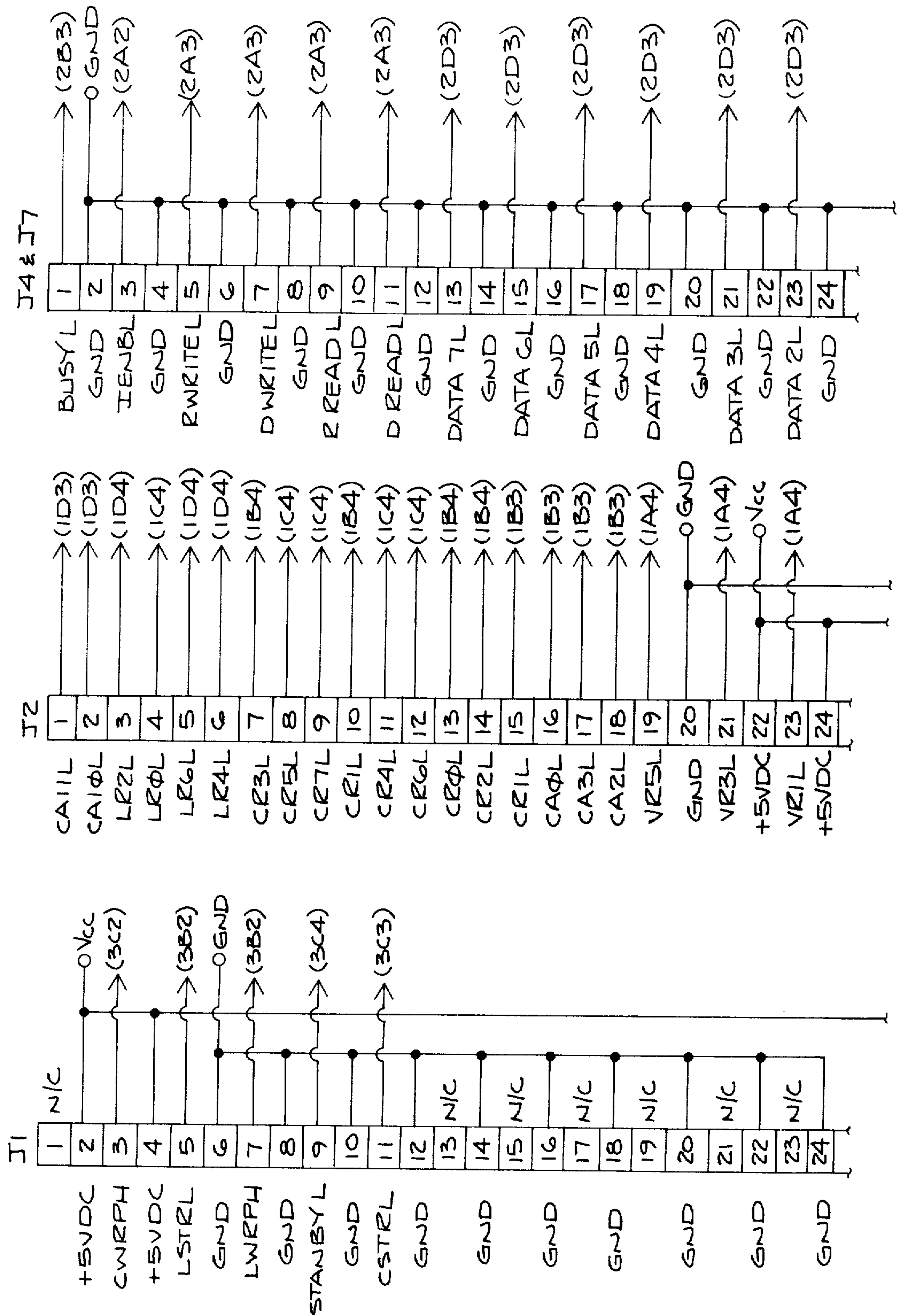
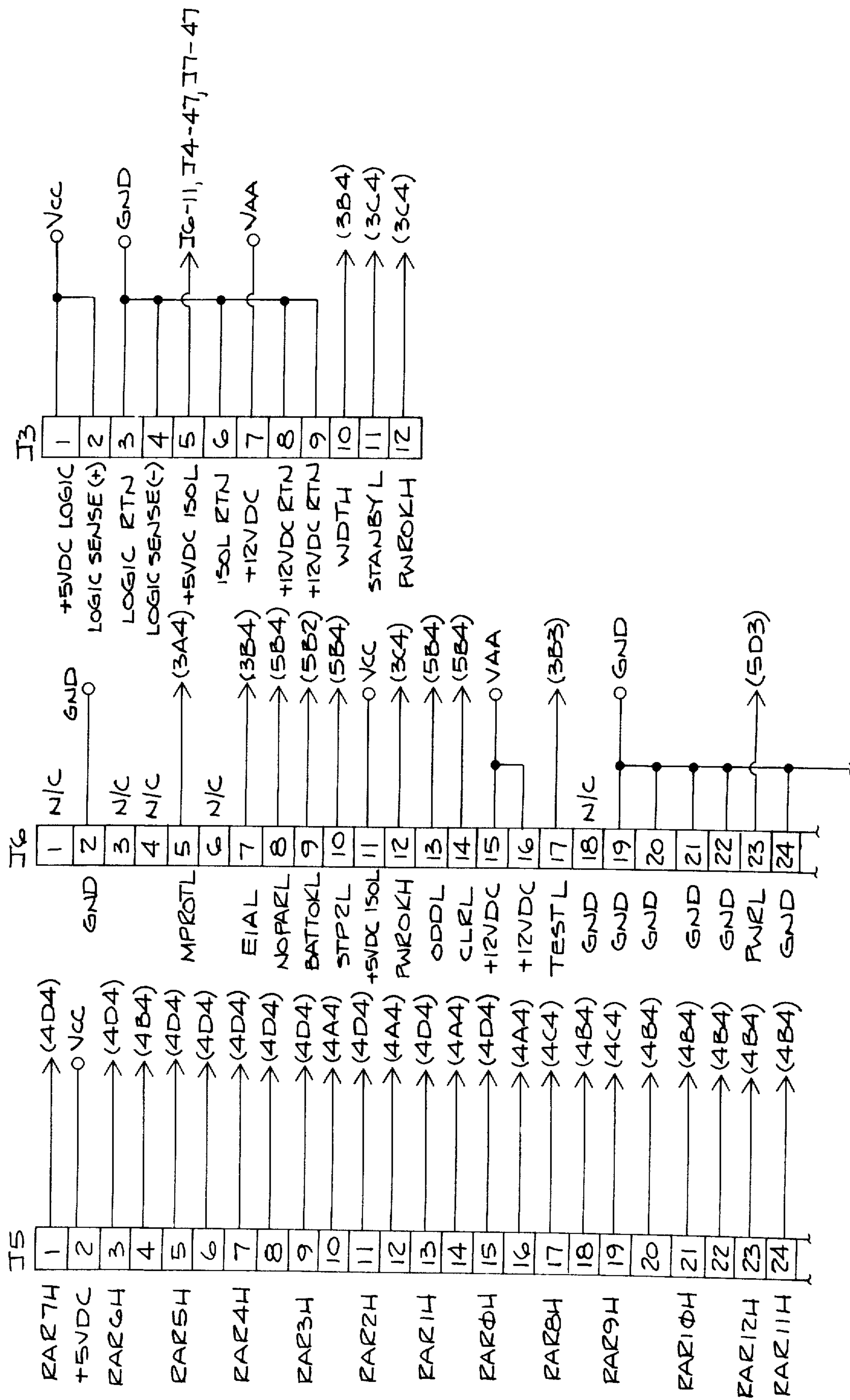


FIG. 18B



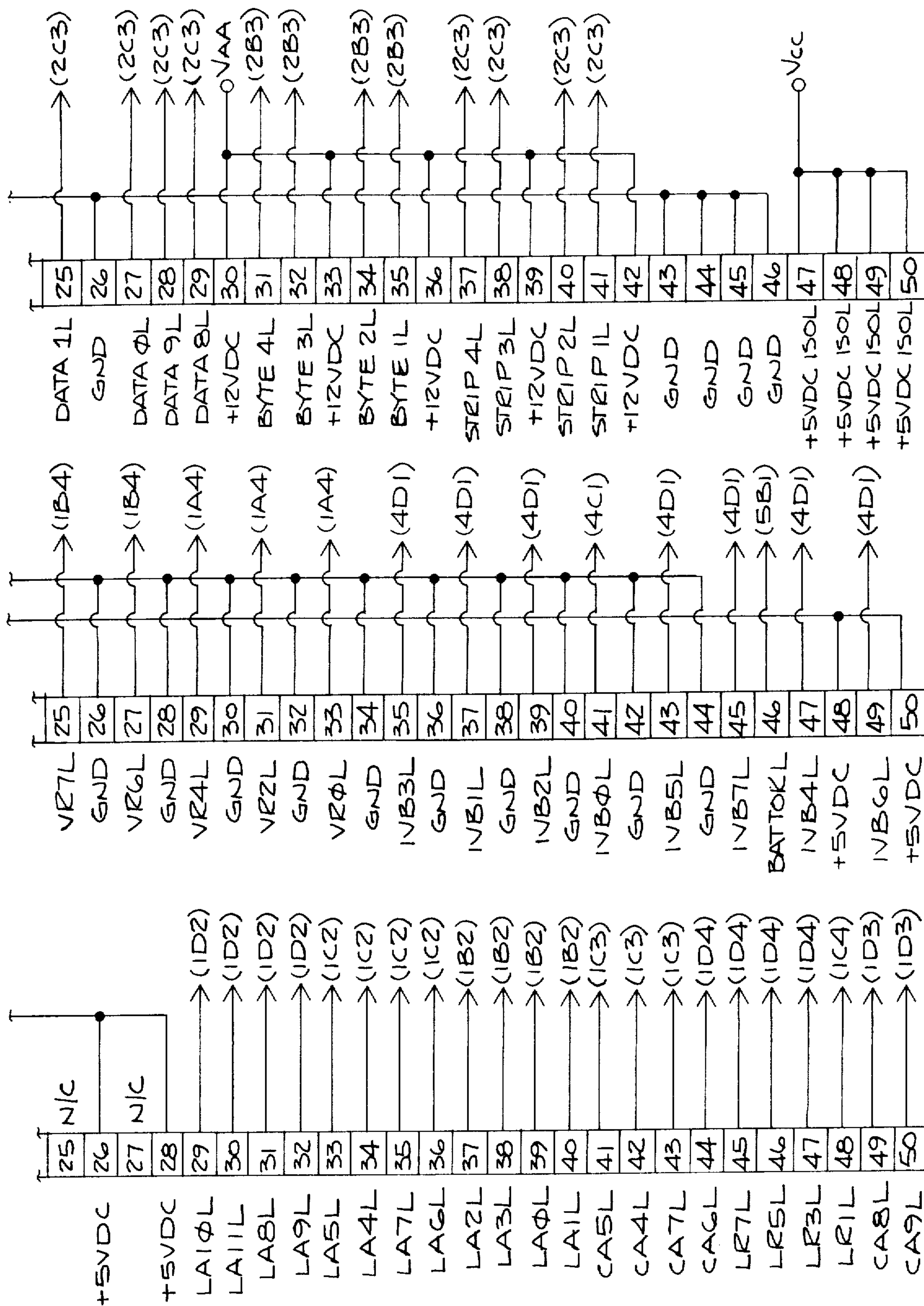


FIG. 18C

FIG. 18D

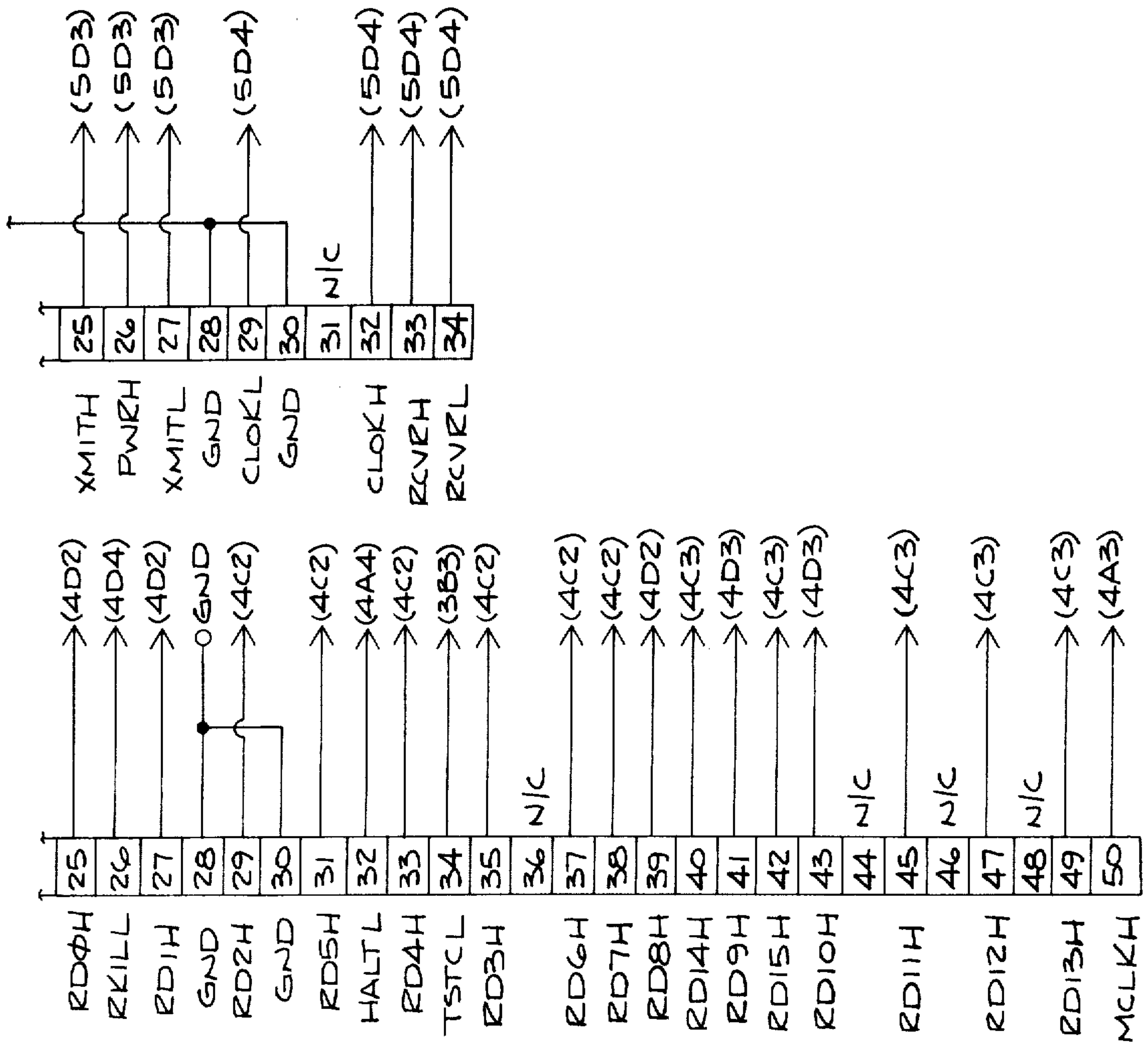


FIG. 19A

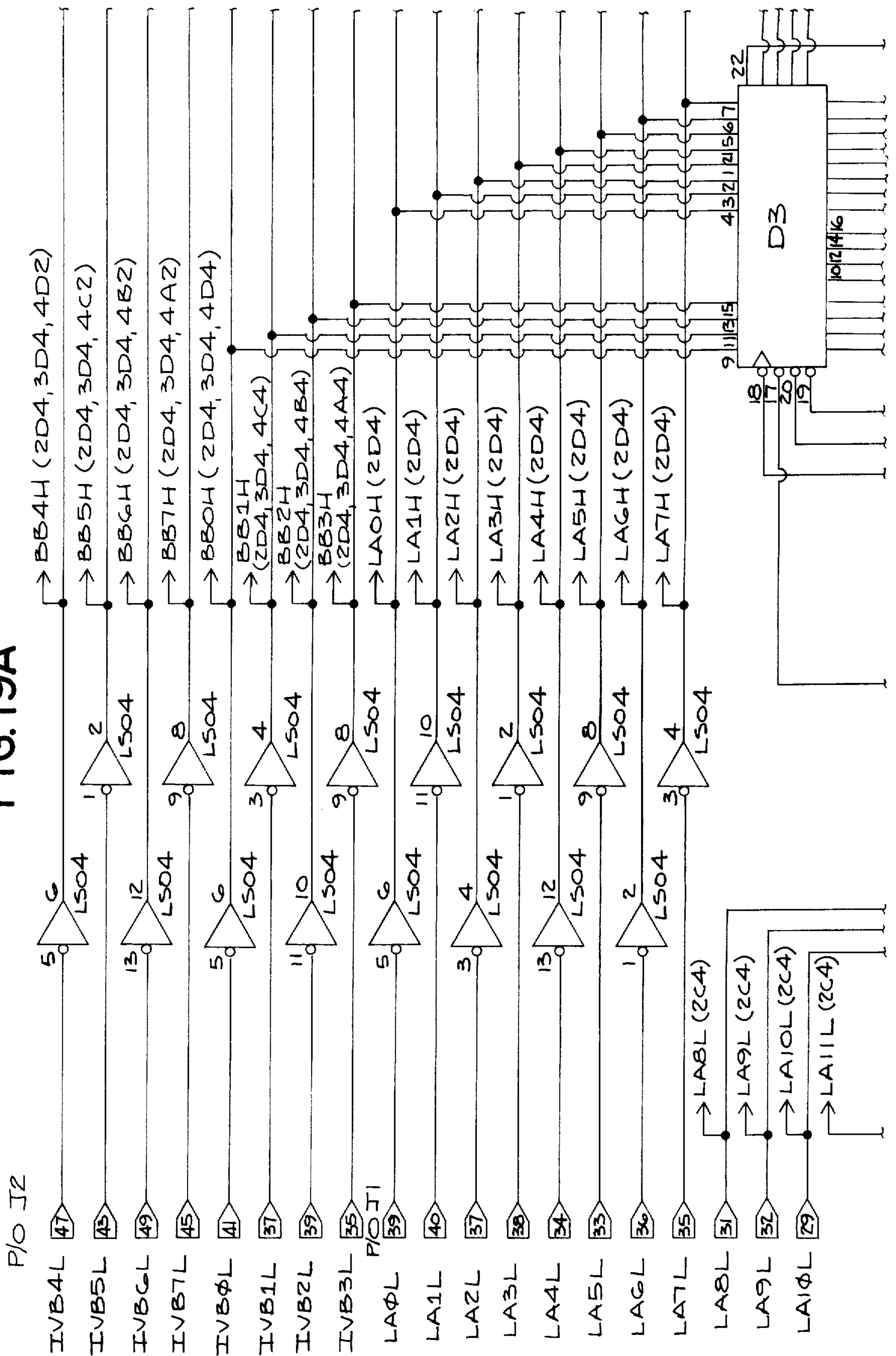
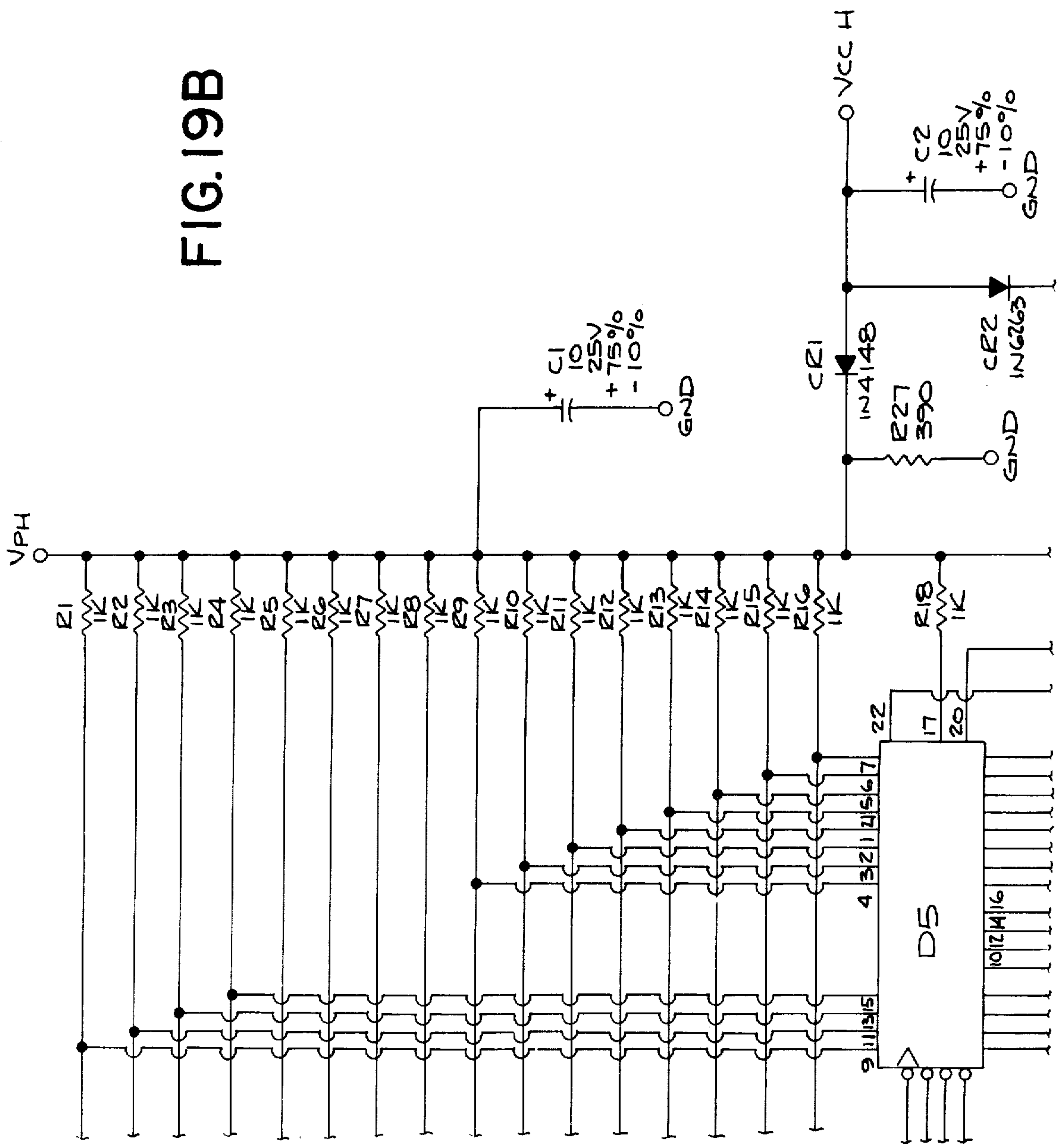


FIG. 19B



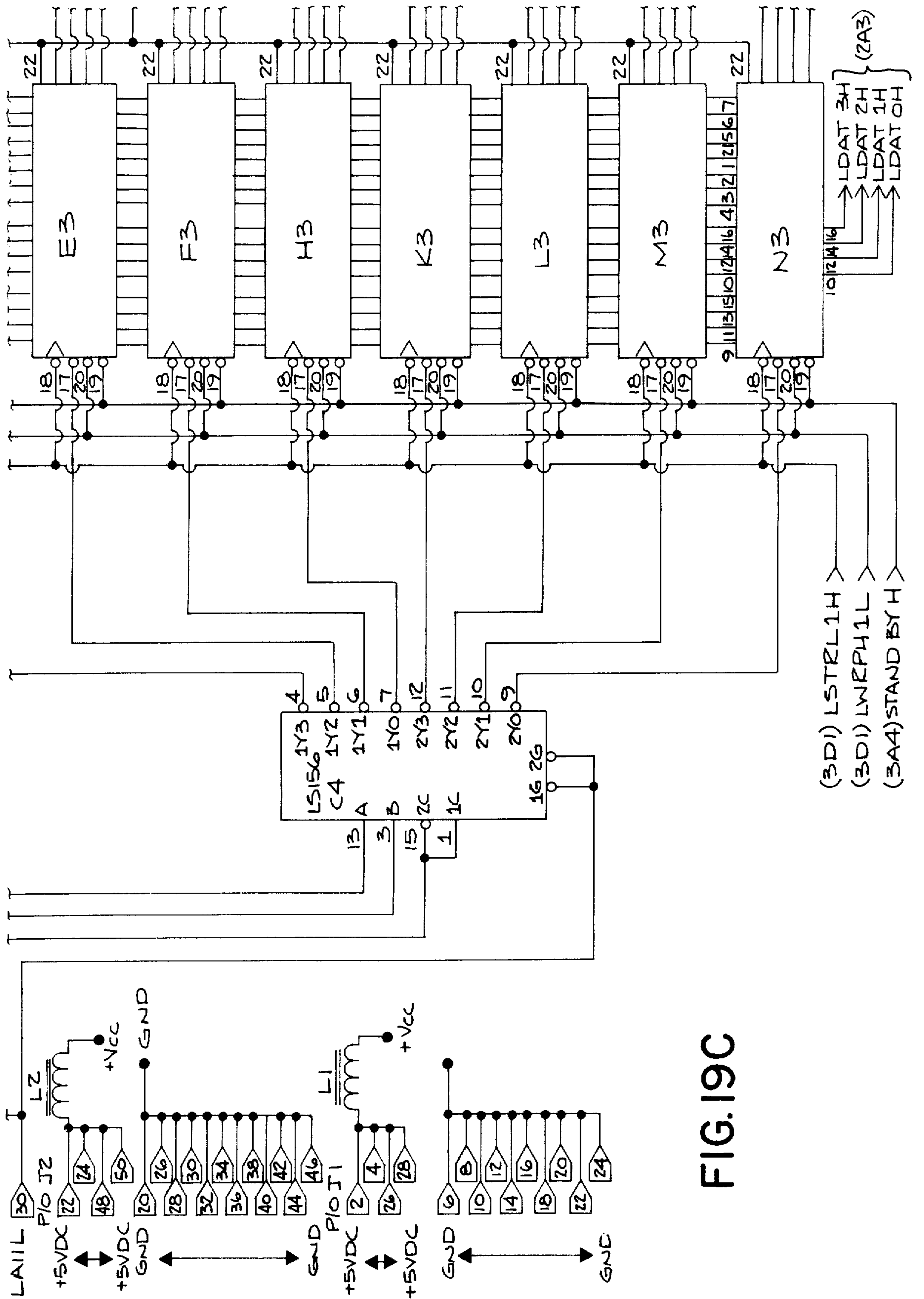


FIG. 19C

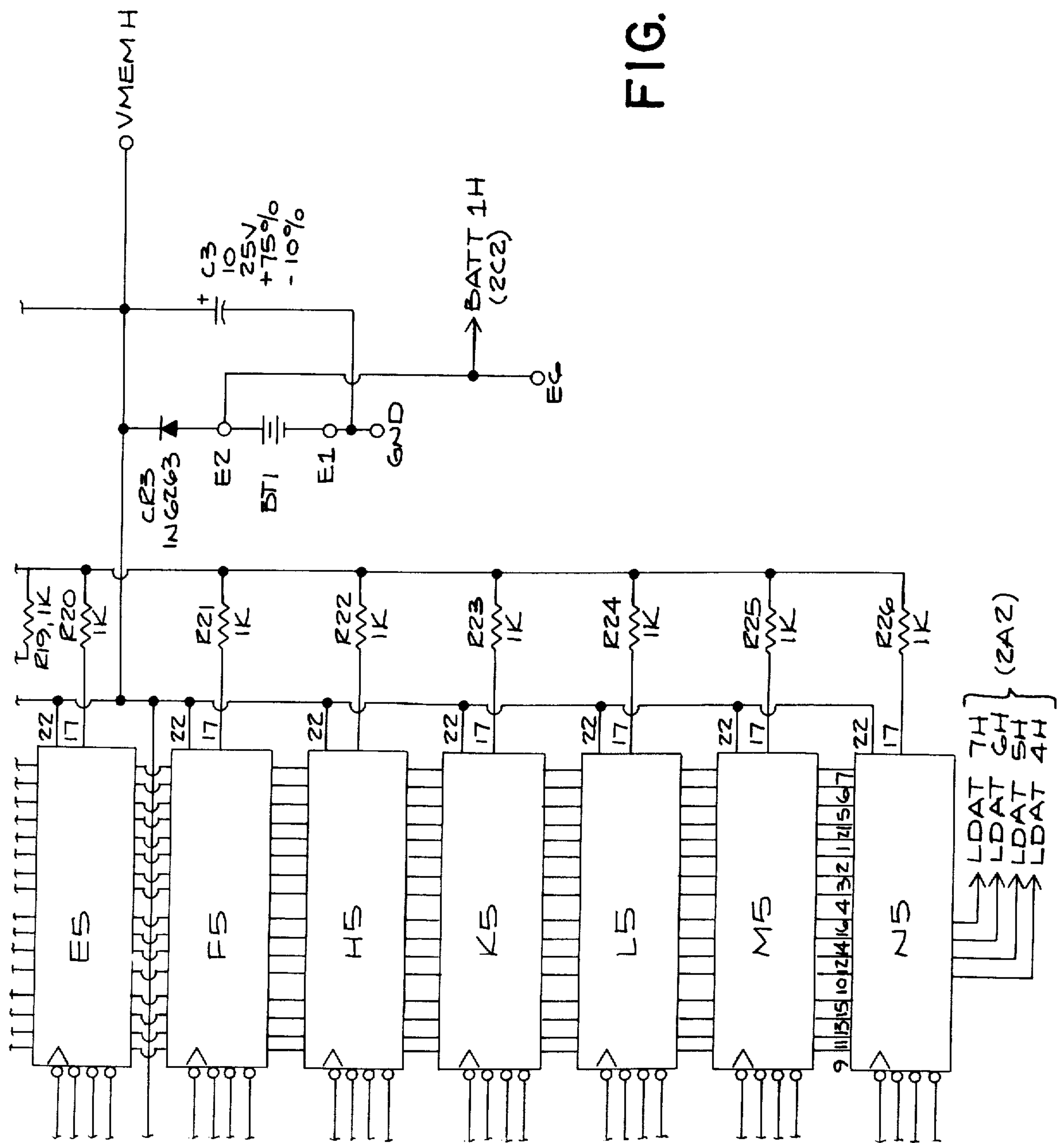


FIG. 19D

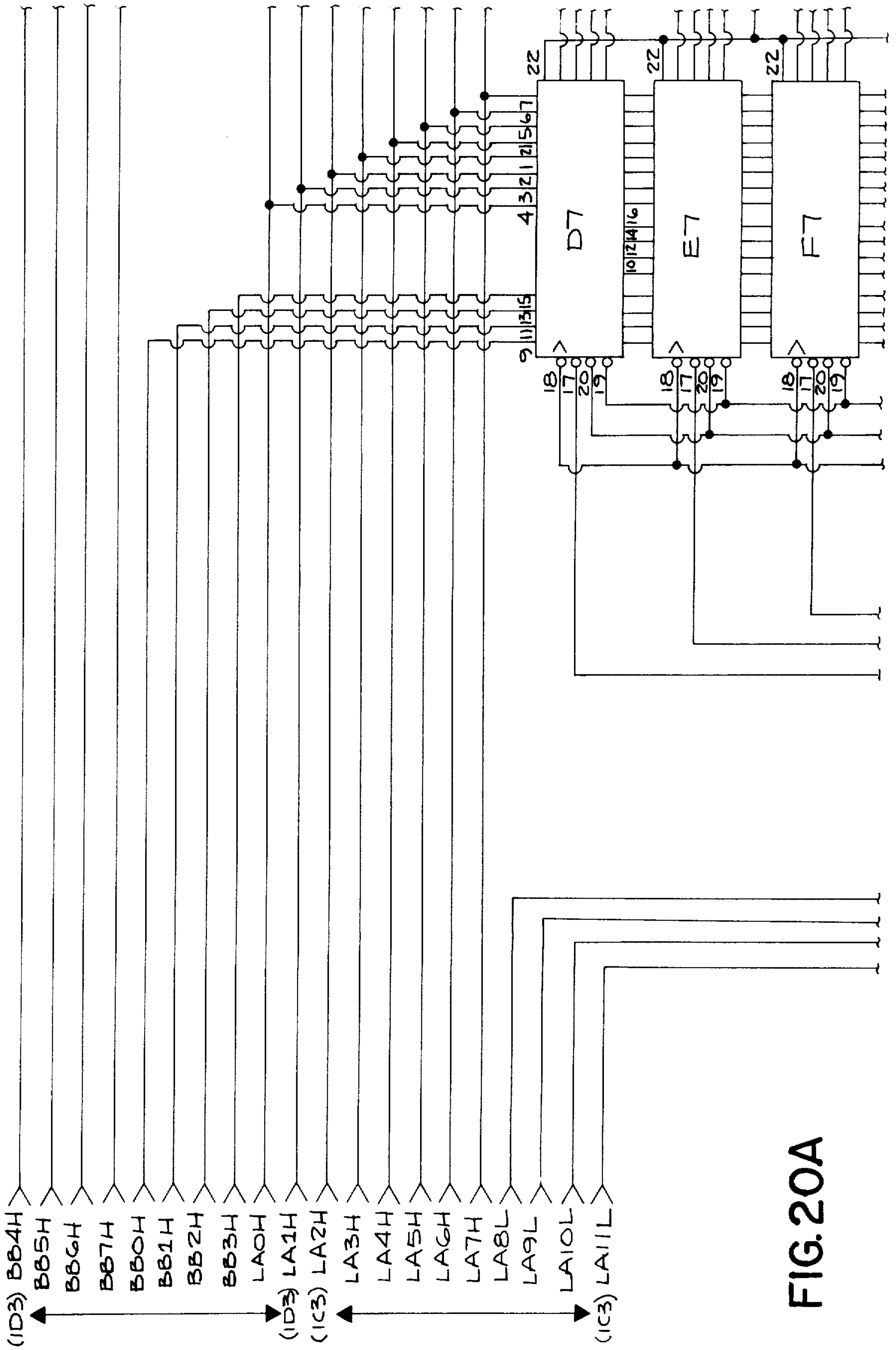


FIG. 20A

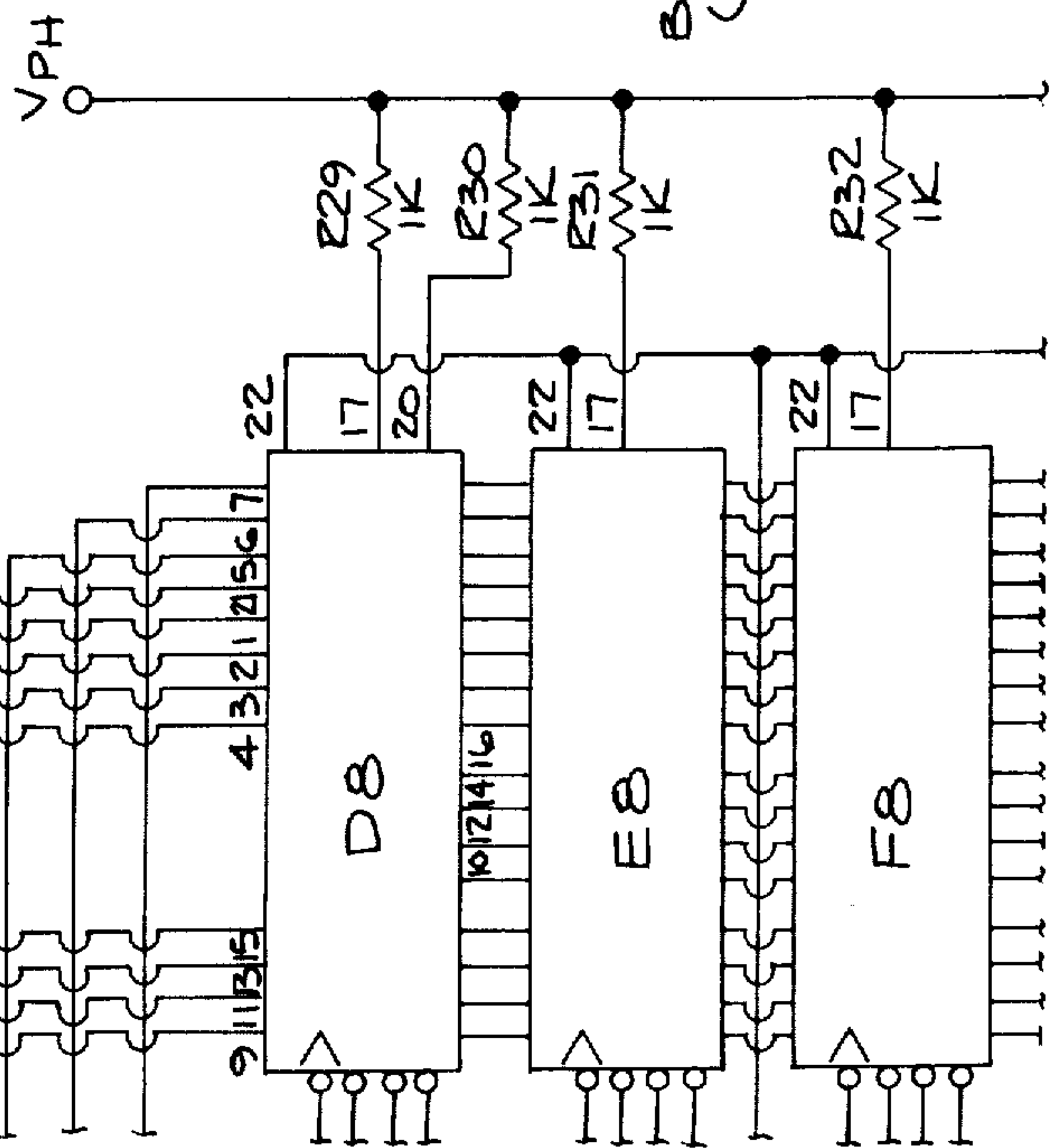
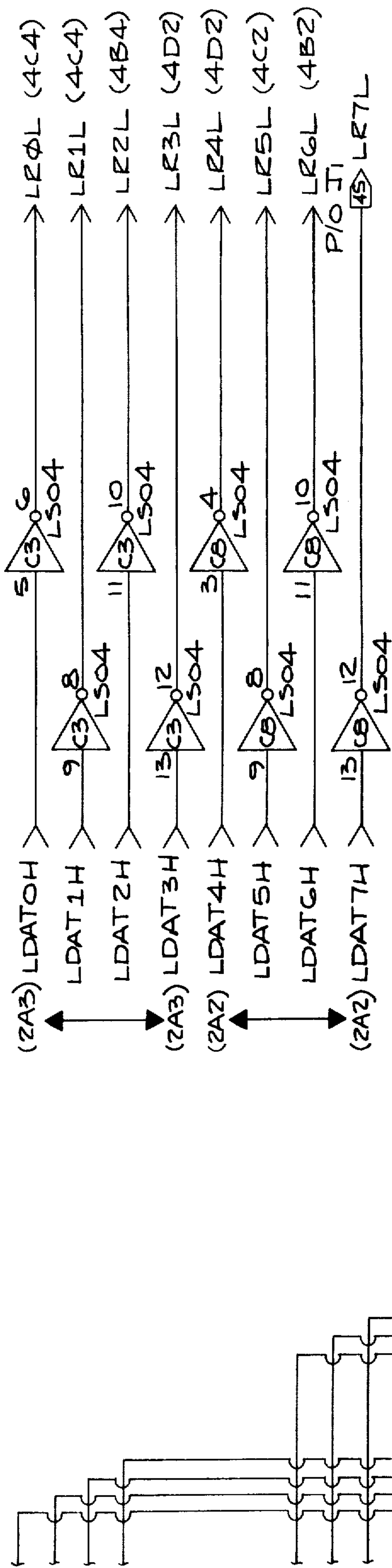


FIG. 20B

P/O J2
BATTOKL

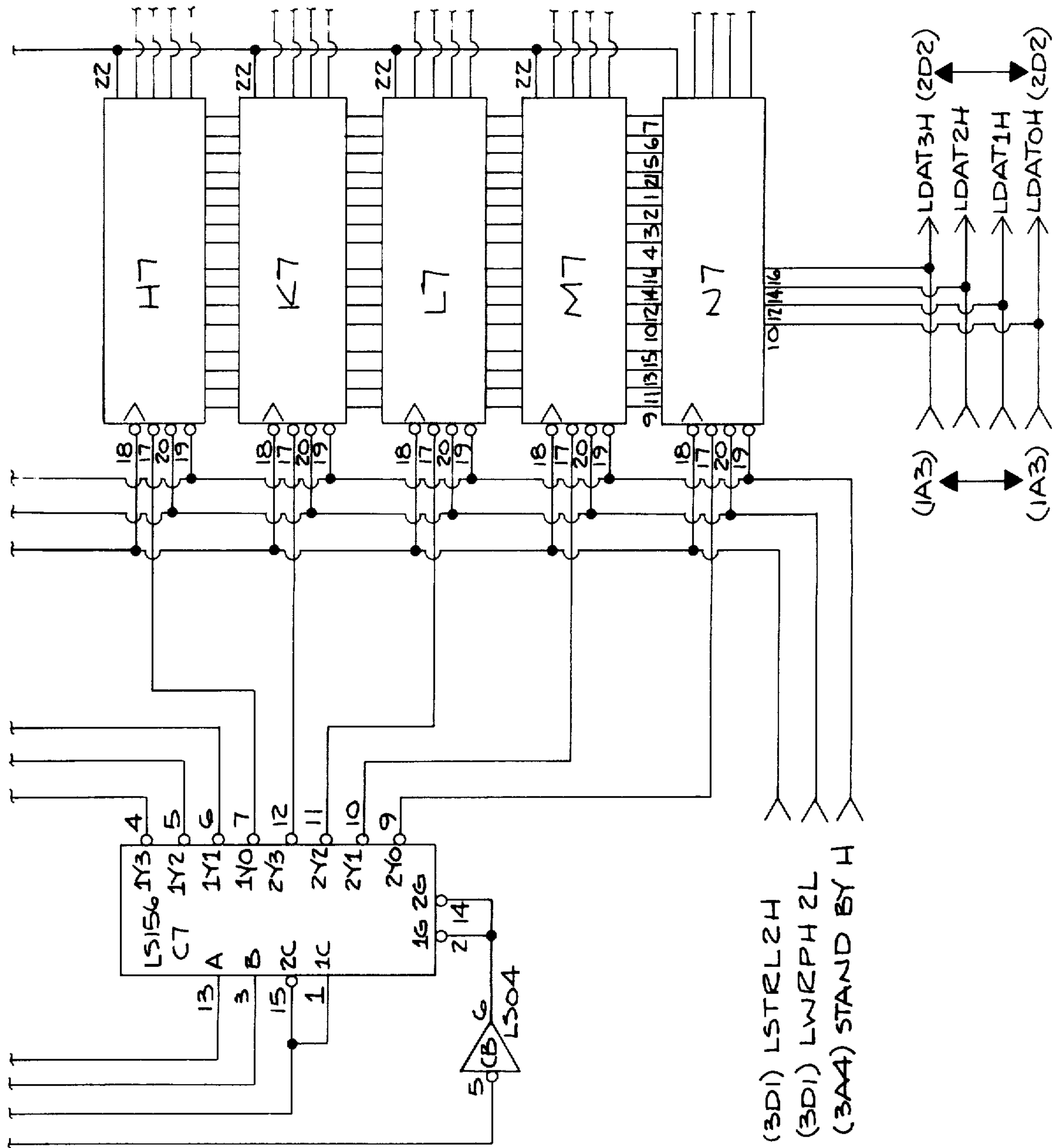


FIG.20C

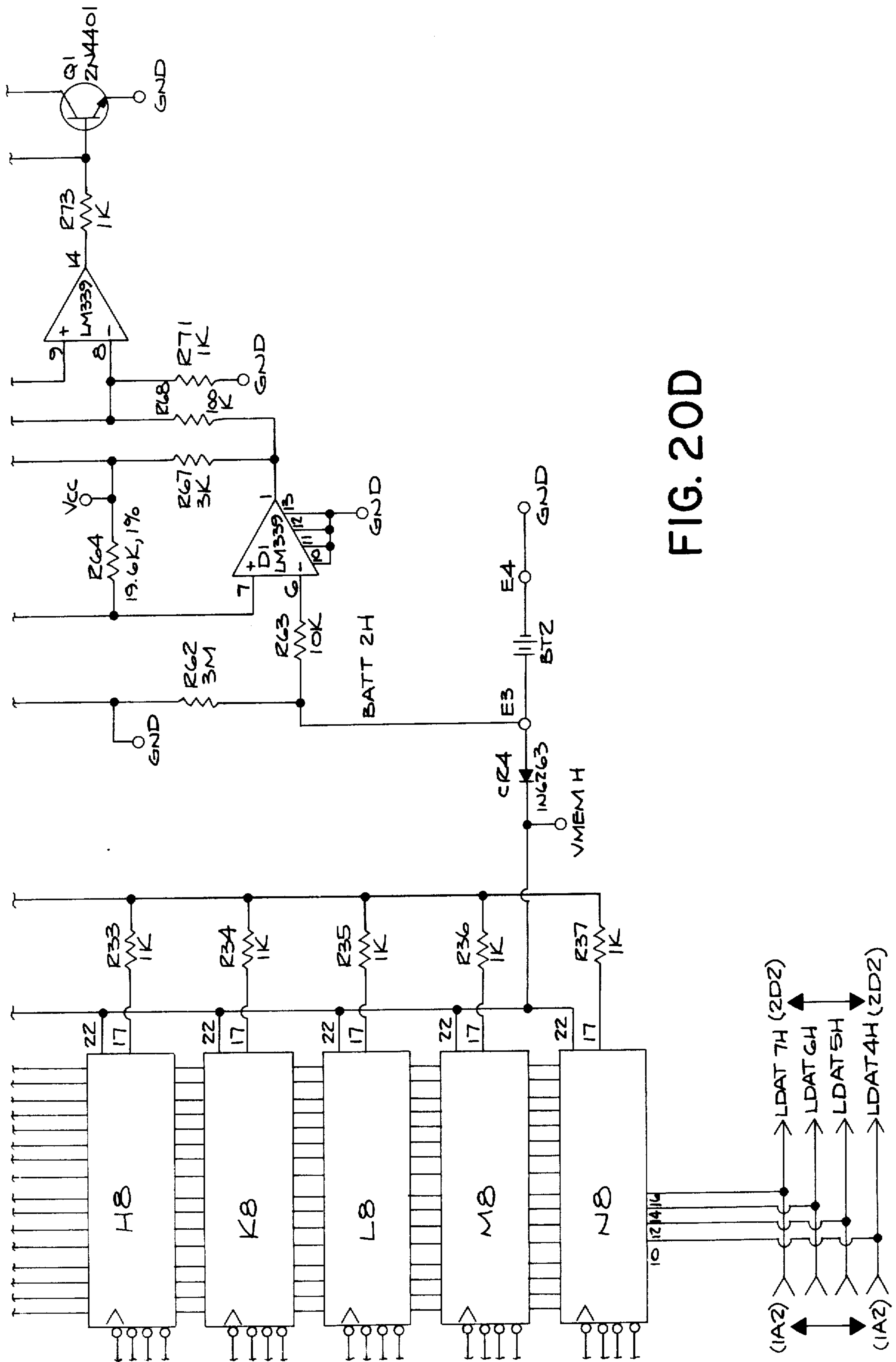


FIG. 20D

FIG. 21A

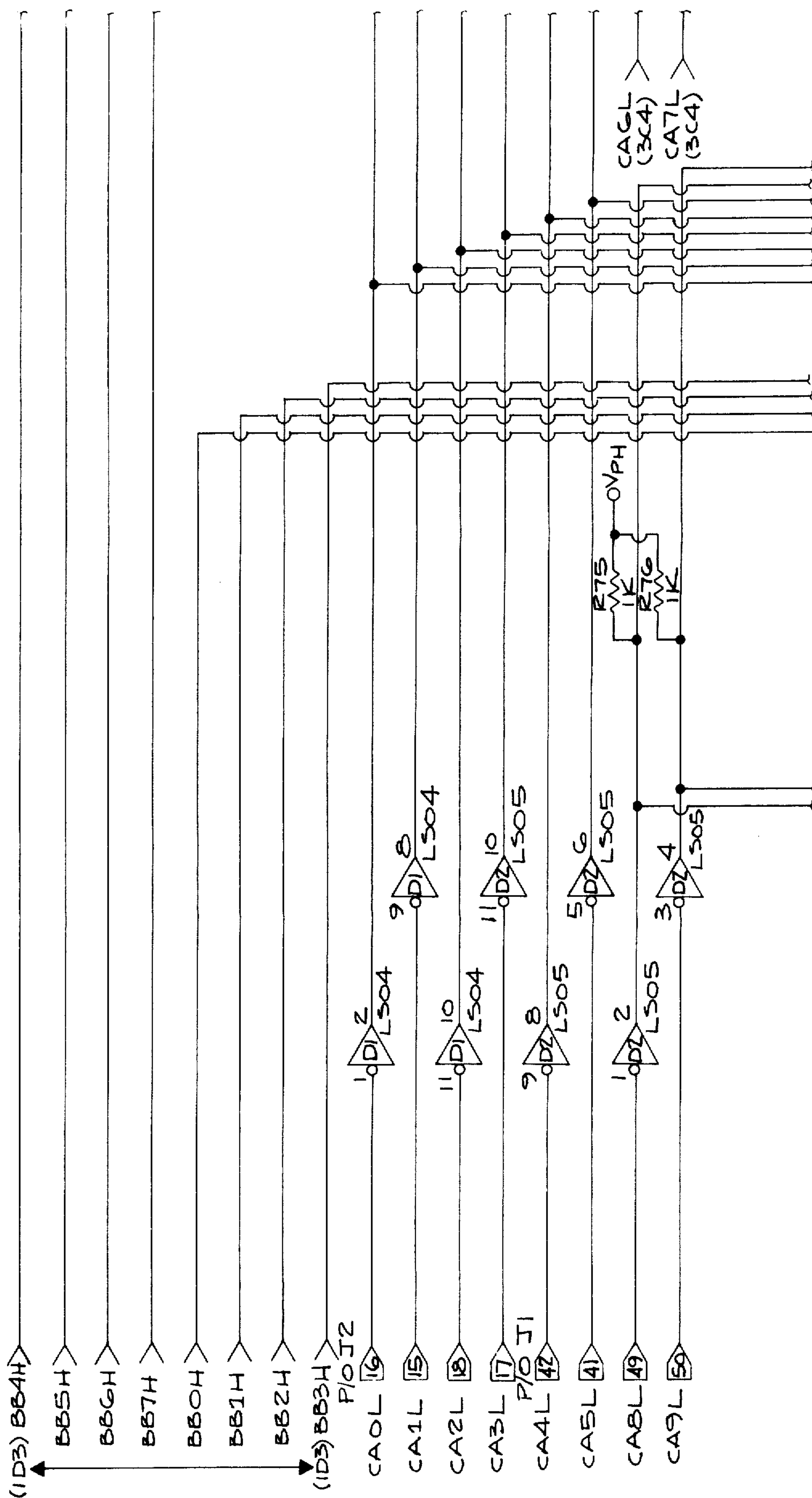
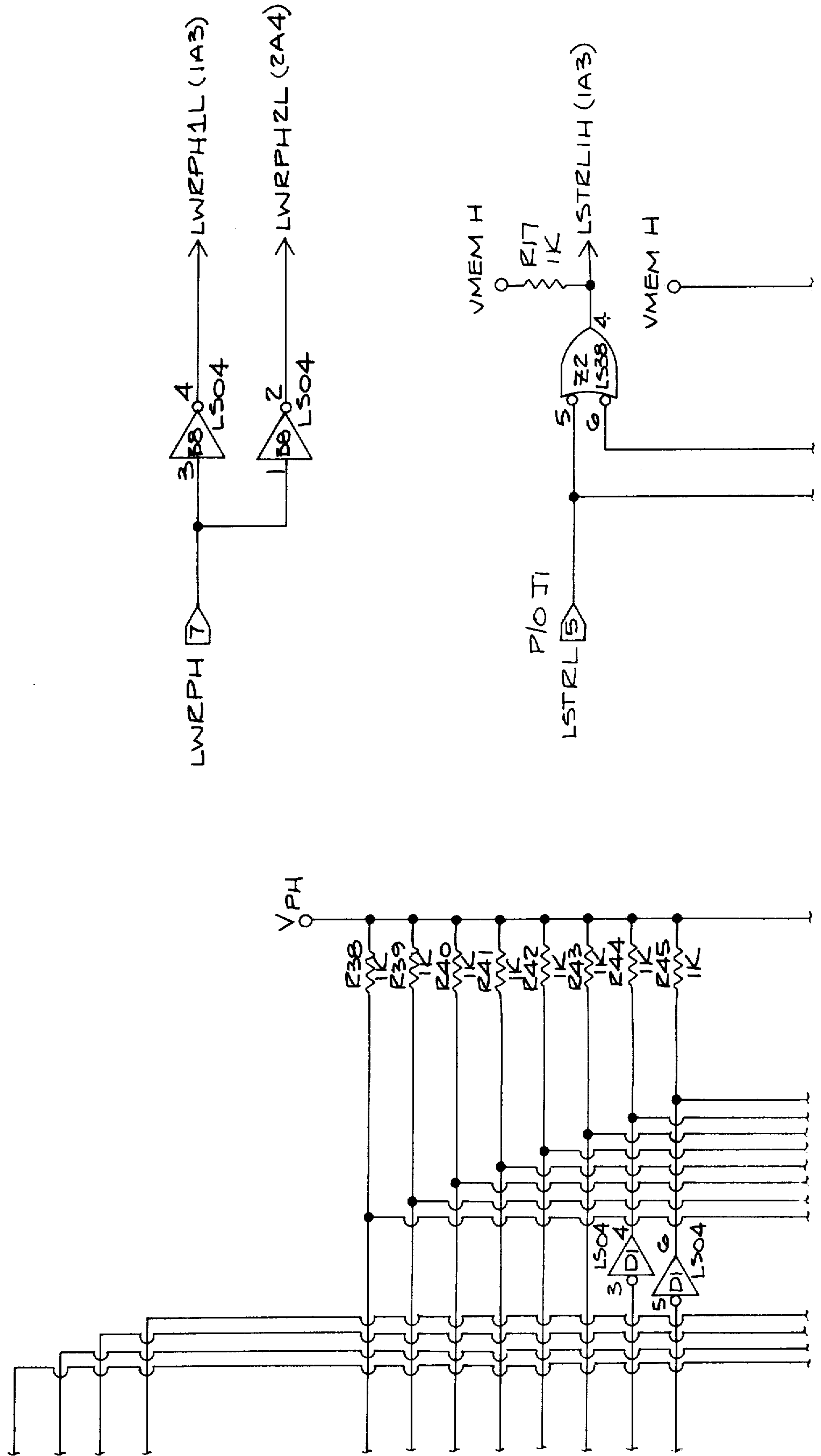


FIG. 21B



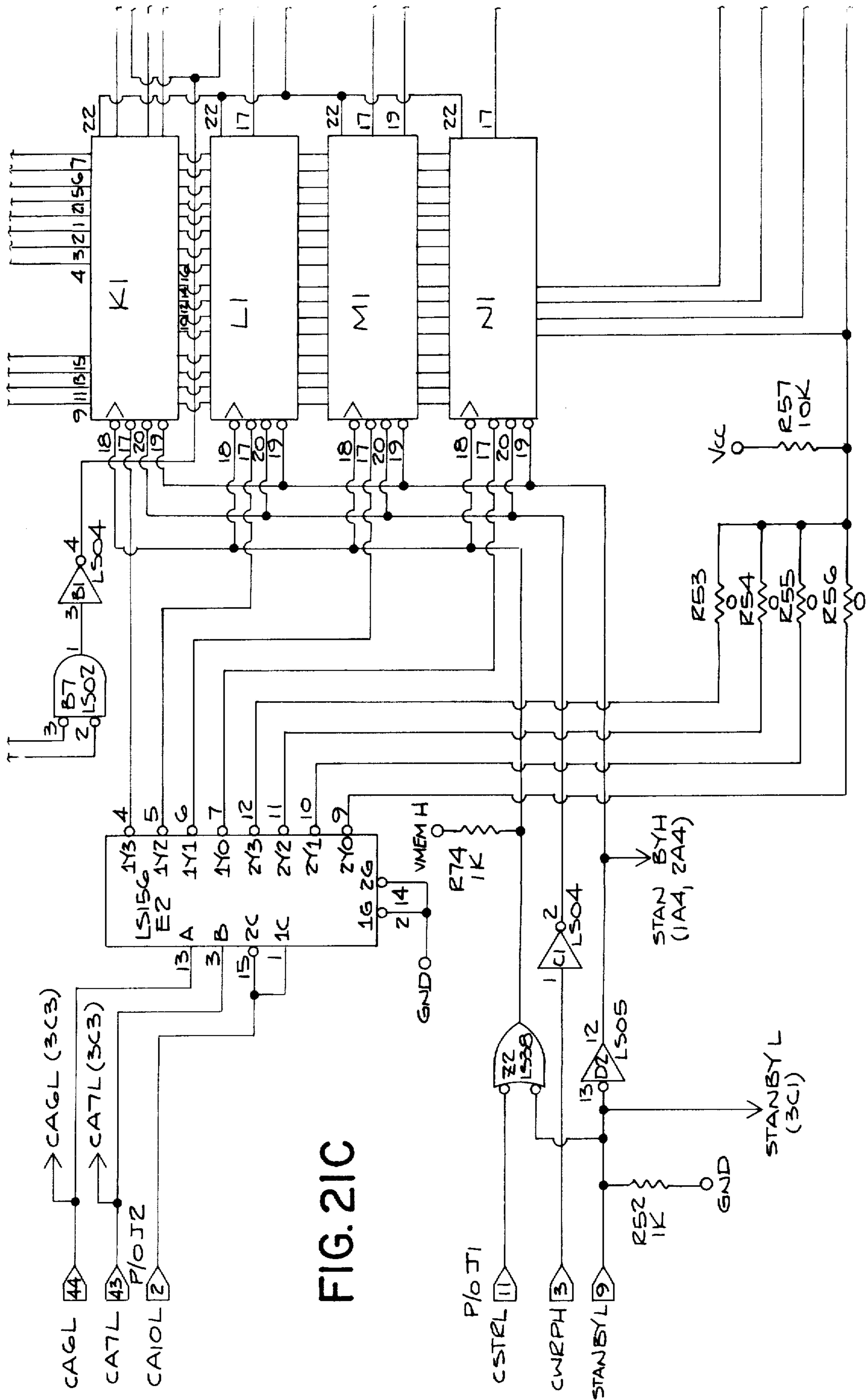


FIG. 21C

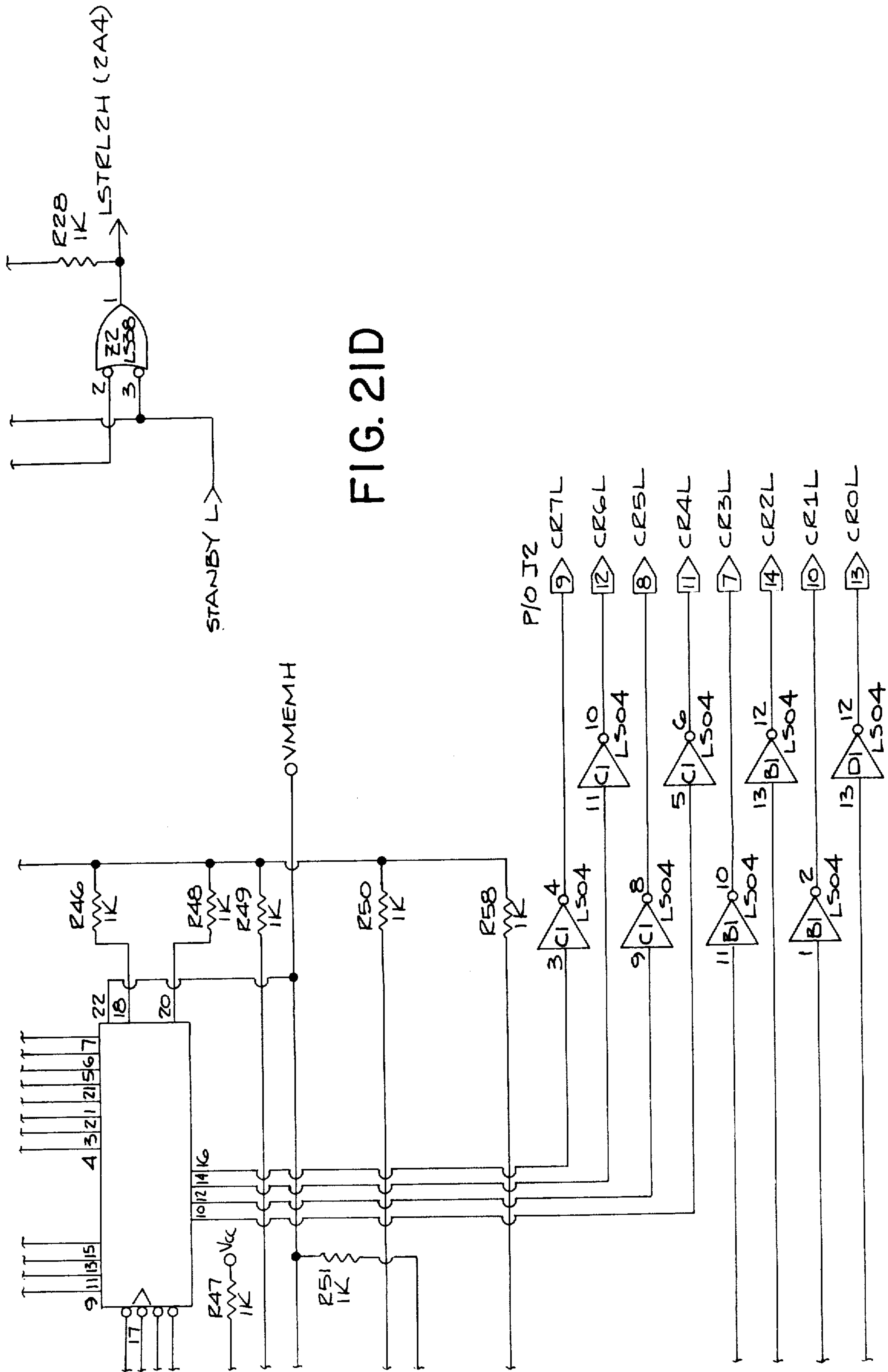


FIG. 21D

FIG. 21A	FIG. 21B
FIG. 21C	FIG. 21D

FIG. 21E

FIG. 22A	FIG. 22B
FIG. 22C	FIG. 22D

FIG. 22E

FIG. 23A	FIG. 23B
----------	----------

FIG. 23C

FIG. 24A	FIG. 24B
FIG. 24C	FIG. 24D

FIG. 24E

FIG. 22A

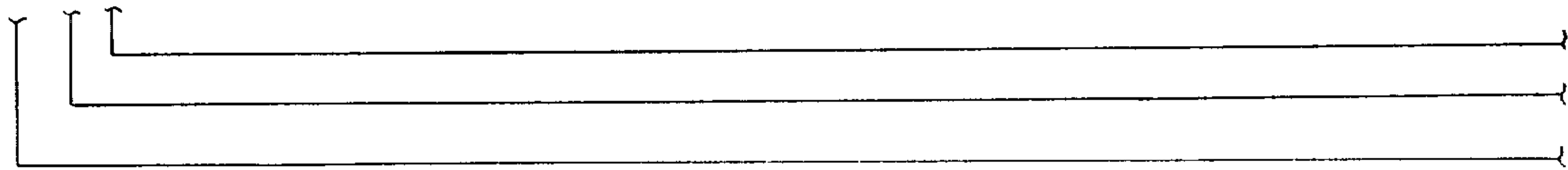
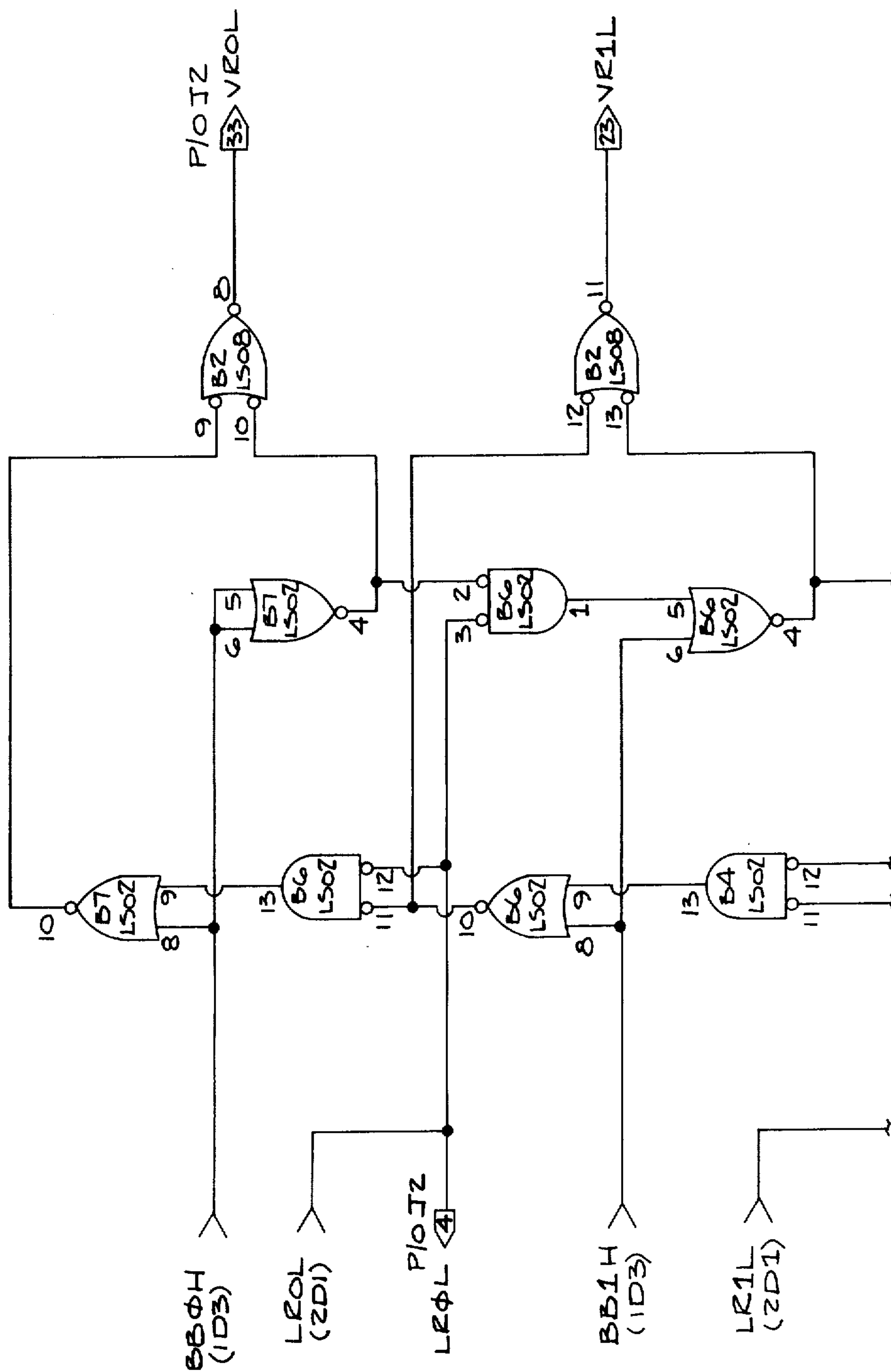
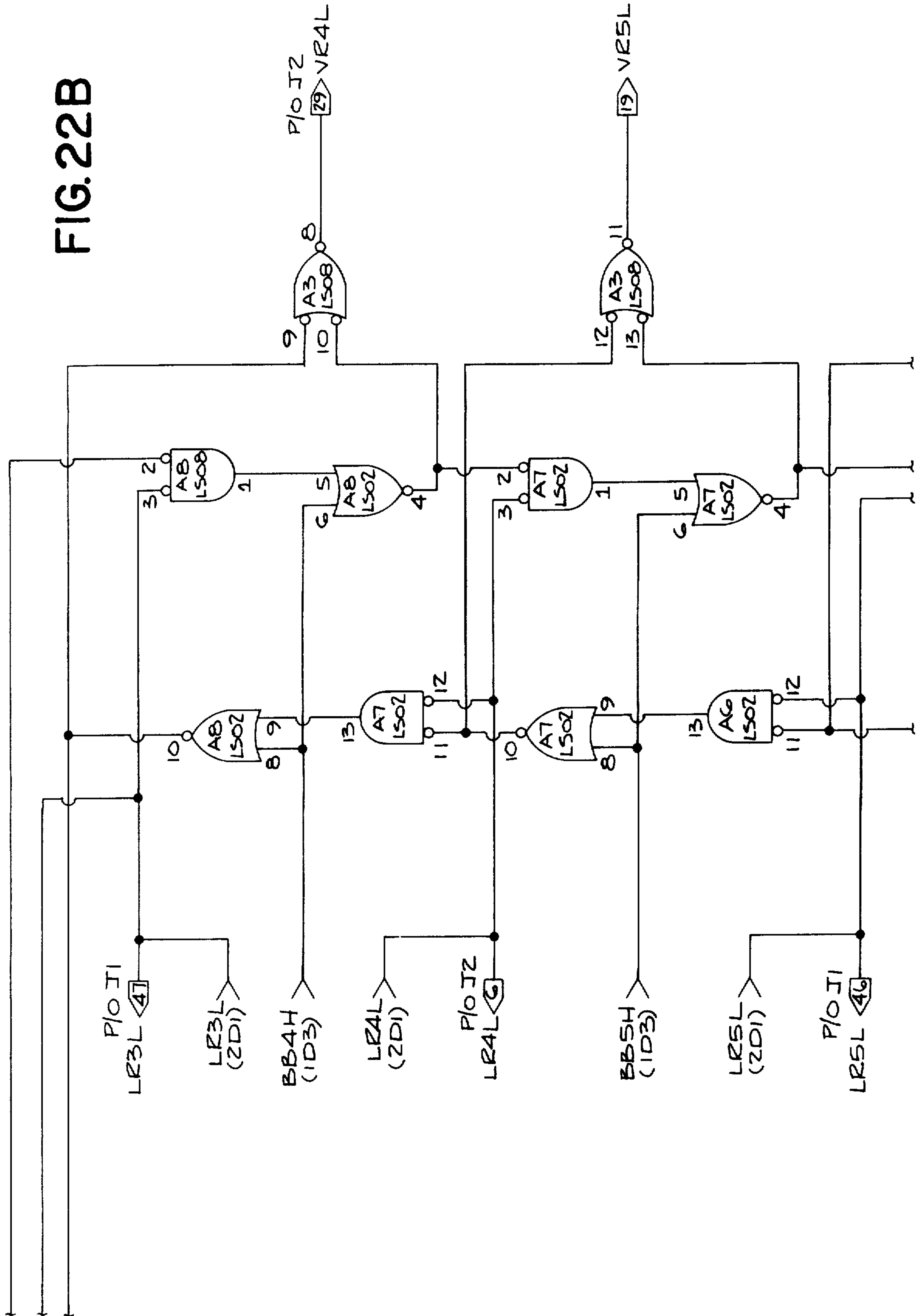


FIG. 22B



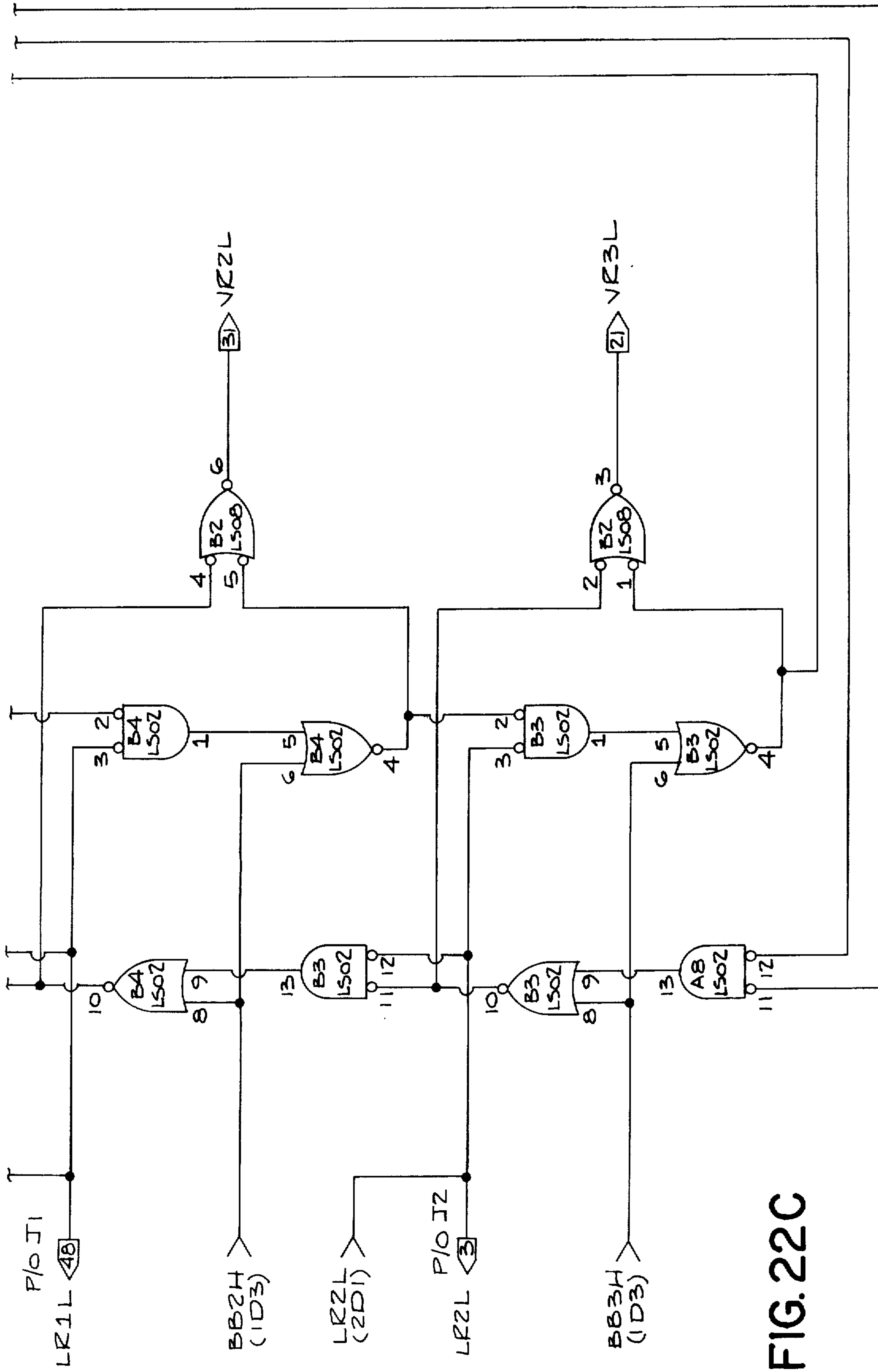


FIG. 22C

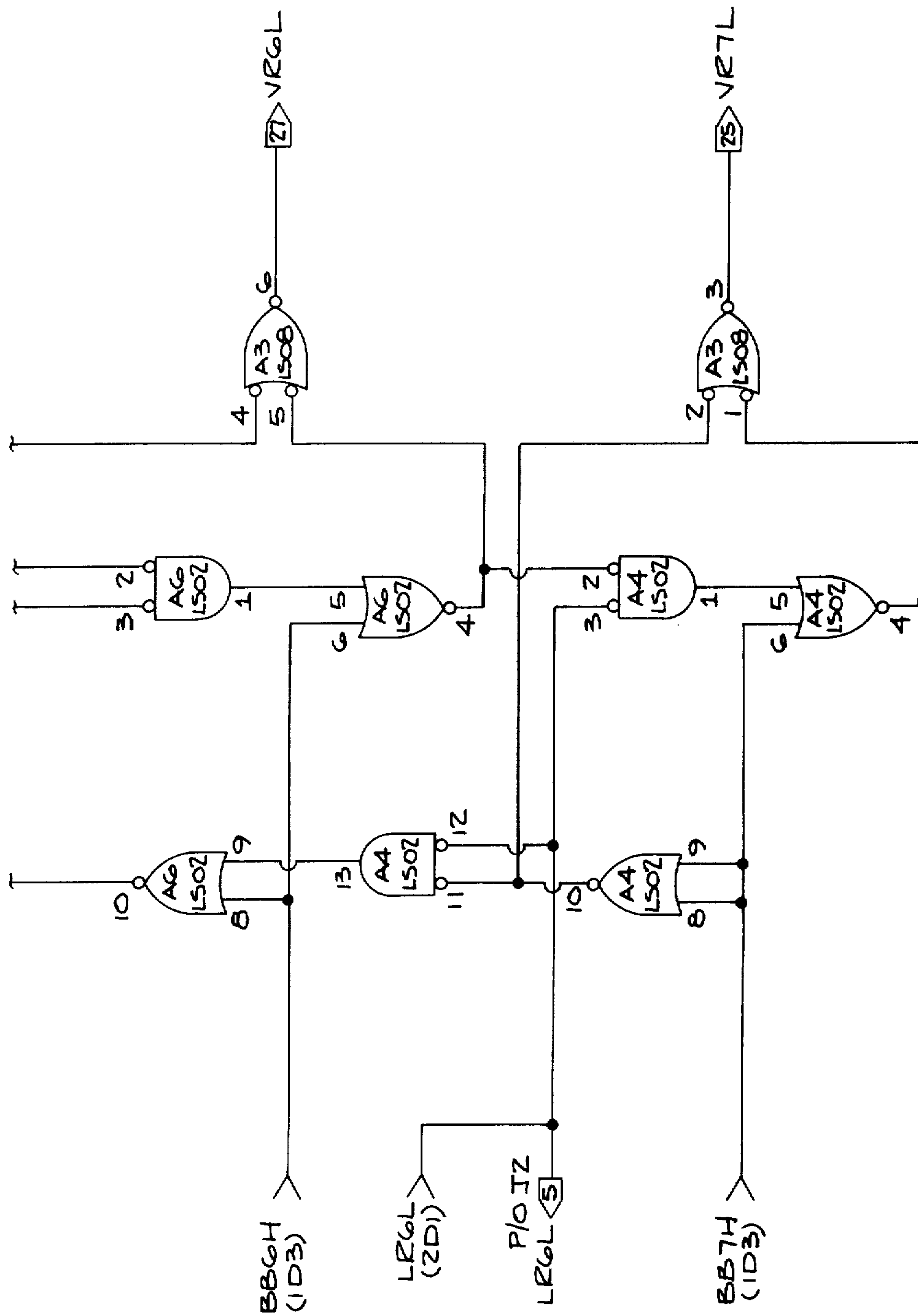
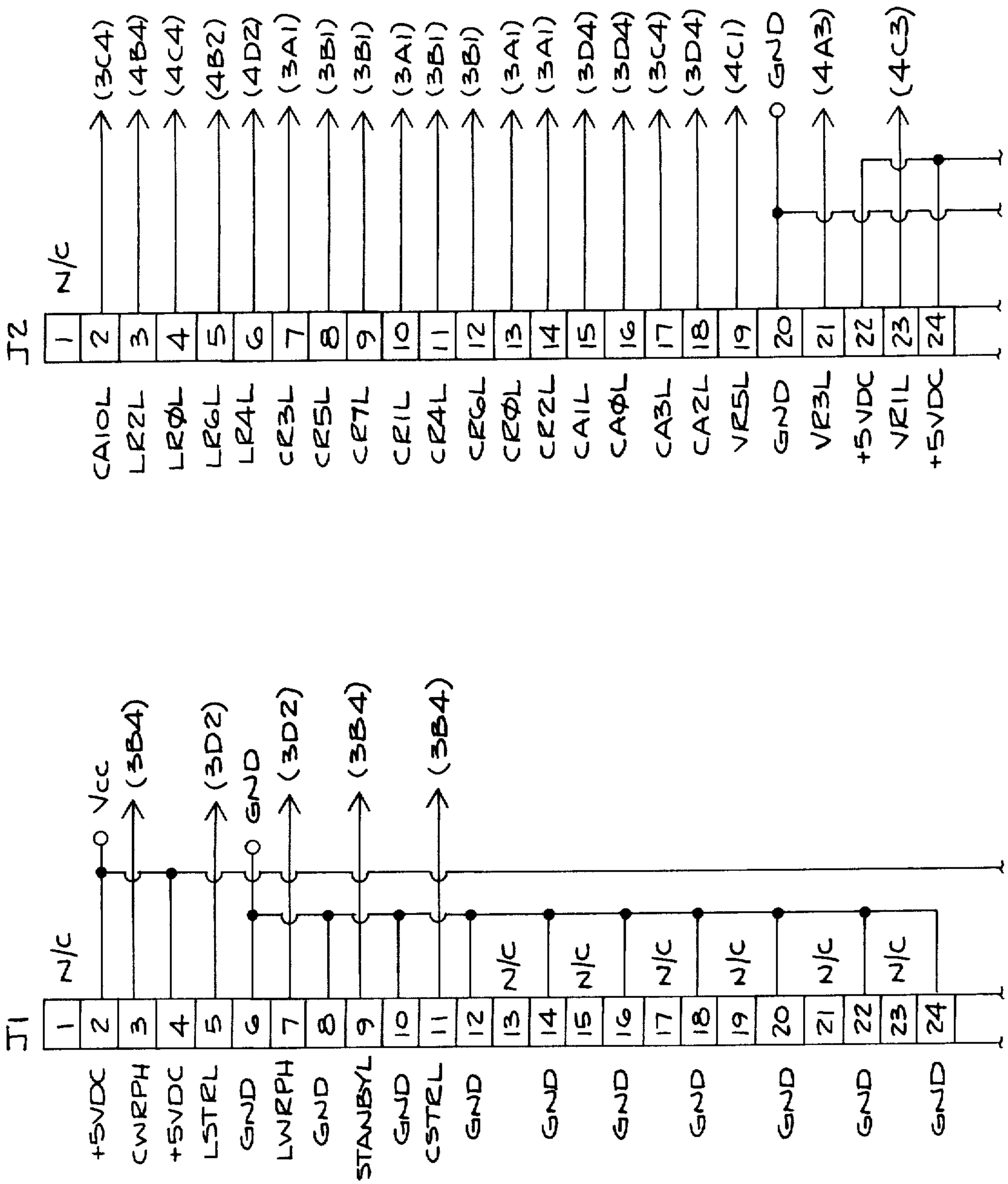


FIG. 22D

FIG. 23A



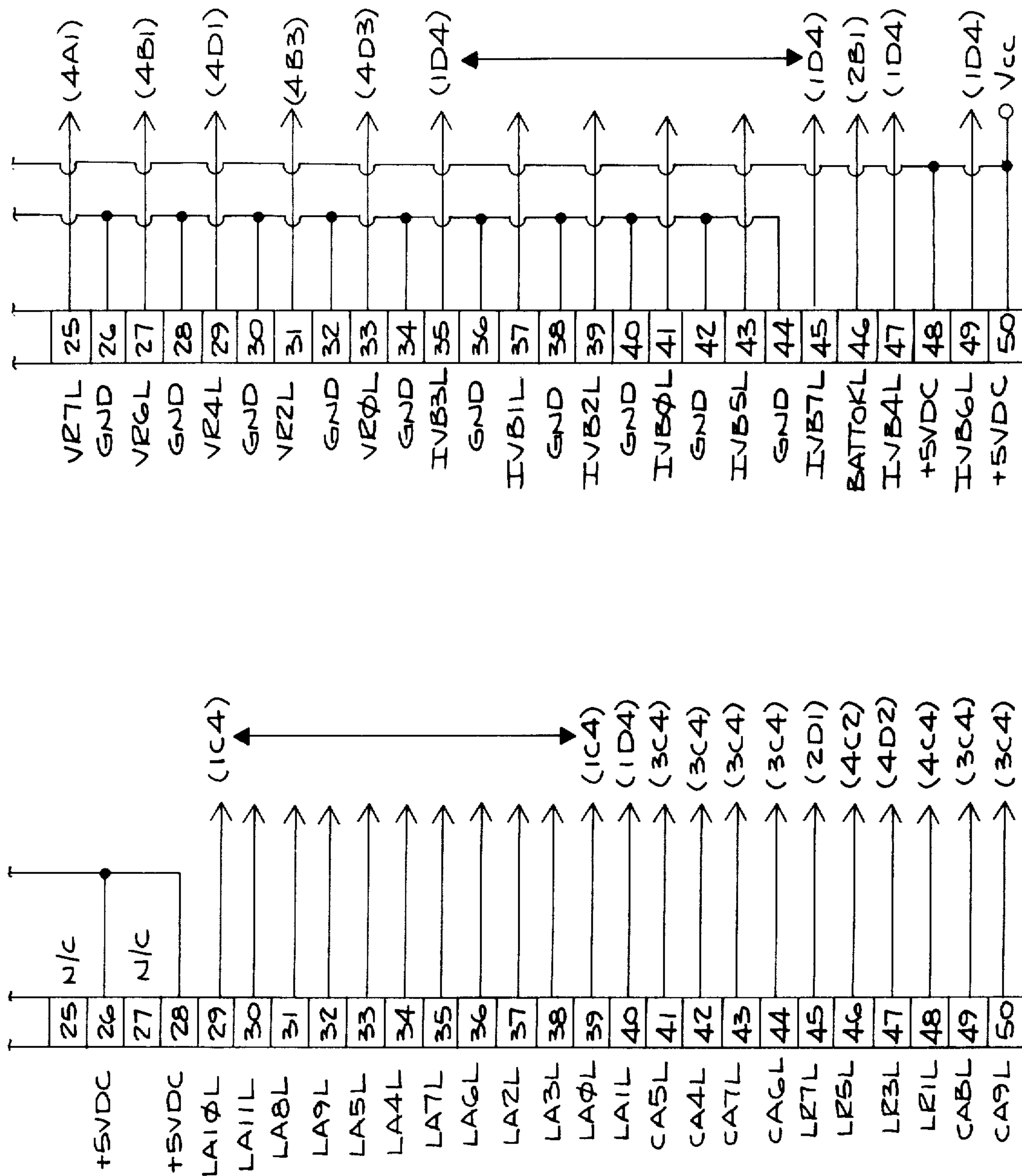


FIG. 23B

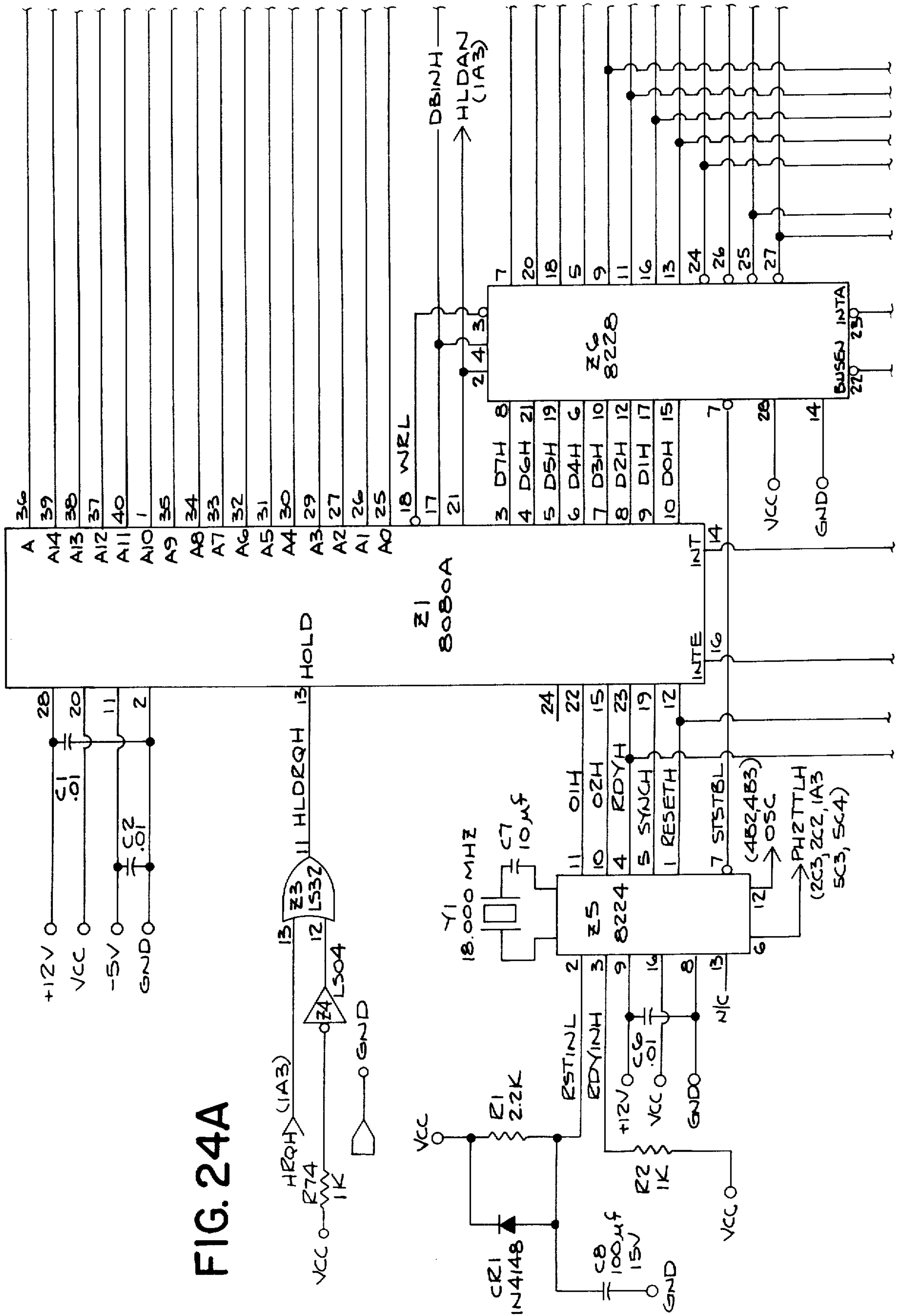
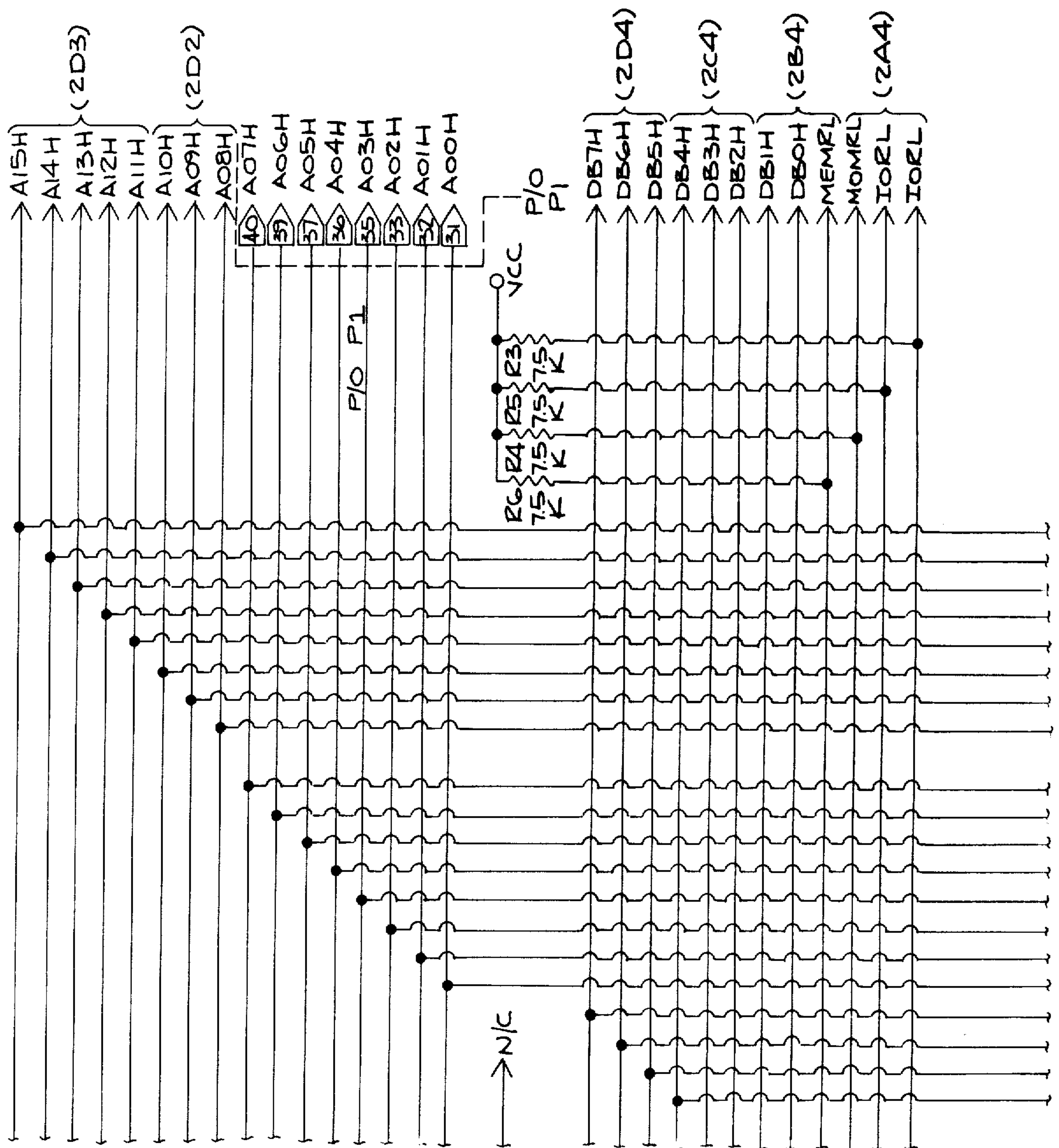
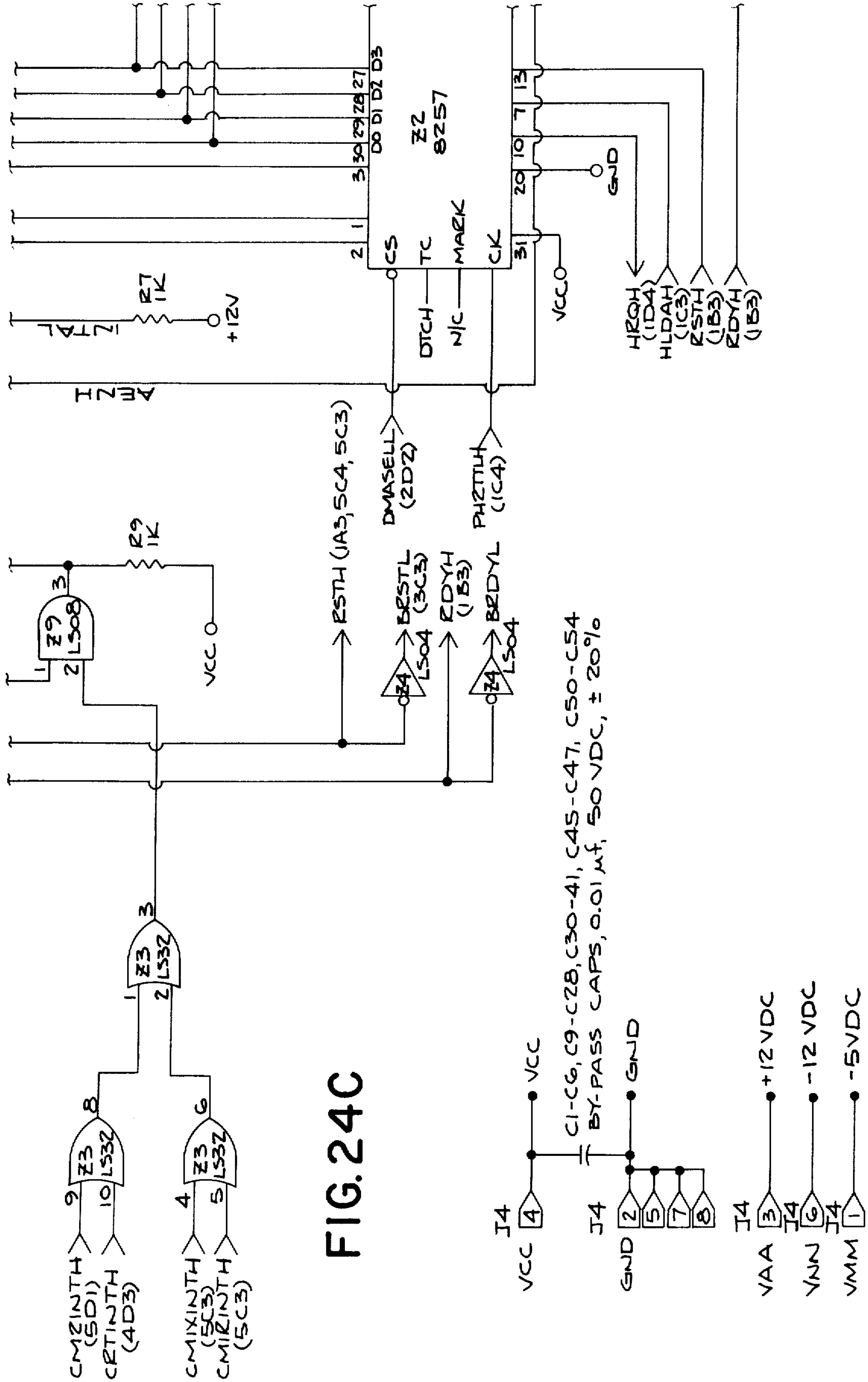


FIG. 24A

FIG. 24B





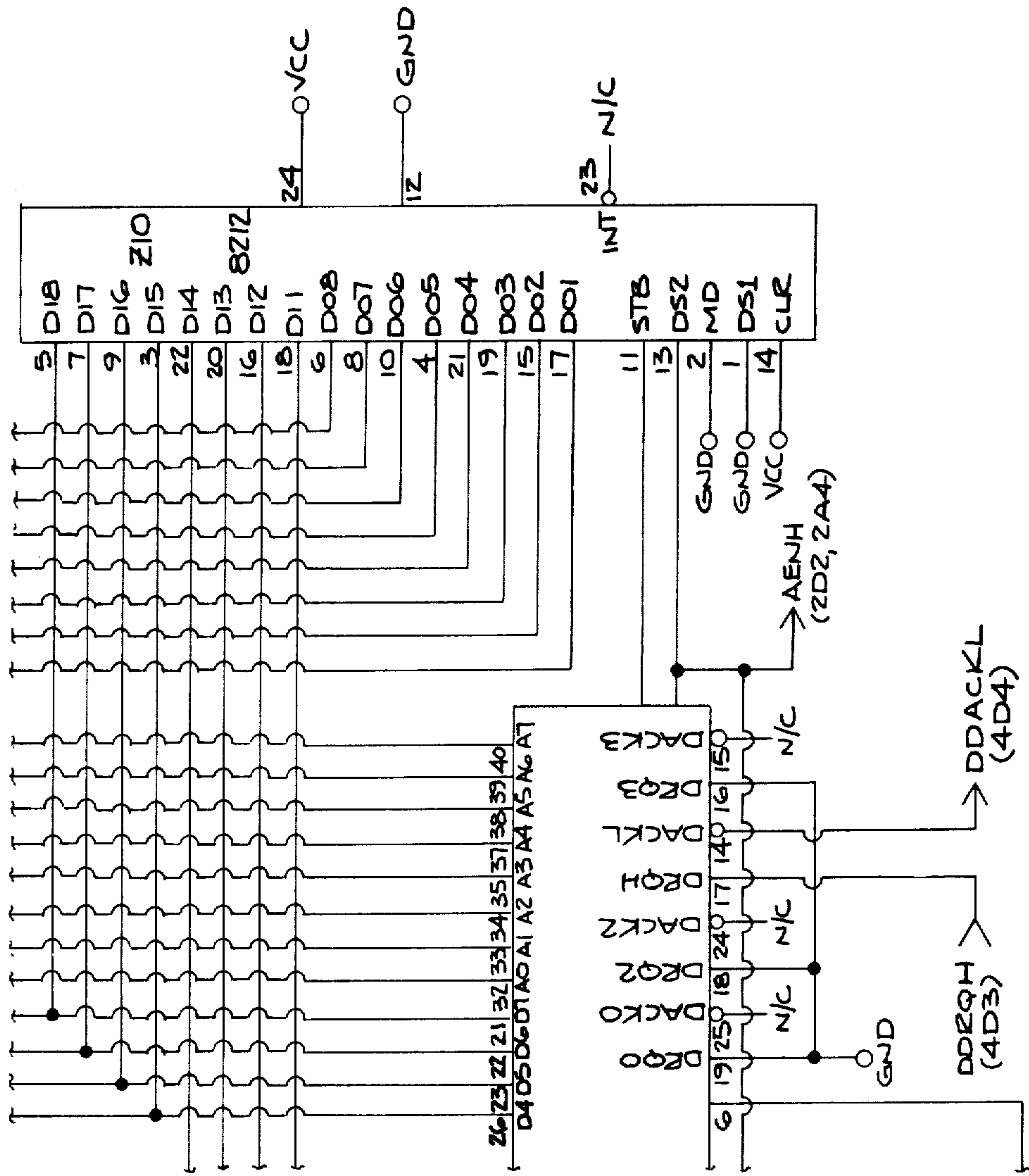


FIG. 24D

FIG. 25A

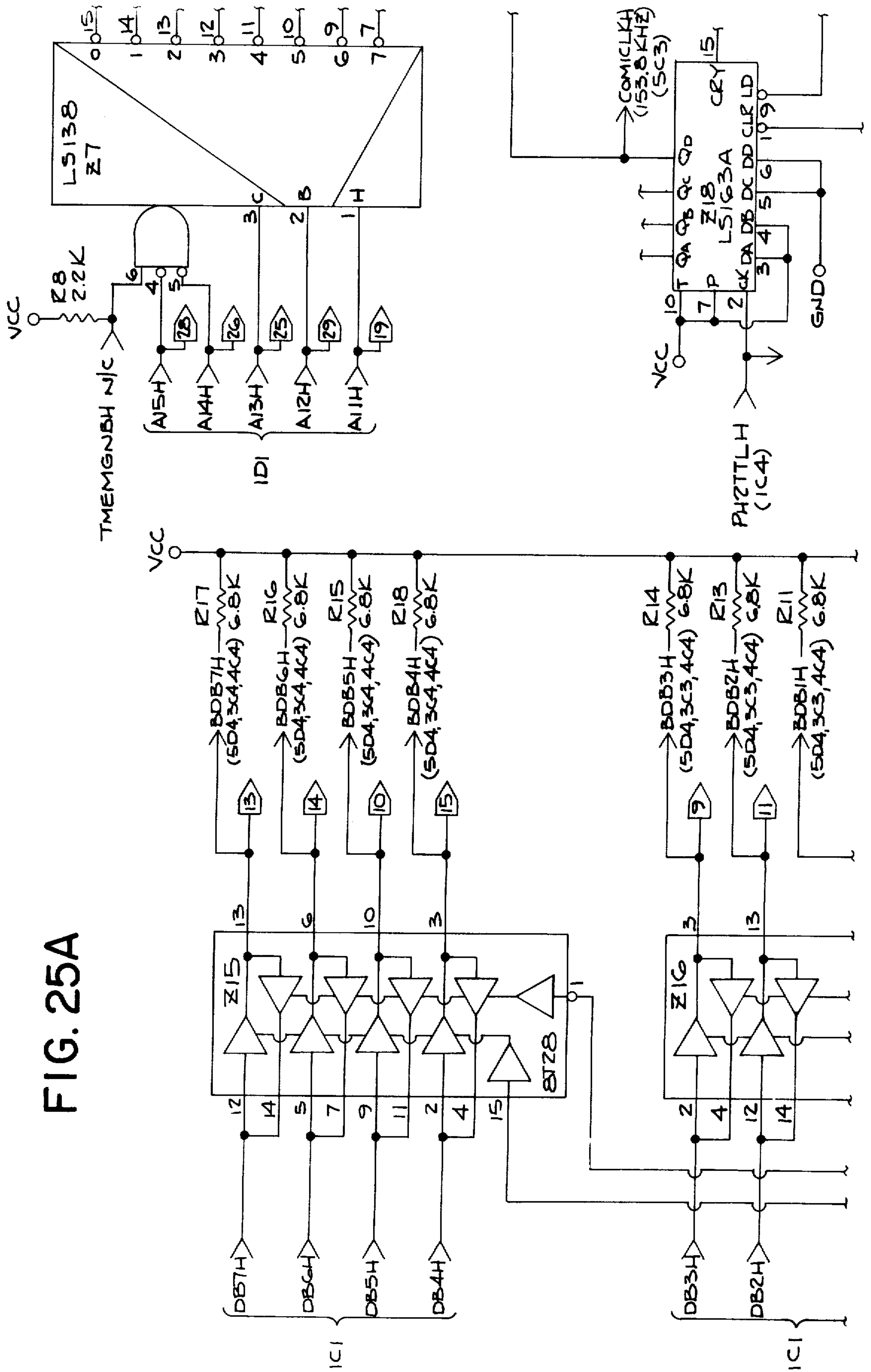
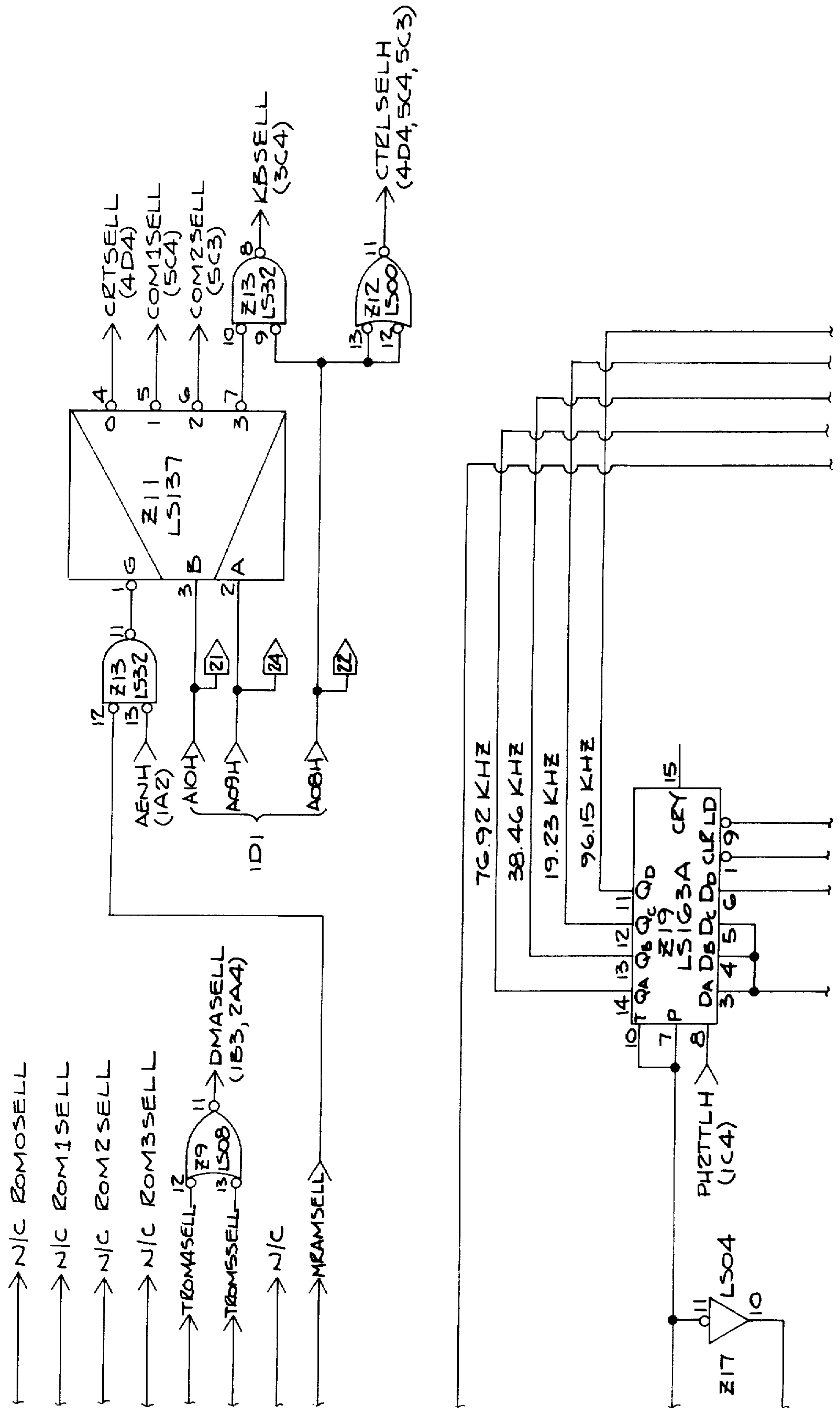
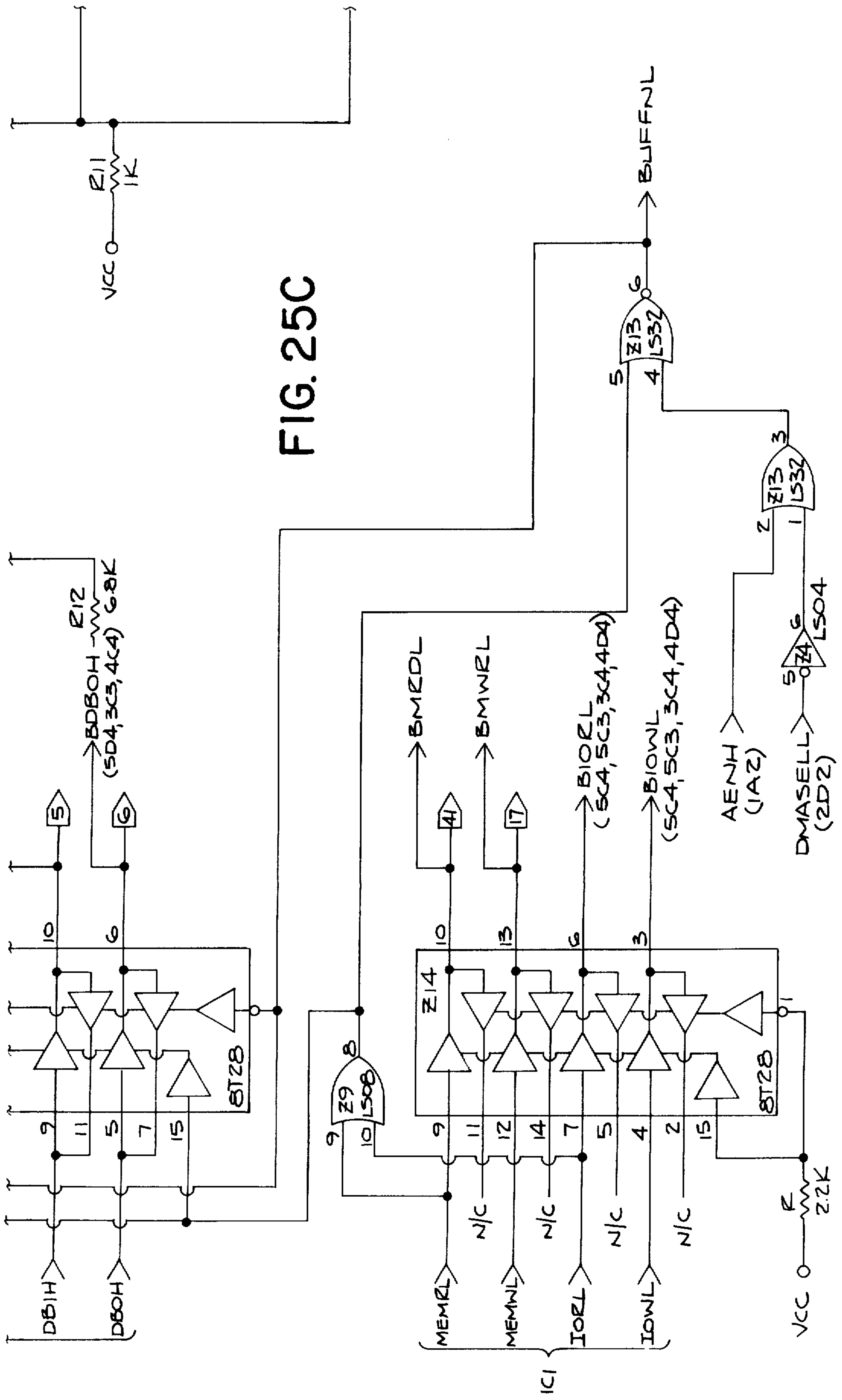


FIG. 25B





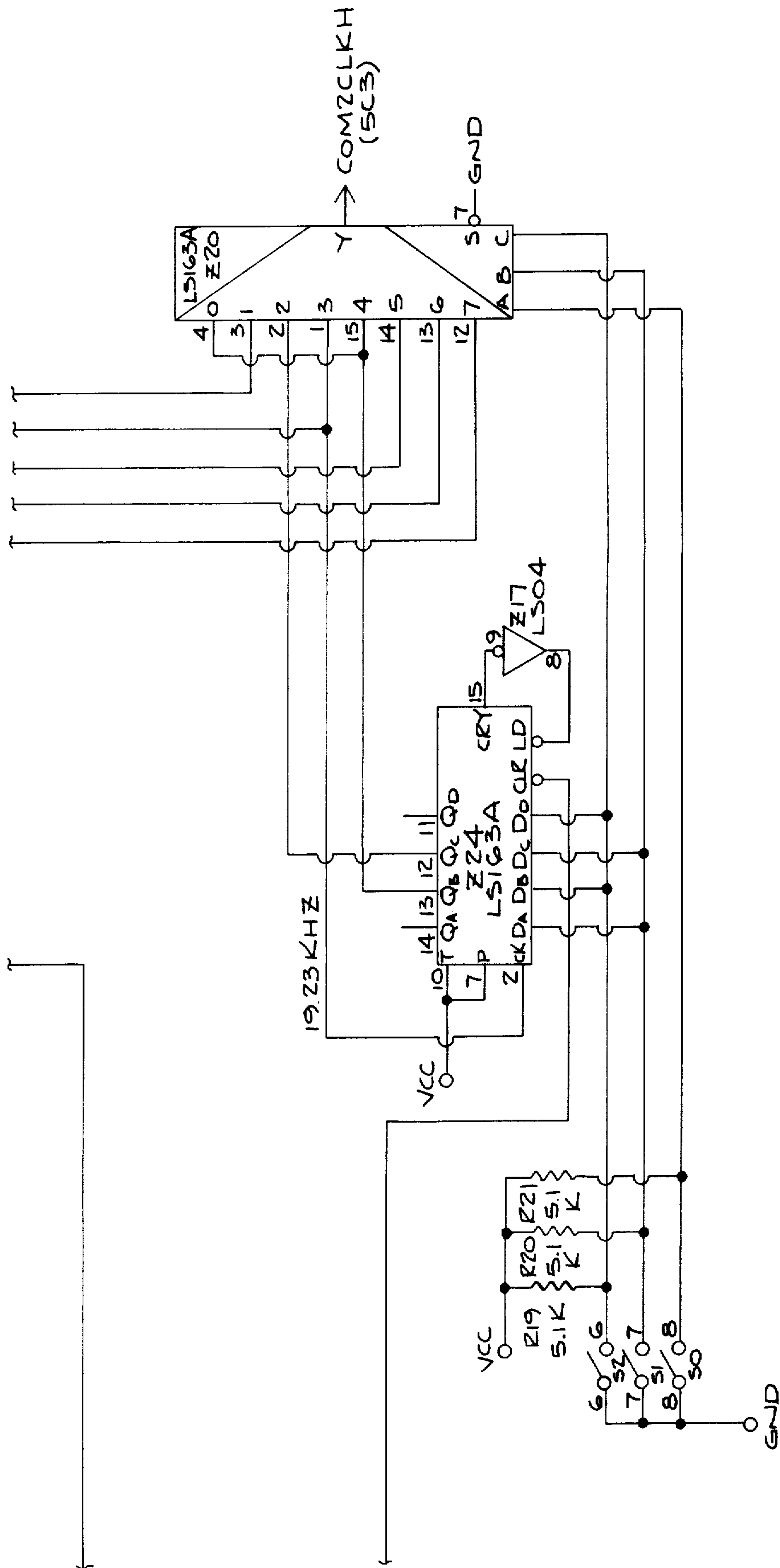


FIG. 25D

FIG. 25A	FIG. 25B
FIG. 25C	FIG. 25D

FIG. 25E

FIG. 26A	FIG. 26B
FIG. 26C	FIG. 26D

FIG. 26E

FIG. 27A	FIG. 27B
FIG. 27C	FIG. 27D

FIG. 27E

FIG. 28A	FIG. 28B
FIG. 28C	FIG. 28D

FIG. 28E

FIG. 26A

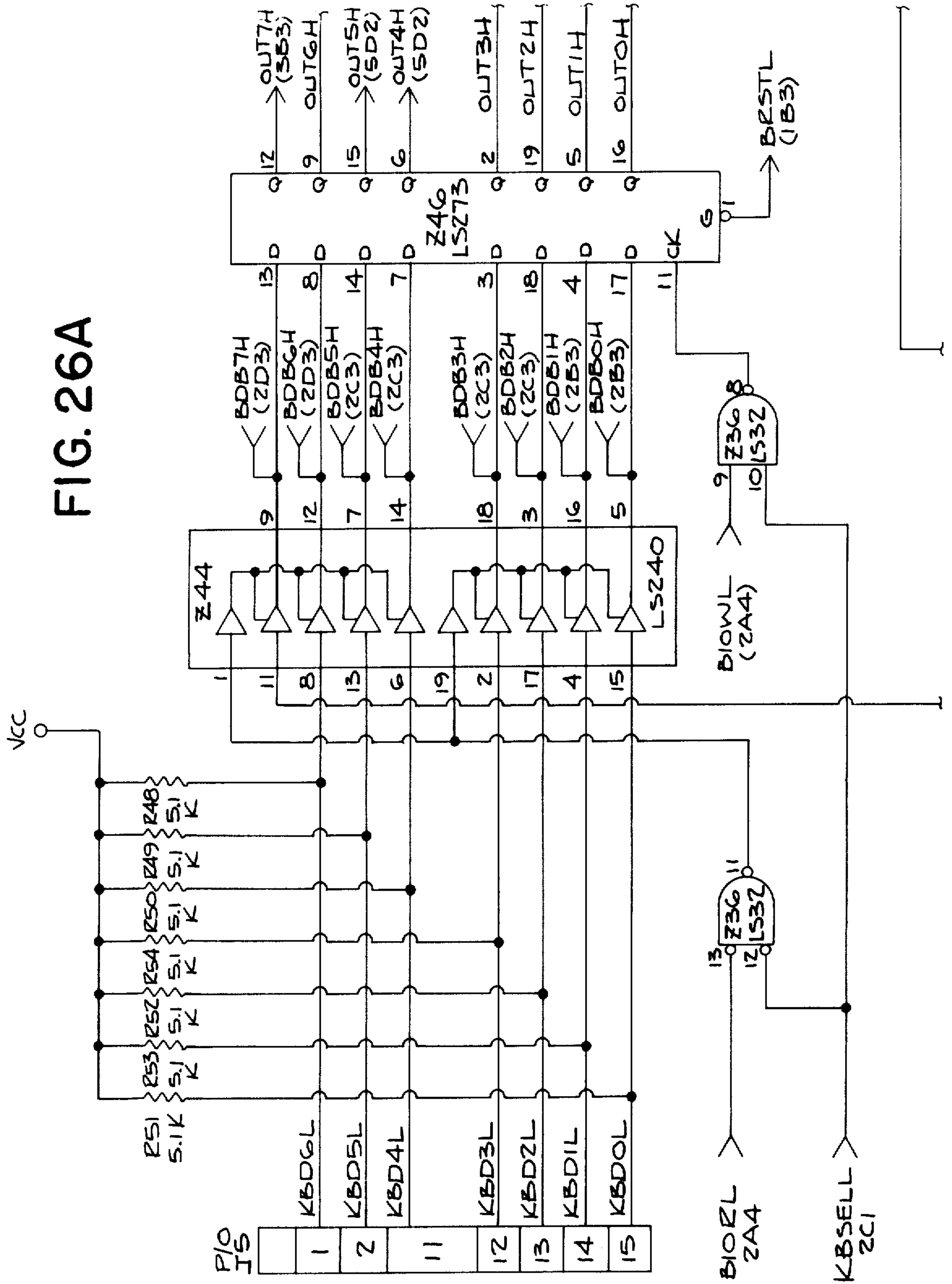
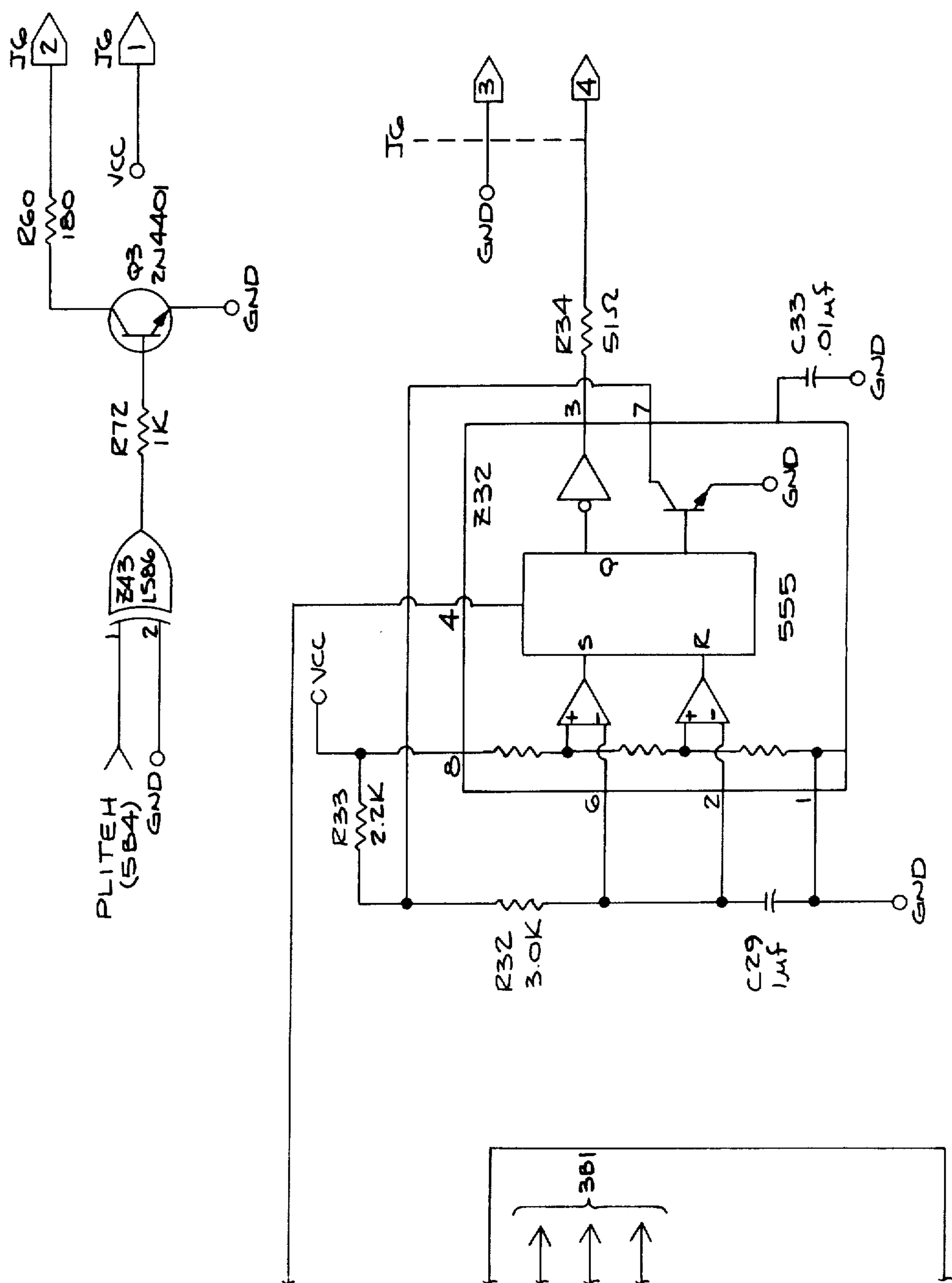


FIG. 26B



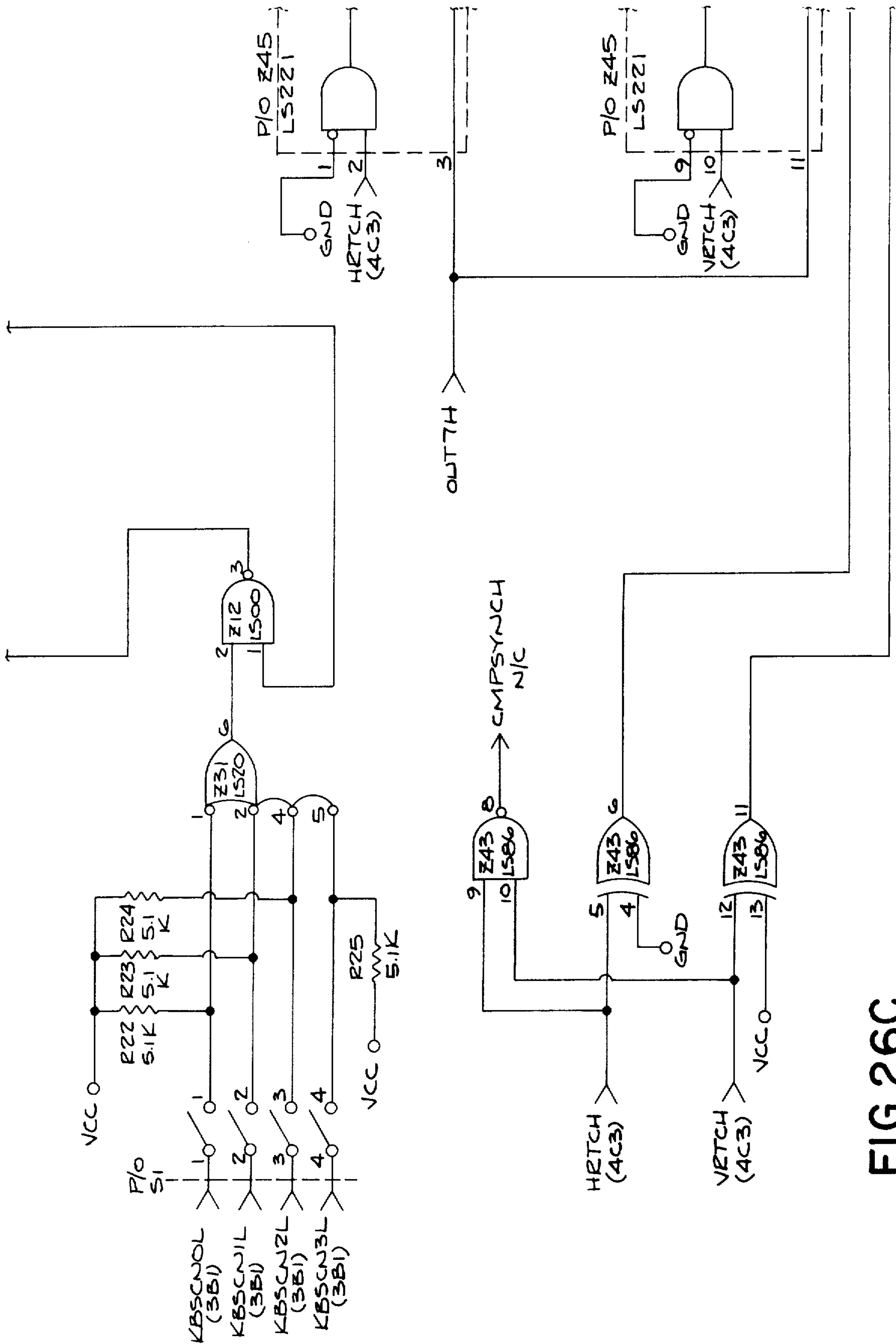


FIG. 26C

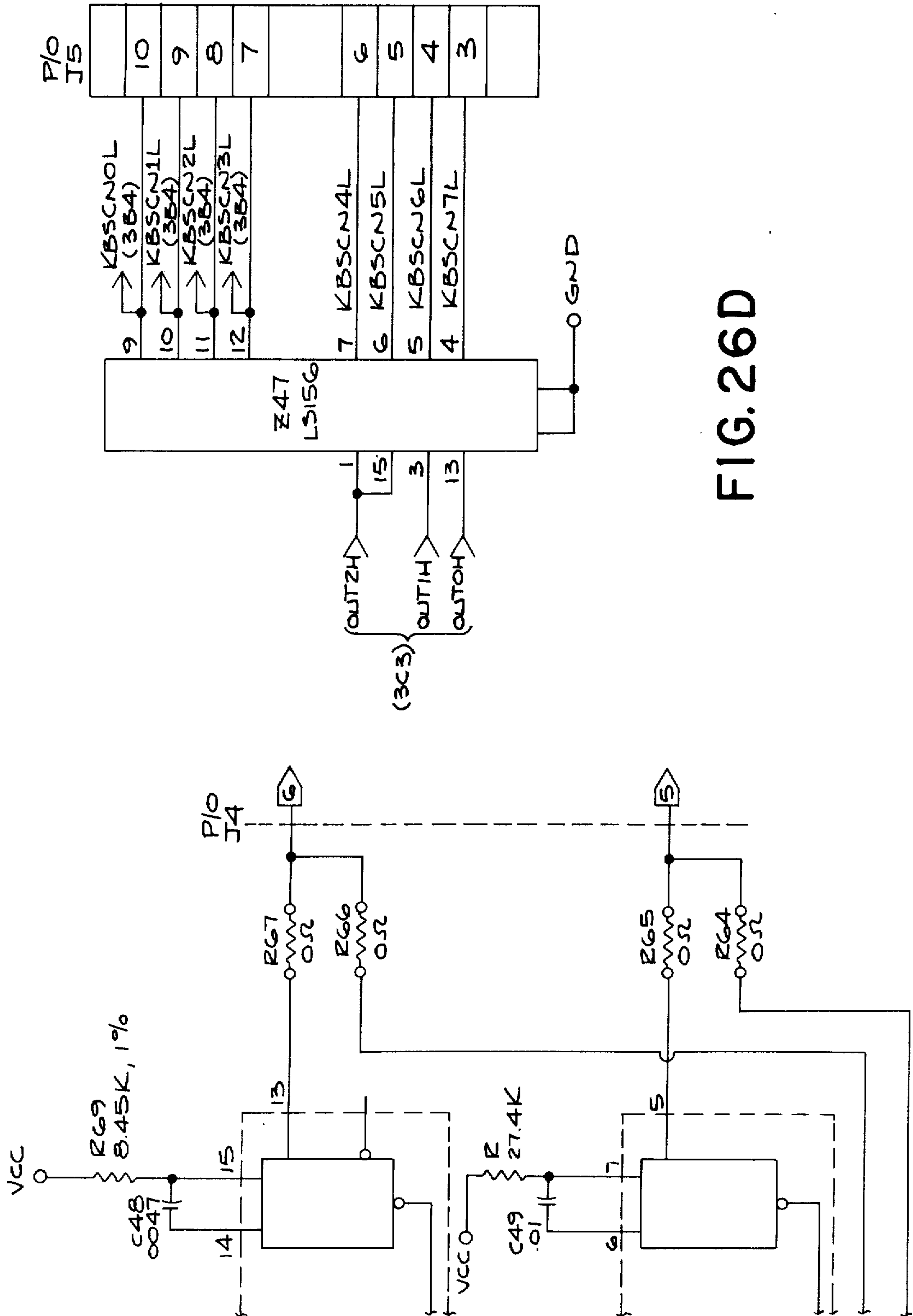
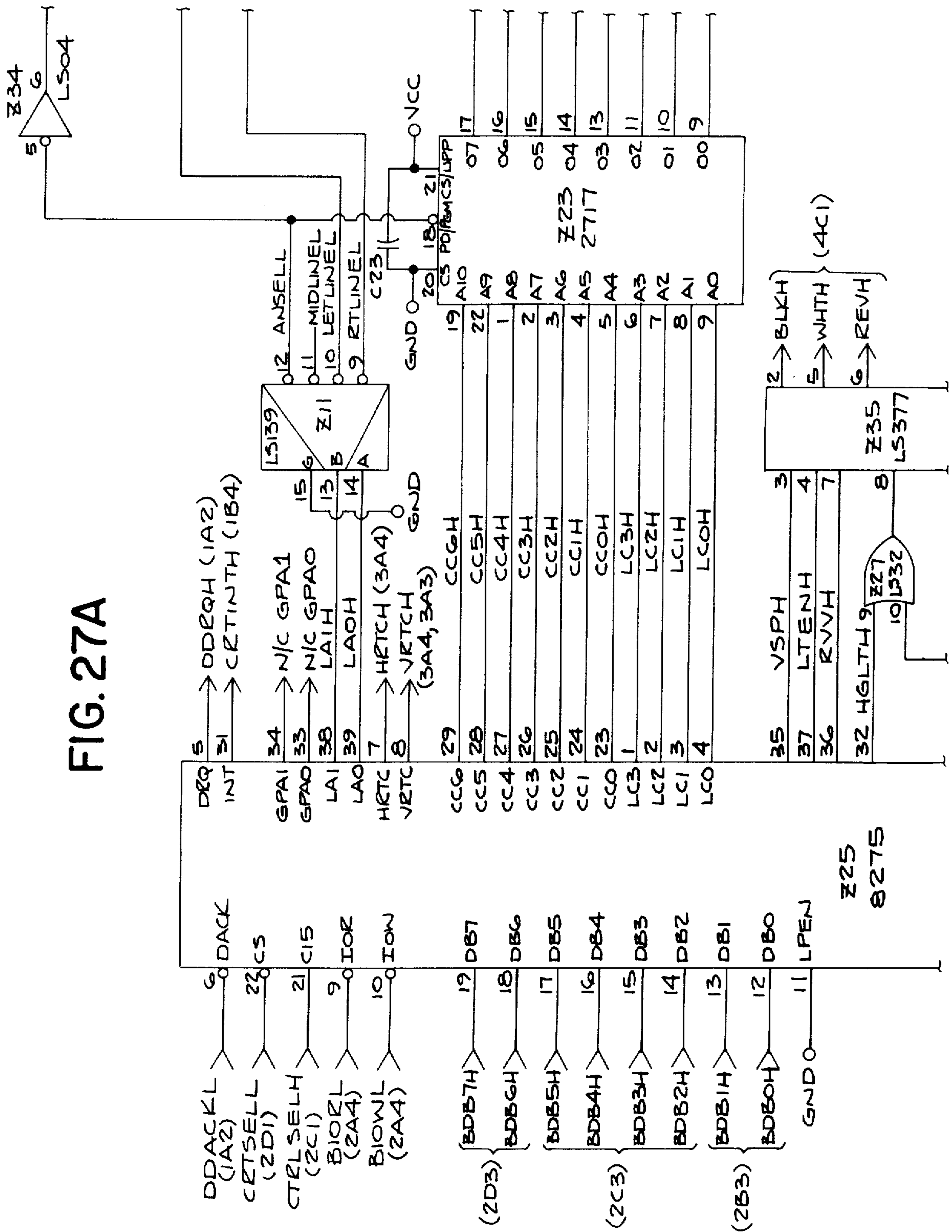


FIG. 26D

FIG. 27A



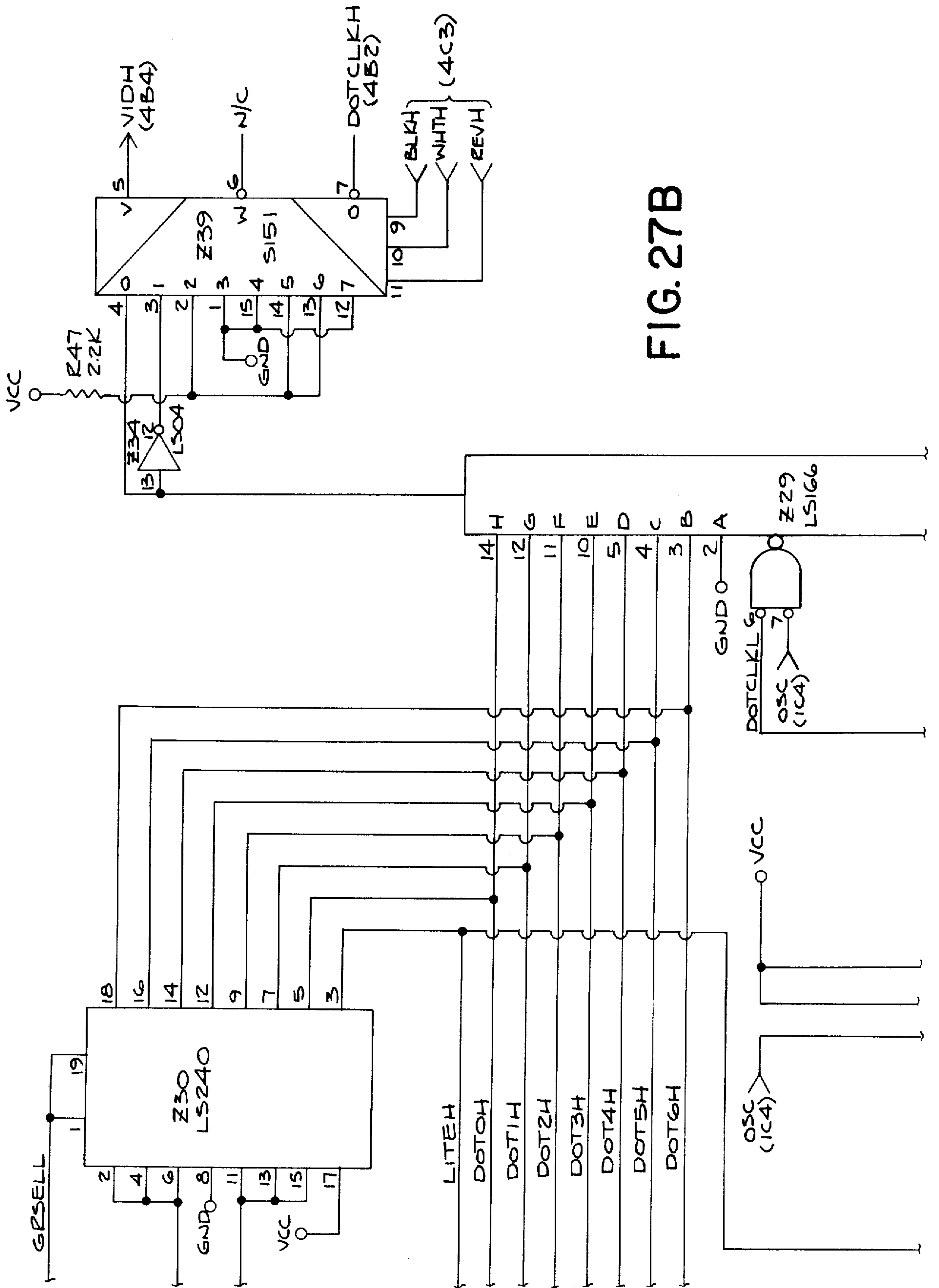


FIG. 27B

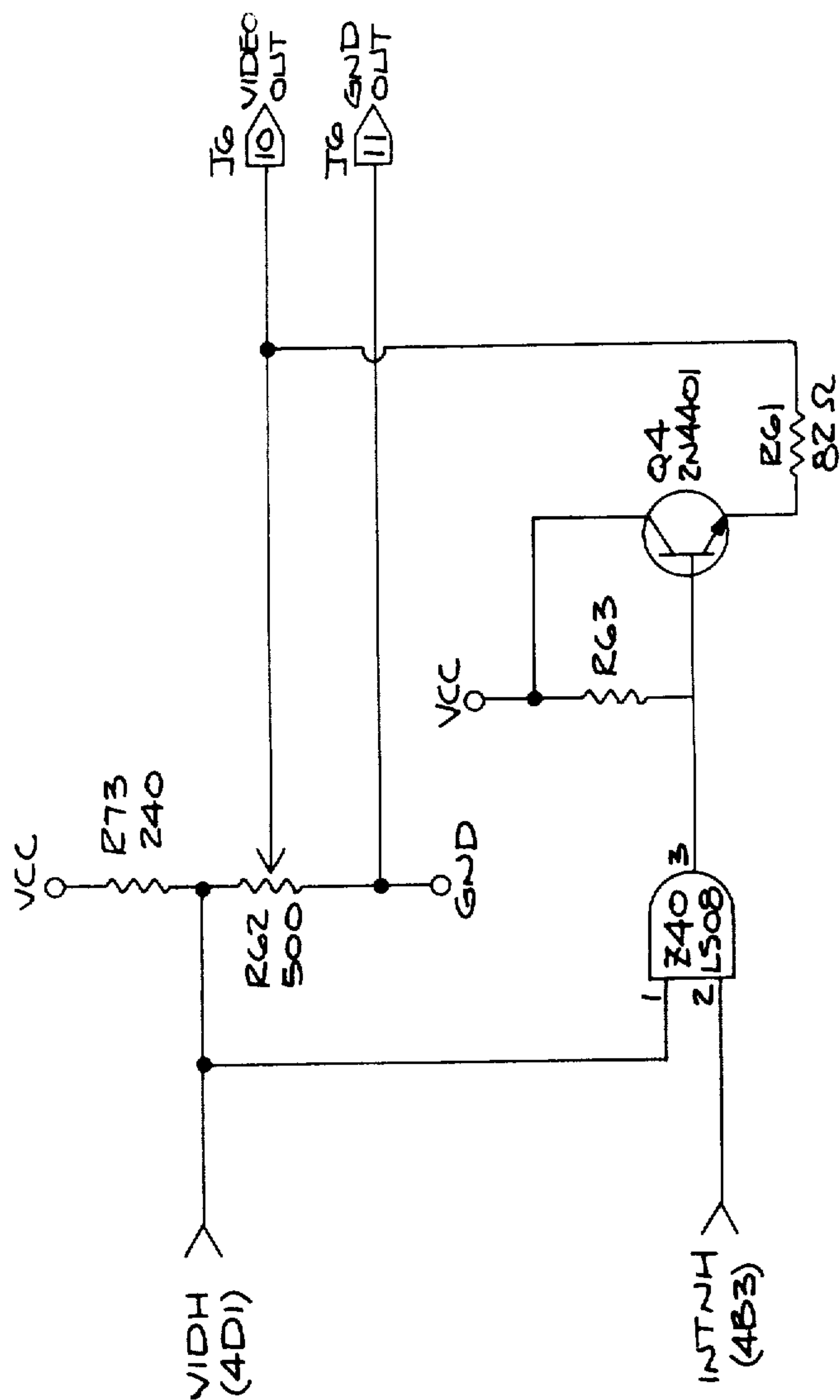
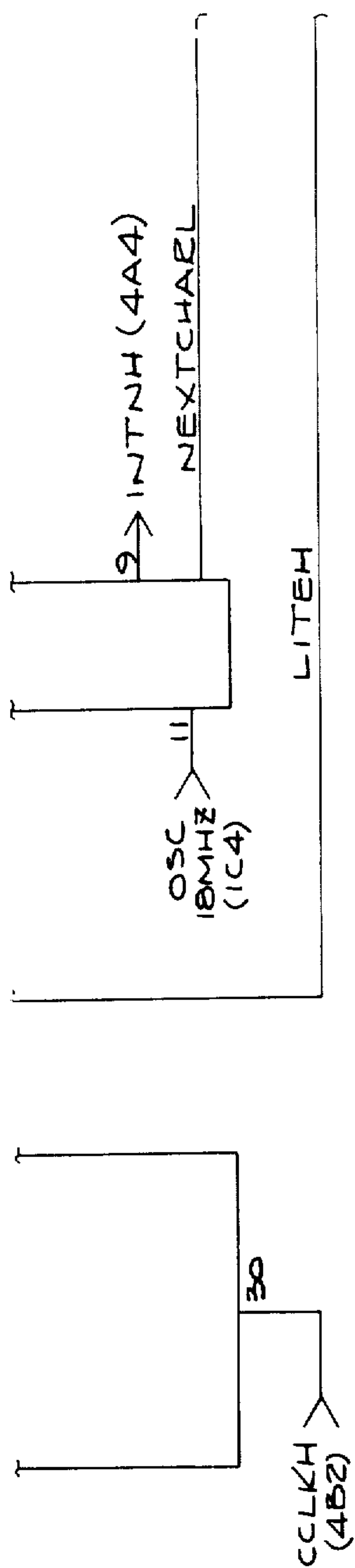


FIG. 27C

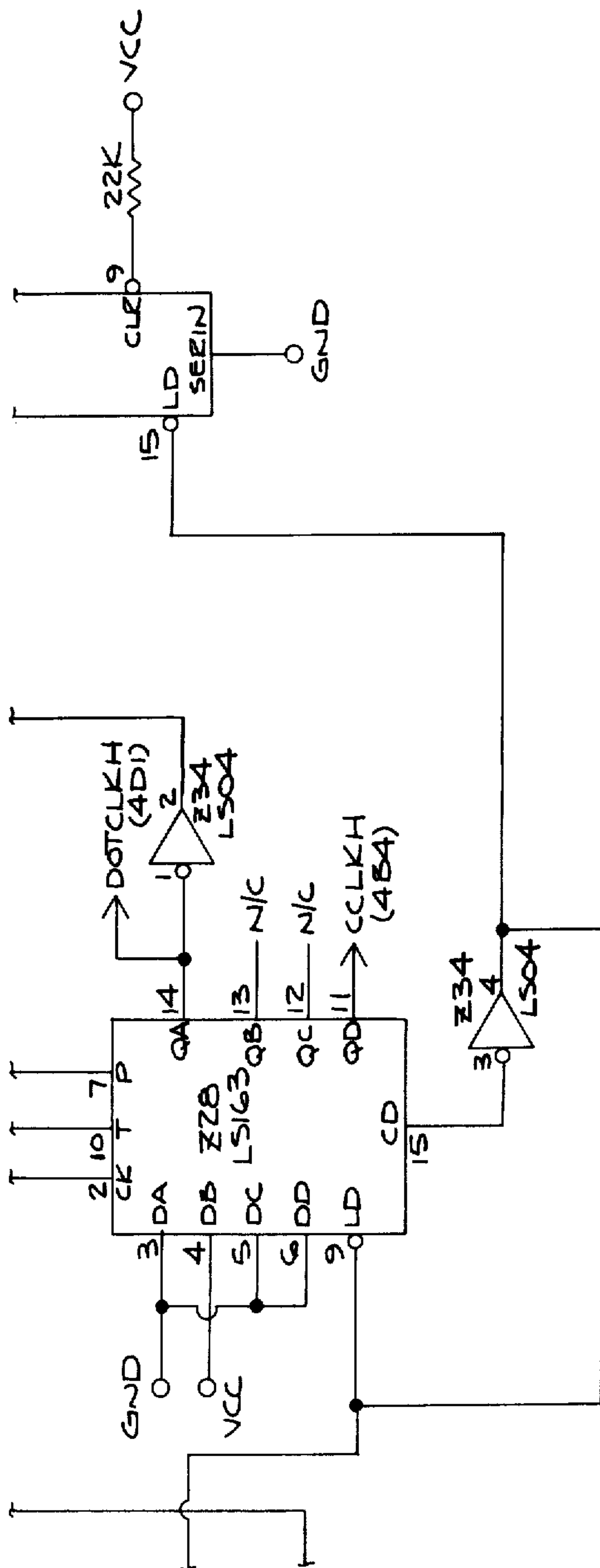


FIG. 27D

FIG. 28A

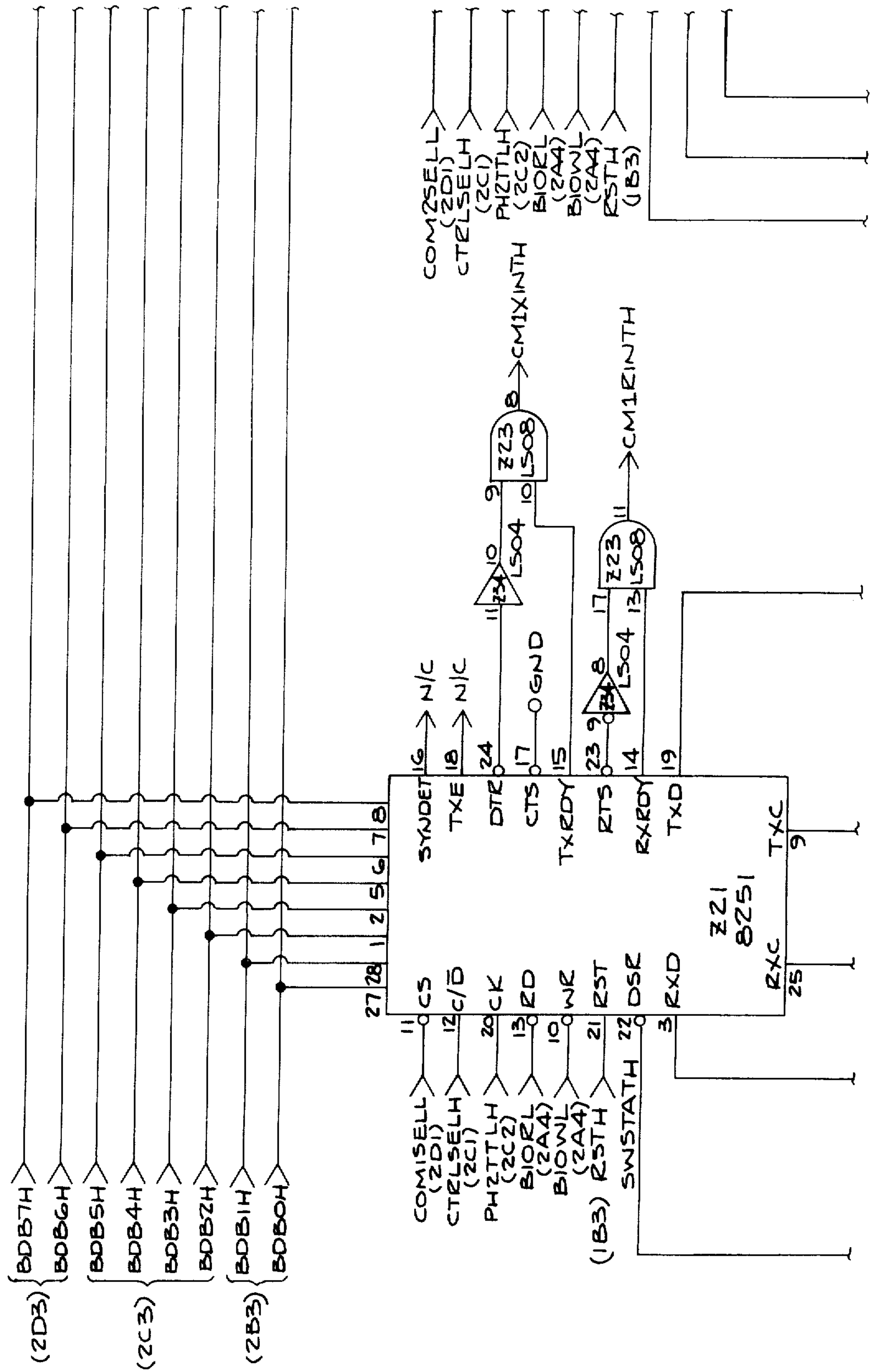
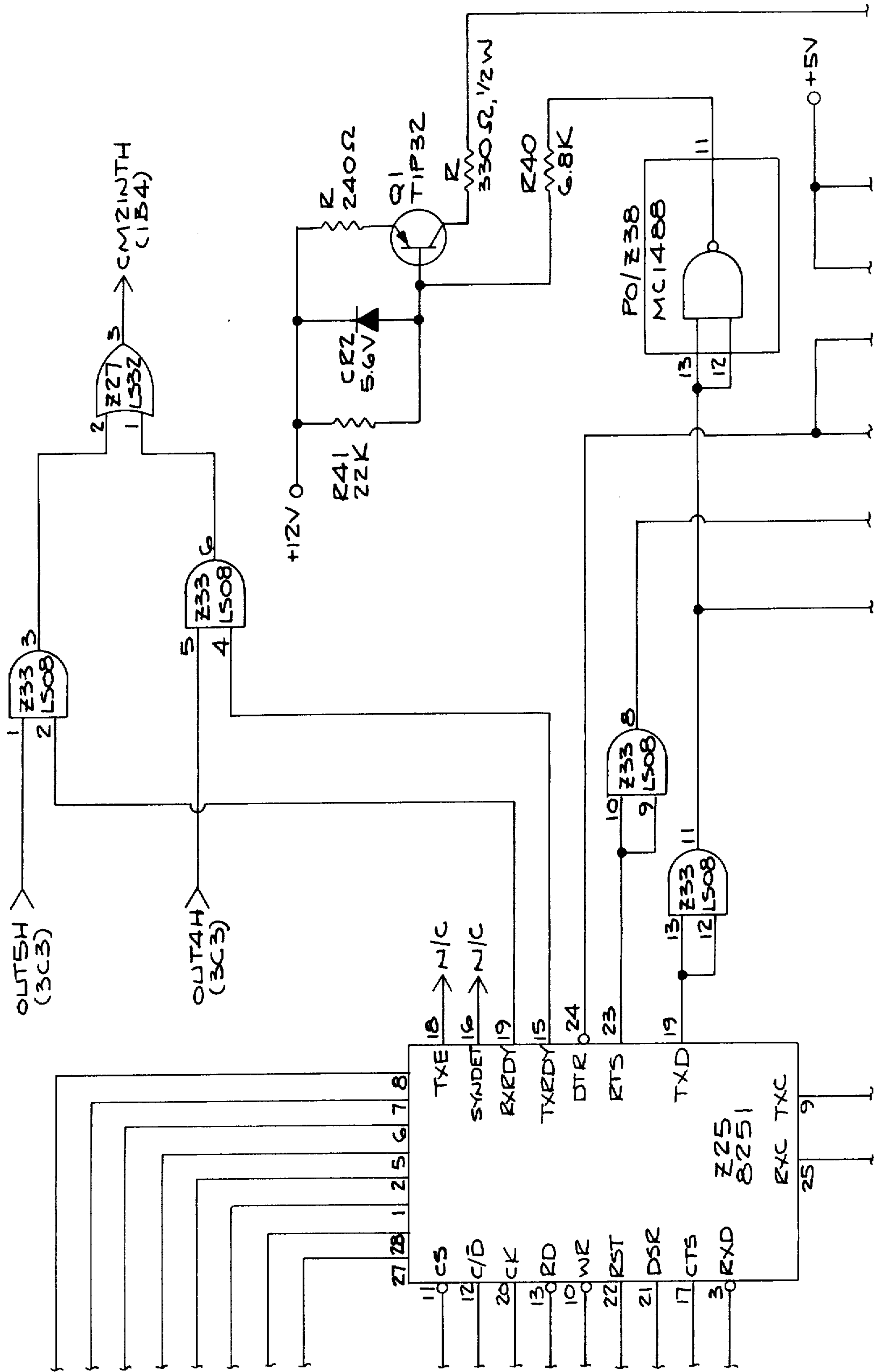


FIG. 28B



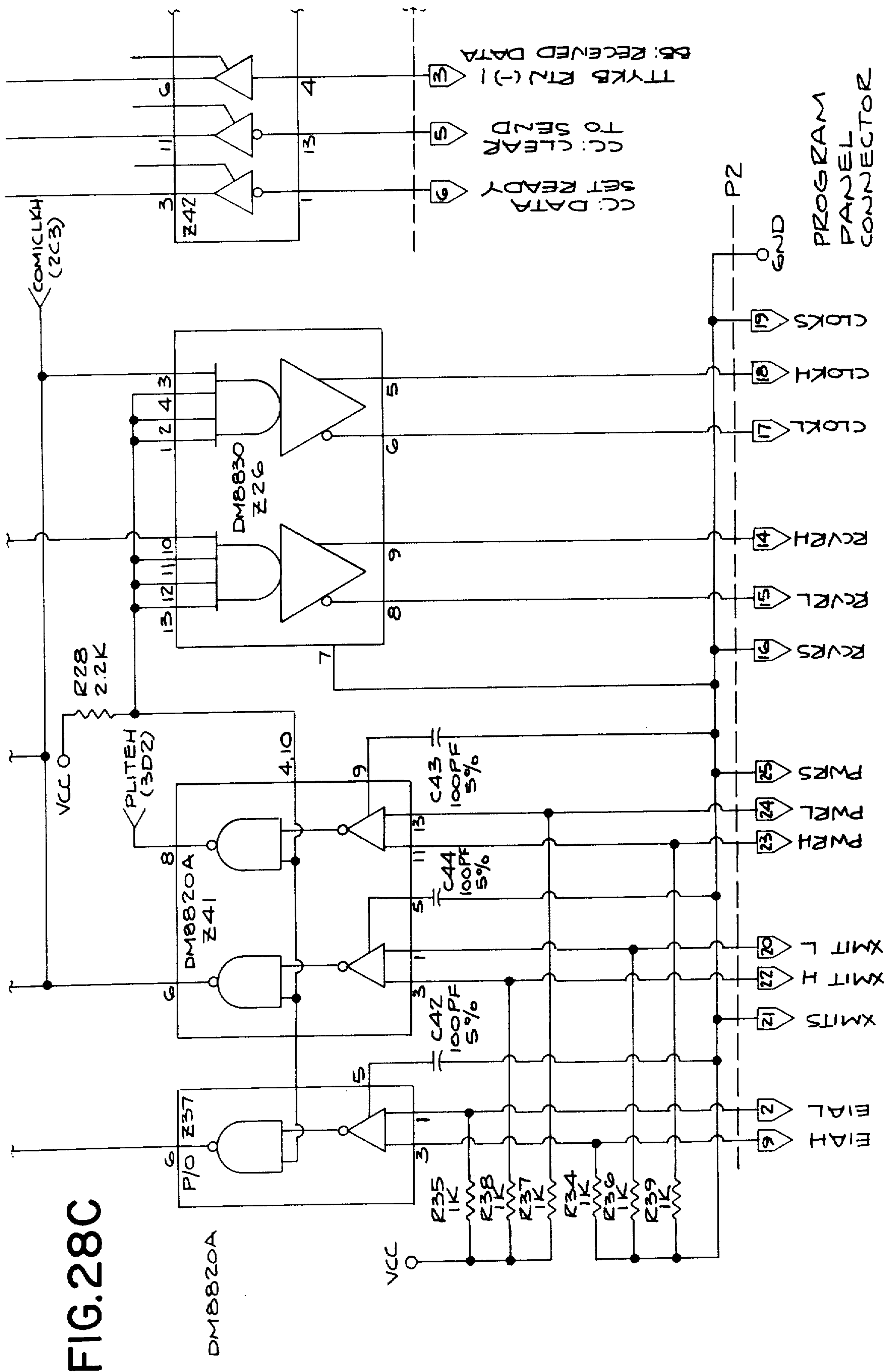


FIG. 28C

CC: DATA SET READY
CC: CLEAR TO SEND
TYKB RTN (-) 1
SS: RECEIVED DATA

PROGRAM PANEL CONNECTOR

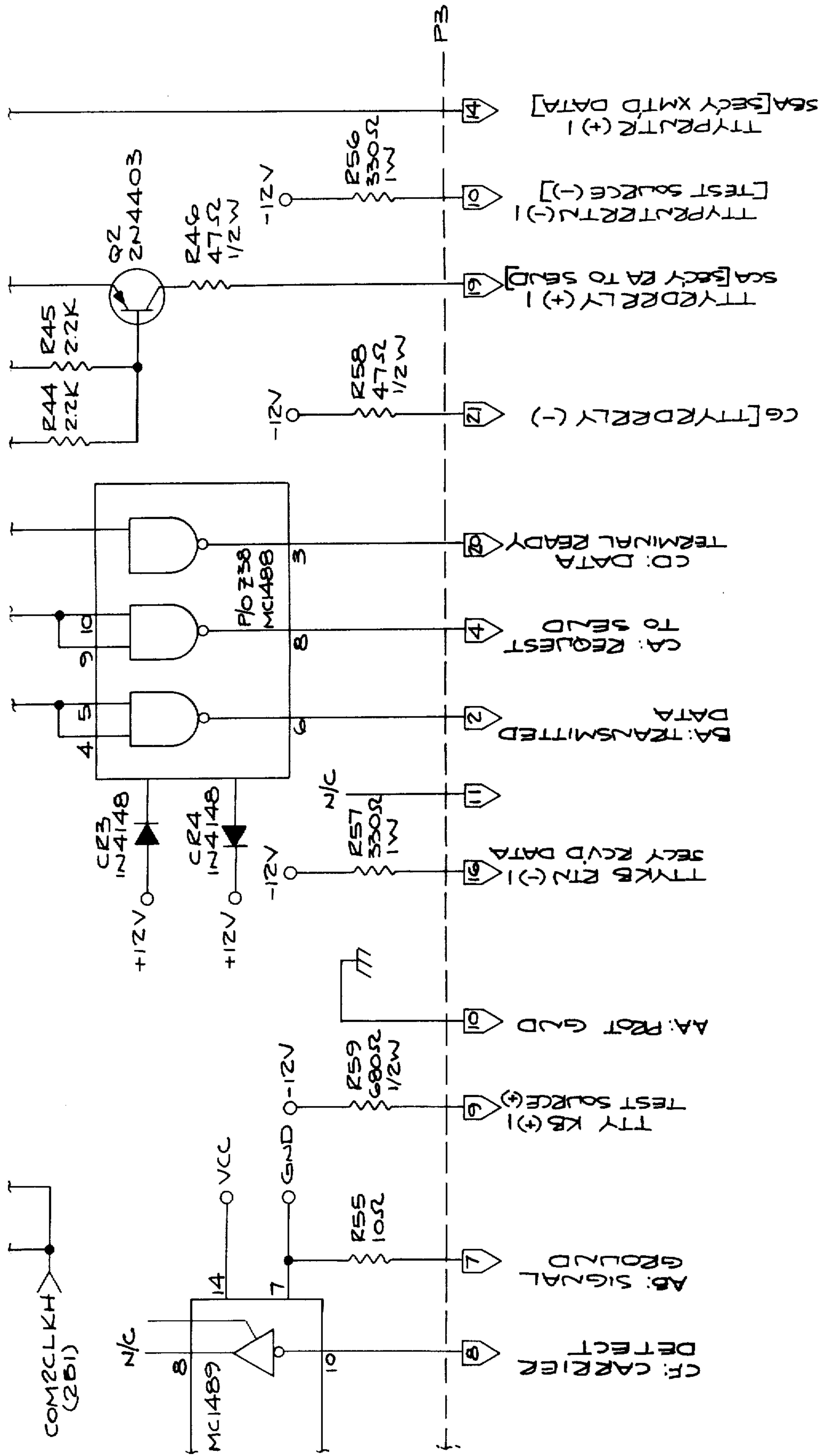


FIG. 28D

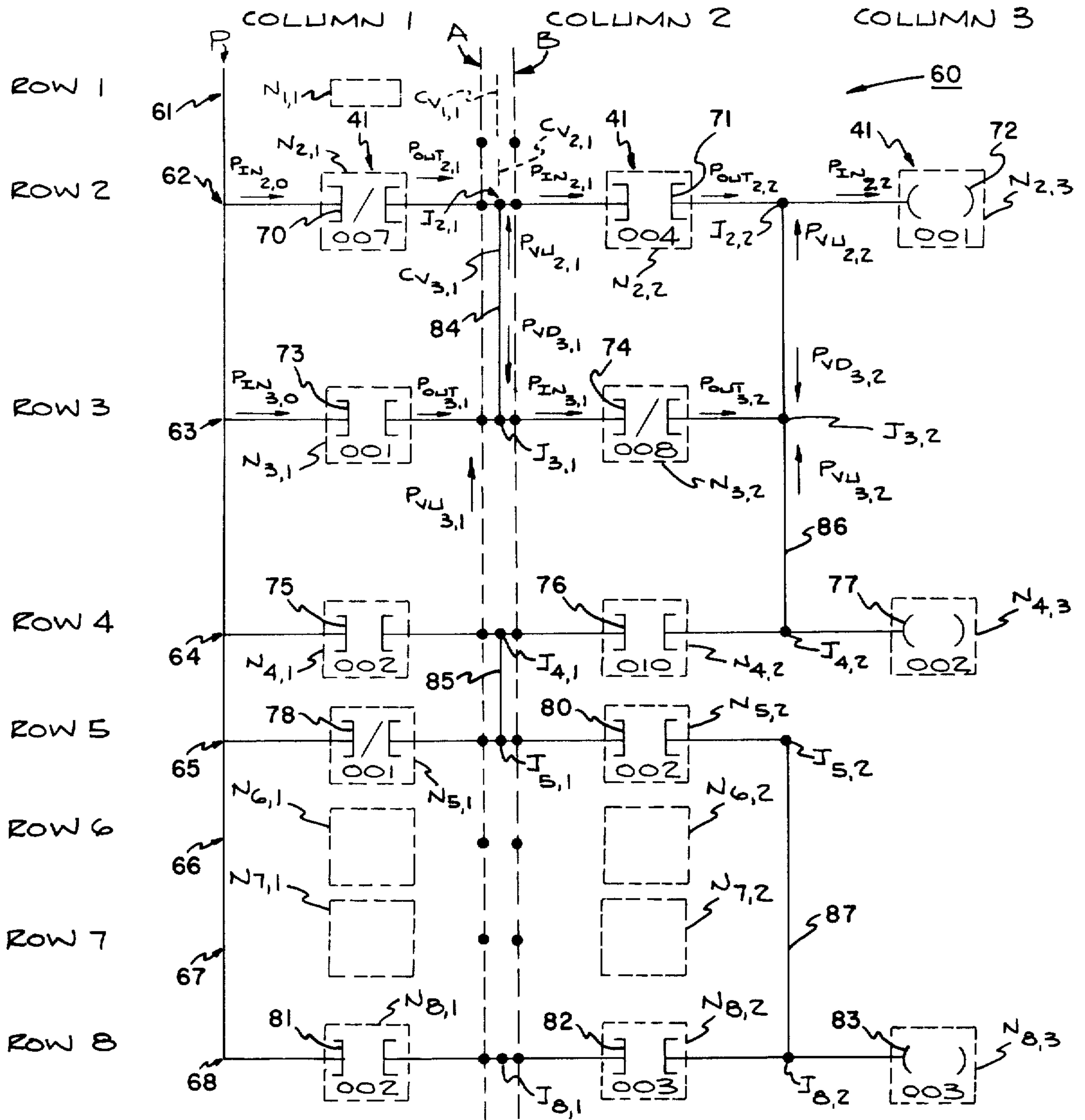


FIG. 29

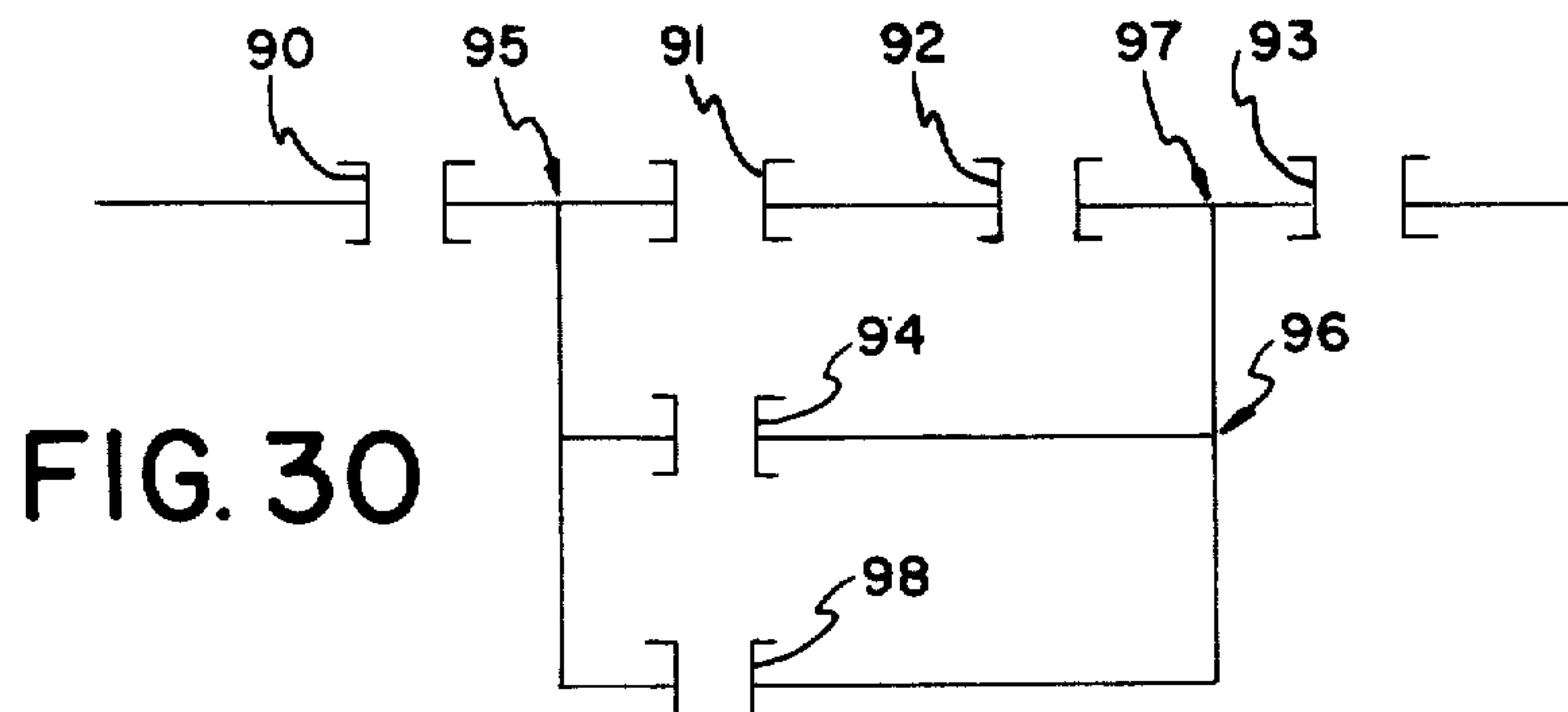


FIG. 30

PROGRAMMABLE CONTROLLER

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to programmable controllers used in industrial control applications such as those found to control material handling, metal cutting, packaging, assembly, batch sequencing, grinding, welding, polymer blending and handling, as well as energy management.

2. Description of the Prior Art

Since the advent of programmable controllers in the early 1970's (such as that disclosed in U.S. Pat. No. 3,686,639), these devices have been able to replace the hard wire relay logic control systems used in many industrial control applications. In the ensuing years, they have become more powerful, replacing not only relay ladder-type control programs, but also performing non-relay functions such as timing and counting, as well as performing data manipulation and transfer such as that disclosed in U.S. Pat. No. 3,930,233. Indeed, programmable controllers have become so powerful in recent years, controlling virtually thousands of outputs and performing many diverse and complicated data manipulation and transfer operations that they in many circumstances can replace the minicomputer for controlling complex industrial control systems. The Modicon 1084 Programmable Controller disclosed in pending U.S. patent applications Ser. No. 646,412 filed Jan. 2, 1976, now abandoned, and divisional application Ser. No. 873,407 filed Jan. 30, 1978, now U.S. Pat. No. 4,162,536 are characteristic of these large, high-powered controllers/data processors.

It has also been found during the relatively short history of the programmable controller that a need existed for small, low cost programmable controllers to replace control programs that would normally utilize eight or more hard-wired relays. It has further been found that it is at times desirable to allow the control engineer to program not only ladder-type control programs with each rung of the ladder representing an electrical circuit line having one or more nodes or contacts and a coil output which may be referenced to other nodes, but also a network of logic lines with interconnections between nodes of adjacent lines. Some companies such as Texas Instruments and Allen-Bradley have provided programmable controllers with programming panels capable of being programmed with control networks which can have interconnections between adjacent lines within the network. However, it has been found that, due to the type of solution employed by these programmable controllers, constraints had to be placed upon the user in terms of the number of vertical connections that could be placed between adjacent lines as well as the number of nodes that could be encompassed within two vertical lines of the control program. The present invention eliminates these problems in prior art programmable controllers by providing a control network without any limitations on the user in terms of the number of vertical interconnections that can be made within the network nor in the arrangement of nodes between vertical interconnections of the network. This is achieved by the utilization of what is called a "column solver" which for each network solves the vertical power flow in both the up and down directions for each node in a column.

The present invention also provides a programmable controller with improvements not found in prior art programmable controllers, such as the capability of inserting one or more networks between two existing networks so as to effectively re-number the remaining networks and thereby insure correct sequential solution of the networks where such a solution is desired.

The output point in the I/O system to which the coil output of a user line references, is assignable by the user and not dictated by line number. This further reduces the constraints placed on the user in formulating his or her control program.

The present invention also provides a programmable controller that has multiple discrete outputs on some calculate functions. These multiple outputs facilitate use of the result of the calculate function by the control engineer. Furthermore, the present invention not only provides for discrete input/output but also register input/output on the same I/O modules for the transferral of data to and from the programmable controller and interconnected devices such as other programmable controllers in a hierarchical control arrangement. In addition, the present invention provides a cursor display on its CRT which allows the user to have the real-time display of power status at any particular node in any selected line of the ladder-diagram network. Specialized search features are also present to the user.

In addition, the present programmable controller is housed in a unique modular arrangement suitable to a rugged industrial environment. The various features of the mechanical aspects of the present invention are disclosed and claimed in a co-pending patent application filed simultaneously with the present patent application; namely, U.S. patent application Ser. No. 883,277, filed May 3, 1978, U.S. Pat. No. 4,215,386.

All of the improvements synergistically combine to provide a low cost, flexible, and easily viable programmable controller.

SUMMARY OF THE INVENTION

An improved programmable controller according to the present invention comprises a power supply and central processing unit (CPU) and memory forming a mainframe enclosed in a first housing, and an input/output assembly having an input/output (I/O) bus interconnected to the mainframe at one end and to one or more I/O housings in a daisy chain fashion. Depending on their length, each I/O housing has from one to four or from one to eight I/O modules. Each I/O module has either four discrete input points or four discrete output points. There are separate I/O modules for AC and DC inputs and outputs. The I/O bus is housed in an I/O duct which provides easy installation as well as effective electromagnetic interference (EMI) protection.

Insertion of a user generated control program is performed by an interconnectable programming panel which allows for the generation of electrical ladder diagram networks up to seven rows in length and eleven columns in width, representing up to 77 nodes. The programming panel in conjunction with the mainframe allows the user to move a cursor to any node in the network with an associated light-emitting diode (LED) on the programming panel indicating the real-time power status of that node.

The CPU further comprises a column solver which solves the vertical power status between adjacent nodes in different lines or rows on a column-by-column basis

interacting with the solution of the nodes by other portions of the mainframe.

The programming panel allows the user to insert one or more networks between two existing networks in such a manner that the networks below the inserted network are effectively pushed down not only on the CRT display but also in the solution order as performed by the mainframe. This feature coupled with the user assignability of coil outputs to any I/O point allows for more effective user programming, especially where solution order of the program is important.

Finally, the programming panel in conjunction with the memory has a percentage memory feature and an associated check count which is stored during a power-down sequence and compared with the count obtained during a power-up sequence in order to prevent the operation of the controller in solving the user networks if the two check counts do not match. This prevents the use of incorrectly stored data in memory in a power-up sequence.

OBJECTS OF THE INVENTION

Therefore, it is a principal object of the present invention to provide an improved programmable controller which is able to generate and solve multi-node electrical ladder-diagram networks in conjunction with a column solver for the rapid and efficient columnar solving of interconnections between adjacent lines of the ladder-diagram network;

It is a further object of the present invention to provide an improved programmable controller of the above description utilizing a CRT programming panel which displays the user generated ladder-diagram networks and which has a user movable cursor that can be placed at any node within the ladder-diagram network for displaying on an associated LED the real-time power status of that node as it is solved by the CPU;

Another object of the present invention is to provide an improved programmable controller of the above character capable of performing calculate functions with multiple outputs so as to facilitate use of the resultant output in other portions of the control program;

A still further object of the present invention is to provide an improved programmable controller of the above character in which the I/O system incorporates one or more I/O housings, each housing connecting with one or more input or output modules which can communicate with the mainframe not only discrete input/output data but also register input/output data for data processing purposes;

Another object of the present invention is to provide an improved programmable controller of the above description which has a programming panel and associated mainframe which allows the user to insert networks between existing networks and which provides for the sequential solution of the inserted networks;

An additional object of the present invention is to provide a programmable controller of the above character having coil I/O assignability independent of its line and network location;

Another object of the present invention is to provide a programmable controller of the above character having specialized search techniques to facilitate monitoring and de-bugging of the user program.

A still further object of the present invention is to provide a programmable controller of the above character which generates a check count during a power-down sequence indicative of the contents of memory

and to generate a second check count during a power-up sequence representative of the same status of the memory and to prevent operation of the controller if the two check counts are not the same;

Other objects of the present invention will in part be obvious and will in part appear hereinafter.

THE DRAWINGS

For a fuller understanding of the nature and objects of the present invention, reference should be made to the following detailed description and the accompanying drawings, in which:

FIG. 1 is a perspective view of the programmable controller according to the present invention illustrating the housing enclosing the mainframe comprising the central processing unit, memory and power supply, the I/O duct housing the I/O bus for communicating between the CPU and the illustrated I/O housings interconnected to the I/O bus and in turn housing up to eight I/O modules, each module being either an input or an output module and intercommunicating at four points with external devices, and further illustrating the programming panel interconnected to the mainframe housing by a front mounted connector for user monitoring, programming and debugging of the control program as generated by the user on the programming panel;

FIG. 1A is a perspective view of a portion of the mainframe housing and I/O system showing the I/O duct with its front cover removed and illustrating interconnection of the I/O bus with the I/O housings.

FIG. 1B is a diagrammatic block diagram of the programmable controller shown in FIG. 1;

FIG. 2 is a plan view of the keyboard, LED, and portion of the CRT display of the programming panel shown in FIG. 1;

FIG. 3 is an illustration of the top level subsystem hierarchy of the programmable controller shown in FIG. 1;

FIG. 4 illustrates a typical electrical ladder-diagram network that may be programmed by a control engineer with the programming panel shown in FIG. 1;

FIG. 5 illustrates another typical electrical ladder-diagram network that may be programmed on the programming panel;

FIG. 6A illustrates the CRT format for both the user network and status/assembly areas;

FIG. 6B illustrates the status/assembly area for a normally open contact with a vertical interconnection;

FIG. 6C illustrates the status/assembly area for a normally open contact and a START function;

FIG. 6D illustrates the status/assembly area for a normally open contact with memory protect;

FIG. 6E sets forth the legend for the symbols used in FIGS. 6A-6D;

FIG. 7 illustrates the displays generated by the programming panel CRT for a selected node when various changes to the node are made by the user;

FIGS. 8A-8H illustrate the assembly portion of the CRT display when a search function is desired utilizing various parameters of the control program;

FIG. 9 is a control flow diagram of the mainframe software of the programmable controller shown in FIG. 1;

FIG. 10A is a data flow diagram of the mainframe software during normal operation of the programmable controller following startup;

FIG. 10B, is a data flow diagram similar to that shown in FIG. 10A representing the data flow during power-down and power-up operations;

FIG. 11 is a timing diagram for the mainframe of the programmable controller shown in FIG. 1;

FIG. 12 is a software state diagram for the mainframe of the programmable controller shown in FIG. 1;

FIGS. 13A-13D are schematic diagrams of the memory addressing counters and read gates of the central processing unit shown in FIG. 1;

FIG. 13E is a diagram showing how FIGS. 13A-13D are placed together;

FIGS. 14A-14D are schematic diagrams of the I/O interface of the CPU;

FIG. 14E is a diagram showing how FIGS. 14A-14D are placed together;

FIGS. 15A-15D are schematic diagrams of the control select logic of the CPU;

FIG. 15E is a diagram showing how FIGS. 15A-15D are placed together;

FIGS. 16A-16D are schematic diagrams of the processor and program ROM interface of the CPU;

FIG. 16E is a diagram showing how FIGS. 16A-16D are placed together;

FIGS. 17A-17D are schematic diagrams of the peripheral port and scratchpad of the CPU; and

FIG. 17E is a diagram showing how FIGS. 17A-17D are placed together;

FIGS. 18A-18D are schematic diagrams of the connectors used in the central processing unit shown in FIG. 1;

FIG. 18E is a diagram showing how FIGS. 18A-18D are placed together;

FIGS. 19A-19D, 20A-20D, 21A-21D, 22A-22D, and 23A-23B are schematic diagrams of the memory boards for storing the user ladder-diagram network, coil data and register data, this memory schematic diagram forming a portion of the central processing unit of the programmable controller shown in FIG. 1;

FIGS. 19E, 20E, 21E, 22E, and 23C are diagrams showing how FIGS. 19A-19D, 20A-20D, 21A-21D, 22A-22D, and 23A-23B are respectively put together;

FIGS. 24A-24D, 25A-25D, 26A-26D, 27A-27D, and 28A-28D are schematic diagrams of the programming panel shown in FIG. 1;

FIGS. 24E, 25E, 26E, 27E, and 28E are diagrams showing how FIGS. 24A-24D, 25A-25D, 26A-26D, 27A-27D, and 28A-28D are respectively put together;

FIG. 29 is a diagrammatic view of a user network illustrating how the column solver functions; and

FIG. 30 is a diagrammatic view of another user network which can pose difficulties for prior art programmable controllers.

DETAILED DESCRIPTION

GENERAL DESCRIPTION

As best seen in FIGS. 1-1A, 1B and 2, a programmable controller 20 according to the present invention includes a housing 22 enclosing a mainframe 39 comprising a central processing unit 31, memory 21, and a power supply 37 for providing DC power to the remainder of the programmable controller. The housing includes a power indicator 23, a run indicator 24, a memory protect key lock switch 25, a utility AC connector 26, and a peripheral port connector 27. As shown on FIG. 1A, a battery low light 51 may also be used to show when battery backup power is low. The peripheral port connector provides intercommunication

between the programmable controller and a programming panel 29 by means of a cable (not shown).

The programmable controller further includes an I/O system 28 comprising an I/O duct 30, I/O bus 32, I/O housings 33, and I/O modules 34. I/O duct 30 houses the input/output bus 32 (see FIGS. 1A and 1B) which interconnects the mainframe with each of the interconnected I/O housings 33 depending from the I/O duct. Each I/O housing incorporates from one to eight I/O modules 34 each module being an input module or output module for either AC or DC voltages. Each I/O module has four output points or input points for interconnection with discrete external devices or, when operating in a register I/O mode, with data processing devices such as minicomputers or hierarchical programmable controllers. The programmable controller in its maximum configuration can control 256 discrete outputs and respond to up to 256 discrete inputs. These additional I/O points are provided by additional I/O modules housed on additional I/O housings not shown in FIG. 1. Indeed, the duct 30 may be extended on both the sides shown in FIG. 1 as well as below housing 22 in order to provide for the additional I/O housings and modules. In addition to the programming panel 29 that may be interconnected to the peripheral port connector 27, a tape loader, other CRT programming panels, and a monitoring computer may all be connected through connector 27 by means of a peripheral port adapter 35.

The full range of the programmable controller is diagrammatically shown in FIG. 3 which illustrates the various subsystems of the controller and the various interconnections between the subsystems and the external world.

The mainframe is an integral assembly within housing 22 containing a processor, E5 (see FIGS. 16A-16D), read-only memory (ROM), a resident executive program, battery backed up random access memory (RAM), a resident user program and interfaces to the I/O programming panel 29, other peripheral devices and to the I/O system 28. As best seen in FIGS. 1 and 2, the programming panel 29 consists of a cathode ray tube (CRT) 36, a keyboard 38, and an LED power status light 40, all of which is supported by a microprocessor (see FIGS. 22A-28D) as more fully discussed later. The programming panel displays the user generated program in terms of one or more networks such as shown in FIGS. 4 and 5, each network comprising up to seven electrical ladder-diagram rows or rungs containing nodes comprising user selected elements which may be interconnected vertically as more fully described later. The programming panel further displays the power status and register contents and permits changes to the control program.

Thus, the basic programmable controller according to the present invention performs logic solution processing which interfaces to I/O, a programming panel and other peripherals. The mainframe memory 21 includes a minimum of 256 bytes of user memory which allows the user to nominally program 96 nodes in his or her electrical ladder-diagram networks including 64 discrete inputs, 64 discrete outputs, 64 internal coils, and 62 holding registers. Registers are represented as 12-bit binary quantities in the CPU and are converted to three decimal digits for display on programming panel 29 and to three binary coded decimal digits (BCD) for I/O via a register multiplexer. For limited register data transferal discrete I/O modules may be used with the CPU

software making the necessary BCD to binary and binary to BCD conversions. The user instruction set includes relays, latches, timers, counters, all represented on a multi-node seven row by eleven column program format per network as best seen in typical networks shown in FIGS. 4 and 5. The programmable controller can additionally perform register I/O up to 32 input and 32 output registers and transitional contacts sensing true to false or false to true transitions as well as calculate functions with multiple outputs and step sequencers. The user memory can also be extended from 256 bytes up to 4,096 bytes.

Functional Description

CONTROLLER MAINFRAME

The controller mainframe 39 within housing 22 performs the processing necessary to convert inputs to outputs in accordance with the user's control program. It contains an interface to the I/O bus 32 and a serial interface 27 for communication with peripherals such as programming panel 29. Control and indicators consist of the run light 24, a power O.K. light 23, a battery low light 51 and a memory protect switch 25. Physically, the mainframe is approximately six inches deep, fifteen inches wide and eighteen inches high and can hang vertically from mounting screws and is normally intended for installation within an eight inch NEMA cabinet. It is packaged in a drip-proof enclosure and cooled by convection; thereby making it suitable for harsh industrial environments. The mainframe CPU scans and solves the user program once every twenty milliseconds maximum, and the system can support up to 256 discrete inputs, 256 discrete outputs and register I/O. The CPU software, as described more fully later, cycles continuously. Appendix A sets forth the entire mainframe software.

In each cycle it reads all field inputs, executes a logical transfer function defined by the user entered program which relates inputs to outputs, and generates field outputs accordingly. In addition, the software interfaces the CPU to the programming panel and/or additional EIA devices via the peripheral port adapter 35. This interface accommodates changes to the user entered program and provides output status information for display on the programming panel 29. The user program represented on the programming panel is in the form of a relay ladder-diagram network having nodes including normally open and normally closed switches, open and shorted connections both vertically and horizontally, timers and counters, transitional contacts, arithmetic functions including add, subtract, multiply and divide, sequencers, and binary-to-BCD and BCD-to-binary converts.

The field inputs consist of up to 256 discrete points, four per input module 34, each with a state of ON or OFF, plus of up to 32 words of register data. Each word of register data represents a binary number in the range of ϕ to 999 (base 10). These values are read into the controller from the I/O bus 32. BCD to binary conversion is made by the register multiplexer. All inputs are read at least once every 20 milliseconds.

Field outputs consist of up to 256 discrete points each with the state ON or OFF plus up to 32 ten-bit words of register data. These values are sent from the controller to the I/O bus 32 and are generated at least once every 20 milliseconds based on completing execution of the user program.

The mainframe contains a peripheral port 27 whose purpose is to interface to the programming panel or via a peripheral port adapter 35 to any EIA protocol device. The CPU accepts commands and data from this port whose purpose is to modify the user program residing in the controller, to alter the controller's state or to extract data from the controller. This data may either be a portion of the user program or the state of the programmable controller.

For all transfers of information, the peripheral device such as the programming panel 29 initiates a command and the controller mainframe responds thereto. This is true even for power data. Redundant bits are transmitted to aid in detecting transmission errors.

In addition, the mainframe displays operational and non-operational status via the run light 24. This light is ON whenever the executive program within the controller is being executed properly and is OFF when the executive program is halted due to a power failure, failure of onboard diagnostics, or other intermediate failures. All discrete outputs are turned OFF in the event of such failure and remain OFF until primary power has been cycled on in the power-up sequence.

The mainframe senses the status of the memory protect keylock switch 25. If the memory protect is engaged, attempts to change the user program by the programming panel are not permitted and result in transmission of error code.

The mainframe displays proper power supply output via the power O.K. light 23.

The basic CPU processing can be set forth in five systems:

- (1) power-up, power-down,
- (2) logic solving;
- (3) peripheral port I/O handling;
- (4) field I/O handling; and
- (5) onboard diagnostics.

Upon power-up, the CPU executes a set of appropriate diagnostic tests to insure that the hardware is functioning properly. If these tests fail, the system halts, leaving data in predetermined locations of memory identifying what has failed. If these tests are passed, then the following sequence occurs:

- (1) all outputs are set OFF with the exception of latches and disabled outputs which were ON when power was last removed, these outputs retain their ON state;

- (2) read all inputs; and

- (3) illuminate the run light 24 and start solving the user logic.

Upon an indication of imminent power failure, appropriate parameters are stored to permit orderly start-up of the programmable controller.

The CPU interprets the user program data base and generates field outputs based on field inputs as determined by the contents of the data base. The instruction set and syntax of the interpretive language used to represent the user's relay ladder-diagram networks in the data base is set forth below. Details of the operation and representation of various instructions, addressing conventions, and range constraints also appear below.

INSTRUCTION SET

The instruction set of the programmable controller includes the following:

- (1) relays-normally open, relays-normally closed, horizontal open, horizontal short, vertical short, vertical open;

(2) timers, 0.1 second, 1 second, and ϕ . ϕ 1 seconds, 3 BCD digit magnitude;

(3) counters, 3 BCD digit magnitude;

(4) coil, latched or unlatched; may be disabled ON or OFF;

(5) transitional relay contacts conduct ON with a transition from OFF and ON or conduct ON on a transition from OFF to ON of the designated reference;

(6) sequencer stepping switches;

(7) binary-to-BCD and BCD-to-binary converts;

(8) calculate $B + C = D$;

(9) calculate; $B - C = D$; three discrete outputs; one output ON if B greater than C, a second output ON if $B = C$, a third output ON if B is less than C;

(10) $B \times C = D$; one discrete output always equal to the logical value of input I1; (see Table 10C)

(11) $B \div C = D$; one discrete output ON if the division is proper, a second discrete output ON if there is a dividend overflow, and a third discrete is ON if the divisor is equal to zero.

The syntax for the instructions is a ladder-diagram network of a maximum size of eleven column by seven rows as best seen in FIGS. 4 and 5. Coils appear only in the right-most column of the network on any or all of the rows. All coils are latchable and coils and inputs may be disabled ON and OFF from the programming panel. Coil designations for output I/O points is independent of the line or network number.

An important aspect of the present invention is the order of solution of the user program. The user program is solved in a sequential network basis and is from left to right by column within each network. This left-to-right column solution is performed in part by a column solver described more fully later which defines the input power status to the next node in a line based upon the output power from the node to its immediate left as well as any power transferred by vertical interconnections to that line from adjacent lines.

The I/O serviced at the end of each scan solving all of the user networks and includes an update of both inputs and outputs. The network order is under the control of the user and thus, a network may be inserted between networks in a situation where the sequential order of the solving of the networks is important to the control engineer.

The CPU performs data validity checking necessary to insure that all register values, address, and reference number values are within valid ranges and that all operation codes are valid. An invalid instruction is prevented from being entered into the user memory by the CPU. If, in the process of executing the user program an invalid instruction or an invalid random access memory check sum or a stuck I/O bit is encountered, the CPU processing is halted; i.e., discrete outputs are dropped and logic solution ceases.

PROGRAMMING PANEL SUBSYSTEM

As shown in FIG. 1, the programming panel 29 provides the primary operator/user interface for determining the functions to be performed by the programmable controller. The programming panel is a small portable device having a rugged CRT display 36 and a small dedicated function keyboard 38. The CRT displays one or more networks representing relay ladder-diagrams. The display shows a seven by eleven array of nodes containing contacts or function blocks. The system provides near real-time power display for one network at a time; however, since the network is updated less

frequently than the scan time of the CPU for solving the network, it is possible that beating between the CRT refresh rate and the scan rate can result in spurious displays of power for an oscillating contact. This is overcome by the programming panel having a true real-time power display light-emitting diode (LED) 40 which displays the power for a selected contact in the displayed network as selected by the user with a cursor. The network includes a numeric key pad and a set of function buttons enabling the user to enter, edit and delete portions of his or her program.

The programming panel enables the user to enter, modify and delete logic networks as well as to monitor registers and discrete I/O points.

FIGS. 4 and 5 illustrate how a network of the control program is displayed on the CRT. Each line of the userlogic program uses two rows of display on the CRT. The lower of the two rows indicates the contact type inserted at a particular column within a particular line by the user. The two lines define a series of nodes 41, each node including a contact type element such as normally open contact 42 in the lowermost row of the display and a reference number to that contact in the uppermost row such as the number 1 shown for the upper left-handmost node of FIG. 4. The references to the elements within nodes 41 can be any coil and need not be in the sequential order shown in FIGS. 4 and 5. Horizontal connections between adjacent nodes is made by dashed lines 43 while vertical interconnections between adjacent nodes in different lines is made by dashed vertical lines 44. By use of the dashed vertical lines, it is readily apparent that user programs need not have a coil output for each line but may reference nodes from one line to vertically higher or lower nodes of other lines.

A cursor 47 (shown by dashed slanted lines) is available under user control by means of switches 45 (see FIG. 2) to move the cursor from node to node on the network. The cursor is displayed by a reverse shading with respect to the remainder of the CRT display. The "current network" is defined as that network on the programming panel CRT which is identified by having the cursor positioned somewhere within the network. If the cursor is not positioned on any network, no network is current. Power flow is indicated by an intensified vertical and horizontal power connections and is displayed for the current network. The start of a network as indicated by a break in the left hand power rail 46 as shown in FIGS. 4 and 5.

It is readily apparent that networks need not be rectangular in shape due to the vertical interconnections available. However, they will occupy a rectangular area on the CRT display. Thus, a network whose largest column is five elements deep (that is it includes five rows) requires an eleven-by-five array on the screen. Unused elements in a network are displayed as blank areas. Vertical opens and horizontal opens are defined as used elements.

Networks are displayed on the screen only if the entire network can fit on the screen. As scrolling causes networks to shift on the screen, any network that cannot be completely displayed is blanked out from the screen.

The programmable controller does not allow the user to insert via the programming panel more data than the controller has memory to hold. Any attempt to do so results in an error code displayed on the CRT.

The lower two lines of the CRT screen form the status/assembly area. The status/assembly area consists of seven sections; all sections arranged vertically. Typical status assembly area format is shown in FIGS. 6A, 6B, 6C, 6D and 6E.

As also shown in FIGS. 6A-6E, one of the status/assembly areas displays discrete data which allows up to a maximum of six data values to be displayed from the programmable controller as shown by the six groups of NNNN. The first line is labeled "REF" and contains the reference numbers for the items being displayed. The second line is the current value of those reference elements and is labeled "VAL". Reference elements may be holding registers, input registers, discrete inputs and outputs, or internal coils. If the reference is for a register value, the current contents of the register are displayed as a four digit value. If the reference is for an I/O point, the first position of the value field contains either a D or a blank. The D indicates that the contact is disabled. The other three characters in the field are either OFF or ON which is the state of the contact. References are placed in the discrete display area via the cursor which may be placed on any of the six reference locations.

A second status/assembly area is designated "USED" with a number beneath it which indicates the number of bytes of memory that is filled by the user's control program. This number is automatically updated as changes are made in the user data base.

Another of the seven areas displays a step number (Step #) and is the position or number of the current network shown on the CRT display. It indicates the order of solution of this user network with respect to the other networks. A step number of "N" implies that there are "N-1" networks which precede this network in the data base and in the solution order.

A fourth area is the error field. It is normally blank. It is used only when the panel has an error message to display as shown in the status/assembly area by "EEEEEEEEEEEE". The error field is cleared by the first error reset key 48 shown in FIG. 2. A fifth area is the advisory field shown by "AAAAAAAAAA". It is used to display a status message. The message indicates to the user that activity is taking place during extended execution time such as a search or enter function as explained later in this description. It also indicates that the programming panel is waiting on the availability of a peripheral port. The advisory field is cleared when the message is no longer applicable.

The SHIFT field is a sixth area of the status/assembly and is shown by "S" which is normally blank. It contains the letter "S" only after the shift key 49 (FIG. 2) has been struck. It remains on the screen for only the next key stroke. It indicates that the next key stroke will be interpreted as a shifted key stroke as shown by the upper level indicia on some of the keys of keyboard 38.

The last area is the assembly area. This area is on the extreme lower left-hand side of a six-by-two character array which is used to build the contact-type, reference number and vertical connections of a node. It is shown in FIG. 6A as "CCCCVRRRRV"; as defined in the legend of FIG. 6E.

The LED 40 shown in FIG. 2 generates a real-time display of the status of the power output of any one node in the current network as selected by the cursor position.

As shown in FIG. 2, the keyboard 38 is the user input device of the programming panel. It consists of a set of dedicated keys and a set of keys which may be used in

conjunction with shift key 49. The keys may be divided into three basic types; data keys, 40, cursor control keys 45, control keys 52 and function keys 54.

DATA KEYS

The data keys, 50 shown in FIG. 2 are defined as those keys which are entered into the assembly area. They consist of contact types and numbers. The data keys are set forth in Table 1 with an indication of the key that is used, its name and the symbol on the CRT display.

The assembly area is a six-ty-two array of characters which represents the contact, reference number and vertical connection currently being keyed by the user. The assembly area is not entered into the controller memory until a proper FUNCTION key is struck.

Data is keyed into the assembly area in a simple manner. Numerics cause the current reference number to be shifted left one position and a new character to enter the least significant digit. Contact-type and vertical connectors replace the current value in the assembly area for that type. The data in the assembly area is retentive; i.e., it is not cleared unless the CLEAR key is struck. The reference data area is filled with leading zeros when a new numeric key is depressed following operation of any function key that uses a numeric argument from the assembly area as discussed later in this specification.

TABLE 1

KEY	NAME	SYMBOL
0-9	Numeric	0-9
-{ }	Normally Open Relay	-{ }
-{ }-	Normally Closed Relay	-{ }-
-{ ↑ }	Positive Going Contact	-{ ↑ }
-{ ↓ }	Negative Going Contact	-{ ↓ }
-(-)	Coil	-(-)
-(L)-	Latch	-(L)-
:	Vertical Open	:
!	Vertical Short	!
..	Horizontal Open	..
.-	Horizontal Short	.-
Shift 0	Counter	CTR
Shift 1	Timer - 0.01 sec.	T.01
Shift 2	Timer - 0.10 sec.	T 0.1
Shift 3	Timer - 1.0 sec.	T 1.0
Shift 7	Add	+
Shift 4	Subtract	-
Shift 9	Multiply	×
Shift 6	Divide	÷
Shift 8	Convert	CON

CURSOR CONTROL KEYS 45

The programming panel supports four cursor control keys as set forth in Table 2 below.

The cursor 41 (see FIG. 4) wraps around horizontally on the CRT screen but does not have vertical wrap-around.

If the cursor crosses from one network to another, the new network is re-fetched from the controller and becomes the current network.

Unrestricted cursor movement is permitted throughout the user logic display and the discrete display area. The cursor location is indicated by a reverse video image of the cursor location. Each cursor position is a six-by-two array of characters on the screen.

FUNCTION KEYS

Function keys cause activity to occur within the programmable controller. Table 3 describes the function keys and the key stroke or keystrokes used to generate them.

The ENTER function moves data from the assembly area to the cursor position on the screen and updates the controller memory. No changes are made on the screen until the change is made in the controller memory. Three restrictions are imposed:

- (1) reference numbers must be valid for the node type and controller capacity;
- (2) certain node replacements are not valid; and
- (3) placement of nodes along a network has certain restrictions.

TABLE 2

KEY	NAME
↑	Move cursor up one position
↓	Move cursor down one position
→	Move cursor right one position
←	Move cursor left one position

TABLE 3

KEY	FUNCTION	SYMBOL
ENTER	Move data from assembly area to position indicated by cursor.	ENTER
START NEXT	Create a new network in the controller following the current network.	START
DELETE	Delete node at cursor position.	DELETE NODE
SHIFT DELETE	Delete current network from data base.	DELETE NETWORK
SEARCH	Using data in assembly area, search for a match beginning with the first network.	SEARCH
SHIFT SEARCH	Using data in assembly	SEARCH

	area, search for a match beginning at the current cursor position and network.	CONTINUE
GET NEXT	Fetch the network following the current network to the panel.	GET NEXT
GET PREV	Fetch the network preceding the current network to the panel.	GET PREVIOUS
CLEAR	Blank the assembly area.	CLEAR
SHIFT CLEAR	Blank the entire screen.	CLEAR ALL
GET	Fetch the status of the contact or register specified by reference part of the assembly area.	GET
DISABLE	Invert the status of the enable/disable flag for an input, output coil, or internal coil indicated by the cursor.	ENABLE/ DISABLE
FORCE	Invert the state of the contact specified by the cursor if disabled.	CHANGE STATE

TABLE 3-continued

KEY	FUNCTION	SYMBOL
SUPERVISORY	Enter supervisory state.	SUPERVISORY
ERROR RESET	Resets error condition	ERROR RESET

When a modification of an existing node is attempted, only that data currently in the assembly area is used. A field which has not been defined is not modified. An undefined field is maintained as null reversed video in the assembly area. A defined field reverts to normal video at the start of entry. FIG. 7 illustrates the display in the assembly area, the contact at the cursor, and the result at the cursor when modifications to an existing node are made.

Reference numbers must be valid for the node type and the controller capacity. For example, if a controller has 62 registers and an attempt is made to reference register 4063, an error code is generated. Valid references are defined for discrete I/O and register space for each programmable controller. The controller validates all changes before changing any user logic. Changing contact types is allowed under the rules set forth in Table 4.

Because programming is performed on line (that is, while the controller is operating) and because even partially entered programs must be interpretable by the controller, there are some restrictions on the order of entering nodes in a network. Thus, the first node programmed must always be at the top left-hand corner of the network. The next node programmed may be either adjacent below or adjacent to the right of the first node. Programming thus continues, observing the following rules:

TABLE 4

OLD CONTACTS	NEW CONTACTS	RULES
Non-CTR/TMR/CALC	Non-CTR/TMR/CALC	No Restrictions
Non-CTR/TMR/CALC	CTR/TMR/CALC	Allowed at node (row) if node (I + 1,J) and node (I + 2,J)* are blank horizontal open, or horizontal short and 1 + 2.LE.8.
CTR/TMR/CALC	Non-CTR/TMR/CALC	Not Allowed
CTR/TMR/CALC	CTR/TMR/CALC	One for one replacement allowed.

*for CALC only

(1) there may be no unprogrammed nodes to the left of the rightmost programmed node in the top row;

(2) for any programmed node in the top row, a column may be extended below it without regard for the presence of nodes in the column to the right or left.

If the cursor is positioned in the reference display area of the screen, the ENTER key will move the reference number to the VALUE area and update the reference register in the controller. ENTER may be used only with a register already referenced in the reference area. The ENTER key does not function if memory protect is enabled.

START NEXT

The START NEXT key is used to create a new network in the controller memory. Networks are inserted into the data base after the current network. If the cursor is on a network whose network (step) number is N, the network number of the new network is N + 1. Networks are inserted at the beginning of the

logic data base by using the CLEAR key to reset the network number and then the START NEXT places the new network at the start of the data base. The new network has a network number of 1. When START is depressed, the START INDICATOR in the status area is loaded with the word "START" and space is made on the CRT display for the new network. If the insertion takes place at other than the start of logic, the network is built on the screen after the current network. A blank line is preserved with the cursor pointing to the leftmost position of the line. If there is a network on the screen after the old current network having a step number that does not immediately follow the old current network, it is shifted down one line if possible. If this causes part of the network to disappear, this entire network is removed. If the old network is at the bottom of the screen and occupies the last row, the screen is shifted up to create space. Only if the old current network occupies seven rows is it removed from the screen. Insertions at the start of the data base have an empty screen on which to compose logic as this is accomplished by the CLEAR key.

When a new network is created, the network number on the CRT is updated and the new network is then designated the current network for power display purposes. The START key does not function if memory protect is enabled.

DELETE

The DELETE key removes the current node from the data base in the controller. Nodes may be deleted only at the bottom of a column. A node in the top row may be deleted if there are no contacts to the right of it. This is necessary to preserve the integrity of the data base. Deleting a multi-node contact (TIMER/COUNTER/CALCULATE) results in all the nodes of that contact being deleted. The deletion may take place only in the PRESET node for timers and counters and the "B" node for calculate functions.

A user may delete all contacts in a network and still not delete the network itself. The DELETE NETWORK function must be used to delete the entire network. A network with no nodes is displayed as a line with a START OF NETWORK indicator and null nodes across the remainder of screen. A null network occupies one line on the screen. The DELETE key does not function if memory protect is enabled.

DELETE NETWORK

The DELETE NETWORK function removes the current network from the logic data base. The current network is removed from the data base and the area on the screen occupied by the network is blank. The cursor remains in the blank space. The remainder of the screen is not altered. The network number is set to zero. The DELETE NETWORK key does not function when memory protect is enabled.

SEARCH

The SEARCH function is used to fetch networks satisfying specified parameters to the panel. The SEARCH function is implemented using the contents of the assembly area to form a mask and object data. SEARCH commences at the start of the logic data base and continues sequentially until either a match is found or the end of user logic is reached. The elements of the assembly area form the search arguments. Any element left blank is assumed to be not important in finding a

match. The elements which are defined are compared against the user logic until a match is found. Examples of assembly areas that are used to clarify these SEARCH functions are set forth in FIGS. 8A-8H and indicate that a search can be made for the first node, for the first occurrence of a particular contact-type, for the first occurrence of a particular reference number, for the first occurrence of a vertical connector, for the first occurrence of a contact-type having a vertical connector, for the first occurrence of a contact-type with a particular reference number, for the first occurrence of a particular reference number with a vertical interconnection, and for the first occurrence of a particular node.

If the SEARCH is successful, the network containing the matched node is put on the bottom of the CRT screen along with its network number. The network is designated as the current network and a power display is activated for it. The screen display of other networks is shifted upwards to make room for the new network. The cursor is placed on the node which was the match for the search. If the search fails, an error code is displayed in the error code section of the CRT/assembly area.

The SEARCH function thus provides a powerful tool to the control engineer when a control program is first generated and for later monitoring and de-bugging. It is an improvement over prior art controllers that allowed the user to scroll through the control program lines or to trace to a line to which a node in a current line was referenced. Such trace and scroll functions are disclosed in U.S. Pat. No. 3,944,984.

SEARCH CONTINUE

The SEARCH CONTINUE function performs the same function as the SEARCH function except that the search is started at the cursor position. The search operates in a top-to-bottom scan down each column and moves from left to right in a network. All search arguments and return codes are the same as for the SEARCH function.

GET NEXT

The GET NEXT key causes the network following the current network in sequence of solution to be fetched to the panel and treated as the current network. If there are no networks on the screen, the first network in the data base is retrieved.

A check is first made to ascertain whether the network to be fetched is already on the screen. If it is, the cursor is placed on that network. It is also re-fetched from the controller to verify its contents. If the network is not already on the screen, it is fetched from the controller data base. If there are no more networks in the controller, an error code is returned. Placement of the next network on the screen is determined by the following rules:

Unless already on the screen, the next network is placed below the old current network on the screen. If any network exists on the screen below the old current network having a step number (network number) that does not immediately follow the old current network, it is pushed downward to make room. If any part of this network disappears, the entire network is removed from the screen. If the next network fills the portion of the screen below the old current network and more space is required, the old current network and any networks above it on the screen are pushed

upward. Only complete networks are allowed on the screen.

The GET NEXT function causes the network number to be updated on the screen. Power display is made for the network. The cursor is placed in the upper-left-hand corner of the network.

GET PREVIOUS

The GET PREVIOUS key causes the network before the current network in sequence solution to be fetched to the panel and treated as the current network. If there are no networks on the screen, the last network in the data base is fetched. A check is first made to ascertain whether the network to be fetched is already on the screen. If it is, the cursor is moved to that network. The network is also re-fetched from the data base to verify its contents. If the network is not already on the screen, it is fetched from the controller data base. If the current network is the first network, an error code is generated to indicate that there are no more previous networks. Placement of the new work on the screen follows this rule:

Unless it is already on the screen, the previous network is placed on the screen above the old current network. If any networks exist on the screen above the old current network having a step number that does not immediately precede the old current network, they are shifted upward to make room. If any part of these networks disappear, the entire network is removed from the screen. If the previous network fills the space above the old current network, the old current network and any networks beneath it are shifted down. If any part of these networks disappear, the entire network is removed from the screen. As mentioned earlier, only complete networks are displayed.

The GET PREVIOUS key causes the network number to be updated on the screen. The power display for the new network is initiated. The cursor is placed in the upper-left-hand corner of the network.

CLEAR

The CLEAR key is used to blank the assembly register. All previous contents of the composition area are removed. The assembly register is returned to reversed video, nulled condition. No other portion of the display is affected.

SHIFT CLEAR

The SHIFT CLEAR key is used to reset the entire display. The assembly area is blank. The error code is cleared. The user logic space on the screen is set to all blank. The network number is set to zero. The cursor is placed in the top left corner of the screen.

Following a SHIFT CLEAR key depression, certain keys have different functions as defined in TABLE 5.

The SHIFT CLEAR key has no affect on the controller data base. It is a panel command only that returns it to a virgin state.

TABLE 5

Key	Function
START NEXT	Insert network at start of data base.
GET NEXT	Fetches first network from data base.
GET PREV	Fetches last network from

TABLE 5-continued

Key	Function
	data base.

TABLE 6

1 - EXIT
2 - STOP
3 - GO
4 - INITIALIZE
5 - DUMP
6 - LOAD
7 - VERIFY

GET

The GET key permits references to be monitored. The GET key requires that a proper reference number exist in the reference portion of the status/assembly area. The reference number is moved to the discrete display REF line specified by the cursor and the referenced value is then updated at the screen refresh rate. The GET function does not change any data base values. The cursor must be positioned in the discrete display area or an error code is generated. The display of sequences step references (2xxx) is not allowed, although the sequencer register may be monitored. If the reference is to a register (3xxx or 4xxx), the number below is the contents of the register. If the reference is to a contact, a "D" in the first position indicates that the point is disabled. The words ON or OFF then refer to the current state of the contact.

DISABLE

The DISABLE key is used to enable and disable discrete I/O points. Each input point and each output point may be enabled or disabled. If a point is enabled, its state is that which is determined by the controller. An input is the sense of the input channel as determined during the I/O sweep.

A disabled point cannot be changed automatically by the system. It may be changed via the FORCE key. Disabled points retain their state through power failure. Disabled coils are indicated by a " ~ " in the network. An input point enabled/disabled is enabled/disabled globally.

The DISABLE key complements the disable state of the point. If the point was enabled, it is disabled. If the point was disabled, it is enabled. The point is indicated by the cursor. The cursor must be pointing to an I/O reference in the discrete display area. All points are initially enabled. The DISABLE key does not function if memory protect is enabled.

FORCE

The FORCE key is used to change the state of discrete I/O points. It is designed to be used with the DISABLE key. An I/O point may be forced unless it is disabled. Enabled points are redefined by the next controller I/O sweep.

FORCE complements the state (ON/OFF) of the discrete point indicated by the cursor. It works only on relay or coil type nodes. Reference to other node types or to relays not disabled causes an error code to be generated.

If the discrete point is ON it is turned OFF. If the discrete point is OFF it is turned ON. FORCE does not function if memory protect is enabled.

SUPERVISORY

The SUPERVISORY key places the programming panel in a supervisory state. Table 6 is displayed on the CRT display when the SUPERVISORY key is depressed. The programming panel remains in the supervisory state until an exit function is executed. A function is executed by striking the numeric key corresponding to the function. All other keys are invalid.

ERROR RESET

When the programming panel detects an error during normal operations, a message is displayed in the error message portion of the status/assembly area (see FIGS. 6A and 6E). The keyboard is then locked out from the user until the ERROR RESET key is struck. This clears the error message and allows normal processing to resume.

REFERENCE NUMBER CONVENTIONS

Reference numbers are used to identify I/O points, internal coils, sequencer states, input registers, and holding registers. By convention, the reference number is four digits long except when used as a constant in which case it is three digits long. Table 7 defines the reference number conventions.

PROGRAMMING PANEL NODE TYPES

Relays/Coils/Shorts/Opens/Sequencers

Relays, coils, shorts, opens and sequencers are single node elements in the programmable controller. They are called single node elements because all information about them is expressed in one node in the data base. Tables 8A through 8H respectively define a normally open relay, a normally closed relay, a positive transitional relay, a negative transitional relay, a coil, a latch, a horizontal short, and a horizontal open.

TABLE 7

RANGE	USE
0001-0256	DISCRETE OUTPUTS
0257-0512	INTERNAL COILS
1001-1256	DISCRETE INPUTS
2YXX	SEQUENCER STATES - STEP XX OF SEQUENCER Y.
3XXX	INPUT REGISTERS - NUMBER XXX
4XXX	HOLDING REGISTERS - NUMBERS XXY

TABLE 8A

RELAY - Normally Open				
Symbol: --] [--	XXXX	Input Power	XXXX State	Result
		0	0	0
		0	1	0
		1	0	0
		1	1	1
XXXX = 0001-0256 DISCRETE OUTPUT*				
0257-0512 INTERNAL COIL				
1001-1256 DISCRETE INPUT				
2YXX SEQUENCER STATE				

TABLE 8B

3.5.4.1.2 RELAY - Normally Closed				
Symbol: --] [--	XXXX	Input Power	XXXX State	Result
		0	0	0
		0	1	0
		1	0	1
		1	1	0

TABLE 8B-continued

3.5.4.1.2 RELAY - Normally Closed				
Symbol: --] [--	XXXX	Input Power	XXXX State	Result
		0	X	X
		1	0	X
		1	1	0
		1	1	1
XXXX = 0001-0256 DISCRETE OUTPUT				
0257-0512 INTERNAL COIL				
1001-1255 DISCRETE INPUT				
2YXX SEQUENCER STATE				

TABLE 8C

RELAY - Positive Transitional					
Symbol: --] ↑ [--	XXXX	Input Power	XXXX State	XXXX Previous State	Result
		0	X	X	0
		1	0	X	0
		1	1	0	1
		1	1	1	0
XXXX = 0001-0256 DISCRETE OUTPUTS					
0257-0512 INTERNAL COILS					
1001-1256 DISCRETE INPUTS					
2YXX SEQUENCER STATE					

TABLE 8D

RELAY - Negative Transitional					
Symbol: --] ↓ [--	XXXX	Input Power	XXXX State	XXXX Previous State	Result
		0	X	X	0
		1	0	0	0
		1	0	1	1
		1	1	X	0
XXXX = 0001-0256 DISCRETE OUTPUTS					
0257-0512 INTERNAL COILS					
1001-1256 DISCRETE INPUTS					
2YXX SEQUENCER STATE					

TABLE 8E

COIL			
Symbol: - () -	XXXX	Input Power	Result
Enabled			
		0	0
		1	1
Disabled			
		0	0
		1	1
XXXX = 0001-0256 DISCRETE OUTPUT			
0257-0512 INTERNAL COIL			

TABLE 8F

LATCH			
Symbol: - (L) -	XXXX	Input Power	Result
Enabled			
		0	0
		1	1
Disabled			
		0	0
		1	1
XXXX = 0001-0256 DISCRETE OUTPUT			
0257-0512 INTERNAL COIL			

TABLE 8G

HORIZONTAL SHORT		
Symbol:	Input Power	Result
	0	0
	1	1

TABLE 8H

HORIZONTAL OPEN		
Symbol:	Input Power	Result
	0	0
	1	0

SEQUENCER REFERENCE

Normally opened, normally closed, and relay transitional contacts may refer to a sequencer. The referencing of sequencers is in the following form:

2YXX where Y, in the range of 1 to 8 represents the sequencer register (405Y). The XX is in the range of 01 to 32 and is the sequence step. 8 sequencer registers are provided in the programmable controller numbered 4051 through 4058.

When a reference to a sequencer is encountered, the "XX" portion of the node is compared to the proper sequencer register (defined by "Y"). If the two values are equal, the solution is true (normally open nodes pass power, normally closed nodes do not pass power). Otherwise the solution is false (normally open nodes do not pass power, normally closed nodes do pass power). If the contents of the sequencer register is zero or is greater than 32, all references are false.

TIMERS AND COUNTERS

Timers and counters are 2-node elements. The symbol for the counter is shown in Table 9A and the symbol for the timer is shown in Table 9B. The nodes are arranged vertically. The top node is the preset value while the bottom node is the holding register where counts are accumulated. Each element has two inputs and two outputs. When input EI is activated the holding register is incremented for a counter and clock pulses accumulated for a timer. Input RI is the reset line. When RI is false, the holding register is cleared regardless of the state of EI. Output EO is true if the contents of the holding register is greater than or equal to the preset value. Output RO is always false.

TABLE 9A

COUNTER		
Symbol:	Input Power	Result
	0	0
	1	0

RI	BI	REGISTER ACTION	Ro	Eo
0	X	4XXX ← 0	0	0
1	0	No Change	0	0
			0	0 1 if 4XXX . GE . Preset
1	1	4XXX ← 4XXX + 1	0	0 If 4XXX . LT . Preset
			0	1 If 4XXX . GE . Preset
PRESET	XXXX = 0000-0999	NUMERIC CONSTANT		
	3XXX	INPUT REGISTER		
	4XXX	HOLDING REGISTER		

If PRESET is a numeric content, it is compared directly against the contents of the holding

TABLE 9A-continued

register.
If PRESET is a register (3XXX or 4XXX), the contents of the register are compared against the contents of the holding register.

TABLE 9B

TIMER		
Symbol:	Input Power	Result
	0	0
	1	0

RI	BI	REGISTER ACTION	Ro	Eo
0	X	4XXX ← 0	0	0
1	0	No Change	0	0
			0	0 1 if 4XXX . GE . Preset
1	1	4XXX + No. of ticks since Last pass	0	0 If 4XXX . LT . Preset
			0	1 If 4XXX . GE . Preset
PRESET	XXXX = 0000-0999	NUMERIC CONSTANTS		
	3XXX	INPUT REGISTER		
	4XXX	HOLDING REGISTER		
HOLDING REGISTER	= 4XXX			
TIMER VALUE	TXXX = 1.0 One Second Timer			
	0.1 Tenth Second Timer			
	.01 Hundredths Second Timer			
	If PRESET is a numeric value, it is compared directly against the contents of the holding register.			
	If PRESET is a register value, its contents are compared against the contents of the holding register.			

CALCULATE FUNCTIONS

All calculate functions (add, subtract, multiply and divide) are 3-node elements. Tables 10A, 10B, 10C and 10D describe the add, subtract, multiply and divide functions respectively. The top node of each function is the "B-node" and must reference a register. The middle or "C-node" may be either a register or a constant. The bottom or "D-node" is a register reference. The general format for a calculate function is that the B node is operated on by the C node with the result placed in the D node.

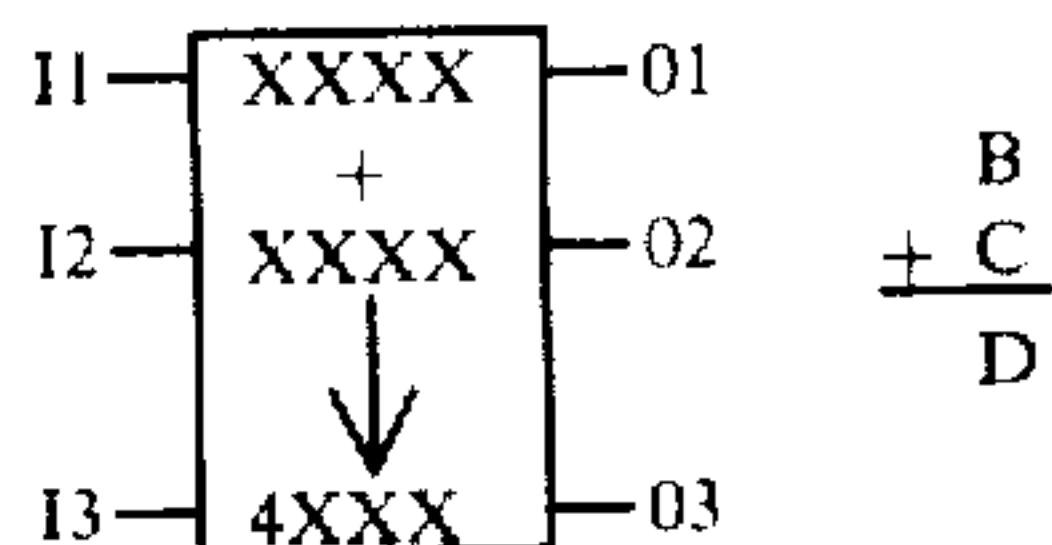
Each element has three possible input lines and three possible discrete output lines. Input I1, when true, activates the function. Inputs 2 and 3 are ignored. Such multiple output calculate functions are unique in the programmable controller art. By use of multiple discrete outputs the user is able to more easily and definitively utilize the result of a calculate function in his or her control program. Thus, for example, in the subtract mode, the three discrete outputs—only one of which may be true at any particular time depending upon the result of the subtract operation—may be used to indicate to other portions of the control program the result of the calculation by means of binary on and off states.

Similarly, in the division function the first output indicates whether the division was proper while the second and third outputs indicate whether or not various kinds of input errors have occurred. When output 2 is true there is a dividend overflow and when output 3 is true the divisor equals zero. Thus the multiple outputs gives the user more information than just the value of the result of the calculate function as stored in the D register.

TABLE 10A

ADD

Symbol:



01 = 0 = > B + C . LE . 999
 = 1 = > B + C . GT . 999
 02 = No function, always false
 03 = No function, always false

B-NODE = 0XXX NUMERIC CONSTANT
 3XXX INPUT REGISTER
 4XXX HOLDING REGISTER

C-NODE = 0XXX NUMERIC CONSTANT
 3XXX INPUT REGISTER
 4XXX HOLDING REGISTER

D-NODE = 4XXX HOLDING REGISTER

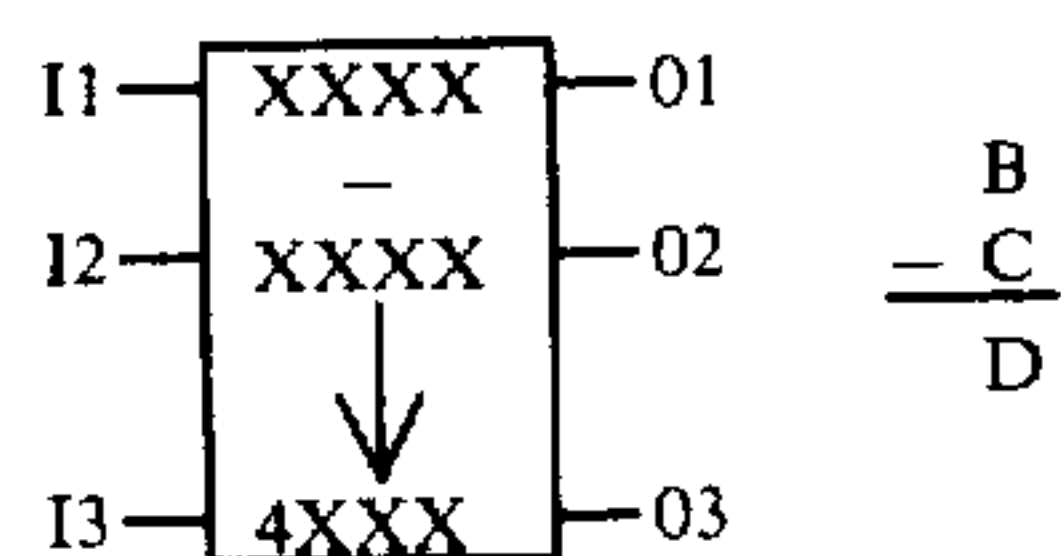
If B + C > 999, the D-Node register receives the result module 1000. For example:

B = 700	B = 700
C = 450	C = 291
B + C = 1150	B + C = 991
D = 150	D = 991
01 = 1	01 = 0

TABLE 10B

SUBTRACT

Symbol:



01 = 0 = B . LE . C
 = 1 = B . G . T . C .
 02 = 0 = B . NE . C
 = 1 = B . EG . C
 03 = 0 = B . GE . C
 = 1 = B . LT . C

B-NODE = 0XXX NUMERIC CONTACT
 3XXX INPUT REGISTER
 4XXX HOLDING REGISTER

C-NODE = 0XXX NUMERIC CONSTANT
 3XXX INPUT REGISTER
 4XXX HOLDING REGISTER

D-NODE = 4XXX HOLDING REGISTER

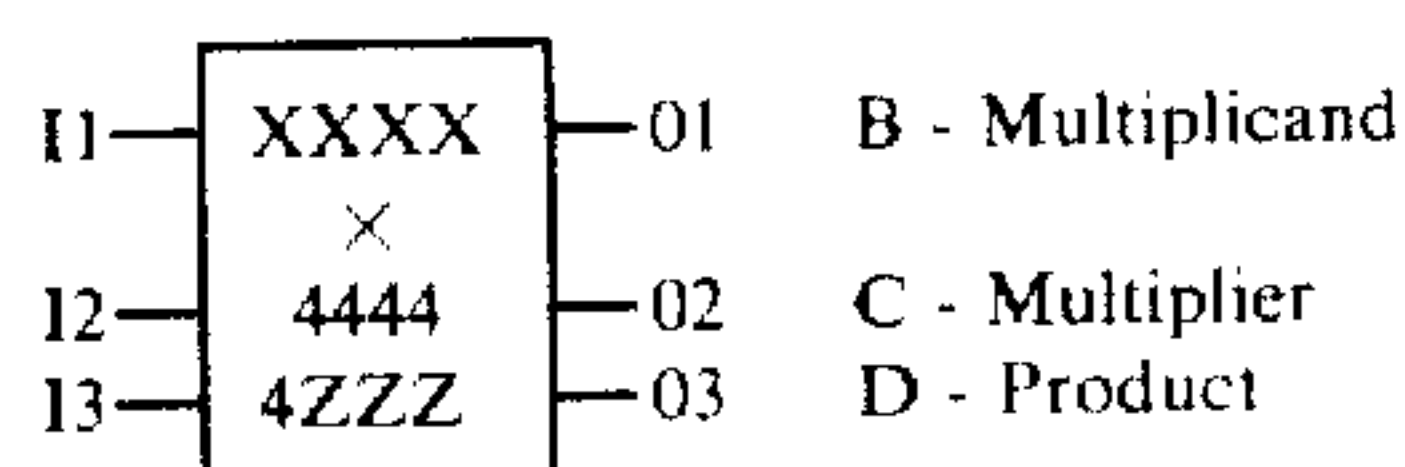
If B . LT . C, the D-Node register contains the absolute value of the result. For example:

B = 700	B = 450	B = 300
C = 450	C = 700	C = 300
B - C = 250	B - C = 250	B - C = 0
D = 250	D = 250	D = 0
01 = 1	01 = 0	01 = 0
02 = 0	02 = 0	02 = 1
03 = 0	03 = 1	03 = 0

TABLE 10C

MULTIPLY

Symbol:



Multiplicand: XXXX = 000-999 NUMERIC CONSTANT
 = 3XXX INPUT REGISTER
 = 4XXX HOLDING REG.

If MULTIPLICAND is a NUMERIC CONSTANT, it's value is used in the multiply. If it is a REGISTER, the

TABLE 10C-continued

contents of the REGISTER are used in the multiply.

Multiplier: YYYY = 000-999 NUMERIC CONSTANT
 3YYY INPUT REGISTER
 4YYY HOLDING REGISTER

If MULTIPLIER is a NUMERIC CONSTANT, it's value is used directly in the multiply. If it is a REGISTER, the contents of the REGISTER are used in the multiply.

Product 4ZZZ Specifies the first of 2 consecutive HOLDING REGISTERS which will contain the Product. Must be HOLDING REGISTER, can NOT BE THE LAST HOLDING REGISTER. The 484/P180 will disallow entry of the Last HOLDING REGISTER as the product register on the multiply node.

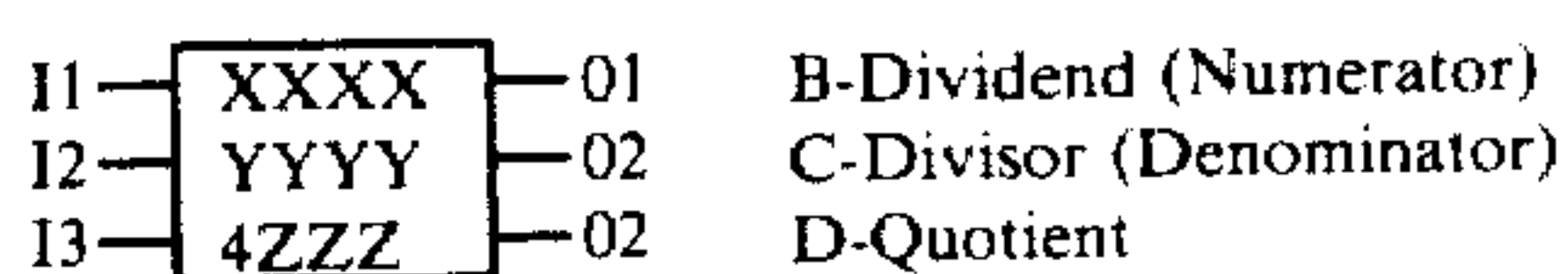
01 is always equal to I1. I2, I3, 02, and 03 are unused.
 Function: When I1 is ON (= 1), multiply the Single Register MULTIPLICAND VALUE by the Single Register MULTIPLIER VALUE. This yields a Double Register (Double Precision) PRODUCT. The most significant three digits (with leading zeros) are stored in REGISTER 4ZZZ, the least significant three digits are stored in REGISTER 4ZZZ + 1.

When I1 is OFF (0) the product is unaffected.

TABLE 10D

DIVIDE

Symbol:



Dividend: The DIVIDEND is a Double Precision (double register) Value.

XXXX = 000-999 NUMERIC CONSTANT
 = 3XXX INPUT REGISTER
 = 4XXX HOLDING REGISTER

If the DIVIDEND is a NUMERIC CONSTANT, the value is used as the LOW ORDER DIVIDEND, with the High ORDER DIVIDEND assumed to be ZERO (0). (i.e. a NUMERIC CONSTANT DIVIDEND is in the range 000000-000999, inclusive.)

If the DIVIDEND is a REGISTER (3XXX or 4XXX), then the REGISTER specified is the first of two REGISTERS to contain the Double Precision DIVIDEND.

The first REGISTER (3XXX or 4XXX) contains the HIGH ORDER DIVIDEND (the most significant three digits), the second REGISTER (3XXX + 1 or 4XXX + 1) contain the LOW ORDER DIVIDEND (the least significant three digits). The REGISTER specified CAN NOT be the last INPUT REGISTER or the last HOLDING REGISTER. The 484/P180 will disallow their use as the Dividend Register on

Dividend Node.
 Divisor: The DIVISOR is a Single Precision (single register) Value.

YYYY = 000-999 NUMERIC CONSTANT
 = 3YYY INPUT REGISTER
 = 4YYY HOLDING REGISTER

Quotient: 4ZZZ = HOLDING REGISTER only.
 The QUOTIENT is a Single Precision.

I1 is ENABLE, I2 and I3 are unused.

01 is DIVISION OR.
 02 is DIVIDEND Overflow.
 03 is DIVISOR = 0.

FUNCTION: When I1 is ON (1), DIVIDE the Double Precision DIVIDEND by the Single Precision DIVISOR, giving a Single Precision QUOTIENT. No remainder or fractional part is kept.

RULES: The DIVISOR > 1000 must be greater than the DIVIDEND. AND, The DIVISOR must be NON-ZERO.

TABLE 10D-continued

Output indications when I1 is ON:	
01 = 1 if DIVIDE performed OK.	
02 = 1 if DIVISOR × 1000 . LE . DIVIDEND, QUOTIENT ← 0.	5
03 = 1 if DIVISOR . EQ . 0. QUOTIENT ← 0.	
If I1 is OFF, the Quotient will be unaffected, and 01, 02, and 03 will be OFF (0).	

ERROR CODES

Error codes are displayed in the error section of the screen. A code is displayed when the programming panel detects an error condition. The code is displayed until the RESET key is struck. The error section on the screen is normally blank (see FIGS. 6A and 6E).

On power-up the programming panel performs certain internal diagnostics to verify that it is capable of functioning. The system software is verified via a ROM check sum test. The RAM in the programming panel is tested via several diagnostics. A mini-instruction test is also performed. If any of these tests fail, the system keeps the screen blank and attempts to sound the system alarm.

SYSTEM ERROR CODE

Systems errors are defined as those error conditions which are internal to the programmable controller and not the result of any user action. They are displayed when they are detected. Table 11 defines the system error codes.

TABLE 11

Code	Meaning
CN	Controller not responding; two seconds have elapsed without a response to a command from the controller.
CE	Communications error; a hard communications failure (16 retries) exist.
TE	Trap error; an internal processor error has been detected.
IK	Illegal keystroke; an illegal keystroke has been sensed.

FUNCTION KEY ERRORS

The function key errors have a lower priority than system level errors. They indicate a malfunction with an attempted function key operation. Table 12 defines the function key errors.

Thus the functionality of the programmable controller according to the present invention has been defined in the preceding pages. It is readily apparent that this programmable controller not only performs those functions found earlier in the programmable controller art but also is able to perform several new functions such as the search function, the multiple output calculate function, the real time power display of a selected node on the CRT panel, and the ability to allow the user to form a multi-node control program with minimal constraints on the format of the network. The circuitry and software necessary for allowing the programmable controller and programming panel to perform these functions is next described.

TABLE 12

Function Key	Code	Meaning
ENTER	MP	Memory protect; memory protect feature is enabled.

TABLE 12-continued

Function Key	Code	Meaning
	IR	Illegal reference number; the reference number is illegal for the node type.
	NC	Not configured; the element referenced is not configured in the controller.
	BR	Bad replacement; the element type in the assembly area can not be used as a replacement for the element type at the cursor.
	DI	Date incomplete; an attempt to replace a null node with a contact has failed because the contact was not fully defined.
	BP	Bad position; an attempt to replace a null node with a contact has failed because the column is not defined fully above the cursor.
	FU	Full; the controller data base is full and no further inserts may be made until some logic is deleted.
	TC	Two coils; an attempt has been made to place a second coil or a line.
START NEXT	MP	Memory protect; see ENTER key.
	FU	Full; see ENTER key.
	DI	Data incomplete; see ENTER key.
DELETE	MP	Memory protect; see ENTER key.
	MC	Middle of column; deletion not allowed in middle of columns.
	MN	Middle of Node; deletion not allowed in middle of calculate or timer/counter nodes.
DELETE NETWORK	MP	Memory protect; see ENTER key.
SEARCH	NF	Not found; target data was not found in data base.
SEARCH CONTINUE	NF	Not found; target data was not found in portion of data base searched.
GET NEXT	EL	End of Logic; user is at end of logic data base.
GET PREV	BL	Beginning of Logic; user is at beginning of logic data base.
CLR		No codes.
SHIFT CLR		No codes.
GET	IR	Illegal reference number; see ENTER key.
	NC	Not configured; see ENTER key.
DISB	MP	Memory protect; see ENTER key.
	IN	Illegal contact; contact type at cursor may not be disabled.
FORCE	IN	Illegal contact; see DISB key.
LOAD REG	TL	Too Large; value is greater than 999.
	NR	No register; no register has been specified in the register display area.

MAINFRAME HARDWARE DESCRIPTION

The central processing unit and memory which in conjunction with the power supply form the mainframe enclosed within housing 22 shown in FIG. 1 is set forth in detail in FIGS. 13A-18D for the CPU and FIGS. 19A-23D for the memory. The power supply is not detailed since its implementation would be well known to one of ordinary skill in electronics. The only requirements on the power supply are that it provide the necessary direct current power to drive the CPU and mem-

ory. The schematic diagrams for the CPU and memory, and programming panel schematics (FIGS. 24A-28D), designate each component with a reference number and further identify the values of discrete components and identify the type of integrated circuits used (for example discrete capacitor C5 shown in FIG. 16C). Inputs and outputs are identified so that all interconnections between the various figures is readily ascertainable. Unless otherwise noted, all resistive values are in ohms, 1/2 watt, 5%, all capacitors are in microfarads, 50 VDC, 20%, all IC's are of the 74 series except components E2, F2-F4, C1, C2, C11, C12, E5, A2, A3, and H1 for the CPU schematics and components D3-N3, D5-N5, D7-N7, D8-D8, H1 and N1 for the memory boards. These components are identified with other numbers well known to those skilled in the art so as to specify the type of integrated circuit component used.

Destination of interrupted circuit runs are indicated in parentheses in the schematic drawings by a sheet number and zone. The sheet number must be increased by the numbers set forth in Table 13 in order to find the proper drawing to which the signal is directed to or from. The zone number is a letter followed by a number within the parentheses which corresponds to the perimeter letters and numbers about the figures. The zone number is used to find the precise location for that signal, similar to finding a geographical location in an atlas.

Thus, referring to FIG. 13A at its upper lefthand corner, the signal LRSELL is from a location designated as "(3C1)". Thus, the sheet number within the parentheses is "3". Referring to Table 13, this number is converted to 15, representing FIG. 15A-D. Referring to FIGS. 15A-D, it is seen that zone "C1" refers to FIG. 15B where the signal "LRSELL" is found having designated destination (1D4) corresponding to the upper lefthand corner of FIG. 13A.

References to components within these schematic diagrams is made by the part number associated with schematic diagram. Thus, referring to FIG. 13D, capacitor C1 refers to the 10 microfarad 35 volt capacitor shown in the lefthand portion hereof. Integrated circuit components are referred to by the letter-number combination shown within or near the block designating the IC component. Again referring to FIG. 13D, an integrated circuit is shown having outputs LA3L through LA0L designated as "A7".

TABLE 13

FIGURE	NUMBER TO BE ADDED TO PARENTHESIS SHEET NUMBER
13A-18D	12
19A-23D	18
24A-28D	23

This IC component is of the "74" series with component number "LS169A". For designating integrated circuit components with multiple components within the IC component, reference is made to the output lead number of that particular component within the integrated circuit component. Thus in FIG. 13A, integrated circuit component H6 has eight drivers. If the uppermost driver is referred to, it would be identified as H6-9; the number "9" referring to lead 9 of the output associated with that driver.

In addition, logic gates are defined by the part number and output line. Referring to FIG. 15C, the lower lefthand nand gate would be referred to as H2-8.

MAINFRAME HARDWARE

FIGS. 13A-18D are schematic diagrams fully illustrating the central processing unit 31 (see FIG. 1B) utilized in the mainframe 39 of programmable controller 20. As best seen in FIG. 16B, a signetics 8X300 microprocessor E5 serves as the processor. A 1K by 16-bit program ROM (components F1, F2, F3 and F4) contains the control software. Additional functionality can be provided by replacing the 1K ROM with a larger ROM. The contents of the program ROM is not directly accessible to the control software. It is available at test points for diagnostic and system testing.

INTERFACE VECTORS

The Signetics 8X300 has no random storage as an integral part of the processor. All interfacing to the processor E5 is done via the interface vectors (IV) on interface vector lines IV0-IV7. There are two sets of interfaces vectors, one on the "left bank" and one on the "right bank". Each bank can support 256 vectors. The right bank is used for the scratchpad memory, logic RAM read and coil RAM low address. The scratchpad memory is shown in FIGS. 17B and 17C as integrated circuit components A2 and A3 and driver B2. The left bank of the interface vectors have the registers, status and control information, the column solver (discussed later), and the peripheral port interface. Since the architecture of processor E5 allows for simultaneous input and output port utilization, interbank data movement is possible on the same instruction. That is, data can be moved from the left bank to the right bank, or vice versa during the instruction.

SCRATCHPAD RAM

As noted above, the scratchpad RAM is shown in FIGS. 17B and 17C as integrated circuit components A2 and A3 and driver B2. The scratchpad RAM provides 256 bytes of temporary data storage. It is not retentive through a power failure. It is located on the right interface vector bank register. The following timing restrictions are applicable to accessing the scratchpad:

Load address register to read data	1 Instruction Wait
Write data to load address register	1 Instruction Wait
Write data to read data	2 Instruction Wait
Write data to write data	1 Instruction Wait

LOGIC RAM

The logic RAM is fully shown in FIGS. 19A-19B and 20A-20D. In addition to the actual RAM memory shown by integrated circuit components D3, D5, E3, E5, F3, F5, H3, H5, K3, K5, L3, L5, M3, M5, N3, N5 in FIGS. 19A-D and components D7, D8, E7, E8, F7, F8, H7, H8, K7, K8, L7, L8, M7, M8, N7, N8 in FIGS. 20A-D, the other addressing and driving circuitry shown in FIGS. 19A-D and 20A-D all comprise what is broadly called the logic RAM.

The logic RAM is used to store the user program. It resides on the left bank for writing and right bank for reading. It has two address registers which are concatenated to form the physical address. A signal to increment the address registers is available. The contents of the logic RAM are retentive through power failure.

The following timing restrictions apply to the logic RAM:

Load address register to read data	3 Instruction Wait
Load address register to write data	1 Instruction Wait
Write data to read data	2 Instruction Wait
Write data to Write data	1 Instruction Wait

COIL/REGISTER RAM

The coil/register RAM is shown in FIGS. 21A-21D. Like the logic RAM, the coil/register RAM in addition to the memory integrated circuit components K1, H1, L1, M1 and N1 also encompasses addressing and buffer circuitry as shown in FIGS. 21A through 21D. The coil/register RAM is used to store input, output data, and register values. Its data is retentive through a power failure, and it has two address registers which are concatenated to form the physical address. There is a memory address increment function available. The basic size of the coil/register RAM is 256 by 4 bits. The coil/register RAM is on the left bank and it has the same timing restrictions as the logic RAM.

REAL-TIME CLOCK

The real-time clock is shown in FIGS. 15A and 15B and comprises integrated circuit components H8, H7, H6, H5 and H11. This real-time clock generates a pulse at a fixed rate of once every ten milliseconds. The pulse sets a bit in the status sense register (discussed later). The software within the processor acknowledges the real-time clock via the control register (discussed later). The clock continues to generate pulses regardless of whether it is acknowledged.

WATCHDOG TIMER

The watchdog timer is shown in FIG. 15C as integrated circuit component E7 and generates a watchdog timer signal (WDTH) which is enabled by the software as part of the end-of-sweep (or scan) processing. If the software fails to enable the watchdog timer signal at least once every 50 milliseconds, the mainframe run light 24 (see FIG. 1) goes off and the I/O outputs are shut down. The state of the watchdog timer is also available through the status sense register.

PERIPHERAL PORT INTERFACE

The peripheral port interface shown in FIGS. 17A and 17C provides a serial input to the mainframe. This interface is used by the programming panel 29 and a peripheral port adapter 35 (see FIG. 1). Status information is available on the interrupt sense register and the status sense register. The peripheral port adapter provides input data from peripherals and transmits data back to those peripherals.

INPUT/OUTPUT

FIGS. 14A-14B show the electrical circuitry for performing input/output transferrals of data from the mainframe to the I/O bus 32 forming part of the I/O system 28 (see FIG. 1). There are two types of I/O in the programmable controller. Discrete I/O is used to interface to input points and output points on the I/O bus via the I/O modules. Word I/O can be obtained by use of the discrete I/O modules and converted from typical binary coded decimal (BCD) format to the binary format utilized by the controller for reading data from external registers. Binary output data is also con-

verted by software to BCD data for writing data into external registers. The higher level code describing the conversions is shown in Table 32. Register I/O in 10 bit words can also be accommodated by the controller via Register Multiplexer Modules.

SYSTEM CONTROL

The system includes the control register and interrupt sense register and is shown in FIGS. 15 A, B, C and D as integrated circuit components C6, E9, D8, H11, F10, F9, F8, H10, H9, D4, E12, F12, F11, and E11. The system control including the control register is used to trigger control pulses which are signals activated when the control register is loaded. The contents of the control register is decoded as follows:

Code	Pulse
7	Reset Processor
6	Acknowledge Real-Time Clock
5	Watchdog Timer
4	Clear peripheral port interface receiver ready
3	Not used
2	Not used
1	Increment coil address register and 0 increment logic register

Interrupt Sense Register

The interrupt sense register is shown in FIGS. 15C and 15D as integrated circuit components C4 and C5. The interrupt sense register is used to provide a sensory mechanism for the four real-time system activities; power-failure detection, real-time clock tick, peripheral port interface receiver ready, and peripheral port interface transmitter ready. There is no true interrupt structure in that software must check for any of these conditions at an interval which guarantees that data will not be lost (See Appendix A).

The interrupt sense register provides two additional signals which indicate when the I/O test connector and the CPU tester (MOT) are attached. The interrupt sense register is decoded as follows:

Bit	Condition
7	I/O tester connected
6	CPU tester connected
5	I/O busy
4	Not used
3	Peripheral port interface transmitter ready
2	Peripheral port interface receiver data ready
1	Real-time clock (100 hertz)
0	Power failure

Status Sense Register

The status sense register utilizes the same integrated circuit components as the interrupt sense register and is part of the interrupt sense system. The status sense register is used to provide hardware status information to the mainframe software. The contents of the status sense register are decoded as follows:

Bit	Status
7	Not used

-continued

Bit	Status
6	Peripheral port interface status (EIA = 1)
5	No overrun error in peripheral port interface
4	Parity/framing error in peripheral port interface
3	Watchdog timer RUN (WDT RUN = 1)
2	Memory protect
1	Register I/O Input - Bit 9
0	Register I/O Input - Bit 8

Software Overview

The mainframe software overview is presented in its entirety in Appendix A. The software block diagram is shown in FIG. 9. It indicates that the executive program (EXEC) stored in the microprocessor ROM communicates with the logic solver, peripheral port handler, I/O handler and on-line diagnostics as well as power up and power down sequences. Likewise, the interrupt handler communicates to and from the logic solver peripheral port handler, I/O handler and on-line diagnostics. The power up sequence also communicates with the CPU tester (MOT monitor).

FIGS. 10A and 10B show the data flow paths for the software. FIG. 10A is directed to the normal operation of the programmable controller while FIG. 10B illustrates the software data flow paths during power up and power down sequences.

FIG. 11 illustrates the general timing during power up, executive, interrupt handling, I/O handling, logic solving, command handling, and on-line diagnostics with information in letters within pulses explained at the bottom portion of FIG. 11.

FIG. 12 is a state diagram of the software, showing the interrelationship of the powerup and power down sequences, the normal scan in which the users' networks are solved, the error stop and halt routines as well as the CPU tester (MOT).

The actual executive program for the processor E5 (FIG. 16C) as stored in the control ROM is set forth in Appendix A to this patent application. This software in conjunction with the mainframe hardware and programming panel hardware (FIGS. 24A-28D) and programming panel software (Appendix B) performs the functions of the programmable controller as set forth in Table 14.

TABLE 14

1.	Power-up diagnostics
2.	Power-down functions
3.	Executive
4.	I/O interrupt handling including a real-time clock, peripheral port interface and power-down,
5.	Logic solutions using a multi-node 7×11 format including <ul style="list-style-type: none"> (A) relays, normally open, normally closed, and transitional contacts, (B) coils, latches, internal coils, disabled coils, and disabled latches, (C) counters, (D) timers, 1.0, 0.1, 0.01 seconds, (E) calculate with multiple outputs (add, subtract, multiply and divide) and (F) sequencers
6.	I/O handling, 128 inputs and outputs, register I/O, and extension to 256 discrete inputs and outputs.
7.	Peripheral port interface for the programming panel and the peripheral port adaptor for other types of peripherals including a computer interface.

TABLE 14-continued

8.	On-line diagnostics.
5	<h3>Scan Time</h3> <p>The maximum scan time including logic solution, I/O handling and peripheral port service and on-line diagnostics is no more than 20 milliseconds.</p>
10	<h3>I/O Service Time</h3> <p>All field I/O is serviced once per scan.</p>
15	<h3>Peripheral Port Interface Response Time</h3> <p>All characters are read before data overrun occurs. Data overrun is a system error condition. Once a command has been received, a response is initiated in no more than 1 second after receipt of a complete request.</p>
20	<h3>INPUTS</h3> <p>This section describes the inputs to the mainframe software.</p>
25	<h4>User Logic</h4> <p>User logic is the input to the logic solution module. It consists of the user program formed as entered via the programming panel or other peripheral device. All entries in the user logic data consist of two-byte nodes, each byte having 8 bits. Node format is described later.</p>
30	<p>The user logic is solved sequentially by the logic solver with processing beginning with the first node and terminating with the end-of-logic node.</p>
35	<h4>Discrete Inputs</h4> <p>A discrete input is the state of an input point which is located on an I/O input module interconnected to the I/O bus (see FIG. 1). It is either true or false which is indicated by a "1" or a "0" respectively. Discrete inputs are specified in a user program by reference designation 1 followed by three X's. A discrete input may be disabled which means that its state is not updated during each I/O scan.</p>
40	<h4>Register Input</h4> <p>Register inputs of a limited number can be transferred to the mainframe by the discrete I/O modules. Mainframe software performs the conversion from BCD to binary and binary to BCD for reading and writing register information to and from external devices. Register I/O Modules transfer 10 bit binary words to and from the mainframe directly, allowing a greater number of I/O registers.</p>
45	<h4>Communication Peripherals</h4> <p>The peripheral port interface allows a set of devices to be interfaced to the mainframe. A programming panel 29 and the peripheral port adaptor 35 interface directly to the mainframe. A tape loader and other types of programming panels can be interfaced to the peripheral port adaptor. An EIA type computer interface may also be interconnected to the peripheral port adaptor. These devices communicate using the mainframe communications protocol.</p>
50	<h4>Real-Time Clock</h4> <p>A real-time clock operating at 100 hertz frequency provides an interrupt signal via the interrupt sense regis-</p>
55	
60	
65	

ter. The clock is used to provide a time base for timers and internal clocking functions.

Power-Failure Sense

The power failure sensing is available in the interrupt status register. Five milliseconds of power are required to execute the power-down fail routine. Following completion of power failure processing, the reset processor command is issued via the control register.

Watch-Dog Timer Sense

The watch-dog timer sense provides a mechanism for checking the status of the watch-dog timer. If the software fails to enable the watch-dog timer at least once every 50 milliseconds, it expires and causes the outputs to shut down and the run light to turn off.

OUTPUTS

This section covers the outputs generated by the controller's software in response to inputs and internal processing.

Discrete Outputs

A discrete output is the state of an output point on an I/O output module interconnected to the I/O bus (see FIG. 1). This state is determined in one of two ways: first, the state of the coil as determined by the network driving the coil; and second, a disabled coil is not changed by the logic. A coil that is latched maintains its state through power failure. Discrete outputs are updated once per scan.

Register Outputs

A set of register values may be transferred to the discrete I/O modules via the software which converts the binary data used in the mainframe processing to BCD data for use with data processing external devices. Register I/O modules receive 10 bit binary register values directly from the mainframe allowing a greater number of output registers.

Communication Peripherals

Via the peripheral port interface, the mainframe sends data to peripherals attached to it. These communications take place using the mainframe communications protocol.

Real-Time Clock Acknowledge

This is a signal which clears the real-time clock sense bit in the interrupt sense register enabling the next clock pulse to be detected.

Watch-Dog Timer

The watch-dog timer pulse (WDT) is a control signal issued by the processor once per scan to indicate that the system is running. Before issuing a watch-dog timer pulse, the controller checks the watch-dog timer sense input to verify that the system is still functioning properly. The watch-dog timer controls all discrete outputs in that it must be on for outputs to be electronically enabled.

DATA BASE

Address Assignments

The address assignments are set forth in Appendix A.

External Access Conventions

This section defines the mechanisms and conventions used to access the various memories, data registers, address registers, and control registers in the mainframe.

Interface Vector Bus

All activity takes place on the interface vector (IV) bus (see processor E5, FIG. 16C). Addressing on the IV bus is via the IV left bank and registers. IVL (interface vector left) and IVR (interface vector right) select one of the 256 address locations on the left bank and the right bank respectively.

The mainframe's architecture permits 4 points to be selected simultaneously: input left, output left, input right, and output right. This is controlled via the IVL selection mechanism. Once the IVL or IVR address is loaded, the data is available on the left bank (LB) and the right bank (RB) or in sub fields as defined by the instruction set.

Scratchpad Access

Scratchpad access is by the right IV bank. The IV register must be loaded with the proper select information to allow either scratchpad read or scratchpad write as needed. Once the IV register has been loaded with the address, a "1" instruction wait time is needed to allow the address and data to settle on the bus for the operation to be read. A write takes place on the next operation with no wait. Example:

Read	XMT	ADDR, IVR		Load Address
	XMT	00010000B,	IVL	Select Read (wait cycle)
	MOV	RB, R1		Read Data
Write	XMT	00000001B,	IVL	Select Write
	XMT	ADDR, IVR		Load Address
	MOV	R1, LB		Write Data

Logic RAM and Coil/Register RAM Access

The access mechanisms for the logic RAM and the coil/register RAM are similar. First, the address to be accessed is loaded into the memory address register. The memory address register is loaded in two pieces, the lower eight bits and the upper eight bits. This is done using the IVL select to locate the proper item on the bus. When the address has been loaded, a three instruction wait is required for read operation and the one instruction wait for the write operation. An example is shown in Table 15.

Peripheral Port Interface

The peripheral port interface is a serial data channel offering full duplex communications. During the interrupt sense check, the state of the two peripheral port interface status lines are checked. If the receiver ready signal (INTRRCVR) is true, the peripheral port interface has a character ready for processing and the receiver handler is used to read the data from the interface so as to do some preliminary processing of the data prior to buffering the character. If the transmitter ready signal (INTRXMIT) is true, the transmitter is capable of sending a character. If there is data in the transmitter buffer, the next character is loaded to the interface.

TABLE 15

Read	XMT	00000011B,	IVL	Select Lo-order Addr
	XMT	ADDRLO, LB		Load Addr. Low
	XMT	00000100B	IVL	Select High-order Addr
	XMT	ADDRHI, LB		Load ADDR High
	XMT	00000000B,	IVL	Select Logic Input
	MOP			Wait 2
	NOP			Wait 3
	MOV	RE, R1		READ DATA
Write	XMT	00000011B,	IVL	Select Lo-Order Addr
	XMT	ADDRLO, LB		Load Addr. Low
	XMT	00000100B,	IVL	Select High-order Addr
	XMT	ADDRHI, LB		Load Addr. High
	XMT	00001001B,	IVL	Select Output Data
	MOV	DATA, LB		Write Data

Discrete I/O

Discrete I/O is serviced once per scan for each I/O address on the I/O bus. Once the I/O address register is loaded, the input enable is turned on. A wait of 35 instructions is required before data is available. During this time period, the output data is assembled from the coil/register RAM and packed into a byte for the output points corresponding to the input points. The input data is read and output data is loaded. The output enable is turned on and the output strobe follows 17 instructions later. During this time, the input data is decoded and stored in the coil/register RAM. The output strobe is cleared and the output enable is turned off. This cycle is repeated for each of the 8 I/O points in the system.

Register I/O

Register I/O follows the same sequence as discrete I/O except the register enables are used. Similar timing inserts are used.

Memory Organization

Scratchpad

The scratchpad organization is set forth in Appendix A.

Logic RAM Organization

The first ten bytes of the logic RAM are reserved for system status information as set forth in Table 16.

Coil/Register RAM Organization

The I/O information is allocated one 4 bit nibble per I/O point as set forth in Table 17. This table also sets forth the history extension and the register information arrangement.

Node Types

Node type arrangement is set forth in Table 18 and the node format set forth in Table 19.

Communications Protocol

I/O Assignments

The I/O assignments are set forth below:

Bit	Pinout
0	1
1	2
3	2
3	4
4	5
5	6
6	7
7	8

Strip and byte select on the I/O bus is a 1-of-4 code as set forth in Table 20.

This format gives a maximum of 16 data byte addresses with 8 points per data byte; i.e., 128 I/O points.

TABLE 16

MAIN MICROCONTROLLER CROSS ASSEMBLER VER 1.1				
* ***LOGIC RAM BIT ASSIGNMENTS * * * ***SYSCONF1 * ***MASK DEFINITIONS *				
000200	SYS4096M	EQU	10000000B	4096 BYTE LOGIC RAM
000100	SYS2048M	EQU	01000000B	2048 BYTE LOGIC RAM
000040	SYS1024M	EQU	00100000B	1024 BYTE LOGIC RAM
000020	SYS0512M	EQU	00010000B	0512 BYTE LOGIC RAM
000010	SYS0256M	EQU	00001000B	0256 BYTE LOGIC RAM
*		EQU	00000100B	NOT USED
*		EQU	00000010B	NOT USED
*		EQU	00000001B	NOT USED
* ***BIT DEFINITIONS *				
000 7 1	SYS4096B	RIV	0,7,1	4096 BYTE LOGIC RAM
000 6 1	SYS2048B	RIV	0,6,1	2048 BYTE LOGIC RAM
000 5 1	SYS1024B	RIV	0,5,1	1024 BYTE LOGIC RAM
000 4 1	SYS0512B	RIV	0,4,1	0512 BYTE LOGIC RAM
000 3 1	SYS0256B	RIV	0,3,1	0256 BYTE LOGIC RAM
*		RIV	0,2,1	NOT USED
*		RIV	0,1,1	NOT USED
*		RIV	0,0,1	NOT USED
* ***SYSCONF2 * ***MASK DEFINITIONS				

TABLE 16-continued

MAIN MICROCONTROLLER CROSS ASSEMBLER VER 1.1				
*				
000200	SYSC256M	EQU	10000000B	256 I/O POINTS
000100	SYSC192M	EQU	01000000B	192 I/O POINTS
000040	SYSC128M	EQU	00100000B	128 I/O POINTS
000020	SYSC064M	EQU	00010000B	064 I/O POINTS
*		EQU	00001000B	NOT USED
000004	SYSTRANM	EQU	00000100B	TRANSITIONAL OPTION
000002	SYSENHM	EQU	00000010B	ENHANCED EXECUTIVE
*		EQU	00000001B	NOT USED
***BIT DEFINITIONS				
*				
000 7 1	SYSC256B	RIV	0,7,1	256 I/O POINTS
000 6 1	SYSC192B	RIV	0,6,1	192 I/O POINTS
000 5 1	SYSC128B	RIV	0,5,1	128 I/O POINTS
000 4 1	SYSC064B	RIV	0,4,1	064 I/O POINTS
*		RIV	0,3,1	NOT USED
000 2 1	SYSTRANB	RIV	0,2,1	TRANSITIONAL OPTION
000 1 1	SYSEHNB	RIV	0,1,1	ENHANCED EXECUTIVE
*		RIV	0,0,1	NOT USED
***STATE VECTOR				
*				
***MASK DEFINITIONS				
*				
000200	SYSSRUNM	EQU	10000000B	RUN STATE
000100	SYSSPUPM	EQU	01000000B	POWER-UP STATE
000040	SYSSPDNM	EQU	00100000B	POWER-DOWN STATE
000020	SYSSTOPM	EQU	00010000B	STOP STATE
000017	SYSCODEM	EQU	00001111B	ERROR CODE MASK
***BIT DEFINITIONS				
*				
000 7 1	SYSSRUNB	RIV	0,7,1	RUN STATE
000 6 1	SYSSPUPB	RIV	0,6,1	POWER-UP STATE
000 5 1	SYSSPDNB	RIV	0,5,1	POWER-DOWN STATE
000 4 1	SYSSTOPB	RIV	0,4,1	STOP STATE
000 0 4	SYSCODEB	RIV	0,0,4	ERROR STATE CODE
*		RIV	0,2,0	
*		RIV	0,1,0	
*		RIV	0,0,0	
***ERROR STATE CODES				
*				
000001	SYSEOVR	EQU	1	COMMUNICATIONS OVERRUN
000002	SYSELCHK	EQU	2	MEMORY CHECKSUM FAILED
000003	SYSENODE	EQU	3	INVALID NODE TYPE FOUND
000004	SYSEIO	EQU	4	I/O PORT ERROR
000005	SYSESPD	EQU	5	SCRATCHPAD DIAGNOSTIC FAILED
000006	SYSECCHK	EQU	6	COIL RAM CHECKSUM FAILED
000007	SYSEDIAG	EQU	7	CPU DIAGNOSTIC FAILED
000010	SYSEMEM	EQU	8	ILLEGAL MEMORY CONFIGURATION
000011	SYSERTC	EQU	9	REAL-TIME CLOCK NOT FUNCTIONING
000012	SYSEWDT	EQU	10	WATCH-DOG TIMER EXPIRED
000013	SYSECOL	EQU	11	ILLEGAL COLUMN DETECTED
000014	SYSEEOL	EQU	12	NO END-OF-LOGIC NODE
*		EQU	13	NOT USED
*		EQU	14	NOT USED
*		EQU	15	NOT USED

TABLE 17

I/O information is allocated one nibble per I/O point as follows:	
Bit Name	Use
3 CRINDISB	Input disable (1=DISABLED, 0=ENABLED)
2 CRINPUT	Input state (1=ON, 0=OFF)
1 CROUTPUT	Output State (1=ON, 0=OFF)
0 CRINTRNL	Internal Coil State (1=ON, 0=OFF)
History extension is as follows:	
Bit Name	Use
7 —	Not Used
6 CRINHIS	Input History (1=ON, 0=OFF)
5 CROUTHIS	Output History (1=ON, 0=OFF)
4 CRINTHIS	Internal History (1=ON, 0=OFF)

Register information is arranged in three 4-bit nibble

TABLE 17-continued

as follows:		
Nibble	Name	Use
n	CRREGHI	Register value - Bits 11-8
r+256	CRREGMID	Register value - Bits 7-4
r+512	CRREGLOW	Register value - Bits 3-0

TABLE 18

Index	Name	Use
0	NODESON	Start of network
1	NODEEOL	End-of-Logic
2	NODEEOC	End-of-column
3	NODENULL	Null node
4	NODESKIP	Skip node
5	NODEOREL	Normally-open relay
6	NODECREL	Normally-closed relay

TABLE 18-continued

Index	Name	Use
7	NODEPOST	Positive-going transitional
8	NODENEGT	Negative-going transitional
9	NODECOIL	Coil
10	NODELATC	Latch
11	NODEDCOL	Disabled coil
12	NODEDLAT	Disabled latch
13	NODEHOZO	Horizontal Open
14	NODEHOZS	Horizontal Short
15	NODECPRE	Preset constant
16	NODERPRE	Preset register value
17	NODECTR	Counter
18	NODET100	Timer - 1.00 secs
19	NODET010	Timer - 0.10 secs
20	NODET001	Timer - 0.01 secs
21	NODEBCON	Calculate - B node constant
22	NODEBREG	Calculate - B node register
23	NODECCON	Calculate - C node constant
24	NODECREG	Calculate - C node register
25	NODECALC	Calculate node
26		
27		
28		
29		
30		
31		

TABLE 19

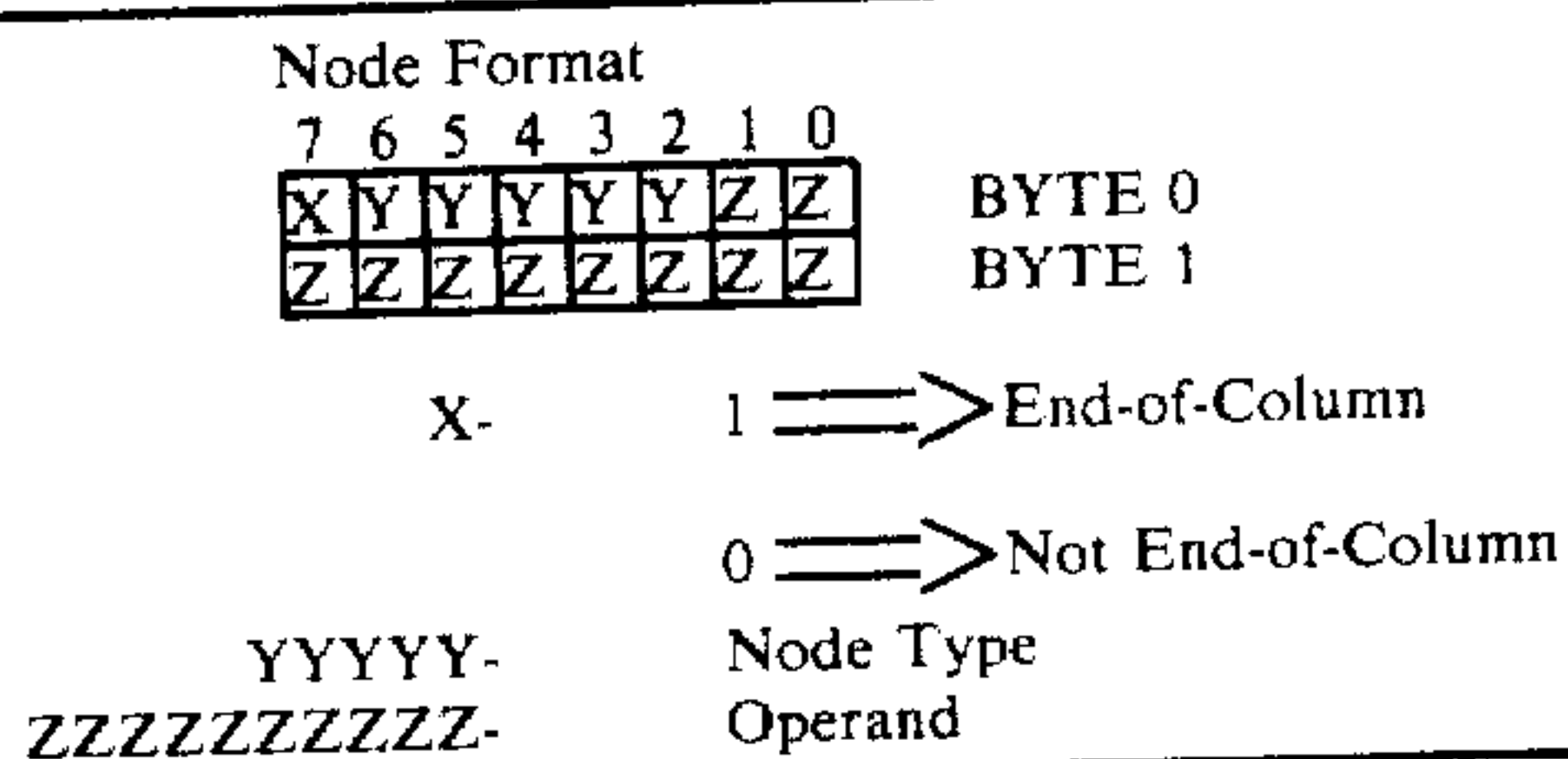


TABLE 20

Bit	Select	Name
7	STRIP D	IOSTRIPD
6	STRIP C	IOSTRIPC
5	STRIP B	IOSTRIPB
4	STRIP A	IOSTRIPA
3	BYTE 3	IOBYTE3
2	BYTE 2	IOBYTE2
1	BYTE 1	IOBYTE1
0	BYTE 0	IOBYTE0

Register I/O and extended discrete I/O can take place through the register address space as set forth in Table 21.

DATA FORMAT CONVENTIONS

The low order bit of all address and data buses is numbered "0" with the number increasing by 1 for each higher order bit. Thus, the high order bit of the several buses are:

- IV Bus=7
- Instruction Data=15
- Instruction Address=12

This is not consistent with the Signetics 8X300 processor manufacturing conventions and is consequently compensated for in the CPU hardware (See FIGS. 13A-18D).

BUS ASSIGNMENTS

When the destination address field of an instruction defines the IVR Register (17), the eight bit operand is loaded into the scratchpad addressing register. All fu-

ture references to the scratchpad memory are made to the word (1) of 256) selected by this operand.

IV Bus Addressing

Instructions specifying the IVL register (07) as the destination address send an eight bit operand to the IV select register. This operand specifies which registers and data ports are to be accessed on the IV bus by the CPU on all future references to registers 2N and 3N.

TABLE 21

Bit	Name	Use
7-0	IOWORDSL	Word Select

The CPU instructions read from either the "left bank" (2N) or the "right bank" (3N). The four choices are defined by the eight bit operand sent to the IVL register.

Output Assignments

The output assignments are set forth in Table 22.

IV Input Assignments

The IV input assignments are set forth in Table 23.

Control Pulse, Bit Assignments

The control pulses are decoded from the low order three bits of the control register as set forth in Table 24.

Status Input Assignments, Interrupt Sense

The status input assignments, interrupt sense is set forth in Table 25.

Status Sense

The status sense assignments are set forth in Table 26.

MEMORY TIMING

The scratchpad, logic, and coil RAM's operate at lower speeds than the CPU and thus require wait cycles (instructions not affecting the memory) between some operations. The instructions affecting memory are address (A), read (R), and write (W). The wait cycles are set forth in Table 27.

The address cycles are those that load the scratchpad address, or increment or load either the byte of the coil address or the logic address.

TABLE 22

IVL Register	Left Bank (Reg 2N)	Right Bank (Reg 3N)
X XXX 0000	Control Pulses	Coil Low Address
X XXX 0001	Coil High Address	Scratchpad Write
X XXX 0010	Coil Write Data	"
X XXX 0011	Logic Low Address	"
X XXX 0100	Logic High Address	"
X XXX 0101	Interface Data	"
X XXX 0110	Interface Address	"
X XXX 0111	Interface Control	"
X XXX 1000	Peripheral Data	"
X XXX 1001	Logic Write Data	"
X XXX 1010	Column Solver Pwr	"
X XXX 1011	"	"
X XXX 1100	"	"
X XXX 1101	"	"
X XXX 1110	"	"
X XXX 1111	"	"

TABLE 23

IVL Register	Left Bank (Reg 3N)	Right Bank (Reg 3N)
X 000 XXXX	Coil Read Data	Logic Read Data

TABLE 23-continued

IVL Register	Left Bank (Reg 3N)	Right Bank (Reg 3N)
X 001 XXXX	Column Solver	Scratchpad Read
X 010 XXXX	Status Sense	"
X 011 XXXX	Interrupt Sense	"
X 100 XXXX	Interface Input	"
X 101 XXXX	Peripheral Data	"
X 110 XXXX		"
X 111 XXXX		"

TABLE 24

Code	Pulse
7	= Reset Processor
6	= Acknowledge RTC
5	= Pulse WDT
4	= Clear Prog. Pnl. ROV Ready
3	=
2	=
1	= Increment Coil Address
0	= Increment Logic Address

TABLE 25

Bit #	Input
7	= I/O Tester Connected
6	= CPU Tester Connected
5	= I/O Busy
4	=
3	= Peripheral XMT Ready
2	= Peripheral RCV Ready
1	= Real Time Clock (100 HZ)
0	= Power Down Warning

TABLE 26

Bit #	Input
7	=
6	= EIA Peripheral Device
5	= Peripheral Not Overrun
4	= Peripheral Comm Err
3	= WDT Run
2	= Memory Protect
1	= Interface Data Bit 9
0	= Interface Data Bit 8

TABLE 27

	A	A	A	W	W	W	R	R	R
	to	to	to	to	to	to	to	to	to
	R	W	A	A	R	W	A	W	R
Logic/Coil	3	1	0	0	2	1	0	0	0
Scratchpad	1	0	0	1	2	1	0	0	0

The write cycle to any one of the three memories, the peripheral interface, or vertical column solver has at least one wait cycle before another write cycle to any of these devices.

POWER DOWN

A warning signal is provided to the status sense whenever power has turned off or a failure on the power line occurs. The controller is able to function for five milliseconds after the warning occurs. The software completes its pass within five milliseconds of the warning signal and issues a "reset processor" instruction.

During a power dip, the warning signal may go on and off several times with a warning occurring during the power up routine. For this reason, the maximum time from power up to the time the warning is polled

plus the power down routine time is less than five milliseconds.

On power up the instruction in location zero of the instruction ROM is executed immediately after power up stabilization. If a "reset processor" instruction is executed when the warning signal is off, the instruction is treated as a non-operation. Due to this treatment, and due to the possibility of bounce on the warning signal, the instruction after "reset processor" is the jump instruction to zero.

WATCHDOG TIMER

The watchdog timer (WDT) drives the run light 24 (see FIG. 1) and allows the interface outputs to turn on. The WDT remains enabled as long as the CPU updates it with the "pulsed WDT" control pulse more often than once every 50 milliseconds.

INTERFACE CONTROL

The interface control register is loaded by the CPU with an 8 bit byte as set forth in Table 28.

TABLE 28

Bit #7	=	Programming Panel Power Light
6	=	
5	=	Register Input Enable
4	=	Register Output Strobe
3	=	Discrete Input Enable
2	=	Discrete Output Strobe
1	=	Interface Data Bit
0	=	Interface Data Bit

VERTICAL COLUMN SOLVER

As shown in FIGS. 4, 5, and 29 the user networks allow for vertical interconnections between adjacent nodes in adjacent lines. The solving of the user networks by the mainframe of the programmable controller incorporates both hardware and software so as to perform the solution on a column-by-column basis from left to right. After the power flow is determined across each node from left to right; that is, whether or not a particular contact is passing power due to the condition of the reference element, the vertical conductivity power flow is determined by a hardwired vertical column solver 60 shown in FIGS. 22A-22D inclusively. This vertical column solving could, like any other logical operation, be performed by an appropriately programmed data processor.

This vertical column solver is shown in detail in FIGS. 22A-22D for a typical relay logic ladder diagram network such as that shown in FIG. 29. The user's ladder diagram is programmed into the controller in the form of a nodal matrix or network where each node embodies some logic element in the user's diagram. The nodes in FIG. 29 are uniquely identified by their row and column position in the network. For example, the node in the second row and first column is identified as "N_{2,1}". In general, each node is identified as "N_{i,j}", where "i" is an integer representing the row number and "j" is an integer representing the column number of the node. These logic elements can comprise, among others, normally-closed or normally-open contacts or switches, counters, timers or coils. The logical solution of each line of the ladder diagram or each row of the matrix is displayed in an output coil node corresponding to that line. Any node within the matrix can be referenced to any output coil in order to utilize the logical state of that output coil as an input to a node. The nodal

matrix in the preferred embodiment of the present invention has a maximum size of eight rows and eleven columns. Of course, it would be obvious to use either a larger or smaller network nodal matrix size.

The method of solving of the relay logic ladder diagram will now be described. Referring to FIG. 29, there is shown a typical programmed relay logic ladder diagram network 60 comprising eight rows 61, 62, 63, 64, 65, 66, 67 and 68. Logic Rows through 68 each comprise a series of nodes 41 where each node comprise an input, an output and a logic element of the type previously described, located between the input and the output. The output of one node connects to the input of the next sequential node in a junction area.

Row 61 has not been programmed and consequently is blank. Row 62 comprises a normally closed contact 70 in node $N_{2,1}$, normally open contact 71 in node $N_{2,2}$ and coil 72 in node $N_{2,3}$. Row 63 comprises normally open contact 73 in node $N_{3,1}$ and normally closed contact 74 in node $N_{3,2}$. Row 64 comprises normally open contact 75 in node $N_{4,1}$, normally open contact 76 in node $N_{4,2}$ and coil 77 in node $N_{4,3}$. Row 65 comprises normally closed contact 78 in node $N_{5,1}$ and normally open contact 80 in node $N_{5,2}$. Rows 66 and 67 are blank, and row 68 comprises normally open contact 81 in node $N_{8,1}$, normally open contact 82 in node $N_{8,2}$ and coil 83 in node $N_{8,3}$. Each of the previously described contacts and coils represents a logic element of a node in the relay logic ladder diagram. It should be noted that many more nodes may be programmed into each row.

Additionally, each row may be interconnected with adjacent rows. Such interconnections occur within the junction areas between nodes. In FIG. 29 there is shown a connection 84 between rows 62 and 63, a connection 85 between rows 64 and 65, a connection 86 between rows 62, 63, and 64 and a connection 87 between rows 65, 66, 67 and 68. These connections can be referred to by their placement in the network. Thus connections 84 can be referred to as the logic true state for variable " $C_{V3,1}$ "; that is, a connection between the output of node $N_{3,1}$ and $N_{2,1}$.

As shown in FIGS. 22A-D, the CPU of the programmable controller uses a hardware column solver 59 for performing an algorithm to solve equations for the power flow across a nodal junction area on a column-by-column basis for the entire network. Thus power flow equations for the nodal junctions in the first column are solved first followed by the nodal junctions, in the second column etc. This column solving approach is unique to the present invention and provides high speed network solving.

The column solver incorporated into the CPU of the programmable controller employs a concept called connectivity in solving the network power flow equations; that is whether variable C_V is true between adjacent nodes in the same column. Connectivity defines whether there is a connection between adjacent rows in the same column. If there is connectivity, power can flow in either direction; i.e., from the upper row line to the lower row or from the lower row to the upper row. Since the connections between rows occur at the junction between nodes of the network, the CPU solves the power flow equations for each line by determining whether or not power is present just to the right of each nodal junction $J_{i,j}$, where "i" and "j" define the junction location by row and column respectively. Thus for example, the power input status to node $N_{2,2}$ is defined by discrete variable $P_{IN2,1}$; that is, the power input

status from node $N_{2,1}$ taking into account any vertical power flow. The presence of power is determined as a function of the power status just to the left of the nodal junction; that is " P_{OUT} " from the node, logically ORed with the connectivity power state relating to power flow from interconnected lines.

In FIG. 29, in order to illustrate the column solving technique, phantom line A is placed just to the left of the first column's nodal junctions and represents the power output status for each node in column 1. Line B is placed just to the right of the first column's nodal junctions and represents the power input status for each node to the right from the node to the left in combination with any vertical power flow. If we assume that power is applied to all lines at power rail P shown in FIG. 29 and that normally open contacts close when their reference is ON and open when their reference is OFF; it is seen that P_{OUT} from a node is true if there is input power to the node and the node contact is closed. This can be stated generally by the following equation: $P_{OUT_{i,j}} = P_{IN_{i,j-1}} C_{i,j}$ where $C_{i,j}$ is the conductivity state of node $N_{i,j}$. Other elements in the nodes conduct depending upon the states of their references. Thus a normally closed switch conducts if the reference is OFF, etc. These conducting states are set forth in Tables 8A-8H, 9A-9B, and 10A-10D. The output power from the node is coupled with the vertical output status at the junction between two adjacent nodes in the same column. Thus the junction between nodes $N_{3,1}$ and node $N_{3,2}$ is junction $J_{3,1}$. The power to junction $J_{3,1}$ is the power output from node $N_{3,1}$ —that is, $P_{OUT_{3,1}}$ —plus the vertical power down—that is $P_{VD_{3,1}}$ —due to vertical connector 84 (alternatively designated $C_{V_{3,1}}$) and vertical power up—that is $P_{VU_{3,1}}$. Vertical power up or down is true if there is a corresponding vertical connection and if a power out is true to the connection from an interconnected node. Thus for junction $J_{3,1}$ vertical power down— $P_{VD_{3,1}}$ is true because a connector 84 ($C_{V_{3,1}}$) exists (is true) and power out from node $N_{2,1}$ is true assuming element 70 is conducting).

Thus the power in from node $N_{3,1}$ is the power out from node $N_{3,1}$ ($P_{OUT_{3,1}}$) logically ORed with the vertical down power ($P_{VD_{3,1}}$) and the vertical up power ($P_{VU_{3,1}}$). In Boolean logic, this statement can be set forth for any node in the user network by the following equation:

$$P_{IN_{i,j}} = P_{OUT_{i,j}} + P_{VU_{i,j}} + P_{VD_{i,j}} \quad (1)$$

where

$$P_{OUT_{i,j}} = P_{IN_{i,j-1}} C_{i,j} \quad (2)$$

where $C_{i,j}$ is the conductivity state, of node $N_{i,j}$, where

$$P_{VU_{i,j}} = P_{IN_{i+1,j}} C_{U_{i,j}} \quad (3)$$

where $C_{U_{i,j}}$ is the connectivity state between the output of node $N_{i,j}$ and node $N_{i+1,j}$, and where

$$P_{VD_{i,j}} = P_{IN_{i-1,j}} C_{D_{i,j}} \quad (4)$$

where $C_{D_{i,j}}$ is the connectivity state between the output of node $N_{i,j}$ and node $N_{i-1,j}$.

Alternatively, since power vertical is equal to the logically "anding" of power out and vertical connectors, the following Boolean equations can define the power input to the next horizontal node from the node to its left:

$$P_{INij} = P_{OUTij} + P_{OUTi-1j} C_{V1j} + P_{OUTi-2j} C_{V1-1j} C_{V1j} + \dots + P_{OUT1j} C_{V2j} C_{V3j} \dots C_{Vij} + P_{OUTi+1j} C_{V1+1j} + P_{OUTi+2j} C_{V1+2j} C_{V1+1j} + \dots + P_{OUT1j} C_{V1j} C_{V1-1j} \dots C_{V1+1j} \quad (1)$$

Visually, the power status for each of the lines 61 through 68 at line A shown in FIG. 29 is determined as follows: in this discussion a "1" indicates the presence of power and "0" indicates the absence of power. The power status of row 61 at line A is obviously 0 since no connection exists between the power rail P and line A in row 61. Since the normally closed contact 70 of row 62 is false if reference "007" is true, the power status at line A ($P_{OUT2,1}$) for row 62 is also 0. The normally open contact 73 in row 63 will close when reference 001 is true. Since power in ($P_{IN3,0}$) is true, the power out status ($P_{OUT3,1}$) at line A will be 1. Similarly, the power status at line A for row 64 will also be 1 if reference 002 is true. Row 65 is similar to row 62, and the power status at line A will be 0 if reference 001 is true. The power status at line A will also be 0 for lines 66 and 67 since no nodes exist. Row 68 is similar to rows 63 and 64 and therefore the power status at point A will be 1 if reference 002 is true. This resultant series of 1's and 0's is the output power status at point A and is referred to as a power byte. This power byte is generated by the software within the mainframe and is transferred to the column solver 59 (FIGS. 22A-22D) as signals BB0H through BB7H. The power byte at line A is shown in Table 29 for rows 1-8 from left to right.

TABLE 29
0 0 1 1 0 0 0 1

The next step is to determine the connectivity between the rows for the first column. In determining connectivity a 1 indicates a connection to the row above the row in question, and a 0 indicates no connection to the row above. Referring again to FIG. 29, it can be seen that for row 61 there is no row above so consequently the connectivity status for row 61 at column one (C_{V11}) is always zero and therefore is shown as a blank on Table 30 below. It can be seen that there is no connection between rows 62 and 61 so the connectivity status of row 62 ($C_{V2,1}$) is 0. The connectivity status of row 63, however, is 1 since there is a connection to row 62. Similarly, the connectivity for row 64 is a 0, for row 65 is a 1, and for rows 66, 67, and 68 are all 0. This result is illustrated in TABLE 30 for rows 1-8 from left to right. The previous data comprising the power status at line A and connectivity is determined by the software of the programmable controller.

TABLE 30
- 0 1 0 1 0 0 0

This data is stored as part of the logic data within the mainframe and is transferred to the column solver (FIGS. 22A-22D) as signals LR0L through LR6L.

The CPU of the controller then solves the power flow equations for each row at phantom line B. Power can be present at line B for each row in one of three ways. (1) power can flow directly through the row from line A if the node is in the conducting state. (2) power can flow from line A of a row above through a connection to the row being solved; and (3) power can

flow from line A of a row below through a connection to the row being solved. In the example shown in FIG. 29 it can be seen that the power status at line B of row 61 is 0. The power status at line B of row 62, however, is 1 since power can flow from line A of row 63 up through connection 84. The power status at line B of row 64 is 1 since power flows directly from line A to line B if contact 73 is closed (reference 001 is true). Power also flows from line A of row 64 down through connector 85 to line B of row 65 making the power status at line B of row 65 also 1. Since it can be seen that no connections exist, the power status at line B of rows 66 and 67 is 0. It can also be seen that the power status at line B of row 68 is 1 since power can flow directly from line A to line B of row 68. The solution to the power flow equation for each column is an input power byte, such as that shown in Table 31 for rows 1 through 8 from left to right.

TABLE 31
0 1 1 1 1 0 0 1

This data is generated by the column solver 59 shown in FIGS. 22A-22D on output lines VR0L through VR7L.

The software of the programmable controller (Appendix A) furnishes the column solver with information of the power input to the left of the nodal junction and the connectivity data relating to connections between lines. The column solver then determines the input power byte just to the right of the nodal junction. The software then uses this input power byte to determine power flow through the next node, in order to get the power input at the following nodal junction. The column solver of the controller continues to solve the lines in this columnar manner in a left to right fashion until the overall power status of the network is determined, resulting in a power byte for each output coil for the entire network (nodes $N_{2,3}$, $N_{4,2}$, and $N_{8,3}$ for FIG. 29).

The logic hardware implementation that performs the column solving is shown in FIGS. 22A-D. For the sake of simplicity, only the logic steps involved in determining the output for one line of any particular column is described. Referring to FIG. 22C, there is shown a number of logic elements or gates. Lines to and from the logic gates are referred to by an alphanumeric number comprising the component and the input or output line number. Also shown in FIG. 22C are input lines BB2H, BB3H, LR1L and LR2L and output lines VR2L and VR3L. Input line BB2H carries the input power data for row 2. Input line BB3H carries the input power data for row 3. Line LR6L carries the connectivity data relating to connectivity between rows 1 and 2, and line LR5L carries connectivity data relating to connectivity between rows 2 and 3. In terms of this particular logic arrangement, a logical 1 on lines BB2H and BB3H indicates power and a logical 0 on lines LR1L and LR2L indicates connectivity between the respective lines. Line VR6L carries the output power data relating to row 2 and shows a logical 0 when power is present. The input power data and connectivity data is supplied to the hardware column solver from the software of the CPU.

The method of determining the power status on output line VR2L will now be described. From the previous discussion, it is apparent that output line VR2L can exhibit a logical 0 indicating the presence of power

when any of three situations occurs: (1) when power flows directly from input line BB2H; (2) when power flows down from the row above; or (3) when power flows up from a row below. The case where power is present on input line BB2H will now be examined. If power is present on line BB2H, a logical 1 will appear on line B4-6. When either line B4-6 or line B4-5 is a logical 1, line B4-4 becomes a logical 0. This output also appears on line B2-5. Whenever the status of line B2-4 or B2-5 or both is a logical 0, line B2-6 becomes a logical 0. In that way, it can be seen that when input line BB2H carries a logical 1, output line VR2L will be a logical 0 indicating power is present.

The case of power flowing down from a row above will now be examined. It can be seen that when the input line BB3H carries a logical 1, line B3-10 will be a logical 0 making line B3-11 also a logical 0. Line B3-12 is connected to input line LR2L which carries the connectivity data relating to connectivity between rows 6 and 5. If connectivity exists, this line will carry a logical 0 also making line B3-12 a logical 0. When both lines B3-11 and B3-12 are a logical 0, line B3-13 becomes a logical 1. It can be seen that when power exists at input line BB3H, and there is connectivity between that line and line BB2H, the presence of the logical 1 on line B3-13 also applied to line B4-9, will in turn cause a logical 0 at the output VR2L indicating the presence of power.

In a similar fashion, lines B4-1 and B4-4 determine whether there is a connection to the row below and also whether there is power flowing in that row. It is clear that both conditions of power flowing and connection between rows must be true in order for the output line VR2L to show a logical 0 indicating the presence of power.

It should be noted that the column solver is not limited to the specific hardware implementation shown, or to any hardware implementation. The function of the column solver could easily be done by a software program or by other hardware construction.

Any software program or hardware implementation that performs the following logic algorithm would accomplish the result of the column solver of the present invention.

$$D_n = O_{n+1} \cdot D_{n+1} + P_n$$

$$U_n = O_{n-1} \cdot U_{n-1} + P_n$$

$$O_n = D_n + U_n$$

where

D_n = power flowing up from below to line n.

U_n = power flowing down from above to line n.

O_n = power output on line n.

O_{n+1} = power output on line below (line n+1)

D_{n+1} = connectivity from line below (line n+1)

O_{n-1} = power output on line above (line n-1)

U_{n-1} = connectivity from line above (line n-1)

P_n = power input on line n.

The concept of column solving as embodied in the present invention is superior to other techniques utilized by other programmable controllers in the solving of network ladder diagrams. Prior to the present invention, ladder diagrams were solved on a line-by-line basis. This technique would often create problems for the programmer who would often have to rewrite his ladder diagrams in order to conform to a specified programming format.

FIG. 30 illustrates a network that is easily solvable by the column solver of the present invention, but presents difficulties to the conventional line solver controller.

A conventional controller using prior art line solving technology would solve the relay logic ladder diagram shown in FIG. 30 in the following fashion. The power flow for node 90 would be solved first followed by the solving of nodes 91 and 92. Prior to solving nodes 91 and 92, however, the results of the power flow through node 90 would be stored in a register for later use. This stored value would correspond to the power status at point 95. The power flow solution to nodes 91 and 92 would also have to be stored in a register. This value would correspond to the power status at point 97. After storing the status of point 97, the conventional controller would return to point 95 and using the previously stored power status value, it would then solve for node 94. This value would be stored for the power status at point 96, and the controller would return to solve for node 98. The solution to node 98 would be basically ORed with the stored value at point 96 and the resultant value ORed with the value at point 97. The results from this would be then used to solve the power flow for node 93. It is quite apparent that for a complicated network having many node branches, a large amount of register storage is required in order to hold intermediate power status values while other nodes are solved. This storage space requirement in prior art controllers necessitates limitations on the format of the user network so as to limit the number of logically ORed nodes. The column solver of the present invention, however, is not adversely affected in its execution of such logic functions and is therefore faster and more efficient than prior art controllers.

NETWORK INSERTION

The programmable controller via its software allows for the insertion of networks between two sequentially adjacent existing networks. Since the networks are solved sequentially in the order of their step number (see the status/assembly area in FIG. 6A), the sequential solution order of the programmable controller can be altered by network insertion. The portion of the software for implementing this network insertion is set forth in Appendix A.

COIL DESIGNATION

The programmable controller not only allows the user to insert networks between two existing networks in his or her control program but also allows the user to designate any desired output point in the I/O system for any line within any network. Thus user lines may be inserted anywhere within the control program without affecting other lines within the control program or their coil numbers. In prior art programmable controllers employing the user line concept, each user line had a fixed coil number representing its logical output state. Thus for example it was not possible to change user line "6" to have an output coil designated "9" or any other number, other than "6".

TABLE 32

/* Convert Node */
/* Direction and Type of Convert is Specified in Bits 1-0 of R1 */
/* IF R1(1-0) = 00B, Discrete Source Node */
/* IF R1(1-0) = 01B, Register Source Node */
/* IF R1(1-0) = 10B, Convert to BCD, Store in Discrete */
/* IF R1(1-0) = 11B, Convert to Binary, Store in Register */

TABLE 32-continued

```

/* Discrete Source is always a Discrete Input */
/* Discrete Destination is always a Discrete Output
   (i.e. Not an Internal Coil) */
/* Registers are always Holding Registers */
/* GET Type of Convert Node */
R11 = R1.AND.3
/* Vector OFF R11 */
/* If R11 = 00, then Discrete Source Node */
/* Coil ADDR REG has been Loaded */
/* GET Coil Increment Code */
R11 = CTR LINCC
/* Clear Assembly Area for Data */
[R5,R6] = 0
/* Set Up Count */
R2 = -1210
FOR I = 1, 12 (using R2 for counting).
/* SHIFT Discrete Bits */
[R5,R6] = [R5,R6].Rotate Left. 1
/* Bring in Next Discrete Input */
[R5,R6] = [R5,R6].OR.CRINPUT
IVOCTRL < = R11
NEXT I
/* Store Source Data */
[CONVSRCH, CONVSRCL] = [R5,R6]
Go To Logic 020 /* Solve Next Node */
/* If R11 = 01B, Then Register Source Node */
/* Save R1 */
Save R1 = R1
Call REGVAL
/* Save Source Data */
[CONVSRCH, CONVSRCL] = [R5,R6]
/* Solve Next Node */
/* If R11 = 10B, Then Discrete Destination, with Binary
   to BCD Convert */
/* Save R1 */
Save R1 = R1
Save R3,R4 = Save R3, Save R4
/* GET Binary Source Data */
[R5,R6] = [CONVSRCH, CONVSRCL]
/* Set Up Count A */
BCD Value = 0, [R3,R4] = 0
R1 = -4
Do R1 To 0, Step 1
/* Set Up Count B */
R2 = -4
Multiply BCD Value By 2, [R3,R4] = 2.*[R3,R4]
Do R2 to 0, Step 1
/* Subtract 800 from BIN Value */
[Aux,R11] = [R5,R6] - 80010
If [Aux,R11].GE.0
Then Do
/* Replace BIN Value */
[R5,R6] = [Aux,R11]
/* Add one to BCD Value */
[R3,R4] = [R3,R4] + 1
Else:
ENDIF
ENDDO
/* Divide BCD Value By 16 */
[R3,R4] = [R3,R4].Rotated Right.4
/* Mult BIN Value By 10 */
[R5,R6] = [R5,R6] * 10
ENDDO
/* BCD Value is In [R3,R4]
/* Future Rotates with [R3,R4] will be Wrap-Around
   With Carryout = > Carryin */
/* Coil Addr Reg is set, GET INCR Code */
R11 = CTRLINCC
/* Set Count */
R2 = -1210
/* Rotate First Bit into Position */
[R5,R6] = [R5,R6].Rotate Left 5
DO R2 to 0 Step 1
/* Output a Bit */
CROUTPUT = [R4]
/* Rotate Next Bit into Position */
[R3,R4] = [R3,R4].Rotate Left.1
ENDDO
/* Restore Registers */
R1 = Save R1
R3 = Save R3
R4 = Save R4

```

TABLE 32-continued

```

/* Set Power */
R3 = (R3.AND.3).OR.2
/* Solve Next Node */
5 /* If R11 = 11B, Register Destination with BCD to
   Binary Convert */
/* Save R1, R2, R3, and R4 */
Save R1 = R1
Save R2 = R2
Save R3 = R3
10 Save R4 = R4
/* Get Source Data */
[R1,R2] = [CONVSRCH, CONVSRCL]
/* Set Bin to 0 ]
[R3,R4] = 0
1 Set Count */
R11 = -3
15 Do R11 to 0, Step 1
/* Multiply Bin Value by 10 */
[R3,R4] = [R3,R4] * 10
/* Add Next Digit to Bin Value */
[R3,R4] = [R3,R4] + R1
/* Move next Digit into Position */
20 [R1,R2] = [R1,R2]. Rotate Left. 4. AND.7777
ENDDO
/* Save Bin Data */
[CONVSRCH, CONVSRCL] = [R3,R4]
/* Restore R1, R2 */
R1 = Save R1
25 R2 = Save R2
/* GET REG ADDR */
CALL REGVAL
/* GET Bin Data */
[R1,R2] = [CONVSRCH, CONVSRCL]
CALL STORE
30 /* Restore Registers */
R1 = Save R1
R3 = Save R3
R4 = Save R4
/* Set Power */
R3 = (R3.AND.NOT.3).OR.2
35 /* Solve Next Node */

```

In the present invention, the output coil of any user line is identifiable with any number within the output address state of the I/O system. Thus it is not necessary that the coil numbers of outputs within user networks be equal to the number of the line in that network. For example, in network number one (step number one) the first line output can reference any I/O point from "1" to the maximum number of I/O points in the I/O system; typically, 256. Similarly, the second line of that network need not have a coil output numbered "2" but can be any number within the I/O output field. Therefore, the present invention is unlike prior art programmable controllers where the solution order of the user program was the same as the line number order. In the present programmable controller, the line number order can be designated arbitrarily by the user while the solution order of his or her program is by the step number (network number) of the networks in the control program. The software for programming the user line outputs is set forth in Appendix B while the software used by the mainframe 39 to solve the user program and setting output points in the I/O system is set forth in Appendix A.

PROGRAMMING PANEL

The programming panel 29 shown in FIG. 1 is presented in detail in FIGS. 24A-28D. As shown in FIG. 24A and 24C, it incorporates an Intel 8080A microprocessor Z1 and associated circuitry. The software controlling the microprocessor is set forth in Appendix B. The resultant programming panel in conjunction with the hardware and software of the mainframe allow

the user to program, monitor and debug his or her control program. Furthermore, the programming panel in conjunction with the mainframe allows for the realtime display of a node as selected by the cursor control keys of the programming panel, the insertion of networks between two existing networks within the control program as well as allowing the user to assign the coil output state of any line within any network without being constrained by the line number of the line within the network. The programming panel in combination with the mainframe provides a real-time output on LED (see FIG. 1) for any node or CRT screen 36 as selected by cursor 47 (see FIGS. 2, 4, and 5). It also allows the user to perform specialized searches of the control program.

Thus what has been described is an improved, low cost programmable controller intended to replace from 8 to 256 hardwired relays used in typical industrial control applications. This improved programmable controller allows the user to enter his or her control program via a programming panel having a CRT display and utilizing networks comprising up to seven rows by eleven columns of user selected elements. The solution order of the control program is performed in sequence by the network number associated with each of the user networks. In this manner, in situations where the solution order is important to the proper functioning of the control system, the user can have the programmable controller perform the solution of one network before another network. This capability is enhanced by the programmable controller, in conjunction with its programming panel, allowing the user to insert a network between two existing networks in the control program.

Furthermore, the programmable controller described in this application allows the user to designate the output coil associated with a user line without being constrained by the line number of the user line. This gives the user more freedom in generating the control program since the output coil numbers are not fixed by the line numbers of the control program. This, in conjunction with the capability of inserting networks between existing networks within the control program, further

helps the user obtain a desired control system.

In addition, the programmable controller described in this application allows the networks of the user control programs to have vertical interconnection between adjacent lines in the network. Although such vertical connections have existed in prior art programmable controllers, the present invention overcomes a problem in the prior programmable controllers in having a column solver which eliminates most of the constraints on the user in setting up the network and also which greatly reduces the hardware and software requirements of the programmable controller to solve the network. The present programmable controller thus solves each user network on a column-by-column basis with one portion of the solution being the state of the element within the nodes within a particular column of the network and the next step being the determination of vertical power flow from any line to any adjacent line due to vertical interconnections.

Lastly, the programmable controller as described in this application has a CRT display which utilizes a cursor which when placed on any node of the user generated networks displays on an LED the real-time power status of that node. Furthermore, the programming panel in conjunction with the mainframe of the programmable controller, allows the user to perform various search operations of the user program so as to facilitate monitoring and debugging of the user program.

The combination of these various features and improvements yields an advance in the state of the art of programmable controllers.

It will thus be seen that the objects set forth above, and those made apparent from the preceding description, are efficiently attained and, since certain changes may be made in the above construction without departing from the scope of the invention, it is intended that all matter contained in the above description or shown in the accompanying drawings shall be interpreted as illustrative and not in a limiting sense.

Appendix A includes a listing of mainframe software and Appendix B includes a listing of program panel software.

```

PROG  MAIN                                *CAP EX3 MOD 01 REV 0
1      PROG  MAIN
2      LIST  S,C,P
3
4      ***OBJECT FORMAT STATEMENT HERE (DEFAULT IS 'MP' FORMAT).
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

36 *
37 *
38 *
39 *
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *

```

02. 10/07/77 REARRANGED PWRUP TO SAVE CODE CLEANSING INPUT REGISTERS
ADD ERROR HALT ON SCRATCHPAD DIAGNOSTIC FAILURE
T. STODDLEY

03. 10/20/77 PWRUP - CODE
PWRDN - CODE
EXEC - CODE
INTRP - CODE
LOGIC - CODE
FLDIO - STOP
CMDS - STOP
DIAGS - STOP
SUBS - CODE
NEW CODE RAM CHECKSUM
NEW LOGIC RAM CONFIGURATION
T. STODDLEY

04. 11/05/77 PWRUP - CODE
PWRDN - CODE
EXEC - CODE
FLDIO - CODE
LOGIC - CODE
CMDS - STOP
DIAGS - STOP
INTRP - CODE
SUBR - CODE

ADDED FIELD I/O MODULE
T. STODDLEY

05. 11/09/77 PWRUP - CODE
PWRDN - CODE
EXEC - CODE
FLDIO - CODE
LOGIC - CODE
CMDS - STOP
DIAGS - CODE
INTRP - CODE
SUBR - CODE

ADDED DIAGNOSTIC MODULE
T. STODDLEY

06. 11/18/77 PWRUP - CODE
PWRDN - CODE
EXEC - CODE
FLDIO - CODE
LOGIC - CODE
CMDS - STOP
DIAGS - CODE
INTRP - CODE
SUBR - CODE

CHANGES TO LOGIC, GLOBAL AND INTRP
GLOBAL- TO DEFINE NEW CONSTANTS FOR LOGIC
LOGIC- TO CORRECT BUGS
INTRP- TO CORRECT PAGING ERRORS IN M211 USAGE

J.VAN SCHALKWYK

07. 11/23/77 PWRUP - CODE
PWRDN - CODE
EXEC - CODE
FLDIO - CODE
LOGIC - CODE
CMDS - STOP
DIAGS - CODE
INTRP - CODE
SUBR - CODE

MORE CLEANUP ON INTRP FOR PAGING ERRORS
JVS

08. 12/05/77 PWRUP - CODE
PWRDN - CODE
EXEC - CODE
FLDIO - CODE
LOGIC - CODE
CMDS - CODE
DIAGS - CODE
INTRP - CODE
SUBR - CODE

ADDED CMDS MODULE
T. STODDLEY

09. 12/06/77 MODIFIED CMDS MODULE
T. STODDLEY

10. 12/09/77 MODIFIED CMDS MODULE
FIXES TO INTRP


```

127 *
128 *
129 *
130 *
131 * 11. 12/10/77 MODIFY LOGIC MODULE
132 * J.VAN SCHALKWYK
133 *
134 * 12. 12/13/77 NEW COMMAND HANDLER
135 *
136 *
137 * ADDITIONAL SUBROUTINES
138 * T. STODDLEY
139 *
140 *
141 * 13. 12/15/77 MODIFY LOGIC AND GLOBAL TO ALLOW
142 * CONDITIONAL ASSEMBLY OF ENHANCED INSTRUCTION SET.
143 * FLAG 'ENHANCE' IS SET TO '0' FOR BASIC SET,
144 * SET TO '1' FOR ENHANCED SET.
145 * J. VAN SCHALKWYK
146 *
147 *
148 * 14. 12/19/77 MODIFY LOGIC TO HAVE SEQUENCER REFERENCES
149 * INCLUDED IN CONDITIONAL ASSEMBLY
150 * J. VAN SCHALKWYK
151 *
152 *
153 * 15. 12/27/77 ALLOW PROPER WAIT STATES FOR SCRATCHPAD WRITE
154 * CHANGES TO CMDS MODULE
155 * T. STODDLEY
156 *
157 *
158 * 16. 12/29/77 TIMING PROBLEMS WITH SCRATCHPAD ACCESS
159 * T. STODDLEY
160 *
161 *
162 * 17. 12/30/77 CLEAN-UP FOR DEMO
163 * T. STODDLEY
164 *
165 *
166 * 18. 1/18/78 CLEAN UP TIMER/COUNTER IN LOGIC MODULE
167 * J.VAN SCHALKWYK
168 *
169 *
170 * 19. 1/30/78 EDITS AND MORE EDITS
171 * ALL UPDATED FILES CHANGED TO '.L1' EXTENSIONS
172 * T. STODDLEY
173 *
174 *
175 * 20. 1/31/77 ADDITIONAL EDITS
176 * GLOBAL, LOGIC, AND SUBROUTINE ARE AT .L2 LEVEL
177 * -JVS-
178 *
179 *
180 * 21. 12/14/78 CHANGE SCRATCH PAD ALLOCATION,
181 * MULTIPLY AND DIVIDE AT DOUBLE PRECISION
182 * ALL CURRENT REVISION MODULES WILL BE AT '.L0'
183 * EXTENTION LEVEL, OLDER REVS WILL GO TO '.L1', '.L2', ETC
184 * -JVS-
185 *
186 *
187 * 22. 12/23/78 CHANGE COMMAND HANDLER AND INTERRUPT HANDLER
188 * TO USE NEW PROTOCOL.
189 * -RAD-
190 *
191 *
192 * CHANGE VALIDATE MODE TO USE CONDITIONAL ASSEMBLY TO
193 * DISTINGUISH BASIC SET FROM ENHANCED SET.
194 * JVS
195 *
196 *
197 * 24. 1/31/78 DELETE NULLS ON POWER UP
198 * SLIGHT CHANGE TO VALIDATE MODE
199 * -RAD- -JVS-
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *

```

***LOCAL FLOW

* MODULE NAME: GLOBAL
* THIS MODULE SUPPLIES THE GLOBAL DEFINITIONS FOR THE 484 CONTROLLER

* LEFT BANK/RIGHT BANK

	LR	RR	EGG	20H	30H	LEFT BANK	RIGHT BANK
210	000020						
211	000030						
* DEFINE BITS FOR EACH BANK							
211	000 0 1	LRBIT0	LIV	0,0,1		LEFT BANK - BIT 0	
211	000 1 1	LRBIT1	LIV	0,1,1		LEFT BANK - BIT 1	
212	000 2 1	LRBIT2	LIV	0,2,1		LEFT BANK - BIT 2	
213	000 3 1	LRBIT3	LIV	0,3,1		LEFT BANK - BIT 3	
214	000 4 1	LRBIT4	LIV	0,4,1		LEFT BANK - BIT 4	
215	000 5 1	LRBIT5	LIV	0,5,1		LEFT BANK - BIT 5	
216	000 6 1	LRBIT6	LIV	0,6,1		LEFT BANK - BIT 6	
217	000 7 1	LRBIT7	LIV	0,7,1		LEFT BANK - BIT 7	
218							
219	000 0 1	RRBIT0	RIV	0,0,1			RIGHT BANK - BIT 0
220	000 1 1	RRBIT1	RIV	0,1,1			RIGHT BANK - BIT 1
221	000 2 1	RRBIT2	RIV	0,2,1			RIGHT BANK - BIT 2
222	000 3 1	RRBIT3	RIV	0,3,1			RIGHT BANK - BIT 3
223	000 4 1	RRBIT4	RIV	0,4,1			RIGHT BANK - BIT 4

```

224 000 5 1 RRRIT5 RIV 0,5,1 RIGHT BANK - BIT 5
225 000 6 1 RRRIT6 RIV 0,6,1 RIGHT BANK - BIT 6
226 000 7 1 RRRIT7 RIV 0,7,1 RIGHT BANK - BIT 7
227 *
228 ***ENHANCED INSTRUCTION SET INDICATOR.
229 *
230 ENHANCE EQU 1
231 *
232 * SET 'ENHANCE' TO '0' FOR BASIC INSTRUCTION SET.
233 * SET 'ENHANCE' TO '1' FOR ENHANCED SET.
234 * THIS IS USED IN CONDITIONAL ASSEMBLY STATEMENTS.
235 *
236 *
237 * STARTING ADDRESS OF MACHINE OPEN TEST (MOT)
238 *
239 * THE MOT WILL START AT LOCATION 4001 (OCTAL).
240 * MCSIM WILL NOT ALLOW ILLEGAL REFERENCES.
241 * MOT WILL CAUSE A JMF * TO BE EXECUTED
242 *
243 004001 MOTTEST EQU 4001H START OF MOT
244 *
245 * I/O REGISTER DEFINITION
246 *
247 * OUTPUT CONTROL (BITS 3-11)
248 *
249 000000 IVOCTRL EQU 00000000H CONTROL PULSES LEFT
250 000001 IVOCRLO EQU 00000001H COIL ADDRESS LOW RIGHT
251 000002 IVOCRHI EQU 00000010H COIL ADDRESS HIGH LEFT
252 000003 IVHSPD EQU 00000011H SCRATCHPAD OUTPUT DATA RIGHT
253 000004 IVOCKDAT EQU 00000100H COIL WRITE DATA LEFT
254 000005 IVOIRLO EQU 00000110H LOGIC ADDRESS LOW LEFT
255 000006 IVOIRHI EQU 00000101H LOGIC ADDRESS HIGH LEFT
256 000007 IVOIDATA EQU 00001010H I/O INTERFACE DATA LEFT
257 000008 IVOIADDR EQU 00001100H I/O INTERFACE ADDRESS LEFT
258 000009 IVOICTL EQU 00001110H I/O INTERFACE CONTROL LEFT
259 000010 IVOIPDAT EQU 00010000H PERIPHERAL PORT OUTPUT DATA LEFT
260 000011 IVOIRDAT EQU 00010010H LOGIC RAM WRITE DATA LEFT
261 000012 IVOICOL EQU 00010100H COLUMN SOLVER LEFT
262 * EQU 00010110H NOT USED
263 * EQU 00011000H NOT USED
264 * EQU 00011010H NOT USED
265 * EQU 00011100H NOT USED
266 *
267 ***INPUT CONTROL (BITS 6-4)
268 *
269 000000 IVICPDAT EQU 00000000H COIL RAM READ DATA LEFT
270 000001 IVILRDAT EQU 00000001H LOGIC RAM READ DATA RIGHT
271 000002 IVICOLIN EQU 00010000H COLUMN SOLVER INPUT DATA LEFT
272 000003 IVISPD EQU 00010001H SCRATCHPAD INPUT DATA RIGHT
273 000004 IVISTAT EQU 00100000H STATUS SENSE REGISTER LEFT
274 000005 IVIINTSP EQU 00110000H INTERRUPT SENSE REGISTER LEFT
275 000006 IVIDDATA EQU 01000000H I/O INTERFACE INPUT DATA LEFT
276 000007 IVIPDAT EQU 01010000H PERIPHERAL PORT INPUT DATA LEFT
277 * EQU 01100000H NOT USED
278 * EQU 01110000H NOT USED
279 *
280 ***BIT 7 NOT USED
281 *
282 ***CONTROL REGISTER VALUES
283 *
284 000000 CTRLINCL EQU 00 INCREMENT LOGIC RAM ADDRESS
285 000001 CTRLINCC EQU 01 INCREMENT COIL RAM ADDRESS
286 * EQU 02 NOT USED
287 * EQU 03 NOT USED
288 000004 CTRLRCLR EQU 04 CLEAR PERIPHERAL PORT RECEIVER READY
289 000005 CTRLWDT EQU 05 PULSE WATCHDOG TIMER
290 000006 CTRLRTC EQU 06 ACKNOWLEDGE REAL-TIME CLOCK
291 000007 CTRLPROC EQU 07 RESET PROCESSOR
292 *
293 000 0 3 CTRLREG LIV IVOCTRL,0,3 IVOCTRL IS A 3 BIT REGISTER AND CAN BE
294 * XMTED TO
295 *
296 ***INTERRUPT SENSE REGISTER DEFINITIONS
297 *
298 ***MASK DEFINITIONS
299 *
300 000001 INTRPWFH EQU 00000010H POWER-FAIL WARNING
301 000002 INTRRTCH EQU 00000011H REAL-TIME CLOCK
302 000003 INTRRYM EQU 00000100H PERIPHERAL PORT RECEIVER READY
303 000004 INTRTRYM EQU 00000101H PERIPHERAL PORT TRANSMITTER READY
304 * EQU 00010000H NOT USED
305 000005 INTRIOBM EQU 00100000H I/O BUSY
306 000006 INTRMOTM EQU 01000000H CPU TESTER (MOT) ATTACHED
307 000007 INTRIOCTM EQU 10000000H I/O TESTER (IOCT) ATTACHED
308 *
309 ***BIT DEFINITIONS
310 *
311 000 0 1 INTRPWFH LIV 0,0,1 POWER-FAIL WARNING
312 000 1 1 INTRRTCH LIV 0,1,1 REAL-TIME CLOCK
313 000 2 1 INTRRYM LIV 0,2,1 PERIPHERAL PORT RECEIVER READY
314 000 3 1 INTRTRYM LIV 0,3,1 PERIPHERAL PORT TRANSMITTER READY
315 * LIV 0,4,1 NOT USED
316 000 5 1 INTRIOBM LIV 0,5,1 I/O BUSY
317 000 6 1 INTRMOTM LIV 0,6,1 CPU TESTER (MOT) ATTACHED
318 000 7 1 INTRIOCTM LIV 0,7,1 I/O TESTER (IOCT) ATTACHED

```

325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423

```

*
***STATUS SENSE REGISTER DEFINITION
*
***MASK DEFINITIONS
*
000001  STATINR EQU 00000010  WORD INPUT - BIT 8
000002  STATINR EQU 00000010  WORD INPUT - BIT 9
000004  STATMEM EQU 00000100  MEMORY PROTECT ENABLED (1 => ENABLED)
000010  STATWDR EQU 00001000  WDR RUN (1 => RUN)
000020  STATERR EQU 00010000  PARITY/FRAMING ERROR (1 => ERROR)
000040  STATOVR EQU 00100000  NO OVERRUN ERROR (0 => ERROR)
000100  STATEIA EQU 01000000  EIA STATUS (1 => EIA)
* EQU 10000000  NOT USED
*
***BIT DEFINITIONS
*
000 0 1  STATINR LIV 0,0,1  WORD INPUT - BIT 8
000 1 1  STATINR LIV 0,1,1  WORD INPUT - BIT 9
000 2 1  STATMEM LIV 0,2,1  MEMORY PROTECT ENABLED (1 => ENABLED)
000 3 1  STATWDR LIV 0,3,1  WDR RUN (1 => RUN)
000 4 1  STATERR LIV 0,4,1  PARITY/FRAMING ERROR (1 => ERROR)
000 5 1  STATOVR LIV 0,5,1  NO OVERRUN ERROR (0 => ERROR)
000 6 1  STATEIA LIV 0,6,1  EIA STATUS (1 => EIA)
* LIV 0,7,1  NOT USED
*
***I/O INTERFACE CONTROL REGISTER
*
000001  IOCRDTR EQU 00000001  WORD OUTPUT - BIT 8
000002  IOCRDTR EQU 00000010  WORD OUTPUT - BIT 9
000004  IOCRDOUT EQU 00000100  DISCRETE OUTPUT ENABLE
000010  IOCRDIN EQU 00001000  DISCRETE INPUT ENABLE
000020  IOCRDOUT EQU 00010000  WORD OUTPUT ENABLE
000040  IOCRWIN EQU 00100000  WORD INPUT ENABLE
* EQU 01000000  NOT USED
000200  IOCFLED EQU 10000000  PROGRAMMING PANEL LED DISPLAY CONTROL
*
*
***SCRATCHPAD ALLOCATION
*
***CALCULATE SPACE
*
000000  CALCPHI EQU 0  CALCULATE : B-VALUE HI / PRESET HI
000001  CALCPL0 EQU CALCPHI+1  CALCULATE : B-VALUE LO / PRESET LO
000002  CALCCHI EQU CALCPL0+1  CALCULATE : C-VALUE HI
000003  CALCCL0 EQU CALCCHI+1  CALCULATE : C-VALUE LO
000004  CALCDHI EQU CALCCL0+1  CALCULATE : D-VALUE HI
000005  CALCDL0 EQU CALCDHI+1  CALCULATE : D-VALUE LO
000006  CALCADRH EQU CALCDL0+1  CALCULATE : DIVIDEND NODE HI
000007  CALCADRL EQU CALCADRH+1  CALCULATE : DIVIDEND NODE LO
000010  CALCNT EQU CALCADRL+1  CALCULATE : SCRATCH COUNTER AND MASK
000011  DIVDX1KH EQU CALCNT+1  CALCULATE : PARTIAL HI DIVIDEND
000012  DIVDX1KL EQU DIVDX1KH+1  CALCULATE : PARTIAL MIDDLE DIVIDEND
000013  DIVDX1KL EQU DIVDX1KL+1  CALCULATE : PARTIAL LO DIVIDEND
000014  DIVDFLAG EQU DIVDX1KL+1  DIVIDEND OK FLAG FOR VALIDATE NODE
*
***SYSTEM TIMERS
*
000015  MSTPCLK EQU DIVDFLAG+1  MASTER CLOCK
000016  TIMER001 EQU MSTPCLK+1  TIMER ADDER - 0.01 SECS
000017  TIMER010 EQU TIMER001+1  TIMER ADDER - 0.10 SECS
000020  TIMER100 EQU TIMER010+1  TIMER ADDER - 1.00 SECS
000021  TMR010 EQU TIMER100+1  COUNTER - 0.10 TICKS
000022  TMR100 EQU TMR010+1  COUNTER - 1.00 TICKS
*
***REGISTER SAVE SPACE
*
000023  SAVER1 EQU TMR100+1  SAVE LOCATION- R1
000024  SAVER2 EQU SAVER1+1  SAVE LOCATION- R2
000025  SAVER3 EQU SAVER2+1  SAVE LOCATION- R3
000026  SAVER4 EQU SAVER3+1  SAVE LOCATION- R4
000027  SAVER5 EQU SAVER4+1  SAVE LOCATION- R5
000030  SAVER6 EQU SAVER5+1  SAVE LOCATION - R6
000031  SAVER11 EQU SAVER6+1  SAVE LOCATION - R11
000032  SAVERET EQU SAVER11+1  SAVE LOCATION - LINKAGE
000033  SAVSTATE EQU SAVERET+1  SAVE STATE - USED BY CMDL
*
***DIAGNOSTIC ALLOCATION
*
000034  DIAGSHI EQU SAVSTATE+1  ADDRESS HI
000035  DIAGSLO EQU DIAGSHI+1  ADDRESS LO
000036  DIAGCHK EQU DIAGSLO+1  CHECKSUM
000037  DIAGCTR EQU DIAGCHK+1  LOOP COUNTER
*
***POWER DATA
*
000040  POWERHI EQU DIAGCTR+1  NETWORK NUMBER FOR POWER HI
000041  POWERLO EQU POWERHI+1  NETWORK NUMBER FOR POWER LO
000042  NETWORKH EQU POWERLO+1  CURRENT NETWORK NUMBER HI
000043  NETWORKL EQU NETWORKH+1  CURRENT NETWORK NUMBER LO
000044  POWER EQU NETWORKL+1  POWER FLAG FOR CURRENT NETWORK
000045  POWER1 EQU POWER+1  POWER OUTPUT - COLUMN 1
000046  POWER2 EQU POWER1+1  POWER OUTPUT - COLUMN 2
000047  POWER3 EQU POWER2+1  POWER OUTPUT - COLUMN 3
000050  POWER4 EQU POWER3+1  POWER OUTPUT - COLUMN 4
000051  POWER5 EQU POWER4+1  POWER OUTPUT - COLUMN 5
000052  POWER6 EQU POWER5+1  POWER OUTPUT - COLUMN 6

```


424	000053	POWER7	EQU	POWER7+1	POWER OUTPUT - COLUMN 7
425	000054	POWER8	EQU	POWER8+1	POWER OUTPUT - COLUMN 8
426	000055	POWER9	EQU	POWER9+1	POWER OUTPUT - COLUMN 9
427	000056	POWER10	EQU	POWER10+1	POWER OUTPUT - COLUMN 10
428	000057	POWER11	EQU	POWER11+1	POWER OUTPUT - COLUMN 11
429	000058	POWERPTR	EQU	POWER11+1	POINTER TO POWER ARRAY
430		*			
431		***I/O DATA			
432		*			
433	000061	FRSTPASS	EQU	POWERPTR+1	FLAG - FIRST I/O PASS
434	000062	LEDSTATE	EQU	FRSTPASS+1	LED OUTPUT STATE
435	000063	LEFLOC	EQU	LEDSTATE+1	LED COORDINATES (ROW/COL)
436	000064	COILADDR	EQU	LEFLOC+1	CURRENT COIL ADDR
437		*			
438		***MISCELLANEOUS			
439		*			
440	000065	EOLHI	EQU	COILADDR+1	END OF LOGIC ADDRESS HI
441	000066	EOLLO	EQU	EOLHI+1	END OF LOGIC ADDRESS LO
442	000067	CNTRPWR	EQU	EOLLO+1	COUNTER POWER
443	000070	REG4RDHI	EQU	CNTRPWR+1	DUMMY REGISTER HI
444	000071	REG4RDLO	EQU	REG4RDHI+1	DUMMY REGISTER LO
445		*			
446		***COMMUNICATIONS SPACE			
447		*			
448	000072	CMDCOUNT	EQU	REG4RDLO+1	COMMAND COUNT
449	000073	MSGCHECK	EQU	CMDCOUNT+1	MESSAGE CHECKSUM
450	000074	MSGCOUNT	EQU	MSGCHECK+1	MESSAGE BYTE COUNT
451		*			
452	000075	RCVRBLK	EQU	MSGCOUNT+1	RECEIVER BLOCK
453		*			
454	000075	RCVRBASE	EQU	RCVRBLK	BUFFER BASE
455	000076	RCVRIPTR	EQU	RCVRBASE+1	INPUT POINTER
456	000077	RCVROPTR	EQU	RCVRIPTR+1	OUTPUT POINTER
457	000100	RBUFFLEN	EQU	RCVROPTR+1	BUFFER LENGTH
458	000101	RCVRCNT	EQU	RBUFFLEN+1	BYTE COUNT
459	000102	RCVRSTAT	EQU	RCVRCNT+1	STATUS
460	000103	RCVRLEN	EQU	RCVRSTAT+1	LENGTH LEFT
461		*			
462	000104	XMITBLK	EQU	RCVRLEN+1	TRANSMITTER BLOCK
463		*			
464	000104	XMITBASE	EQU	XMITBLK	BUFFER BASE
465	000105	XMITIPTR	EQU	XMITBASE+1	INPUT POINTER
466	000106	XMITOPTR	EQU	XMITIPTR+1	OUTPUT POINTER
467	000107	XBUFFLEN	EQU	XMITOPTR+1	BUFFER LENGTH
468	000110	XMITCNT	EQU	XBUFFLEN+1	BYTE COUNT
469	000111	XMITSTAT	EQU	XMITCNT+1	STATUS
470		*			
471	000050	RCVRBLEN	EQU	40	RECEIVER BUFFER LENGTH
472	000050	XMITBLEN	EQU	40	TRANSMITTER BUFFER LENGTH
473		*			
474	000112	RCVRBUFF	EQU	XMITSTAT+1	RECEIVER BUFFER
475		*			
476	000162	XMITBUFF	EQU	RCVRBUFF+RCVRBLEN	TRANSMIT BUFFER
477		*			
478	000232	CMD01	EQU	XMITBUFF+XMITBLEN	CURRENT COMMAND - BYTE 1
479	000233	CMD02	EQU	CMD01+1	CURRENT COMMAND - BYTE 2
480	000234	CMD03	EQU	CMD02+1	CURRENT COMMAND - BYTE 3
481	000235	CMD04	EQU	CMD03+1	CURRENT COMMAND - BYTE 4
482	000236	CMD05	EQU	CMD04+1	CURRENT COMMAND - BYTE 5
483	000237	CMD06	EQU	CMD05+1	CURRENT COMMAND - BYTE 6
484	000240	CMD07	EQU	CMD06+1	CURRENT COMMAND - BYTE 7
485	000241	CMD08	EQU	CMD07+1	CURRENT COMMAND - BYTE 8
486	000242	CMD09	EQU	CMD08+1	CURRENT COMMAND - BYTE 9
487	000243	CMD10	EQU	CMD09+1	CURRENT COMMAND - BYTE 10
488	000244	CMD11	EQU	CMD10+1	CURRENT COMMAND - BYTE 11
489	000245	CMD12	EQU	CMD11+1	CURRENT COMMAND - BYTE 12
490	000246	CMD13	EQU	CMD12+1	CURRENT COMMAND - BYTE 13
491	000247	CMD14	EQU	CMD13+1	CURRENT COMMAND - BYTE 14
492	000250	CMD15	EQU	CMD14+1	CURRENT COMMAND - BYTE 15
493	000251	CMD16	EQU	CMD15+1	CURRENT COMMAND - BYTE 16
494	000252	CMD17	EQU	CMD16+1	CURRENT COMMAND - BYTE 17
495	000253	CMD18	EQU	CMD17+1	CURRENT COMMAND - BYTE 18
496	000254	CMD19	EQU	CMD18+1	CURRENT COMMAND - BYTE 19
497	000255	CMD20	EQU	CMD19+1	CURRENT COMMAND - BYTE 20
498	000256	CMD21	EQU	CMD20+1	CURRENT COMMAND - BYTE 21
499	000257	CMD22	EQU	CMD21+1	CURRENT COMMAND - BYTE 22
500	000260	CMD23	EQU	CMD22+1	CURRENT COMMAND - BYTE 23
501	000261	CMD24	EQU	CMD23+1	CURRENT COMMAND - BYTE 24
502	000262	CMDCONT	EQU	CMD24+1	COMMAND CONTINUATION BYTE
503	000263	NOWPAGE	EQU	CMDCONT+1	INSERT AND DELETE FUNCTION PAGE POINTER
504	000264	INPAGE	EQU	NOWPAGE+1	INSERT PAGE #
505	000265	INNUM	EQU	INPAGE+1	INSERT AND DELETE FUNCTION PARAMETER
506	000265	DLNUM	EQU	INNUM	***MUST BE EQU INNUM***
507	000263	SADDRHI	EQU	NOWPAGE	SEARCH ADDRESS
508	000264	SADDRLO	EQU	INPAGE	SEARCH ADDRESS
509		*			
510		***SYSTEM CONFIGURATION			
511		*			
512	000275	SYSSTATE	EQU	159	CURRENT SYSTEM STATE
513	000276	SPDCONF1	EQU	SYSSTATE+1	CONFIGURATION BYTE 1
514	000277	SPDCONF2	EQU	SPDCONF1+1	CONFIGURATION BYTE 2
515		*			
516		* INPUT REGISTER SPACE			
517		*			

518	000300	REG3001H EQU	197	INPUT REGISTER - 3001 HIGH
519	000301	REG3001L EQU	REG3001H+1	INPUT REGISTER - 3001 LOW
520	000302	REG3002H EQU	REG3001L+1	INPUT REGISTER - 3002 HIGH
521	000303	REG3002L EQU	REG3002H+1	INPUT REGISTER - 3002 LOW
522	000304	REG3003H EQU	REG3002L+1	INPUT REGISTER - 3003 HIGH
523	000305	REG3003L EQU	REG3003H+1	INPUT REGISTER - 3003 LOW
524	000306	REG3004H EQU	REG3003L+1	INPUT REGISTER - 3004 HIGH
525	000307	REG3004L EQU	REG3004H+1	INPUT REGISTER - 3004 LOW
526	000310	REG3005H EQU	REG3004L+1	INPUT REGISTER - 3005 HIGH
527	000311	REG3005L EQU	REG3005H+1	INPUT REGISTER - 3005 LOW
528	000312	REG3006H EQU	REG3005L+1	INPUT REGISTER - 3006 HIGH
529	000313	REG3006L EQU	REG3006H+1	INPUT REGISTER - 3006 LOW
530	000314	REG3007H EQU	REG3006L+1	INPUT REGISTER - 3007 HIGH
531	000315	REG3007L EQU	REG3007H+1	INPUT REGISTER - 3007 LOW
532	000316	REG3008H EQU	REG3007L+1	INPUT REGISTER - 3008 HIGH
533	000317	REG3008L EQU	REG3008H+1	INPUT REGISTER - 3008 LOW
534	000320	REG3009H EQU	REG3008L+1	INPUT REGISTER - 3009 HIGH
535	000321	REG3009L EQU	REG3009H+1	INPUT REGISTER - 3009 LOW
536	000322	REG3010H EQU	REG3009L+1	INPUT REGISTER - 3010 HIGH
537	000323	REG3010L EQU	REG3010H+1	INPUT REGISTER - 3010 LOW
538	000324	REG3011H EQU	REG3010L+1	INPUT REGISTER - 3011 HIGH
539	000325	REG3011L EQU	REG3011H+1	INPUT REGISTER - 3011 LOW
540	000326	REG3012H EQU	REG3011L+1	INPUT REGISTER - 3012 HIGH
541	000327	REG3012L EQU	REG3012H+1	INPUT REGISTER - 3012 LOW
542	000330	REG3013H EQU	REG3012L+1	INPUT REGISTER - 3013 HIGH
543	000331	REG3013L EQU	REG3013H+1	INPUT REGISTER - 3013 LOW
544	000332	REG3014H EQU	REG3013L+1	INPUT REGISTER - 3014 HIGH
545	000333	REG3014L EQU	REG3014H+1	INPUT REGISTER - 3014 LOW
546	000334	REG3015H EQU	REG3014L+1	INPUT REGISTER - 3015 HIGH
547	000335	REG3015L EQU	REG3015H+1	INPUT REGISTER - 3015 LOW
548	000336	REG3016H EQU	REG3015L+1	INPUT REGISTER - 3016 HIGH
549	000337	REG3016L EQU	REG3016H+1	INPUT REGISTER - 3016 LOW
550	000340	REG3017H EQU	REG3016L+1	INPUT REGISTER - 3017 HIGH
551	000341	REG3017L EQU	REG3017H+1	INPUT REGISTER - 3017 LOW
552	000342	REG3018H EQU	REG3017L+1	INPUT REGISTER - 3018 HIGH
553	000343	REG3018L EQU	REG3018H+1	INPUT REGISTER - 3018 LOW
554	000344	REG3019H EQU	REG3018L+1	INPUT REGISTER - 3019 HIGH
555	000345	REG3019L EQU	REG3019H+1	INPUT REGISTER - 3019 LOW
556	000346	REG3020H EQU	REG3019L+1	INPUT REGISTER - 3020 HIGH
557	000347	REG3020L EQU	REG3020H+1	INPUT REGISTER - 3020 LOW
558	000350	REG3021H EQU	REG3020L+1	INPUT REGISTER - 3021 HIGH
559	000351	REG3021L EQU	REG3021H+1	INPUT REGISTER - 3021 LOW
560	000352	REG3022H EQU	REG3021L+1	INPUT REGISTER - 3022 HIGH
561	000353	REG3022L EQU	REG3022H+1	INPUT REGISTER - 3022 LOW
562	000354	REG3023H EQU	REG3022L+1	INPUT REGISTER - 3023 HIGH
563	000355	REG3023L EQU	REG3023H+1	INPUT REGISTER - 3023 LOW
564	000356	REG3024H EQU	REG3023L+1	INPUT REGISTER - 3024 HIGH
565	000357	REG3024L EQU	REG3024H+1	INPUT REGISTER - 3024 LOW
566	000360	REG3025H EQU	REG3024L+1	INPUT REGISTER - 3025 HIGH
567	000361	REG3025L EQU	REG3025H+1	INPUT REGISTER - 3025 LOW
568	000362	REG3026H EQU	REG3025L+1	INPUT REGISTER - 3026 HIGH
569	000363	REG3026L EQU	REG3026H+1	INPUT REGISTER - 3026 LOW
570	000364	REG3027H EQU	REG3026L+1	INPUT REGISTER - 3027 HIGH
571	000365	REG3027L EQU	REG3027H+1	INPUT REGISTER - 3027 LOW
572	000366	REG3028H EQU	REG3027L+1	INPUT REGISTER - 3028 HIGH
573	000367	REG3028L EQU	REG3028H+1	INPUT REGISTER - 3028 LOW
574	000370	REG3029H EQU	REG3028L+1	INPUT REGISTER - 3029 HIGH
575	000371	REG3029L EQU	REG3029H+1	INPUT REGISTER - 3029 LOW
576	000372	REG3030H EQU	REG3029L+1	INPUT REGISTER - 3030 HIGH
577	000373	REG3030L EQU	REG3030H+1	INPUT REGISTER - 3030 LOW
578	000374	REG3031H EQU	REG3030L+1	INPUT REGISTER - 3031 HIGH
579	000375	REG3031L EQU	REG3031H+1	INPUT REGISTER - 3031 LOW
580	000376	REG3032H EQU	REG3031L+1	INPUT REGISTER - 3032 HIGH
581	000377	REG3032L EQU	REG3032H+1	INPUT REGISTER - 3032 LOW
582		*		
583		*	COMPLETE AVAILABLE SCRATCHPAD SPACE	
584		*		
585	000010	SPDAVAIL EQU	SYSSIZE-INNUM	
586		*		
587		***RECEIVED ALL TRANSMITTED STATUS		
588		*		
589		*		
590		***TASK DEFINITIONS		
591		*		
592				FIA ACTIVE
593	000200	RCVREJAY EQU	01000000H	MESSAGE IN PROGRESS
594	000100	RCVRESGN EQU	01000000H	BUFFER OVERFLOW
595	000040	RCVROVER EQU	00100000H	FUNCTION CODE READ
596	000020	RCVRCODE EQU	00010000H	LENGTH CODE READ
597	000010	RCVRLNTH EQU	00001000H	
598		*		
599	000040	XMITOVR EQU	00100000H	BUFFER OVERFLOW
600		*		
601	102 7 1	RCVREJAY RIV	RCVSTAT,7,1	FIA ACTIVE
602	102 6 1	RCVRESGN RIV	RCVSTAT,6,1	MESSAGE IN PROGRESS
603	102 5 1	RCVROVER RIV	RCVSTAT,5,1	BUFFER OVERFLOW
604	102 4 1	RCVRCODE RIV	RCVSTAT,4,1	FUNCTION CODE READ
605	102 3 1	RCVRLNTH RIV	RCVSTAT,3,1	LENGTH CODE READ
606		*		
607		***TRANSMIT STATUS		
608		*		
609		*	XMITSTAT,7,1	NOT USED
610		*	XMITSTAT,6,1	NOT USED
611	111 5 1	XMITOVR RIV	XMITSTAT,5,1	BUFFER OVERFLOW
612		*	XMITSTAT,4,1	NOT USED
613		*	XMITSTAT,3,1	NOT USED


```

614          *          RIV    XMITSTAT,2,1      NOT USED
615          *          RIV    XMITSTAT,1,1      NOT USED
616          *          RIV    XMITSTAT,0,1      NOT USED
617          *
618          ***BUFFER DEFINITIONS
619          *
620          000000  BFBASE  EQU    0          BASE ADDRESS
621          000001  BFIPTR  EQU    BFBASE+1     INPUT POINTER
622          000002  BFOPTR  EQU    BFIPTR+1     OUTPUT POINTER
623          000003  BFLEN  EQU    BFOPTR+1     BUFFER LENGTH
624          000004  BFUSE  EQU    BFLEN+1      USAGE COUNT
625          *
626          *** INSERT AND DELETE COMMAND FLAGS
627          *
628          263 7 1  ENT1STH RIV    NOWPAGE,7,1   FIRST ENTRY FLAG
629          263 6 1  PASS1STH RIV   NOWPAGE,6,1   FIRST PAGE MOVE FLAG
630          263 0 6  NOWPAGEH RIV   NOWPAGE,0,6   PAGE BEING WORKED ON
631          000200  ENT1STM  EQU    10000000H
632          000100  PASS1STH EQU    01000000H
633          *
634          ***LOGIC RAM SYSTEM ALLOCATION
635          *
636          000000  SYSLRCHK EQU    0          LOGIC RAM CHECKSUM
637          *
638          ***LOGIC RAM ADDRESS ASSIGNMENTS
639          *
640          000000  SYSLRCHH EQU    0          LOGIC RAM CHECKSUM
641          000000  SYSLRCHL EQU    0
642          *
643          000000  SYSUSERH EQU    0          START OF USER LOGIC
644          000002  SYSUSERL EQU    2
645          *
646          ***COIL RAM ADDRESS ASSIGNMENTS
647          *
648          ***10/25/77
649          *
650          000001  SYSCRCHH EQU    1          COIL RAM CHECKSUM
651          000000  SYSCRCHL EQU    0
652          *
653          000001  SYSSTATH EQU    SYSCRCHH     SYSTEM STATE
654          000001  SYSSTATL EQU    SYSCRCHL+1
655          *
656          ***BASE OF REGISTER SPACE
657          *
658          000001  SYSREGHI EQU    1          COIL [256]
659          000002  SYSREGLO EQU    2
660          *
661          ***LOGIC RAM BIT ASSIGNMENTS
662          *
663          ***SYSCONF1
664          *
665          ***MASK DEFINITIONS
666          *
667          000200  SYS4096M EQU    10000000H   4096 BYTE LOGIC RAM
668          000100  SYS2048M EQU    01000000H   2048 BYTE LOGIC RAM
669          000040  SYS1024M EQU    00100000H   1024 BYTE LOGIC RAM
670          000020  SYSC512M EQU    00010000H   512 BYTE LOGIC RAM
671          000010  SYSC256M EQU    00001000H   256 BYTE LOGIC RAM
672          *          EQU    00000100H       NOT USED
673          *          EQU    00000010H       NOT USED
674          *          EQU    00000001H       NOT USED
675          *
676          ***BIT DEFINITIONS
677          *
678          000 7 1  SYS4096B RIV    0,7,1       4096 BYTE LOGIC RAM
679          000 6 1  SYS2048B RIV    0,6,1       2048 BYTE LOGIC RAM
680          000 5 1  SYS1024B RIV    0,5,1       1024 BYTE LOGIC RAM
681          000 4 1  SYSC512B RIV    0,4,1       512 BYTE LOGIC RAM
682          000 3 1  SYSC256B RIV    0,3,1       256 BYTE LOGIC RAM
683          *          RIV    0,2,1           NOT USED
684          *          RIV    0,1,1           NOT USED
685          *          RIV    0,0,1           NOT USED
686          *
687          ***SYSCONF2
688          *
689          ***MASK DEFINITIONS
690          *
691          000200  SYSC256M EQU    10000000H   256 I/O POINTS
692          000100  SYSC192M EQU    01000000H   192 I/O POINTS
693          000040  SYSC128M EQU    00100000H   128 I/O POINTS
694          000020  SYSC064M EQU    00010000H   64 I/O POINTS
695          *          EQU    00001000H       NOT USED
696          000004  SYSTRANM EQU    00000100H   TRANSITIONAL OPTION
697          000002  SYSENHM  EQU    00000010H   ENHANCED EXECUTIVE
698          *          EQU    00000001H       NOT USED
699          *
700          ***BIT DEFINITIONS
701          *
702          000 7 1  SYSC256B RIV    0,7,1       256 I/O POINTS
703          000 6 1  SYSC192B RIV    0,6,1       192 I/O POINTS
704          000 5 1  SYSC128B RIV    0,5,1       128 I/O POINTS
705          000 4 1  SYSC064B RIV    0,4,1       64 I/O POINTS
706          *          RIV    0,3,1           NOT USED

```


712	000 2 1	SYSTRANS RIV	0,2,1	TRANSITIONAL OPTION
713	000 1 1	SYSENHR RIV	0,1,1	ENHANCED EXECUTIVE
714		* RIV	0,1,1	NOT USED
716				
717		***STATE VECTOR		
718		*		
719		***MASK DEFINITIONS		
720		*		
721	000200	SYSSRUNM EQU	10000000	RUN STATE
722	000100	SYSSPUPM EQU	01100000	POWER-UP STATE
723	000040	SYSSPDM EQU	00100000	POWER-DOWN STATE
724	000020	SYSSTOPM EQU	00010000	STOP STATE
725	000017	SYSCEM EQU	00011111	ERROR CODE MASK
726		*		
727		***PIT DEFINITIONS		
728		*		
729	000 7 1	SYSSRUNM RIV	0,7,1	RUN STATE
730	000 6 1	SYSSPUPM RIV	0,6,1	POWER-UP STATE
731	000 5 1	SYSSPDM RIV	0,5,1	POWER-DOWN STATE
732	000 4 1	SYSSTOPM RIV	0,4,1	STOP STATE
733	000 0 4	SYSCEM RIV	0,0,4	ERROR STATE CODE
734		* RIV	0,2,0	
735		* RIV	0,1,0	
736		* RIV	0,0,0	
737		*		
738		***ERROR STATE CODES		
739		*		
740	000001	SYSEOVR EQU	1	COMMUNICATIONS OVERRIDE
741	000002	SYSELCHK EQU	2	MEMORY CHECKSUM FAILURE
742	000003	SYSNODE EQU	3	INVALID NODE TYPE FOUND
743	000004	SYSIO EQU	4	I/O PORT ERROR
744	000005	SYSESPD EQU	5	SCRATCHPAD DIAGNOSTIC FAILED
745	000006	SYSECCHY EQU	6	COIL RAM CHECKSUM FAILED
746	000007	SYSFBIAG EQU	7	CFU DIAGNOSTIC FAILED
747	000010	SYSMEM EQU	8	ILLEGAL MEMORY CONFIGURATION
748	000011	SYSRTC EQU	9	REAL-TIME CLOCK NOT FUNCTIONING
749	000012	SYSWDT EQU	10	WATCH-DOG TIMER EXPIRED
750	000013	SYSECOL EQU	11	ILLEGAL COLUMN DETECTED
751	000014	SYSECOL EQU	12	NO END-OF-LOGIC NODE
752		* EQU	13	NOT USED
753		* EQU	14	NOT USED
754		* EQU	15	NOT USED
755		*		
756		***COIL RAM HIT ASSIGNMENTS		
757		*		
758				
759	000 3 1	CRINDISH LIV	0,3,1	INPUT DISABLE
760	000 2 1	CRINPUT LIV	0,2,1	INPUT STATE
761	000 1 1	CROUTPUT LIV	0,1,1	OUTPUT COIL STATE
762	000 0 1	CRINTNL LIV	0,0,1	INTERNAL COIL STATE
763		*		
764		***TRANSITIONAL EXTENSION		
765		*		
766		* LIV	0,7,1	NOT USED
767	000 6 1	CRINPIS LIV	0,6,1	INPUT HISTORY
768	000 5 1	CROUTHIS LIV	0,5,1	OUTPUT HISTORY
769	000 4 1	CRINTHIS LIV	0,4,1	INTERNAL COIL HISTORY
770		*		
771		***SYSTEM CONFIGURATION SENSE ADDRESS DATA		
772		*		
773	000000	SYSECNLO EQU	0000	LOW-ORDER ADDRESS
774	000004	SYSCONHI EQU	0040	HIGH-ORDER ADDRESS
775	000100	SYSCONIN EQU	1000	INCREMENT TO LOW-ORDER ADDRESS
776		*		
777		***NODE TYPE DEFINITION		
778		*		
779				
780	000000	NODESON EQU	00	START OF NETWORKS
781	000001	NODEECL EQU	01	END OF LOGIC
782	000002	NODEECC EQU	02	END OF COLUMN
783	000003	NODEOREL EQU	03	NORMALLY OPEN RELAY
784	000004	NODECREL EQU	04	NORMALLY CLOSED RELAY
785	000005	NODEPOST EQU	05	POSITIVE-GOING TRANSITIONAL
786	000006	NODENEG EQU	06	NEGATIVE-GOING TRANSITIONAL
787	000007	NODECOIL EQU	07	COIL
788	000010	NODELATCH EQU	08	LATCH
789	000011	NODEDCOIL EQU	09	DISABLED COIL
790	000012	NODEDLATCH EQU	10	DISABLED LATCH
791	000013	NODEHOPEN EQU	11	HORIZONTAL OPEN
792	000014	NODEHSHORT EQU	12	HORIZONTAL SHORT
793	000015	NODECPRE EQU	13	PRESET/CALCULATE-P-NODE CONSTANT
794	000016	NODERPRE EQU	14	PRESET/CALCULATE-R-NODE REGISTER
795	000017	NODECTR EQU	15	COUNTER
796	000020	NODET100 EQU	16	TIMER - 1.00 SEC
797	000021	NODET010 EQU	17	TIMER - 0.10 SEC
798	000022	NODET001 EQU	18	TIMER - 0.01 SEC
799	000023	NODECON EQU	19	CONVERT
800	000024	NODECCON EQU	20	C NODE CONSTANT
801	000025	NODECREG EQU	21	C NODE REGISTER
802	000026	NODECALC EQU	22	CALCULATE
803	000027	NODENULL EQU	23	NULL NODE
804		* EQU	24	NOT USED
805		* EQU	25	NOT USED
806		* EQU	26	NOT USED
807		* EQU	27	NOT USED
808		* EQU	28	NOT USED
809		* EQU	29	NOT USED
810		* EQU	30	NOT USED
811		* EQU	31	NOT USED

813		*		
814		***NODE INFORMATION		
815		*		
816	000200	NOEEOCK EQU	10000000B	END-OF-COLUMN FLAG
817	000 7 1	NOEEOCK RIV	0,7,1	END-OF-COLUMN BIT
818		*		
819	000037	NOBETYPN EQU	00011111B	MASK FOR NODE TYPE, RIGHT JUSTIFIED
820	000003	NOPEHMSK EQU	00000011B	MASK FOR HIGH-ORDER OPERAND
821		*		
822	000000	FLAGINP EQU	0	INPUT
823	000001	FLAGOUT EQU	1	COIL/LATCH
824	000002	FLAGINT EQU	2	INTERNAL COIL/LATCH
825	000003	FLAGSEQ EQU	3	SEQUENCER
826		*		
827	000000	FLAGHREG EQU	0	HOLDING REGISTER
828	000001	FLAGIREG EQU	1	INPUT REGISTER
829	000002	FLAGDREG EQU	2	DUMMY REGISTER (4100)
830		*		
831		***COUNTER POWER HISTORY		
832	000375	CTRPWRM1 EQU	11111101B	MASK
833	000002	CTRPWRM2 EQU	00000010B	MASK
834	000 1 1	CTRPWRHY RIV	0,1,1	TO GET POWER HIT, RIGHT JUSTIFIED
835	023 2 5	SAVENODE RIV	SAVER1,2,5	NODE TYPE
836		*		
837		***SEQUENCER NODE DATA		
838		*		
839		***MASK DEFINITIONS		
840		*		
841	000340	SEQREGM EQU	11100000B	REGISTER ID
842	000037	SEQSTEFM EQU	00011111B	SEQUENCER STEP
843		*		
844		***BIT DEFINITIONS		
845		*		
846	000 5 3	SEQREGM RIV	0,5,3	REGISTER ID
847	000 0 5	SEQSTEFM RIV	0,0,5	SEQUENCER STEP
848		*		
849	000005	SEQSHIFT EQU	5	ROTATE TO ISOLATE REGISTER
850		*		
851	000063	SEQRASH EQU	51	BASE SEQUENCER REGISTER
852		*		
853	000 0 4	REGDATA LIV	0,0,4	NIBBLE OF HOLDING REGISTER DATA
854	000001	REGPASEM EQU	01H	BASE ADDRESS OF REGISTER SPACE - HIGH
855	000002	REGPASEL EQU	02H	BASE ADDRESS OF REGISTER SPACE - LOW
856		*		
857	000000	CALCADD EQU	00H	ADD
858	000001	CALCSUB EQU	01H	SUBTRACT
859	000002	CALCMPX EQU	10H	MULTIPLY
860	000003	CALCDIV EQU	11H	DIVIDE
861		*		
862		***I/O ASSIGNMENTS		
863		*		
864		***STRIP/BYTE SELECT		
865		*		
866				
867	000001	IOBYTE0 EQU	00000001B	BYTE 0
868	000002	IOBYTE1 EQU	00000010B	BYTE 1
869	000004	IOBYTE2 EQU	00000100B	BYTE 2
870	000010	IOBYTE3 EQU	00001000B	BYTE 3
871	000020	IOSTRIPA EQU	00010000B	STRIP A
872	000040	IOSTRIPB EQU	00100000B	STRIP B
873	000100	IOSTRIPC EQU	01000000B	STRIP C
874	000200	IOSTRIPD EQU	10000000B	STRIP D
875		*		
876		***REGISTER I/O ADDRESS SPACE		
877		*		
878	000 5 3	IOUNITID LIV	0,5,3	UNIT ID
879	000 1 4	IOWORDSL LIV	0,1,4	WORD SELECT
880	000 0 1	IOBYTESEL LIV	0,0,1	BYTE SELECT
881		*		
882		*** I/O ID		
883		*		
884		* EQU 0		NOT USED
885	000001	IOUNITID5 EQU	1	EXTENDED DISCRETE I/O
886	000002	IOUNITR1 EQU	2	REGISTER MUX 1
887	000003	IOUNITR2 EQU	3	REGISTER MUX 2
888		* EQU 4		NOT USED
889		* EQU 5		NOT USED
890		* EQU 6		NOT USED
891		* EQU 7		NOT USED
892		***		
893		*		
894		***ASCII CHARACTERS		
895		*		
896				
897	000002	ASCSTX EQU	002H	STX
898	000320	ASCNAK EQU	320H	NAK
899		*		
900		***COMMUNICATIONS DEFINITIONS		
901		*		
902				
903		***COMMANDS		
904		*		
905				
906	000020	REDCMD EQU	00010000B	READ
907	000040	WRITCMD EQU	00100000B	WRITE COMMAND
908	000060	SEFCMD EQU	00110000B	SEARCH
909	000100	PWRCMD EQU	01000000B	POWER
910	000120	INSCMD EQU	01010000B	INSERT
911	000140	DELCMD EQU	01100000B	DELETE

```

912      0001A0  LEDCMD  EQU  01110000B  LED
913      000200  STPCMD  EQU  10000000B  STOP
914      000220  GOCMD   EQU  10010000B  GO
915      000240  INICMD  EQU  10100000B  INITIALIZE
916      000260  INCCMD  EQU  10110000B  INSERT AT END-OF-COLUMN
917      000300  DECCMD  EQU  11000000B  DELETE AT END-OF-COLUMN
918
919      *
920      * COMMAND CONFIGURATIONS
921      *
921      000001  SRCHCNT  EQU  1  SEARCH CONTINUE
922      000002  INSTCNT  EQU  2  INSERT CONTINUE
923      000003  DLETCNT  EQU  3  DELETE CONTINUE
924      000004  PWRCNT   EQU  4  POWER CONTINUE
925
926      *
927      *
928      0003A0  CMDMSK  EQU  11110000B  COMMAND MASK
929      000017  CNTYSK  EQU  00001111B  COUNT MASK
930      000340  ADMSK   EQU  11110000B  ADDRESS FIELD MASK
931
932      235 5 3  ADRFLD  RIV  CMD(4,5,3)  ADDRESS FIELD IN COMMAND
933      233 0 4  LENFLD  RIV  CMD(2,0,4)  LENGTH FIELD IN COMMAND BYTE
934
935      *
936      *
937      *
937      000001  ERPAK   EQU  1  PARITY/FRAMING ERROR
938      000002  ERROVR  EQU  2  OVERRUN ERROR
939      000003  ERCHK   EQU  3  CHECKSUM ERROR
940      000004  ERRADR  EQU  4  ADDRESS OUT-OF-RANGE
941      000005  ERRADI  EQU  5  INVALID ADDRESS
942      000006  ERRCMD  EQU  6  INVALID COMMAND
943      000007  ERRTIM  EQU  7  TIMEOUT
944      000010  FRPMSK  EQU  8  INVALID MASK
945      000011  ERESER  EQU  9  INVALID STEP NUMBER
946      000012  ERKNOD  EQU  10  INVALID NODE
947      000013  FRMEM   EQU  11  MEMORY PROTECT
948      000014  ENRSTP  EQU  12  SYSTEM NOT IN STOP STATE
949      000015  ERLEN   EQU  13  LENGTH ERROR
950      000016  ERACON  EQU  14  NODE NOT A CONTACT
951      000017  ERNPD   EQU  15  NODE NOT IN POWER DISPLAY
952      000020  ERRSUP  EQU  16  NODE NOT SUPPORTED
953      000021  ERRFUL  EQU  17  MEMORY FULL
954
955      *
956      *
957      *
958      000374  NEG1000H EQU  11111100B  -1000 HI
959      000030  NEG1000L EQU  00011000B  -1000 LO
960      000003  CALCTYP EQU  0111B  CALCULATE TYPE MASK
961      000005  K1000H  EQU  0111B  1000 DE  HI
962      000350  K1000L  EQU  11101000B  1000 DE  LO
963      000374  NEG800H  EQU  11111100B  -800 HI
964      000340  NEG800L  EQU  11100000B  -800 LO
965
966      *
967      *
968      *
969      *
970      *
971      *
972      *
973      *
974      *
975      *
976      *
977      *
978      *
979      *
980      *
981      *
982      *
983      *
984      *
985      *
986      *
987      *
988      *
989      *
990      *
991      *
992      *
993      *
994      *
995      *
996      *
997      *
998      *
999      *
1000
1001
1002
1003
1004
1005
1006
1007
1008

```



```

1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102

```

```

*
OR      MACRO  A,E      A = SOURCE, B = DESTINATION
XOR     A,E      SET NON-DUPLICATE BITS
AND     A,AUX    ISOLATE DUPLICATE BITS
XOR     B,E      SET DUPLICATE BITS
ENDM
END-OF-MACRO
*
***POWER-UP ROUTINE
*
***      CHECK FOR POWER-DOWN FLAG AND MOT FLAG
*
PWRUP   XMT     IVINTRP,IVL   SELECT INTERRUPT SENSE REGISTER
N2T     INTRPWR,PWRUP      TEST FOR POWER-FAIL TRUE
N2T     INTRMOT,PWRUP10    BRANCH IF MOT TESTER CONNECTED
JMP     PWRUP20           NORMAL POWER-UP SEQUENCE
*
PWRUP10 JMP     PWRUP10      ***TEMPORARY FOR MCSIM USE***
*
PWRUP20 XMT     IVOCRHI,IVL   SELECT COIL ADDRESS HIGH
XMT     SYSCONHI,R1        R1 <- CONFIGURATION ADDR HIGH
MOV     R1,LR             LOAD ADDRESS REGISTER
XMT     IVOCRLO+IVICRDAT,IVL SELECT COIL ADDR LOW + COIL INPUT
XMT     SYSCONLO,R1       R1 <- CONFIGURATION ADDR LOW
MOV     R1,RB             LOAD ADDRESS REGISTER
CLR     R3                *1 - CLEAR R3
XMT     0,R3
XMT     SYSTRANM,R4       *2 - R4 <- TRANSITION OPTION FLAG
XMT     3,R2              *3 - R2 <- LOOP COUNTER
N2T     LBEIT0,PWRUP30    BRANCH IF NO TRANSITION OPTION
CLR     R4                CLEAR FLAG
XMT     0,R4
*
PWRUP30 XMT     1000,R5      R5 <- BIT PATTERN
CLR     R6                RESET R6
XMT     0,R6
*
PWRUP40 XMT     SYSCONIN,AUX  AUX <- INCREMENT
ADD     R1,R1             UPDATE LOW-ORDER COIL ADDRESS
MOV     R1,RB             LOAD ADDRESS REGISTER
MOV     R5,AUX           *1 - AUX <- MASK
XOR     R6,R6           *2 - SET HIT
MOV     R5(1),R5        *3 - ROTATE MASK
N2T     LBEIT0,PWRUP50    BRANCH IF BIT SET
XOR     R6,R6           CLEAR BIT
*
PWRUP50 XMT     -1,AUX      AUX <- DECREMENT
ADD     R2,R2            DECREMENT COUNTER
N2T     R2,PWRUP40       R2.NE.0 => CONTINUE
XEC     PWRUTAB1(R6),R   EXECUTE LOAD INSTRUCTION
XOR     R3,R3           SET LOGIC RAM SIZE
XOR     R4,R4           SET COIL RAM SIZE
XMT     SYS0256M,AUX     AUX <- MASK
AND     R3,AUX          AUX.NE.0 => 256-BYTE LOGIC RAM
ADD     R4,R4           SHIFT FLAG IN SYSCONF2 IF NECESSARY
*
***CONDITIONAL ASSEMBLY TO SET ENHANCED INSTRUCTION SET FLAG
***IN SYSCONF2.
*
IF      ENHANCE-1
ENDIF
*
IF      ENHANCE
XMT     SYSTEMM,AUX      AUX <- MASK
ENDIF
*
XOR     R4,R4           SET/CLEAR FLAG AS REQUIRED
JMP     PWRUP60         SKIP TABLE
*
***POWER-UP TABLE FOR SYSTEM CONFIGURATION
*
PWRUTAB1 XMT     SYS0256M,AUX  256 BYTE LOGIC RAM
XMT     SYS0512M,AUX        512 BYTE LOGIC RAM
JMP     PWRUP10           768 BYTE LOGIC RAM - ILLEGAL
XMT     SYS1024M,AUX        1024 BYTE LOGIC RAM
JMP     PWRUP10           1536 BYTE LOGIC RAM - ILLEGAL
XMT     SYS2048M,AUX        2048 BYTE LOGIC RAM
JMP     PWRUP10           3072 BYTE LOGIC RAM - ILLEGAL
XMT     SYS4096M,AUX        4096 BYTE LOGIC RAM
*
***SCRATCHPAD MARCHING ZEROS TEST
*
***THIS TEST ENHANCED TO RUN LOOP EIGHT TIMES AND THUS CLEAR THE
***ENTIRE SCRATCHPAD. ROTATE OF PATTERN IS CHANGED TO AN ADD WHICH
***DOES THE SHIFT AND EVENTUALLY CLEARS THE SCRATCHPAD.
*
PWRUP60 XMT     000000010,R11  R11 <- INITIAL PATTERN
CLR     R1              R1 IS SCRATCHPAD ADDRESS
XMT     0,R1
XMT     IVOSPD+IVISPD,IVL   SELECT SCRATCHPAD IN AND OUT
XMT     1,AUX          AUX <- INCREMENT
*
PWRUP70 MOV     R1,IVR      LOAD SCRATCHPAD ADDRESS
MOV     R11,RB         WRITE TO SCRATCHPAD
ADD     R1,R1          *1 - INCREMENT ADDRESS
N2T     R1,PWRUP70     R1.NE.0 => CONTINUE
*

```

```

1103 00066 6 02010 XMT 4,R2 R2 <- LOOP COUNTER
1104 00067 0 11705 MOV R11(7),R5 S1 <- NEXT PATTERN
1105
*
1106 00070 0 01017 PWRUP100 MOV R1,IVR LOAD ADDRESS REGISTER
1107 00071 0 11000 MOV R11,AUX *1 - AUX <- PATTERN
1108 00072 3 37000 XOR R0,AUX AUX <- MATCH TEST
1109 00073 5 00331 NZT AUX,PWRUPE27 AUX.NE.0 => TEST FAILED
1110 00074 0 05037 MOV R5,R0 WRITE NEW PATTERN
1111 00075 6 00001 XMT 1,AUX AUX <- INCREMENT
1112 00076 1 01001 ADD R1,R1 INCREMENT ADDRESS
1113 00077 5 01070 NZT R1,PWRUP080 R1.NE.0 => CONTINUE
1114 00100 0 11000 MOV R11,AUX AUX <- CURRENT PATTERN
1115 00101 1 11011 ADD R11,R11 ROTATE MASK
1116 00102 0 05000 MOV R5,AUX UPDATE PATTERN
1117 00103 1 05005 ADD R5,R5 USING ADD FOR SHIFT
1118 00104 6 00377 XMT -1,AUX AUX <- DECREMENT
1119 00105 1 02002 ADD R2,R2 R2 <- R2 - 1
1120 00106 5 02070 NZT R2,PWRUP080 R2.NE.0 => CONTINUE
1121
*
1122
*
1123 ***LOAD CONFIGURATION DATA TO SCRATCHPAD
1124
*
1125 00107 6 07001 XMT IVOSPD,IVL SELECT SCRATCHPAD WRITE
1126 00110 6 17276 XMT SPDCONF1,IVR LOAD ADDRESS
1127 00111 0 03037 MOV R3,R0 WRITE SYSCONF1
1128
*
1128 00112 0 00000 + MOV AUX,AUX *1 - WAIT
1129 00113 6 17277 XMT SPDCONF2,IVR LOAD ADDRESS
1130 00114 0 04037 MOV R4,R0 WRITE SYSCONF2
1131
*
1132 ***INITIALIZE PERIPHERAL PORT INTERFACE
1133
*
1134 ***NOTE: PREVIOUS SCRATCHPAD DIAGNOSTIC CLEARED ENTIRE SCRATCHPAD
1135 *** TO ZEROS THUS RESETTING ALL CONSTANTS IN PPI AREA.
1136
*
1137 00115 6 01005 XMT RCVRBLK,R1 R1 <- BLOCK ADDRESS
1138 00116 6 02112 XMT RCVRBUFF,R2 R2 <- BUFFER ADDRESS
1139 00117 6 03050 XMT RCVRBLN,R3 R3 <- BUFFER LENGTH
1140 00120 6 11000 CALL BUFINIT INITIALIZE RECEIVER BUFFER
1141 00121 7 05500
1141 00122 6 01104 XMT XMITBLK,R1 R1 <- BLOCK ADDRESS
1142 00123 6 02162 XMT XMITBUFF,R2 R2 <- BUFFER ADDRESS
1143 00124 6 03050 XMT XMITBLN,R3 R3 <- BUFFER LENGTH
1144 00125 6 11001 CALL BUFINIT INITIALIZE BUFFER
1145 00126 7 05500
1145
*
1146 ***CLEAR PERIPHERAL PORT RECEIVER
1147
*
1148 00127 6 07000 XMT IVCTRL,IVL SELECT CONTROL
1149 00130 6 27304 XMT CTRLCLR,CTRLREG
1150
*
1151 ***VALIDATE LOGIC RAM CHECKSUM
1152
*
1153
*
1154 00131 6 11002 CALL LRCHK COMPUTE CHECKSUM
1155 00132 7 05426
1155 00133 6 01000 XMT SYSLRCHH,R1 R1 <- CHECKSUM ADDRHI
1156 00134 6 07004 XMT IVOLRHI,IVL SELECT LOGIC ADDRHI
1157 00135 0 01027 MOV R1,LB LOAD ADDRESS
1158 00136 6 01000 XMT SYSLRCHL,R1 R1 <- CHECKSUM ADDRLO
1159 00137 6 07003 XMT IVOLRLO,IVL SELECT LOGIC ADDRLO
1160 00140 0 01027 MOV R1,LB LOAD ADDRESS
1161 00141 6 07000 XMT IVILRDAT,IVL *1 - SELECT PORT
1162 00142 0 06000 MOV R6,AUX *2 - AUX <- COMPUTED CHECKSUM
1163
*
1163 00143 0 00000 + MOV AUX,AUX *3 - WAIT
1164 00144 3 37000 XOR R0,AUX AUX.EQ.0 => CHECKSUM OKAY
1165 00145 5 00332 NZT AUX,PWRUPE30 AUX.NE.0 => BAD CHECKSUM
1166
*
1167
*
1168 ***COIL RAM CHECKSUM
1169
*
1170 00146 6 11003 PWRUP100 CALL INTRP DO AN INTERRUPT CHECK
1171 00147 7 05103
1171 00150 5 01000 NZT R1,PWRUP RETRY ON ERROR
1172 00151 6 11004 CALL CRCHK DO CHECKSUM ON COIL RAM
1173 00152 7 05350
1173
*
1174 ***FETCH CHECKSUM STORED IN COIL RAM
1175
*
1176 00153 6 01000 XMT SYSCRCHL,R1 R1 <- COIL CHECKSUM ADDR LOW
1177 00154 6 07000 XMT IVOCRLO,IVL SELECT COIL ADDR LOW
1178 00155 0 01037 MOV R1,R0 LOAD ADDRESS REGISTER
1179 00156 6 01002 XMT SYSCRCHH+1,R1 R1 <- COIL CHECKSUM ADDR HIGH
1180 00157 6 07001 XMT IVOCRHI,IVL SELECT COIL ADDR HIGH
1181 00160 0 01027 MOV R1,LB LOAD ADDRESS REGISTER
1182 00161 0 06000 MOV R6,AUX *1 - AUX <- CALCULATED CHECKSUM
1183 00162 6 07001 XMT IVOCRHI+IVICRDAT,IVL *2 - SELECT PORTS
1184 00163 6 01001 XMT SYSCRCHH,R1 *3 - R1 <- NEXT ADDRESS
1185 00164 0 27011 MOV LB,R11 R11 <- HIGH-ORDER CHECKSUM
1186 00165 0 01027 MOV R1,LB LOAD ADDRESS
1187 00166 6 00360 XMT 111100006,AUX *1 - AUX <- PATTERN
1188 00167 2 11411 AND R11(4),R11 *2 - ISOLATE HIGH-ORDER CHECKSUM
1189 00170 0 00400 MOV AUX(4),AUX *3 - ROTATE PATTERN
1190 00171 2 27000 AND LB,AUX AUX <- LOW-ORDER CHECKSUM
1191 00172 3 11000 XOR R11,AUX AUX <- CHECKSUM
1192 00173 3 06000 XOR R0,AUX AUX.EQ.0 => GOOD CHECKSUM
1193 00174 5 00374 NZT AUX,PWRUPE40 AUX.NE.0 => ERROR, BAD CHECKSUM

```



```

1195
1196
1197
1198
1199
1200
1201 00175 6 17276 *
1202 00176 6 07021 *
1203 00177 0 37003 *
1204 00200 6 00037
1205 00201 2 03303
1206 00202 6 04177
1207 00203 6 01002
1208 00204 6 07003
1209 00205 0 01027
1210 00206 6 01000
1211 00207 6 07004
1212 00210 0 01027
1213
1214 00211 6 01000 *
1215 00212 6 07001
1216 00213 0 01027
1217 00214 5 07001
1218 00215 6 17065
1219 00216 0 01037
1220 00217 6 01002
1221 00220 6 17066
1222 00221 0 01037
1223
1224 00222 6 07000 *
1225 00223 0 37001
1226 00224 6 27300
1227 00225 6 06037
1228 00226 2 01206
1229 00227 6 07021
1230 00230 6 17066
1231 00231 6 00002
1232 00232 1 37037
1233 00233 0 10000
1234 00234 6 17065
1235
1236 00235 0 00000 *
1237 00236 1 37037
1238 00237 6 07000
1239 00240 0 37002
1240 00241 6 27300
1241 00242 6 00007
1242 00243 3 06000
1243 00244 5 00255
1244 00245 0 02037
1245 00246 6 00003
1246 00247 2 01006
1247 00250 4 06302
1248
1249 00251 6 07002
1250 00252 2 27027
1251
1252 00253 6 11005 *
1253 00254 7 06702
1254 00255 6 00037
1255 00256 2 01206
1256 00257 6 00001
1257 00260 3 06006
1258 00261 5 06263
1259 00262 7 00306
1260
1261
1262 00263 6 00377 *
1263 00264 3 01000
1264 00265 5 00271
1265 00266 7 00336
1266
1267 00267 6 01040 *
1268 00270 7 00453
1269
1270
1271 00271 6 00377 *
1272 00272 1 04004
1273 00273 5 04222
1274 00274 6 07060
1275 00275 5 27127
1276 00276 6 04200
1277 00277 1 03003
1278 00300 5 03222
1279 00301 7 00340
1280
1281
1282
1283 00306 6 00377 *
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

```



```

1284 00307 6 06112 XMT 74,R6
1285 00310 6 11054 XMT 44,R11 LOOP SHOULD LAST FOR 10 USEC
1286 00311 6 07060 XMT IVIINTRP+IVUCTRL,IVL SELECT INTERRUPTS AND CONTROL REGISTER
1287 00312 6 27306 XMT CTRLRTC,CTRLREG MAKE SURE RTC BIT CAN GO DOWN
1288 00313 5 26115 NZT INTRTCE,PWRUP165 JUMP IF STILL UP
1289 00314 7 00317 JMP PWRUP170 ELSE DO TIMING LOOP
1290 00315 6 27306 PWRUP165 XMT CTRRTC,CTRLREG KNOCK IT DOWN AGAIN
1291 00316 5 26124 NZT INTRTCE,PWRUP160 IF STILL UP, ERROR
1292 00317 5 26126 PWRUP170 NZT INTRTCE,PWRUP180 TIME HOW LONG TO GO UP
1293 00320 1 06006 ADD R6,R6 LOOP
1294 00321 5 06317 NZT R6,PWRUP170
1295 00322 1 11011 ADD R11,R11
1296 00323 5 11317 NZT R11,PWRUP170 IF FALL THRU, ERROR
1297 *
1298 00324 6 01011 PWRUP160 XMT SYSIRTC,R1 SET ERROR CODE
1299 00325 7 00453 JMP EXEC
1300 *
1301 00326 7 00353 PWRUP180 JMP PWRUP190 SHORT BRANCH PROBLEM
1302 *
1303 ***ERROR HANDLERS
1304 *
1305 *
1306 00327 6 01010 PWRUP160 XMT SYSEMEM,R1 ILLEGAL MEMORY CONFIGURATION
1307 00330 7 00453 JMP EXEC EXIT TO EXEC
1308 *
1309 00331 7 00331 PWRUP160 JMP PWRUP190 SCRATCHPAD DIAGNOSTIC FAILED
1310 * DIE IMMEDIATELY
1311 *
1312 00332 6 01002 PWRUP160 XMT SYSELCHK,R1 LOGIC RAM CHECKSUM FAILED
1313 00333 7 00453 JMP EXEC EXIT TO EXEC
1314 *
1315 00334 6 01006 PWRUP160 XMT SYSECCHK,R1 COIL RAM CHECKSUM FAILED
1316 00335 7 00453 JMP EXEC EXIT TO EXEC
1317 *
1318 00336 6 01003 PWRUP160 XMT SYSENODE,R1 ILLEGAL NODE FOUND
1319 00337 7 00453 JMP EXEC EXIT TO EXEC
1320 *
1321 00340 6 01014 PWRUP160 XMT SYSFEOL,R1 NO END-OF-LOGIC NODE
1322 00341 7 00453 JMP EXEC EXIT TO EXEC
1323 *
1324 ***REMOVE NULL NODES FROM LOGIC MEMORY
1325 *
1326 *
1327 *
1328 00342 6 00002 PWRUP220 XMT 2,AUX INCREMENT TO NEXT NODE
1329 00343 1 04004 ADD R4,R4
1330 00344 0 10000 MOV OV0,AUX
1331 00345 0 03003 ADD R3,R3
1332 00346 0 02000 MOV R2,AUX SEE IF DONE SEARCHING FOR NULLS
1333 00347 3 03000 XOR R3,AUX IF 0, R3 = R2
1334 00350 5 00361 NZT AUX,PWRUP200
1335 *
1336 ***EXIT TO EXEC
1337 *
1338 00351 6 01200 PWRUPX XMT SYSSRUNM,R1 SET RUN STATE
1339 00352 7 00453 JMP EXEC EXIT TO EXEC
1340 *
1341 * CODE UP THERE DUE TO SHORT BRANCH PROBLEM
1342 *
1343 PWRUP190 EQU *
1344 00353 6 17065 XMT EOLHI,IVR GET EOL ADDRHI
1345 00354 6 07021 XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ/WRITE
1346 00355 6 00001 XMT 1,AUX
1347 00356 1 37002 ADD R0,R2 R2<- FOLHI+1
1348 00357 6 03000 XMT SYSUSERH,R3 (R3,R4) <- USER LOGIC BEGIN ADDR
1349 00360 6 04002 XMT SYSUSERL,R4
1350 *
1351 00361 6 07004 PWRUP200 XMT IVOLRHI,IVL SET LOGIC ADDRHI
1352 00362 0 03027 MOV R3,LR
1353 00363 6 07003 XMT IVOLRLO+IVILRDAT,IVL SELECT LOGIC ADDRLO AND LOGIC READ
1354 00364 0 04027 MOV R4,LR SET LOGIC ADDRLO
1355 00365 6 00134 XMT NODENULL.L.2,AUX *1 - AUX<- NULLHI
1356 * *2 - WAIT
1357 *
1358 00366 0 00000 * MOV AUX,AUX *3 - WAIT
1359 *
1360 00367 0 00000 * MOV AUX,AUX
1361 00370 3 37000 XOR R0,AUX SEE IF WE HAVE A NULL NODE
1362 00371 5 00342 NZT AUX,PWRUP221 NO BRANCH
1363 00372 0 03001 MOV R3,R1 SAVE R3 IN R1
1364 00373 0 04011 MOV R4,R11 SAVE R4 IN R11
1365 00374 6 00002 XMT 2,AUX FROMADDR = TOADDR+2
1366 00375 1 04006 ADD R4,R6 (R5,R6) <- FROMADDR
1367 00376 0 10000 MOV OV0,AUX
1368 00377 1 03005 ADD R3,R5
1369 00400 6 07004 PWRUP210 XMT IVOLRHI,IVL SELECT LOGIC ADDRHI
1370 00401 0 05027 MOV R5,R5 SET FROMADDR HI
1371 00402 6 07003 XMT IVOLRLO+IVILRDAT,IVL SELECT ADDRLO AND LOGIC READ
1372 00403 0 06027 MOV R6,LR
1373 00404 6 00001 XMT 1,AUX *1 - FOR INCREMENTING
1374 00405 1 06006 ADD R6,R6 *2 - BUMP FROMADDR
1375 00406 0 10000 MOV OV0,AUX *3
1376 00407 1 05005 ADD R5,R5
1377 00410 0 07000 MOV R0,AUX AUX <- DATA
1378 00411 0 04027 MOV R4,LR SET TOADDR LO
1379 00412 6 07003 XMT IVOLRHI,IVL
1380 00413 0 03027 MOV R3,LR SET TOADDR HI
1381 00414 6 07011 XMT IVOLRDAT,IVL *1 - SELECT LOGIC WRITE

```

```

1379 00415 0 00077      MOV     AUX,LR           WRITE DATA
1380 00416 6 00001      XMT     1,AUX
1381 00417 1 04004      ADD     R4,R4           PUMP TOADDR
1382 00420 0 10000      MOV     OVF,AUX
1383 00421 1 03003      ADD     R3,R3
1384 00422 0 02000      MOV     R2,AUX         SEE IF WE ARE THRU MOVING DATA
1385 00423 3 05000      XOR     R5,AUX         IF SO, R5 = R2
1386 00424 5 00000      NZT    AUX,PWRUP210    LOOP UNTIL COMPRESSED ALL
1387 00425 0 01003      MOV     R1,R3         WRESTORE R3 AND R4
1388 00426 0 11004      MOV     R11,R4
1389 00427 7 00361      JMP     PWRUP200

1391
1392
1393
1394 00430 6 11006      PWRDN  CALL    CHCHK    CALCULATE COIL RAM CHECKSUM
      0431 7 05350
1395 00432 6 07000      XMT     IVOCRLO,IVL    SELECT COIL ADDRESS LOW
1396 00433 6 02000      XMT     SYSCRCHL,R2   R2 <- LOW-ORDER ADDRESS
1397 00434 0 02037      MOV     R2,FB        LOAD ADDRESS
1398 00435 6 07001      XMT     IVOCRHI,IVL  SELECT COIL ADDRESS HIGH
1399 00436 6 02001      XMT     SYSCRCHH,R2  R2 <- HIGH-ORDER ADDRESS
1400 00437 0 02027      MOV     R2,LH        LOAD ADDRESS
1401 00440 6 07002      XMT     IVOCRDAT,IVL *1 - SELECT COIL WRITE
1402 00441 0 06027      MOV     R6,LH        WRITE OUT CHECKSUM
1403 00442 6 02002      XMT     SYSCRCHH+1,R2 R2 <- ADDRESS
1404 00443 6 07001      XMT     IVOCRHI,IVL  SELECT PORT
1405 00444 0 02027      MOV     R2,LH        LOAD ADDRESS
1406 00445 0 06406      MOV     R6(4),R6    *1 - ROTATE CHECKSUM
1407 00446 6 07002      XMT     IVOCRDAT,IVL SELECT COIL WRITE
1408 00447 0 06027      MOV     R6,LR        WRITE OUT CHECKSUM
1409
1410
1411
1412 00450 6 07000      XMT     IVOCtrl,IVL   SELECT CONTROL PORT
1413 00451 6 27317      XMT     CTLPROC,CTRLREG RESET THE PROCESSOR
1414 00452 7 00000      JMP     PWRUP        EXIT TO POWER-UP FUNCTION

1416
1417
1418
1419
1420
1421
1422
1423 00453 6 07000      EXEC   XMT     IVOCRLO,IVL  SELECT COIL ADDRESS LOW
1424 00454 6 02001      XMT     SYSSTATL,R2   R2 <- ADDRESS
1425 00455 0 02037      MOV     R2,RR        LOAD ADDRESS
1426 00456 6 07001      XMT     IVOCRHI,IVL  SELECT COIL ADDRESS HIGH
1427 00457 6 02001      XMT     SYSSTATH,R2  R2 <- ADDRESS
1428 00460 0 02027      MOV     R2,LP        LOAD ADDRESS
1429 00461 6 07002      XMT     IVOCRDAT,IVL *1 - SELECT PORT
1430 00462 0 01027      MOV     R1,LR        WRITE DATA
1431 00463 6 02002      XMT     SYSSTATH+1,R2 R2 <- ADDRESS
1432 00464 6 07001      XMT     IVOCRHI,IVL  SELECT PORT
1433 00465 0 02027      MOV     R2,LH        LOAD ADDRESS
1434 00466 0 01401      MOV     R1(4),R1    ROTATE STATE
1435 00467 6 07002      XMT     IVOCRDAT,IVL SELECT PORT
1436 00470 0 01027      MOV     R1,LR        WRITE HIGH-ORDER STATE VECTOR
1437 00471 0 01401      MOV     R1(4),R1    ROTATE STATE HACK
1438 00472 6 00040      XMT     SYSSPDNM,AUX  AUX <- POWER-DOWN MASK
1439 00473 3 01000      XOR     R1,AUX       AUX <- POWER-DOWN CHECK
1440 00474 5 00076      NZT    AUX,EXEC005   AUX.NE.0 => NOT POWER-DOWN
1441 00475 7 00430      JMP     PWRDN        AUX.EQ.0 => POWER-DOWN
1442
1443
1443 00476 6 07021      EXEC05 WSP     SYSSTATE,R1  LOAD NEW STATE TO SCRATCHPAD
+      XMT     IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
1443 00477 6 17275      XMT     SYSSTATE,IVR  LOAD ADDRESS
+      MOV     R1,RR        WRITE DATA
1444 00500 0 01037      XMT     1,R2         R2 <- VALUE
1445 00501 6 02001      XMT     FRSTPASS,IVR  SET FIRST I/O PASS ON STATE CHANGE
1446 00503 0 02037      MOV     R2,RR        SET FLAG
1447
1448 00504 6 11007      EXEC010 CALL   INT&P        DO INTERRUPT CHECK
      00505 7 05103
1449 00506 5 01113      NZT    R1,EXEC015   R1.NE.0 => ERROR STATE
1450 00507 6 17275      XMT     SYSSTATE,IVR  LOAD ADDRESS OF STATE VECTOR
1451 00510 6 07020      XMT     IVISPD,IVL   SELECT PORT
1452 00511 5 30114      NZT    SYSSRNB,EXEC020 BRANCH ON RUN STATE
1453 00512 7 02626      JMP     EXEC030      BRANCH ON NON-RUN STATE
1454
1455 00513 7 00453      EXEC015 JMP     EXEC
1456
1457 00514      EXEC020 EQU     *        FIELD I/O
1459
1460
1461
1462
1463
1464 00514 6 17062      FLDIO  XMT     LEDSTATE,IVR  LOAD SCRATCHPAD ADDRESS
1465 00515 6 07027      XMT     IVISPD+IVOICtrl,IVL SELECT PORTS
1466 00516 6 00003      XMT     IOCRDOUTR+IOCRDOUT9,AUX AUX <- MASK
1467 00517 0 37011      MOV     RB,R11       R11 <- LED STATE
1468 00520 3 11027      XOR     R11,LR       LOAD CONTROL REGISTER
1469 00521 6 00377      XMT     11111111H,AUX AUX <- PATTERN
1470 00522 6 07005      XMT     IVUIDATA,IVL  SELECT PORT

```



```

1471 00523 0 00027      MOV     AUX,LR           WRITE BUS TEST DATA
1472
1473      *
1474      ***WAIT OF 19 INSTRUCTIONS REQUIRED
1475      *
1475 00524 6 01000      +      CLR     R1           *1 - R1 ← FIRST COIL RAMP ADDRESS
1476      XMT     0,R1
1476 00525 6 07021      +      WSP     COILADDR,R1      *2 - WRITE TO SCRATCHPAD
1476 00526 6 17064      +      XMT     IVISPD+IVOSPD,IVL  SELECT SPB READ/WRITE
1476 00527 0 01037      +      MOV     COILADDR,IVR      LOAD ADDRESS
1477 00530 6 07000      XMT     R1,RH           WRITE DATA
1477 00531 0 01037      MOV     R1,RH
1477 00532 6 07001      XMT     IVOCRHI,IVL      *5 - SELECT COIL ADDRESS LOW
1478 00533 0 01027      MOV     R1,LR           *6 - LOAD ADDRESS
1478 00534 6 01021      XMT     IOSTRIP+IOBYTE0,R1 *7 - SELECT COIL ADDRESS HIGH
1478 00535 6 17277      XMT     SPDCNF2,IVR      *8 - LOAD ADDRESS
1478 00536 6 07020      XMT     IVISPD,IVL      *9 - R1 ← INITIAL I/O ADDRESS
1478 00537 6 06002      XMT     2,R6           *10 - LOAD SCRATCHPAD ADDRESS
1478 00540 5 33102      NZT     SYSCON48,FLDI0010 *11 - SELECT PORT
1478 00541 6 06004      XMT     4,R7           *12 - R6 ← LOOP COUNTER
1477 00542 6 07100      FLDI0010 XMT     IVIIDATA,IVL    *13 - SKIP IF ONLY 64 I/O POINTS
1478 00543 6 00377      XMT     1111111H,AUX  *14 - SET LOOP COUNTER FOR 128 I/O
1478      NOP
1478      MOV     AUX,AUX      *15 - SELECT PORT
1478      NOP           *16 - AUX ← MASK
1478      MOV     AUX,AUX      *17 - WAIT
1478      NOP           *18 - WAIT
1478 00544 0 00000      +      MOV     AUX,AUX
1478      MOV     AUX,AUX      *19 - WAIT
1478 00545 0 00000      +      MOV     AUX,AUX
1478      MOV     AUX,AUX      *20 - WAIT PRECAUTION
1478 00547 0 00000      +      XOR     LR,AUX
1478      NZT     AUX,FLDI0040  TEST LOW-ORDER BITS FOR LINES
1478 00551 5 00176      NZT     AUX,FLDI0040  AUX.NE.0 => ERROR
1478 00552 6 07047      XMT     IVIISTAT+IVOICTRL,IVL  SELECT PORTS
1478 00553 0 27003      MOV     LR,R3           R3 ← STATUS SENSE
1478 00554 6 00003      XMT     STATIN8M+STATIN9M,AUX  AUX ← MASK
1478 00555 2 03003      AND     R3,R3           ISOLATE BITS
1478 00556 3 03000      XOR     R3,AUX          AUX.EQ.0 => OKAY
1478 00557 5 00176      NZT     AUX,FLDI0040  AUX.NE.0 => ERROR
1478 00560 0 11027      MOV     R1,LR           CLEAR HIGH-ORDER BITS
1478 00561 6 07005      XMT     IVOIIDATA,IVL    SELECT PORT
1478 00562 0 00027      MOV     AUX,LR           CLEAR LOW-ORDER BITS
1478
1504      *
1505      ***WAIT OF 19 INSTRUCTIONS REQUIRED
1506      *
1507 00563 6 07040      XMT     IVIISTAT,IVL    *1 - SELECT PORT
1508 00564 6 02011      XMT     0,R2           CREATE WAIT LOOP
1509 00565 6 00377      XMT     -1,AUX          AUX ← DECREMENT
1510 00566 1 02002      FLDI0030 ADD     R2,R2           DECREMENT COUNTER
1511 00567 5 02166      NZT     R2,FLDI0030    LOOP UNTIL DONE
1512 00570 6 00003      XMT     STATIN8M+STATIN9M,AUX  AUX ← MASK
1513 00571 2 27003      AND     LR,R3           R3 ← HIGH-ORDER BITS
1514 00572 5 03176      NZT     R3,FLDI0040    R3.NE.0 => ERROR
1515 00573 6 07106      XMT     IVIIDATA+IVOIALDR,IVL  SELECT PORTS
1516 00574 5 27036      NZT     LR,FLDI0040    BRANCH ON ERROR
1517 00575 7 00600      JMP     FLDI0050        CONTINUE
1518
1519 00576 6 01004      FLDI0040 XMT     SYSEIO,R1      I/O TEST FAILED
1520 00577 7 00727      JMP     FLDI00X        GO TO COMMON EXIT
1521
1522      *
1523      ****MAIN PROCESSING LOOP
1524      *
1525 00600 0 01027      FLDI0050 MOV     R1,LR           LOAD I/O ADDRESS
1526 00601 6 17062      XMT     LEDSTATE,IVR    LOAD SCRATCHPAD ADDRESS
1527 00602 6 07027      XMT     IVOICTRL+IVISPD,IVL  SELECT I/O CONTROL
1528 00603 6 00010      XMT     IOCRDIN,AUX      AUX ← DISCRETE INPUT ENABLE
1529 00604 3 37027      XOR     RH,LR           ENABLE DISCRETE INPUTS
1530
1531      *
1532      ***WAIT OF 26 INSTRUCTIONS REQUIRED
1533      ***BUILD FIRST OUTPUT NIBBLE
1534      *
1534 00605 6 03000      +      CLR     R3           INITIALIZE OUTPUT BYTE
1535 00606 6 11010      +      XMT     0,R3
1535 00607 7 05646      CALL    OUTPUT          BUILD FIRST NIBBLE
1536 00610 6 07100      XMT     IVIIDATA,IVL    SELECT PORT
1537 00611 0 27004      MOV     LR,R4           R4 ← INPUT BYTE
1538 00612 6 17062      XMT     LEDSTATE,IVR    LOAD SCRATCHPAD ADDRESS
1539 00613 6 07027      XMT     IVISPD+IVOICTRL,IVL  SELECT PORTS
1540 00614 0 37027      MOV     RR,LR           TURN OFF STROBE
1541 00615 6 11011      CALL    OUTPUT          BUILD SECOND NIBBLE
1541 00616 7 05646
1542 00617 6 07005      XMT     IVOIIDATA,IVL   SELECT I/O WRITE
1543 00620 0 03027      MOV     R3,LR           LOAD DATA TO BUS
1544
1545      *
1546      ***WAIT OF 19 INSTRUCTIONS REQUIRED
1547      *
1547 00621 6 17064      XMT     COILADDR,IVR    LOAD SCRATCHPAD ADDRESS REGISTER
1548 00622 6 07020      XMT     IVISPD+IVOCFLO,IVL  SELECT PORTS
1549 00623 0 37037      MOV     RR,RR           LOAD COIL ADDRESS LOW
1550 00624 6 11012      CALL    INPUT           UNLOAD FIRST NIBBLE
1550 00625 7 05660
1551 00626 6 07007      XMT     IVOICTRL,IVL    SELECT CONTROL
1552 00627 6 00004      XMT     IOCRDOUT,AUX    AUX ← CONTROL PULSE
1553 00630 0 00027      MOV     AUX,LR           ENABLE OUTPUT STROBE
1554
1555      *
1555      ***WAIT OF 10 INSTRUCTIONS REQUIRED

```



```

1556
1557
1557 00631 6 07021 + WSP SAVER1,R1 SAVE I/O ADDRESS
1557 00632 6 17023 + XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
1557 00633 0 01037 + XMT SAVER1,IVR LOAD ADDRESS
1558 WSP SAVER6,R6 SAVE COUNTER
1558 00634 6 07021 + XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
1558 00635 6 17030 + XMT SAVER6,IVR LOAD ADDRESS
1558 00636 0 06037 + MOV R6,RH WRITE DATA
1559 WSP SAVER4,R4
1559 00637 6 07021 + XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
1559 00640 6 17026 + XMT SAVER4,IVR LOAD ADDRESS
1559 00641 0 04037 + MOV R4,RH WRITE DATA
1560 00642 6 11013 CALL INTKP CHECK INTERRUPTS
1560 00643 7 05103
1561 00644 5 01327 NZT R1,FLDIOX EXIT ON ERROR
1562 RSP SAVER4,R4
1562 00645 6 17026 + XMT SAVER4,IVR LOAD ADDRESS
1562 00646 6 07021 + XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
1562 00647 0 37004 + MOV RB,R4 READ DATA
1563 RSP SAVER1,R1 RESTORE I/O ADDRESS
1563 00650 6 17023 + XMT SAVER1,IVR LOAD ADDRESS
1563 00651 6 07021 + XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
1563 00652 0 37001 + MOV RB,R1 READ DATA
1564 RSP SAVER6,R6 RESTORE COUNTER
1564 00653 6 17030 + XMT SAVER6,IVR LOAD ADDRESS
1564 00654 6 07021 + XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
1564 00655 0 37006 + MOV RB,R6 READ DATA
1565 00656 6 17062 XMT LEDSTATE,IVR LOAD SCRATCHPAD ADDRESS
1566 00657 6 07027 XMT IVISPD+IVDICTRL,IVL SELECT CONTROL
1567 00660 0 37027 MOV RB,LB DISABLE OUTPUT STORE
1568
1569 ***WAIT OF 20 INSTRUCTIONS REQUIRED
1570
1571 00661 6 11014 CALL INPUT UNLOAD SECOND NIBBLE
1571 00662 7 05660
1572 00663 6 17064 XMT COILADDR,IVR SELECT COIL ADDRESS
1573 00664 6 00010 XMT 8,AUX AUX <- INCREMENT
1574 00665 6 07021 XMT IVISPD+IVOSPD,IVL SELECT SCRATCHPAD WRITE AND READ
1575 00666 1 37037 ADD RB,RB UPDATE ADDRESS
1576 00667 6 00017 XMT IOBYTE0+IOBYTE1+IOBYTE2+IOBYTE3,AUX AUX <- PATTERN
1577 00670 2 01003 AND R1,R3 ISOLATE BYTE ID
1578 00671 2 03703 AND R3(7),R3 SHIFT LEFT AND MASK
1579 00672 5 03303 NZT R3,FLDIO070 BRANCH IF STILL ON THIS STRIP
1580 00673 6 03001 XMT IOBYTE0,R3 SET UP FOR BYTE 0 AGAIN
1581 00674 6 00377 XMT -1,AUX AUX <- DECREMENT
1582 00675 1 06006 ADD R6,R6 DECREMENT STRIP COUNTER
1583 00676 5 06300 NZT R6,FLDIO060 BRANCH IF STILL WORKING
1584 00677 7 00713 JMP FLDIO080 FINISHED DISCRETE I/O
1585
1586 00700 6 00360 FLDIO060 XMT IOSTRIPA+IOSTRIPB+IOSTRIPC+IOSTRIPD,AUX AUX <- MASK
1587 00701 2 01001 AND R1,R1 ISOLATE STRIP SELECT
1588 00702 0 01701 MOV R1(7),R1 SELECT NEXT STRIP
1589
1590 00703 6 00360 FLDIO070 XMT IOSTRIPA+IOSTRIPB+IOSTRIPC+IOSTRIPD,AUX AUX <- MASK
1591 00704 2 01000 AND R1,AUX AUX <- STRIP SELECT
1592 00705 3 03001 XOR R3,R1 R1 <- NEW ADDRESS
1593 CLR R3 RESET OUTPUT STATE
1593 00706 6 03000 + XMT 0,R3
1594 00707 6 07005 XMT IVUIDATA,IVL SELECT I/O OUTPUTS
1595 00710 0 03027 MOV R3,LB CLEAR LOW-ORDER DATA
1596 00711 6 07006 XMT IVOIADDR,IVL SELECT I/O ADDRESS
1597 00712 7 00600 JMP FLDIO050 CONTINUE PROCESSING
1599 00713 6 07040 FLDIO090 XMT IVISTAT+IVOCTRL,IVL SELECT PORTS
1600 00714 5 24122 NZT STATWDT,FLDIO090 BRANCH IF WDT RUNNING
1601 00715 6 17061 XMT FRSTPASS,IVR LOAD SCRATCHPAD ADDRESS
1602 00716 6 07020 XMT IVISPD+IVOCTRL,IVL SELECT PORTS
1603 00717 5 37022 NZT RB,FLDIO090 FRSTPASS.NE.0 => TURN ON I/O
1604 00720 6 01012 XMT SYSEWDT,R1 FRSTPASS.EQ.0 => EXPIRED - ERROR
1605 00721 7 00727 JMP FLDIOX GO TO EXIT
1606
1607 00722 6 27305 FLDIO090 XMT CTRLWDT,CTRLREG CTRLREG <- WDT PULSE
1608 CLR R1 INDICATE SUCCESS
1608 00723 6 01000 + XMT 0,R1
1609 WSP FRSTPASS,R1 CLEAR FRSTPASS FLAG
1609 00724 6 07021 + XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
1609 00725 6 17061 + XMT FRSTPASS,IVR LOAD ADDRESS
1609 00726 0 01037 + MOV R1,RB WRITE DATA
1610
1611 00727 5 01331 FLDIOX NZT R1,FLDIOX10 ERROR IF R1 .NE. 0
1612 00730 7 00732 JMP FLDIOX20
1613
1614 00731 7 00453 FLDIOX10 JMP EXEC CHANGE STATE
1615
1616 000732 FLDIOX20 EQU *
1618
1619 * LOGIC MODULE
1620
1621 00732 6 00000 LOGIC000 XMT SYSUSERH,AUX START OF LOGIC ADDR -> LOGIC ADDR REG
1622 00733 6 07004 XMT IVOLRHI,IVL
1623 00734 0 00027 MOV AUX,LR
1624 00735 6 00002 XMT SYSUSERL,AUX
1625 00736 6 07003 XMT IVOLRLO,IVL
1626 00737 0 00027 MOV AUX,LR
1627 CLR AUX *1 0->AUX

```

```

1627 00740 6 00000 * XMT 0,AUX
1628 00741 6 17042 XMT NETWORKH,IVR *2 CLEAR NETWORK #
1629 00742 0 00037 MOV AUX,RR *3
1630 00743 6 07001 XMT IVOCRHI,IVL *1 SELECT COIL ADDR HI
1631 00744 6 17043 XMT NETWORKL,IVR
1632 00745 0 00037 MOV AUX,RR 0->NETWORKL
1633 00746 0 00027 MOV AUX,LH 0->COIL ADDR HI
1634
1635 00747 6 07000 * LOGIC005 XMT IVICRDAT+IVOCTRL,IVL SELECT LOGIC READ & CONTRL PULSE
1636
1637 00750 0 37001 LOGIC010 MOV RB,R1 GET FIRST BYTE OF NODE
1638 00751 6 27300 XMT CTRLINCL,CTRLREG INCREMENT LOGIC ADDR
1639 00752 6 00037 XMT NODETYPM,AUX *1
1640 00753 2 01205 AND R1(2),R5 *2 NODE TYPE->R5
1641 00754 6 00003 XMT NODHMSK,AUX *3 - AUX <- MASK FOR REFERENCE TYPE
1642 00755 0 37002 MOV RB,R2 GET 2ND BYTE OF NODE
1643 00756 0 02037 MOV R2,RR LOAD COIL ADDR LO
1644 00757 0 03703 MOV R3(7),R3 ROTATE POWER.
1645 00760 7 00760 JMP LOGIC035 JMP TO XEC TO SOLVE PAGING PROBLEM
1646
1647 00761 6 07000 * LOGIC020 XMT IVICRDAT+IVOCTRL,IVL SELECT LOGIC RAM READ, CONTROL REG
1648 00762 6 27300 XMT CTRLINCL,CTRLREG INCREMENT LOGIC ADDR
1649 00763 6 00200 XMT NODDEECM,AUX *1
1650 00764 2 01011 AND R1,R11 *2 CHECK FOR END OF COLUMN MARK
1651 00765 6 00377 XMT -1,AUX *3 PREPARE TO SUBTRACT FROM ROW COUNT
1652 00766 5 11373 NZT R11,LOGIC030 IF END OF COLUMN THEN GOTO LOGIC 030
1653 00767 1 04004 ADD R4,R4 ELSE, SUBTRACT FROM ROW COUNT
1654 00770 5 04350 NZT R4,LOGIC010 IF ROW COUNT.NE.0 THEN GOTO LOGIC010
1655 00771 6 01013 XMT SYSECOL,R1 ELSE, ERROR, COLUMN TOO LONG
1656 00772 7 02623 JMP LOGICX
1657
1658 00773 6 11015 LOGIC030 CALL PWRDTE ROTATE AND MASK POWER
1659 00774 7 06546 JMP LOGIC005 ELSE SOLVE NEXT NODE.
1660
1661
1662 00776 7 01000 *
1663 01000 4 05001 XEC LOGICTAB(R5),32 VECTOR THRU JUMP TABLE TO SOLVE NODE
1664
1665
1666 01001 7 01042 LOGICTAB JMP LOG00000 START OF NETWORK
1667 01002 7 01111 JMP LOG01000 END OF LOGIC
1668 01003 7 01143 JMP LOG02000 END OF COLUMN
1669 01004 7 01157 JMP LOG03000 NORMALLY-OPEN RELAY
1670 01005 7 01224 JMP LOG04000 NORMALLY-CLOSED RELAY
1671 01006 7 01271 JMP LOG05000 POSITIVE-GOING TRANSITIONAL
1672 01007 7 01271 JMP LOG06000 NEGATIVE-GOING TRANSITIONAL
1673 01010 7 01327 JMP LOG07000 COIL
1674 01011 7 01327 JMP LOG08000 LATCHED COIL
1675 01012 7 01337 JMP LOG09000 DISABLED COIL
1676 01013 7 01337 JMP LOG10000 DISABLED LATCHED COIL
1677 01014 7 01350 JMP LOG11000 HORIZONTAL OPEN
1678 01015 7 00761 JMP LOG12000 HORIZONTAL CLOSED
1679 01016 7 01353 JMP LOG13000 PRESET/CALCULATE-B-NODE CONSTANT
1680 01017 7 01365 JMP LOG14000 PRESET/CALCULATE-B-NODE REGISTER
1681 01020 7 01407 JMP LOG15000 COUNTER
1682 01021 7 01500 JMP LOG16000 TIMER 1.00
1683 01022 7 01500 JMP LOG17000 TIMER 0.10
1684 01023 7 01500 JMP LOG18000 TIMER 0.01
1685 01024 7 01534 JMP LOG19000 CONVERT NODE
1686 01025 7 02065 JMP LOG20000 CALCULATE-C-NODE CONSTANT
1687 01026 7 02074 JMP LOG21000 CALCULATE-C-NODE REGISTER
1688 01027 7 02112 JMP LOG22000 CALCULATE - D NODE
1689 01030 7 02617 JMP LOG23000 NULL NODE
1690 01031 7 02622 JMP LOG24000 UNASSIGNED - ERROR
1691 01032 7 02622 JMP LOG25000 UNASSIGNED - ERROR
1692 01033 7 02622 JMP LOG26000 UNASSIGNED - ERROR
1693 01034 7 02622 JMP LOG27000 UNASSIGNED - ERROR
1694 01035 7 02622 JMP LOG28000 UNASSIGNED - ERROR
1695 01036 7 02622 JMP LOG29000 UNASSIGNED - ERROR
1696 01037 7 02622 JMP LOG30000 UNASSIGNED - ERROR
1697 01040 7 02622 JMP LOG31000 UNASSIGNED - ERROR
1699
1700 * START OF NETWORK NODE
1701
1702 01041 7 02623 LOG00005 JMP LOGICX ERROR EXIT
1703
1704 01042 6 11016 LOG00000 CALL INTRP CALL INTERRUPT PROCESSOR
1705 01043 7 05103 NZT R1,LOG00005 BRANCH ON ERROR
1706
1707 01045 6 07021 LOG00006 XMT IVISPD+IVOSPD,IVL SELECT SCRATCHPAD READ/WRITE
1708 01046 6 17043 XMT NETWORKL,IVR
1709 01047 6 00001 XMT 1,AUX *1
1710 01050 1 37006 ADD RB,R6 INCREMENT NETWORK #
1711 01051 0 06037 MOV R6,RR
1712 01052 0 10000 MOV OVF,AUX *1 GET OVERFLOW
1713 01053 6 17042 XMT NETWORKH,IVR ADD OVERFLOW TO NETWORK HI
1714 01054 6 04011 XMT 9,R4 *1 RESET ROW COUNTER
1715 01055 1 37005 ADD RB,R5 NETWORKH+OVF->R5
1716 01056 0 05037 MOV R5,RR
1717 CLR R11 *1 CLEAR R11 FOR LATER USE
1718 01057 6 11000 XMT 0,R11
1719 01060 6 17040 XMT POWERHI,IVR CHECK FOR POWER DISPLAY
1720 01061 0 05000 MOV R5,AUX *1 NETWORKH->AUX
1721 01062 3 37000 XOR RR,AUX COMPARE POWERHI TO NETWORKH

```



```

1721          ORG      10,32
1722 01063 5 00101  NZT     AUX,LOG00020  IF NETWORKH.NE.POWERHI THEN GO LOGC0C20
1723 01064 6 17041  XMT     POWERLO,IVR    ELSE COMPARE LOW ORDER
1724 01065 0 06000  MOV     R6,AUX        *1 NETWORKL->AUX
1725 01066 3 37006  XOR     RB,R6         COMPARE POWERLO TO NETWORKL
1726 01067 5 06101  NZT     R6,LOG00020  IF NETWORKL.NE.POWERLO THEN GGO LOG00020
1727          *
1728 01070 6 00001  XMT     1,AUX        INCREMENT => AUX
1729 01071 6 11365  XMT     -11,R11      COUNT => R11
1730 01072 6 05045  XMT     POWER1,R5    TABLE START => R5
1731 01073 0 05017  LOG00010 MOV    R5,IVR       GET TABLE WORD
1732 01074 0 06037  MOV     R6,RB        CLEAR IT.
1733 01075 1 05005  ADD     R5,R5        *1 STFP TO NEXT WORD
1734 01076 1 11011  ADD     R11,R11     COUNT DOWN
1735 01077 5 11073  NZT     R11,LOG00010 LOOP UNTIL R11.EQ.0
1736 01100 6 11001  XMT     1,R11       SET POWER DISPLAY FLAG
1737          *
1738 01101 6 00045  LOG00020 XMT    POWER1,AUX  INITIALIZE POWERPTR
1739 01102 6 17060  XMT    POWERPTR,IVR
1740 01103 0 00037  MOV     AUX,RB
1741 01104 6 11001  XMT     1,R11      *1
1742 01105 6 17044  XMT    POWER,IVR
1743 01106 0 11037  MOV     R11,RB
1744 01107 6 03377  XMT     -1,R3      INIT POWER BITS
1745 01110 7 00761  JMP     LOGIC020   SOLVE NEXT NODE
1747          *
1748          *   END OF LOGIC NODE
1749          *
1750          *   UPDATE LED STATE.
1751 01111 6 17063  LOG01000 XMT    LEDLOC,IVR  GET COORDINATES
1752 01112 6 07021  XMT    IVOSPD+IVISPD,IVL
1753 01113 0 37005  MOV     RB,R5
1754 01114 5 05116  NZT     R5,LOG01010
1755 01115 7 01124  JMP     LOG01020   IF COORDINATES.EQ.[0,0] THEN LED <= 0
1756 01116 6 00017  LOG01010 XMT    01111B,AUX  ELSE, GET ROW => R6
1757 01117 2 05406  AND     R5(4),R6
1758 01120 2 05005  AND     R5,R5      COLUMN => R5
1759 01121 6 00044  XMT    POWER1-1,AUX
1760 01122 1 05017  ADD     R5,IVR     POWER WORD ADDR => IVR
1761 01123 4 06133  XEC    LOG01TAB(R6),8 *1 LED STATE => R5
1762 01124 6 17062  LOG01020 XMT    LEDSTATE,IVR  SAVE LED STATE
1763 01125 0 05105  MOV     R5(1),R5   PUT LED STATE IN BIT 7
1764 01126 0 05037  MOV     R5,RB
1765 01127 6 07007  XMT    IVOICTRL,IVL *1 SELECT I/O CONTROL REGISTER
1766 01130 0 05027  MOV     R5,LB     STROBE LED
1767          CLR    R1      CLEAR ERROR INDICATOR
1767 01131 6 01000  XMT     0,R1
1768 01132 7 02623  JMP     LOGICX     EXIT TO EXEC
1769          *
1770          *
1771          *   EXECUTION TABLE TO GET LED STATE
1772          *
1773 01133 6 05000  LOG01TAB XMT     0,R5      ROW.EQ.0 SET LED.EQ.0
1774 01134 0 20105  MOV     LBBIT7,R5   ROW.EQ.1 ROW 1 POWER BIT => R5(0)
1775 01135 0 21105  MOV     LBBIT6,R5   ROW.EQ.2 ROW 2 POWER BIT => R5(0)
1776 01136 0 22105  MOV     LBBIT5,R5   ROW.EQ.3 ROW 3 POWER BIT => R5(0)
1777 01137 0 23105  MOV     LBBIT4,R5   ROW.EQ.4 ROW 4 POWER BIT => R5(0)
1778 01140 0 24105  MOV     LBBIT3,R5   ROW.EQ.5 ROW 5 POWER BIT => R5(0)
1779 01141 0 25105  MOV     LBBIT2,R5   ROW.EQ.6 ROW 6 POWER BIT => R5(0)
1780 01142 0 26105  MOV     LBBIT1,R5   ROW.EQ.7 ROW 7 POWER BIT => R5(0)
1781          *
1782          *
1783          *   END OF COLUMN NODE
1784          *
1785          *
1786 01143 6 00001  LOG02000 XMT     1,AUX
1787 01144 1 04004  ADD     R4,R4
1788 01145 0 03103  MOV     R3(1),R3
1789 01146 6 11017  CALL    PWROTATE   ROTATE AND MASK POWER
1790          XMT    IVOCOL+IVICOLIN,IVL SELECT PORTS
1791          MOV     R3,LB     COLUMN SOLVER <- POWER BITS
1792 01152 6 04011  XMT     9,R4        *1 - R4 <- COLUMN COUNT
1793          NOP
1793          MOV     AUX,AUX   *2 - WAIT
1794          NOP
1794          MOV     AUX,AUX   *3 - WAIT
1795 01155 0 27003  MOV     LB,R3      R3 <- UPDATED POWER
1796 01156 7 00761  JMP     LOGIC020   SOLVE NEXT COLUMN
1798          *
1799          *   NORMALLY OPEN NODE
1800          *
1801 01157 2 01005  LOG03000 AND     R1,R5      GET REFERENCE TYPE TO R5
1802 01160 6 00376  XMT     1111110B,AUX  MASK -> AUX
1803 01161 4 05164  XEC    LOG03TAB(R5),4  SOLVE NODE
1804 01162 2 03003  AND     R3,R3      UPDATE POWER
1805 01163 7 00761  JMP     LOGIC020   SOLVE NEXT NODE
1806          *
1807 01164 3 25100  LOG03TAB XOR    CRINPUT,AUX  INPUT TYPE REFERENCE
1808 01165 3 26100  XOR    CROUTPUT,AUX  OUTPUT TYPE REFERENCE
1809 01166 3 27100  XOR    CRINTRNL,AUX  INTERNAL COIL REFERENCE
1810          *
1811          *   CONDITIONAL ASSEMBLY FOR ENHANCED SET
1812          *   IF ENHANCED SET, SEQUENCER REFERENCE ARE ALLOWED
1813          *   IF NOT ENHANCED SET, ASSEMBLE A NOP FOR SEQ REF.
1814          *   IF ENHANCE-1
1815          *   ENDIF

```



```

1816
1817
1818 01167 7 01170      *      IF      ENHANCE
1819      *      JMP      LOG03010      SEQUENCER TYPE REFERENCE
1820 01170 6 07021      *      LOG03010 XMT      IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
1821 01171 6 17023      XMT      SAVER1,IVR
1822 01172 0 01037      MOV      R1,RB      SAVE 1ST BYTE OF NODE
1823 01173 6 01000      XMT      0,R1      *1
1824 01174 6 17024      XMT      SAVER2,IVR
1825 01175 0 02037      MOV      R2,RB      SAVE 2ND BYTE OF NODE
1826 01176 6 00340      XMT      SEQREGM,AUX      MASK TO GET SEQ GROUP
1827 01177 2 02502      AND      R2(SEQSHIFT),R2
1828 01200 6 00063      XMT      SEQBASE,AUX      BASE OF SEQUENCER REGISTERS
1829 01201 1 02002      ADD      R2,R2
1830 01202 6 11020      CALL     REGVAL      GET DATA FROM REGISTER
1831 01203 7 05700
1831 01204 6 07021      XMT      IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
1832 01205 6 17024      XMT      SAVER2,IVR      GET 2ND BYTE OF NODE
1833 01206 6 00037      XMT      SEQSTEPM,AUX      *1
1834 01207 2 37000      AND      RB,AUX      GET SEQUENCE REF #
1835 01210 6 11001      XMT      1,R11      MAKE SEQ REF # RELATIVE TO '1'
1836 01211 1 11000      ADD      R11,AUX
1837 01212 3 02000      XOR      R2,AUX      COMPARE WITH SEQ REGISTER VALUE
1838 01213 5 00217      NZT     AUX,LOG03020      IF MISCOMPARE OR
1839 01214 5 01217      NZT     R1,LOG03020      IF R1.NE.0 TURN NODE OFF (NO POWR)
1840 01215 6 00377      XMT      1111111B,AUX      ELSE NODE IS PASSING POWER, SET MASK
1841 01216 7 01220      JMP      LOG03030
1842 01217 6 00376      LOG03020 XMT      11111110B,AUX      NODE NOT PASSING POWER, SET MASK
1843
1844 01220 6 17023      *      LOG03030 XMT      SAVER1,IVR      GET 1ST BYTE OF NODE
1845 01221 2 03003      AND      R3,R3      *1 UPDATE POWER WITH SOLUTION OF NODE
1846 01222 0 37001      MOV      RB,R1      RECOVER 1ST BYTE
1847 01223 7 00761      JMP      LOGIC020      SOLVE NEXT NODE
1848      ENDIF
1850
1851      *      NORMALLY CLOSED NODE
1852
1853 01224 2 01005      *      LOG04000 AND      R1,R5      REFERENCE TYPE -> R5
1854 01225 6 00377      XMT      11111111B,AUX      MASK -> AUX
1855 01226 4 05231      XFC     LOG04TAB(R5),4      SOLVE NODE
1856 01227 2 03003      AND      R3,R3      UPDATE POWER
1857 01230 7 00761      JMP      LOGIC020      SOLVE NEXT NODE
1858
1859 01231 3 25100      *      LOG04TAB XOR      CRINPUT,AUX      INPUT TYPE REFERENCE
1860 01232 3 26100      XOR      CROUTPUT,AUX      OUTPUT TYPE REFERENCE
1861 01233 3 27100      XOR      CRINTRNL,AUX      INTERNAL COIL REFERENCE
1862
1863      *      CONDITIONAL ASSEMBLY FOR ENHANCED SET
1864      *      ASSEMBLE SEQUENCER REFERENCE CODE IF ENHANCED SET
1865      *      ASSEMBLE NOP IF NOT ENHANCED SET
1866      IF      ENHANCE-1
1867      ENDIF
1868 01234 7 01235      IF      ENHANCE
1869      JMP      LOG04010      SEQUENCER TYPE REF
1870 01235 6 07021      *      LOG04010 XMT      IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
1871 01236 6 17023      XMT      SAVER1,IVR
1872 01237 0 01037      MOV      R1,RB      SAVE 1ST BYTE OF NODE
1873 01240 6 01000      XMT      0,R1      *1
1874 01241 6 17024      XMT      SAVER2,IVR
1875 01242 0 02037      MOV      R2,RB      SAVE 2ND BYTE OF NODE
1876 01243 6 00340      XMT      SEQREGM,AUX      MASK TO GET SEQ GROUP
1877 01244 2 02502      AND      R2(SEQSHIFT),R2
1878 01245 6 00063      XMT      SEQBASE,AUX      BASE OF SEQUENCER REGS
1879 01246 1 02002      ADD      R2,R2
1880 01247 6 11021      CALL     REGVAL      GET CONTENTS OF SEQUENCER REGISTER
1881 01251 6 07021      XMT      IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
1882 01252 6 17024      XMT      SAVER2,IVR      GET SECOND BYTE OF NODE
1883 01253 6 00037      XMT      SEQSTEPM,AUX      *1
1884 01254 2 37000      AND      RB,AUX      GET SEQ REF #
1885 01255 6 11001      XMT      1,R11      MAKE SEQ REF. # RELATIVE TO '1'
1886 01256 1 11000      ADD      R11,AUX
1887 01257 3 02000      XOR      R2,AUX      COMPARE WITH REGISTER CONTENTS
1888 01260 5 00264      NZT     AUX,LOG04020      IF MISCOMPARE OR
1889 01261 5 01264      NZT     R1,LOG04020      IF R1.NE.0 THEN NODE PASSING POWER
1890
1891      *      XMT      1111110B,AUX      ELSE NODE NOT PASSING POWER SET MASK
1892 01263 7 01265      JMP      LOG04030
1893
1894 01264 6 00377      *      LOG04020 XMT      11111111B,AUX      NODE PASSING POWER, SET MASK
1895
1896 01265 6 17023      *      LOG04030 XMT      SAVER1,IVR      GET FIRST BYTE OF NODE
1897 01266 2 03003      AND      R3,R3      UPDATE POWER WITH SOLUTION OF NODE
1898 01267 0 37001      MOV      RB,R1      RECOVER R1
1899 01270 7 00761      JMP      LOGIC020      SOLVE NEXT NODE
1900      ENDIF
1902
1903      *      ***ASSEMBLE TRANSITIONAL NODES ONLY IF ENHANCED INSTRUCTION SET.
1904
1905      *      IF      ENHANCE
1906
1907      *      TRANSITIONAL NODES
1908      *      BOTH UP AND DOWN TRANSITIONALS ARE HANDLED HERE
1909      *      DIFFERENCES BETWEEN THE TWO ARE HANDLED BY EXECUTION TABLES
1910
1911      *      TRAN UP = (.NOT.HISTORY).AND.CURRENT
1912      *      TRAN DOWN = (.NOT.CURRENT).AND.HISTORY
1913

```

```

1914
1915
1916
1917
1918
1919 01271 2 01005
1920 01272 6 07021
1921 01273 6 17277
1922 01274 6 00000
1923 01275 2 37000
1924 01276 6 07000
1925 01277 5 00301
1926 01300 6 05003
1927
1928 01301 4 05307
1929 01302 4 05317
1930 01303 6 11376
1931 01304 3 11000
1932 01305 2 03003
1933 01306 7 00761
1934
1935 01307 3 25100
1936 01310 3 26100
1937 01311 3 27100
1938 01312 6 00000
1939 01313 3 21100
1940 01314 3 22100
1941 01315 3 23100
1942 01316 6 00000
1943
1944 01317 2 21100
1945 01320 2 22100
1946 01321 2 23100
1947
1947 01322 0 00000
1948 01323 2 25100
1949 01324 2 26100
1950 01325 2 27100
1951
1951 01326 0 00000
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962 001327
1963 01327 6 07002
1964 01330 2 01005
1965 01331 4 05333
1966 01332 7 00761
1967
1968
1968 01333 0 00000
1969 01334 0 03126
1970 01335 0 03127
1971
1971 01336 0 00000
1972
1973
1974
1975 001337
1976 01337 2 01005
1977 01340 6 00376
1978 01341 2 03003
1979 01342 4 05344
1980 01343 7 00761
1981
1982
1982 01344 0 00000
1983 01345 3 26103
1984 01346 3 27103
1985
1985 01347 0 00000
1987
1988
1989
1990 01350 6 00376
1991 01351 2 03003
1992 01352 7 00761
1993
1994
1995
1996 000761
1997
1999
2000
2001
2002 01353 6 07021
2003 01354 6 17000
2004 01355 2 01037

```

```

* REFERENCES TO INPUTS, OUTPUTS, AND INTERNAL COILS ALLOWED.
* SEQUENCER REFERENCES NOT ALLOWED (NODE SET TO 'NOT PASSING
* POWER ON SEQ REF)
*
LOG05000 EQU *
LOG06000 AND R1,R5 R5 <- REFERENCE TYPE
XMT IVISPD+IVOSPD,IVL SELECT SCRATCHPAD READ/WRITE
XMT SPDCONF2,IVR
XMT SYSTRANB,AUX *1 CHECK IF TRANSITIONALS ARE ALLOWED
AND RB,AUX
XMT IVICRDAT+IVOCRTL,IVL SELECT COIL RAM DATA READ
NXT AUX,LOG05020 IF TRANS ALLOWED THEN GO TO LOG05020
XMT 00000011B,R5 SET 'REFERENCE TYPE' TO SEQUENCER
*
LOG05020 XEC LOG05TAB(R5),8
XEC LOG06TAB(R5),8
XMT 1111110B,R11 MASK SO THAT OTHER BITS ARE NOT ALTERED
XOR R11,AUX
AND R3,R3 UPDATE POWER WITH SOLUTION OF NODE
JMP LOGIC020
*
LOG05TAB XOR CRINPUT,AUX 000 TRAN DOWN INPUT REF 'NOT CURRENT'
XOR CROUTPUT,AUX 001 TRAN DWN OUTPUT REF 'NOT CURRENT'
XOR CRINTRNL,AUX 010 TRAN DOWN INTRNL REF 'NOT CURRENT'
XMT 0,AUX 011 SEQUENCER REF 0-> AUX
XOR CRINHIS,AUX 100 TRAN UP INPUT REF 'NOT HISTORY'
XOR CROUTHIS,AUX 101 TRAN UP OUTPUT REF 'NOT HISTORY'
XOR CRINTHIS,AUX 110 TRAN UP INTRNL REF 'NOT HISTORY'
XMT 0,AUX 111 SEQUENCER REF 0-> AUX
*
LOG06TAB AND CRINHIS,AUX 000 TRAN DWN INPUT REF 'AND HISTORY'
AND CROUTHIS,AUX 001 TRAN DWN OUTPUT REF 'AND HISTORY'
AND CRINTHIS,AUX 010 TRAN DWN INTRNL REF 'AND HISTORY'
NOP 011 SEQUENCER REF.
+ MOV AUX,AUX
AND CRINPUT,AUX 100 TRAN UP INPUT REF 'AND CURRENT'
AND CROUTPUT,AUX 101 TRAN UP OUTPUT REF 'AND CURRENT'
AND CRINTRNL,AUX 110 TRAN UP INTRNL REF 'AND CURRENT'
NOP 111 SEQUENCER REF.
+ MOV AUX,AUX
*
ENDIF
***END OF TRANSITIONAL NODE AREA.
* IF ENHANCED SET IS NOT USED, THIS AREA WILL NOT ASSEMBLE
* AND THE NODE VECTORING WOULD GO TO THE UNASSIGNED NODE TYPE
* AREA (WHICH HAS CONDITIONAL ASSEMBLY FEATURES AS WELL)
*
* COIL AND LATCHED COIL NODES
*
LOG07000 EQU *
LOG08000 XMT IVOCRDAT+IVICRDAT,IVL SELECT COIL RAM WRITE
AND R1,R5 GET REFERENCE TYPE
XEC LOG07TAB(R5),4
JMP LOGIC020 SOLVE NEXT NODE
*
LOG07TAB NOP INPUT REFERENCE
+ MOV AUX,AUX
MOV R3,CROUTPUT OUTPUT COIL REF
MOV R3,CRINTRNL INTERNAL COIL REF
NOP SEQUENCER REF
+ MOV AUX,AUX
*
DISABLED COIL OR DISABLED LATCHED COIL
*
LOG09000 EQU *
LOG10000 AND R1,R5 GET REFERENCE TYPE
XMT 1111110B,AUX
AND R3,R3 SET POWER OFF
XEC LOG09TAB(R5),4 GET POWER STATE
JMP LOGIC020 SOLVE NEXT NODE
*
LOG09TAB NOP INPUT REFERENCE
+ MOV AUX,AUX
XOR CROUTPUT,R3 OUTPUT REFERENCE
XOR CRINTRNL,R3 INTERNAL REFERENCE
NOP SEQUENCER REFERENCE
+ MOV AUX,AUX
*
***HORIZONTAL OPEN
*
LOG11000 XMT 1111110B,AUX AUX <- MASK
AND R3,R3 SHORT STOPS POWER
JMP LOGIC020 CONTINUE
*
***HORIZONTAL SHORT
*
LOG12000 EQU LOGIC020 ALWAYS PASSES POWER
*
* CALCULATE B NODE CONSTANT
*
LOG13000 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD R
XMT CALCBHI,IVR STORE HI ORDER
AND R1,R8

```



```

2005      NOP
2005 01356 0 00000 + MOV AUX,AUX *1
2006 01357 6 17001 + XMT CALCBLO,IVR
2007 01360 0 02037 + MOV R2,RB
2008 01361 6 00377 + XMT -1,AUX *1
2009 01362 6 17006 + XMT CALBADRH,IVR SET DIVIDEND SINGLE PRECISION FLAG
2010 01363 0 00037 + MOV AUX,RB USED FOR DIVIDE NODE
2011 01364 7 00761 + JMP LOGIC020
2013 *
2014 * CALCULATE B NODE REGISTER
2015 *
2016 LOG14000 WSP CALBADRH,R1 SAVE NODE DATA FOR DIVIDE NODE
2016 01365 6 07021 + XMT IVISPD+IVOSPD,IVL SELECT SP: READ/WRITE
2016 01366 6 17006 + XMT CALBADRH,IVR LOAD ADDRESS
2016 01367 0 01037 + MOV R1,RB WRITE DATA
2017 WSP CALBADRL,R2
2017 01370 6 07021 + XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
2017 01371 6 17007 + XMT CALBADRL,IVR LOAD ADDRESS
2017 01372 0 02037 + MOV R2,RB WRITE DATA
2018 01373 6 11022 CALL REGVAL GET REGISTER DATA
2018 01374 7 05700
2019 01375 6 07021 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
2020 01376 6 17000 XMT CALCBHI,IVR SAVE HIGH ORDER DATA
2021 01377 0 01037 MOV R1,RB
2022 NOP *1
2022 01400 0 00000 + MOV AUX,AUX
2023 01401 6 17006 XMT CALBADRH,IVR *1
2024 NOP
2024 01402 0 00000 + MOV AUX,AUX
2025 01403 0 37001 MOV RB,R1 RESTORE 1ST BYTE OF NODE
2026 01404 6 17001 XMT CALCBLO,IVR
2027 01405 0 02037 MOV R2,RB SAVE LOW ORDER REGISTER VALUE
2028 01406 7 00761 JMP LOGIC020 SOLVE NEXT NODE
2030 * COUNTER NODE
2031 * PARTS OF THIS NODE SOLVE (FROM LOG15020) ARE
2032 * USED BY TIMER NODE SOLVE.
2033 *
2034 01407 6 07021 LOG15000 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
2035 01410 6 17023 XMT SAVER1,IVR SAVE 1ST BYTE OF NODE
2036 01411 0 01037 MOV R1,RB
2037 01412 6 11023 CALL REGVAL GET REGISTER VALUE, ADDRESS
2037 01413 7 05700
2038 01414 6 17067 XMT CNTRPWR,IVR GET COUNTER POWER HISTORY
2039 01415 6 07021 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
2040 01416 0 36100 MOV CTRPWRH,AUX POWER HISTORY, RIGHT JUSTIFIED -> AUX
2041 01417 3 03100 XOR R3(1),AUX HIS POW (HP).XOR.CURRENT POWER (CP)
2042 01420 2 03100 AND R3(1),AUX TRANS UP = CP.AND.(CP.XOR.HP)
2043 01421 6 11001 XMT 1,R11
2044 01422 2 11000 AND R11,AUX LSB ONLY
2045 01423 1 02002 ADD R2,R2 ADD TRANS UP PULSE TO COUNT
2046 01424 0 10000 MOV OVF,AUX
2047 01425 1 01001 ADD R1,R1
2048 01426 6 00375 XMT CTRPWRM1,AUX RESET POWER HISTORY, MASK => AUX
2049 01427 2 37011 AND RB,R11 CLEAR 'OLD' HISTORY BIT
2050 01430 6 00002 XMT CTRPWRM2,AUX AUX SINGLE OUT 'NEW' HISTORY
2051 01431 2 03000 AND R3,AUX
2052 01432 3 11037 XOR R11,RB UPDATE HISTORY
2053 CLR R11
2053 01433 6 11000 + XMT 0,R11
2054 01434 6 00001 XMT 1B,AUX CHECK FOR RESET
2055 01435 2 03000 AND R3,AUX
2056 01436 5 00042 NZT AUX,LOG15020 IF .NOT.RESET THEN GOTO LOG15020
2057 CLR R1 ELSE CLEAR COUNT
2057 01437 6 01000 + XMT 0,R1
2058 CLR R2
2058 01440 6 02000 + XMT 0,R2
2059 01441 7 01455 JMP LOG15021
2061 01442 6 07021 LOG15020 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ WRITE
2062 * COUNT (OR TIME) IS IN [R1,R2]
2063 * PRESET IS IN SCRATCH PAD
2064 * [CALCBHI,CALCBLO]
2065 * POWER IS IN R3
2066 * IF COUNT.GE.PRESET
2067 * THEN
2068 * COUNT <- PRESET
2069 * POWER <- POWER.AND.(.NOT.3)
2070 * POWER <- POWER.OR.00000010B
2071 * ELSE
2072 * POWER <- POWER.AND.(.NOT.3)
2073 * POWER <- POWER.OR.00000010B
2074 *
2075 * TO DETERMINE IF COUNT.GE.PRESET
2076 * SET AUX=.NOT.COUNTLO (R2)
2077 * SET AUX= AUX + CALCBLO
2078 * SET R11= OVERFLOW LOW
2079 * SET AUX=.NOT.COUNTHI (R1)
2080 * SET AUX= AUX + CALCBHI
2081 * SET R11= AUX + OVERFLOW LOW (R11)
2082 *
2083 * IF R11(7)=1, THEN COUNT.GT.PRESET
2084 *
2085 * IN REGISTERS, THIS FLOW IS:
2086 * AUX=.NOT.R2
2087 * AUX= AUX + SP[CALCBLO]
2088 * R11= OVF
2089 * AUX=.NOT.R1

```



```

2090
2091
2092
2093 01443 6 17001
2094 01444 6 00377
2095 01445 3 02000
2096 01446 1 37000
2097 01447 0 10011
2098 01450 6 17000
2099 01451 6 00377
2100 01452 3 01000
2101 01453 1 37000
2102 01454 1 11011
2103 01455 6 00374
2104 01456 2 03003
2105 01457 6 00001
2106 01460 3 03003
2107 01461 6 00200
2108 01462 2 11011
2109 01463 5 11065
2110 01464 7 01472
2111 01465 0 37001
2112 01466 6 17001
2113 01467 6 00003
2114 01470 3 03003
2115 01471 0 37002
2116
2117 01472 6 11024
      01473 7 05777
2118 01474 6 17023
2119 01475 6 07021
2120 01476 0 37001
2121 01477 7 00761
2123
2124
2125
2126      001500
2127      001500
2128 01500 6 07021
2129 01501 6 17023
2130 01502 0 01037
2131 01503 6 11025
      01504 7 05700
2132 01505 6 07021
2133 01506 6 00001
2134 01507 2 03000
2135 01510 5 00115
2136 01511 6 01000
2137 01512 6 02000
2138 01513 6 11000
2139 01514 7 01455
2140
2141 01515 2 03100
2142 01516 5 00120
2143 01517 7 01442
2144
2145 01520 6 17023
2146 01521 6 00360
2147 01522 1 35500
2148
2149 01523 4 00131
2150 01524 0 02000
2151 01525 1 37002
2152 01526 0 10000
2153 01527 1 01001
2154 01530 7 01442
2155
2156
2157
2158
2159
2160 01531 6 17020
2161 01532 6 17017
2162 01533 6 17016
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175 01534 2 01011
2176 01535 4 11136
2177
2178 01536 7 01542
2179 01537 7 01572
2180 01540 7 01577

```

```

*
*
*
XMT   CALCBL0,IVR
XMT   -1,AUX
XOR   R2,AUX
ADD   RP,AUX
MOV   OVF,R11
XMT   CALCBHI,IVR
XMT   -1,AUX
XOR   R1,AUX
ADD   RB,AUX
ADD   R11,R11
LOG15021 XMT 11111100B,AUX
AND   R3,R3
XMT   01B,AUX
XOR   R3,R3
XMT   10000000B,AUX
AND   R11,R11
NZT   R11,LOG15025
JMP   LOG15030
LOG15025 MOV  RB,R1
XMT   CALCBL0,IVR
XMT   00000011B,AUX
XOR   R3,R3
MOV   RH,R2
*
LOG15030 CALL STORE
XMT   SAVER1,IVR
XMT   IVOSPD+IVISPD,IVL
MOV   RH,R1
JMP   LOGIC020
*
*
*
TIMERS NODE
*
*
DIFFERENCES IN TIME BASE HANDLED BY EXECUTION TABLE
*
LOG16000 EQU *
LOG17000 EQU *
LOG18000 XMT IVOSPD+IVISPD,IVL
XMT   SAVER1,IVR
MOV   R1,RP
CALL  REGVAL
XMT   IVOSPD+IVISPD,IVL
XMT   1,AUX
AND   R3,AUX
NZT   AUX,LOG16010
CLR   R1
XMT   0,R1
CLR   R2
XMT   0,R2
CLR   R11
XMT   0,R11
JMP   LOG15021
*
LOG16010 AND  R3(1),AUX
      NZT  AUX,LOG16020
      JMP  LOG15020
*
LOG16020 XMT  SAVER1,IVR
      XMT  -NODET100,AUX
      ADD  SAVENODE,AUX
      ORG  10,32
      XEC  LOG16TAB(AUX),3
      MOV  R2,AUX
      ADD  RB,R2
      MOV  OVF,AUX
      ADD  R1,R1
      JMP  LOG15020
*
*
EXECUTION TABLE FOR TIMER NODES
*
SELECT PROPER SCRATCH PAD LOCATION
*
CONTAINING TIME IN THE PROPER BASE
*
LOG16TAB XMT  TIMER100,IVR
      XMT  TIMER010,IVR
      XMT  TIMER001,IVR
***CONDITIONAL ASSEMBLY FOR CALCULATE NODES
*
CALCULATE NODES WILL ASSEMBLE ONLY IF THIS IS
*
THE ENHANCED INSTRUCTION SET, AS DEFINED BY 'ENHANCE'
*
IN THE GLOBAL MODULE.
      IF  ENHANCE
*
*
***CONVERT NODE
*
*
LOG19000 AND  R1,R11
      XEC  LOG19TAB(R11),4
*
LOG19TAB JMP  LOG19100
      JMP  LOG19200
      JMP  LOG19300

```

```

AUX= AUX + SPECIALCBHI]
R11= AUX + R11
*1
AUX=.NOT.COUNTLO
AUX= AUX + CALCBL0
R11= OVERFLOW LOW
*1
AUX=.NOT.COUNTHI
AUX= AUX + CALCBHI
R11= AUX + OVERFLOW LOW
SET POWER TO OFF
SET POWER TO COUNT.LT.PRESET
CHECK SIGN OF R11
IF COUNT.LT.PRESET THEN GOTO LOG15030
ELSE, SET COUNT.EQ.PRESET
GET LOW ORDER
SET POWER, R3(2)<-1
STORE NEW COUNT/TIME
SELECT SCRATCH PAD READ/WRITE
RESTORE 1ST BYTE OF NODE
SOLVE NEXT NODE

```

```

LOG16000 EQU *
LOG17000 EQU *
LOG18000 XMT IVOSPD+IVISPD,IVL
SELET SCRATCH PAD READ/WRITE
SAVE 1ST BYTE OF NODE
GET DESTINATION REG ADDR & DATA
SELECT SCRATCH PAD READ/WRITE
CHECK FOR RESET
IF .NOT.RESET THEN GOTO LOG 18010
ELSE, CLEAR TIME
SET COUNT.LT.PRESET FLAG
CHECK FOR ENABLE
IF ENABLED THEN GOTO LOG16020
ELSE, TIME REMAINS SAME, COMPARE PRSET
GET NODE TYPE
*1 MAKE IT RELATIVE: TIMERS TO 0
GET TIME IN PROPER BASE
*1
ADD TO ACCUMULATED TIME
COMPARE TIME VS. PRESET

```

```

NODE TYPE => R11
VECTOR TO CONVERT NODE
DISCRETE SOURCE NODE
REGISTER SOURCE NODE
BINARY => BCD, DISCRETE DESTINATION

```

```

2181 01541 7 01742      JMP      LOG19400      BCD => BINARY, REGISTER DESTINATION
2182
2183
2185
2186
2187 01542 6 07000      LOG19100 XMT      IVOCTRL+IVICRDAT,IVL SELECT CONTROL PULSE & COIL RAM READ
2188                      CLR      R5                      CLEAR ASSEMBLY REGISTERS
2188 01543 6 05000      +      XMT      0,R5
2189                      CLR      R6
2189 01544 6 06000      +      XMT      0,R6
2190 01545 6 00377      XMT      -1,AUX
2191 01546 3 02000      XOR      R2,AUX      CHECK FOR DUMMY REFERENCE
2192 01547 5 02151      NZT     R2,LOG19110
2193 01550 7 01563      JMP      LOG19130      IF DUMMY REFERENCE, GOTO LOG19130
2194 01551 6 02364      LOG19110 XMT      -12,R2      ELSE, SET COUNT
2195 01552 0 06000      LOG19120 MOV      R6,AUX      SHIFT BITS LEFT
2196 01553 1 06006      ADD     R6,R6      [R5,R6]<=[R5,R6].ROTATE LEFT.1
2197 01554 0 10000      MOV     0VF,AUX
2198 01555 1 05705      ADD     R5(7),R5
2199
2200 01556 0 25100      MOV     CRINPUT,AUX
2201 01557 1 06006      ADD     R6,R6      BRING IN NEXT BIT
2202
2203 01560 6 00001      XMT     1,AUX      COUNT DOWN
2204 01561 1 02002      ADD     R2,R2
2205 01562 5 02152      NZT     R2,LOG19120      LOOP UNTIL R2.EQ.0
2206
2207      LOG19130 WSP     CALCBLO,R6      STORE BCD VALUES
2207 01563 6 07021      +      XMT     IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
2207 01564 6 17001      +      XMT     CALCBLO,IVR      LOAD ADDRESS
2207 01565 0 06037      +      MOV     R6,RB      WRITE DATA
2208      WSP     CALCBHI,R5
2208 01566 6 07021      +      XMT     IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
2208 01567 6 17000      +      XMT     CALCBHI,IVR      LOAD ADDRESS
2208 01570 0 05037      +      MOV     R5,RB      WRITE DATA
2209 01571 7 00761      JMP     LOG1CD20      SOLVE NEXT NODE
2210
2211
2212
2213
2214      ***REGISTER SOURCE NODE
2215
2216 01572 6 00374      LOG19200 XMT     1111100B,AUX      SET REFERENCE TYPE TO
2217 01573 2 01001      AND     R1,R1      HOLDING REGISTER
2218 01574 6 00002      XMT     010B,AUX
2219 01575 3 01001      XOR     R1,R1
2220 01576 7 01365      JMP     LOG14000      COUNTER/TIMER PRESET CAN HANDLE REST
2221
2222
2224
2225
2226      ***BINARY TO BCD CONVERT, DISCRETE DESTINATION
2227
2228
2229 01577 6 00002      LOG19300 XMT     010B,AUX      CHECK FOR ENABLE
2230 01600 2 03000      AND     R3,AUX
2231 01601 5 00205      NZT     AUX,LOG19310      IF ENABLED, GOTO LOG19310
2232 01602 6 00374      XMT     1111100B,AUX      ELSE, CLEAR POWER
2233 01603 2 03003      AND     R3,R3
2234 01604 7 01736      JMP     LOG19399      EXIT
2235
2236 01605 6 00377      LOG19310 XMT     -1,AUX      CHECK FOR DUMMY REGISTERS
2237 01606 3 02000      XOR     R2,AUX
2238 01607 5 00211      NZT     AUX,LOG19320
2239 01610 7 01737      JMP     LOG19390      IF DUMMY REF, GOTO LOG19390
2240
2241 01611 6 07021      LOG19320 XMT     IVOSPD+IVISPD,IVL ELSE, SELECT SCRATCHPAD READ/WRITE
2242 01612 6 17025      XMT     SAVER3,IVR
2243 01613 0 03037      MOV     R3,RB      SAVE R3
2244                      CLR     R3      *1 CLEAR HI BCD VALUE
2244 01614 6 03000      +      XMT     0,R3
2245 01615 6 17026      XMT     SAVER4,IVR
2246 01616 0 04037      MOV     R4,RB      SAVE R4
2247                      CLR     R4      *1 CLEAR LO BCD VALUE
2247 01617 6 04000      +      XMT     0,R4
2248 01620 6 17023      XMT     SAVER1,IVR
2249 01621 0 01037      MOV     R1,RB      SAVE R1
2250 01622 6 01374      XMT     -4,R1      SET COUNT FOR LOOP A
2251 01623 6 17001      XMT     CALCBLO,IVR      GET BINARY VALUE
2252                      NOP
2252 01624 0 00000      +      MOV     AUX,AUX
2253 01625 0 37006      MOV     RB,R6      GET LO BINARY VALUE
2254 01626 6 17000      XMT     CALCBHI,IVR
2255                      NOP
2255 01627 0 00000      +      MOV     AUX,AUX
2256 01630 0 37005      MOV     RB,R5
2257
2258 01631 6 02374      LOG19330 XMT     -4,R2      START LOOP A
2259                      SET     COUNT FOR LOOP B
2260                      START LOOP B
2261
2261 01632 0 04000      LOG19340 MOV     R4,AUX      ROTATE BCD VALUE LEFT ONCE
2262 01633 1 04004      ADD     R4,R4
2263 01634 0 10000      MOV     0VF,AUX
2264 01635 1 03703      ADD     R3(7),R3      [R3,R4] <= [R3,R4].ROTATE LEFT.1
2265 01636 6 00340      XMT     NEG800LO,AUX      SUBTRACT 800 FROM BINARY VALUE
2266 01637 1 06011      ADD     R6,R11      [AUX,P11] <= [R5,R6] - 800

```


2267	01640	6	00374		XMT	NEG8DOHI,AUX	
2268	01641	1	10000		ADD	OVF,AUX	
2269	01642	1	05000		ADD	R5,AUX	
2270	01643	5	10245		NZT	OVF,LOG19345	
2271	01644	7	01651		JMP	LOG19350	IF [R5,R6].LT.800 GOTO LOG19350
2272				*			
2273	01645	0	00005	LOG19345	MOV	AUX,R5	ELSE, [R5,R6] <= [R5,R6] - 800
2274	01646	0	11006		MOV	R11,R6	
2275	01647	6	00001		XMT	1,AUX	
2276	01650	1	04006		ADD	R4,R4	SET BCD BIT, R4(0) <= 1
2277				*			
2278	01651	0	06000	LOG19350	MOV	R6,AUX	MULTIPLY BINARY VALUE BY 2
2279	01652	1	06006		ADD	R6,R6	
2280	01653	0	10000		MOV	OVF,AUX	[R5,R6] <= [R5,R6].ROTATE LEFT.1
2281	01654	1	05705		ADD	R5(7),R5	
2282	01655	6	00001		XMT	1,AUX	COUNT DOWN ON LOOP B
2283	01656	1	02002		ADD	R2,R2	
2284	01657	5	02232		NZT	R2,LOG19340	LOOP UNTIL R2.EQ.0
2285				*			
2287				*			
2288	01660	6	00017		XMT	U1111B,AUX	LOOP B FINISHED,
2289	01661	2	05011		AND	R5,R11	DIVIDE BINARY VALUE BY 16
2290	01662	0	05405		MOV	R5(4),R5	[R5,R6] <= [R5,R6].ROTATE RIGHT.4
2291	01663	2	06400		AND	R6(4),AUX	
2292	01664	3	11406		XOR	R11(4),R6	
2293				*			
2294	01665	6	00374		XMT	11111100B,AUX	MULTIPLY BIN VALUE BY 10
2295	01666	2	06611		AND	R6(6),R11	
2296	01667	2	05602		AND	R5(6),R2	
2297	01670	6	00003		XMT	011B,AUX	FIRST [R2,R11]<=[R5,R6].TIMES.4
2298	01671	2	06600		AND	R6(6),AUX	
2299	01672	3	02002		XOR	R2,R2	
2300				*			
2301	01673	0	11000		MOV	R11,AUX	SECOND,
2302	01674	1	06006		ADD	R6,R6	[R5,R6] <= [R5,R6] + [R2,R11] OR,
2303	01675	0	10000		MOV	OVF,AUX	[R5,R6] <= [R5,R6].TIMES.5
2304	01676	1	02000		ADD	R2,AUX	
2305	01677	1	05005		ADD	R5,R5	
2306				*			
2307	01700	0	06000		MOV	R6,AUX	THIRD, MULTIPLY ALL THAT BY 2
2308	01701	1	06006		ADD	R6,R6	[R5,R6] <= [R5,R6].ROTATE LEFT.1
2309	01702	0	10000		MOV	OVF,AUX	
2310	01703	1	05705		ADD	R5(7),R5	
2311				*			
2312	01704	6	00001		XMT	1,AUX	COUNT DOWN ON LOOP A
2313	01705	1	01001		ADD	R1,R1	
2314	01706	5	01231		NZT	R1,LOG19330	LOOP UNTIL R1.EQ.0
2315				*			LOOP A FINISHED
2317				*			
2318	01707	6	01364		XMT	-12,R1	SET COUNT
2319	01710	6	07002	LOG19360	XMT	IVOCRDAT,IVL	SELECT COIL RAM OUTPUT
2320	01711	6	00001		XMT	1,AUX	
2321	01712	2	03400		AND	R3(4),AUX	
2322	01713	0	00126		MOV	AUX,CROUTPUT	WRITE BCD BIT TO OUTPUT POINT
2323	01714	6	07000		XMT	IVOCTRL,IVL	SELECT CONTROL PULSE
2324	01715	6	27301		XMT	CTPLINCC,CTRLREG	INCREMENT COIL RAM ADDR
2325	01716	0	04000		MOV	R4,AUX	SHIFT NEXT BCD BIT INTO POSITION
2326	01717	0	10000		MOV	OVF,AUX	
2327	01720	1	03000		ADD	R3,AUX	[R3,R4] <= [R3,R4].ROTATE LEFT.1
2328	01721	1	03003		ADD	R3,R3	
2329	01722	6	00001		XMT	1,AUX	COUNT DOWN ON LOOP
2330	01723	1	01001		ADD	R1,R1	
2331	01724	5	01310		NZT	R1,LOG19360	LOOP UNTIL R1.EQ.0
2332				*			
2333	01725	6	17026		XMT	SAVER1,IVR	RETRIEVE R1
2334	01726	6	07021		XMT	IVOSPD+IVISPD,IVL	*1 SELECT SCRATCH PAD READ/WRITE
2335	01727	0	37001		MOV	RB,R1	
2336	01730	6	17025		XMT	SAVER3,IVR	RETRIEVE R3 (POWER)
2337	01731	6	00376		XMT	1111110B,AUX	*1 WITH BIT 0 <= 0
2338	01732	2	37003		AND	RB,R3	
2339	01733	6	17026		XMT	SAVER4,IVR	RETRIEVE R4 (ROW COUNT)
2340				*	NOP		*1
2340	01734	0	00000		MOV	AUX,AUX	
2341	01735	0	37004		MOV	RB,R4	
2342	01736	7	00761	LOG19399	JMP	LOGIC02D	SOLVE NEXT NODE
2343				*			
2344				*			
2345	01737	6	00376	LOG19390	XMT	1111110B,AUX	DUMMY REF, SET POWER
2346	01740	2	03003		AND	R3,R3	
2347	01741	7	01736		JMP	LOG19399	
2349				*			
2350				*			
2351							***BCD TO BINARY CONVERT, REGISTER DESTINATION
2352				*			
2353				*			
2354	01742	6	00002	LOG19400	XMT	010B,AUX	CHECK FOR ENABLE
2355	01743	2	03000		AND	R3,AUX	
2356	01744	5	00346		NZT	AUX,LOG19405	
2357	01745	7	02062		JMP	LOG19420	IF .NOT.ENABLED, GOTO LOG19420
2358				*			
2359				LOG19405	WSP	SAVER1,R1	SAVE NODE
2359	01746	6	07021		XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
2359	01747	6	17023		XMT	SAVER1,IVR	LOAD ADDRESS
2359	01750	0	01037		MOV	R1,R6	WRITE DATA
2360					WSP	SAVER2,R2	

2360	01751	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
2360	01752	6	17024	+	XMT	SAVER2,IVR	LOAD ADDRESS
2360	01753	0	02037	+	MOV	R2,RB	WRITE DATA
2361					WSP	SAVER3,R3	SAVE POWER
2361	01754	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
2361	01755	6	17025	+	XMT	SAVER3,IVR	LOAD ADDRESS
2361	01756	0	03037	+	MOV	R3,RB	WRITE DATA
2362					WSP	SAVER4,R4	SAVE ROW COUNT
2362	01757	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
2362	01760	6	17026	+	XMT	SAVER4,IVR	LOAD ADDRESS
2362	01761	0	04037	+	MOV	R4,RB	WRITE DATA
2363	01762	6	11375		XMT	-3,R11	SET COUNT
2364	01763	6	17000		XMT	CALCBHI,IVR	GET BCD SOURCE DATA
2365					CLR	R3	CLEAR BINARY VALUE
2365	01764	6	03000	+	XMT	0,R3	
2366	01765	0	37001		MOV	RB,R1	
2367	01766	6	17001		XMT	CALCBLO,IVR	
2368					CLR	R4	
2368	01767	6	04000	+	XMT	0,R4	
2369	01770	0	37002		MOV	RB,R2	
2370				*			
2372				*			
2373	01771	6	00374		LOG19410 XMT	1111100B,AUX	MULTIPLY BINARY VALUE BY 10
2374	01772	2	04606		AND	R4(6),R6	FIRST, [R5,R6] <= [R3,R4].TIMES.4
2375	01773	2	03605		AND	R3(6),R5	
2376	01774	6	00003		XMT	011H,AUX	
2377	01775	2	04600		AND	R4(6),AUX	
2378	01776	3	05005		XOR	R5,R5	
2379				*			
2380	01777	0	06000		MOV	R6,AUX	SECOND, ADD THAT TO BIN VALUE
2381	02000	1	04004		ADD	R4,R4	[R3,R4] <= [R3,R4] + [R5,R6]
2382	02001	0	10000		MOV	OVF,AUX	OR, [R3,R4] <= [R3,R4].TIMES.5
2383	02002	1	05000		ADD	R5,AUX	
2384	02003	1	03003		ADD	R3,R3	
2385				*			
2386	02004	0	04000		MOV	R4,AUX	THIRD. MULTIPLY ALL THAT BY 2
2387	02005	1	04004		ADD	R4,R4	[R3,R4]<=[R3,R4].TIMES.2
2388	02006	0	10000		MOV	OVF,AUX	
2389	02007	1	03703		ADD	R3(7),R3	
2390				*			
2391	02010	0	01000		MOV	R1,AUX	ADD BCD DIGIT TO BINARY VALUE
2392	02011	1	04004		ADD	R4,R4	
2393	02012	0	10000		MOV	OVF,AUX	
2394	02013	1	03003		ADD	R3,R3	
2395				*			
2396	02014	6	00017		XMT	01111B,AUX	MOVE NEXT BCD DIGIT INTO POSITION
2397	02015	2	02401		AND	R2(4),R1	
2398	02016	0	02402		MOV	R2(4),R2	
2399				*			
2400	02017	6	00001		XMT	1,AUX	COUNT DOWN ON LOOP
2401	02020	1	11011		ADD	R11,R11	
2402	02021	5	11371		NZT	R11,LOG19410	LOOP UNTIL R11.EQ.0
2404				*			
2405					WSP	CALCDHI,R3	SAVE BINARY VALUE
2405	02022	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
2405	02023	6	17004	+	XMT	CALCDHI,IVR	LOAD ADDRESS
2405	02024	0	03037	+	MOV	R3,RB	WRITE DATA
2406					WSP	CALCDLO,R4	
2406	02025	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
2406	02026	6	17005	+	XMT	CALCDLO,IVR	LOAD ADDRESS
2406	02027	0	04037	+	MOV	R4,RB	WRITE DATA
2407					NOP		*1 - WAIT
2407	02030	0	00000	+	MOV	AUX,AUX	
2408					RSP	SAVER1,R1	
2408	02031	6	17023	+	XMT	SAVER1,IVR	LOAD ADDRESS
2408	02032	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
2408	02033	0	37001	+	MOV	RB,R1	READ DATA
2409					RSP	SAVER2,R2	
2409	02034	6	17024	+	XMT	SAVER2,IVR	LOAD ADDRESS
2409	02035	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
2409	02036	0	37002	+	MOV	RB,R2	READ DATA
2410					RSP	SAVER3,R3	
2410	02037	6	17025	+	XMT	SAVER3,IVR	LOAD ADDRESS
2410	02040	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
2410	02041	0	37003	+	MOV	RB,R3	READ DATA
2411					RSP	SAVER4,R4	
2411	02042	6	17026	+	XMT	SAVER4,IVR	LOAD ADDRESS
2411	02043	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
2411	02044	0	37004	+	MOV	RB,R4	READ DATA
2412				*			
2413	02045	6	11026		CALL	PEGVAL	GET REG ADDR
2413	02046	7	05700				
2414				*			
2415					RSP	CALCDHI,R1	GET BINARY VALUE
2415	02047	6	17004	+	XMT	CALCDHI,IVR	LOAD ADDRESS
2415	02050	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
2415	02051	0	37001	+	MOV	RB,R1	READ DATA
2416					RSP	CALCDLO,R2	
2416	02052	6	17005	+	XMT	CALCDLO,IVR	LOAD ADDRESS
2416	02053	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
2416	02054	0	37002	+	MOV	RB,R2	READ DATA
2417				*			
2418	02055	6	11027		CALL	STORE	
2418	02056	7	05777				
2419				*			
2420					RSP	SAVER1,R1	GET FIRST BYTE OF NODE

```

2420 02057 6 17023 * XMT SAVER1,IVR LOAD ADDRESS
2420 02060 6 07021 * XMT IVOSPD+IVOSPD,IVL *1 - SELECT SPD READ
2420 02061 0 37001 * MOV RB,R1 READ DATA
2421 02062 6 00376 LOG19420 XMT 1111110B,AUX SET POWER
2422 02063 2 03003 AND R3,R3
2423 02064 7 00761 JMP LOGIC020 SOLVE NEXT NODE
2424 *
2425 *
2427 *
2428 * CALCULATE C-NODE CONSTANT
2429 *
2430 02065 6 07021 LOG20000 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
2431 02066 6 17002 XMT CALCCHI,IVR
2432 02067 2 01037 AND R1,RB STORE HIGH ORDER
2433 NOP *1
2433 02070 0 00000 + MOV AUX,AUX
2434 02071 6 17003 XMT CALCCL0,IVR
2435 02072 0 02037 MOV R2,RB
2436 02073 7 00761 JMP LOGIC020 SOLVE NEXT NODE
2438 *
2439 * CALCULATE C-NODE REGISTER
2440 *
2441 02074 6 07021 LOG21000 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
2442 02075 6 17023 XMT SAVER1,IVR
2443 02076 0 01037 MOV R1,RB SAVE 1ST BYTE OF NODE
2444 02077 6 11030 CALL REGVAL
2445 02101 6 17023 XMT SAVER1,IVR
2446 02102 6 07021 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD
2447 02103 0 37005 MOV RB,R5
2448 02104 6 17000 XMT CALCBHI,IVR
2449 02105 0 01037 MOV R1,RB SAVE HI ORDER VALUE
2450 02106 0 05001 MOV R5,R1 RESTORE 1ST BYTE OF N
2451 02107 6 17001 XMT CALCBLO,IVR
2452 02110 0 02037 MOV R2,RB SAVE LO ORDER VALUE
2453 02111 7 00761 JMP LOGIC020 SOLVE NEXT NODE
2455 *
2456 * CALCULATE
2457 * ENTER HERE FOR ALL CLCULATE NODES
2458 * IF ENABLED, REGISTERS R1,R2,R3, & R4 WILL BE SAVED
2459 * THEN REGISTERS R3,R4 WILL BE GIVEN THE C-NODE VALUES
2460 * (ADDEND, SUBTRAMEND, MULTIPLIER, OR DIVISOR)
2461 * REGISTERS R1,R2 ARE GIVEN THE B-NODE VALUE
2462 * (AUGEND, MINUEND, MULTIPLICAND, OR DIVIDEND)
2463 * THE PROPER CALCULATION IS EFFECTED USING A EXECUTION TABLE,
2464 * VECTORING OFF THE NODE TYPE
2465 *
2466 * THE CALCULATE ROUTINE LEAVES THE RESULT (THE SUM, THE
2467 * ABSOLUTE VALUE OF THE DIFFERENCE, THE PRODUCT, OR THE QUOTIENT)
2468 * IN REGISTERS R5,R6. AND LEAVES THE POWER (FOR R3(2-0))
2469 * IN REGISTER R11(2-0)
2470 *
2471 * THE RESULT IS STORED IN SCRATCH PAD PRESTHI, CALCBLO AND
2472 * IN THE D NODE REGISTER. POWER (R3) IS UPDATED. REGISTERS
2473 * (R1, UPDATED R3, AND R4) ARE RESTORED, AND EXIT IS MADE
2474 * TO SOLVE NEXT NODE.
2475 *
2476 -02112 6 00144 LOG22000 XMT 0000100,AUX TEST IF NODE IS ENABLED
2477 02113 2 03000 AND R3,AUX
2478 02114 5 00120 NZT AUX,LOG22010 IF NODE ENABLED THEN GOTO LOG22010
2479 02115 6 00370 XMT 11111000B,AUX ELSE, CLEAR POWER
2480 02116 2 03003 AND R3,R3
2481 02117 7 00761 JMP LOGIC020 SOLVE NEXT NODE
2482 *
2483 02120 6 07021 LOG22010 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
2484 02121 6 17023 XMT SAVER1,IVR SAVE FIRST BYTE OF NODE
2485 02122 0 01037 MOV R1,RB
2486 NOP *1
2486 02123 0 00000 + MOV AUX,AUX
2487 02124 6 17024 XMT SAVER2,IVR SAVE SECOND BYTE OF NODE
2488 02125 0 02037 MOV R2,RB
2489 NOP *1
2489 02126 0 00000 + MOV AUX,AUX
2490 02127 6 17025 XMT SAVER3,IVR SAVE POWER BITS
2491 02130 0 03037 MOV R3,RB
2492 NOP *1
2492 02131 0 00000 + MOV AUX,AUX
2493 02132 6 17026 XMT SAVER4,IVR SAVE ROW COUNT
2494 02133 0 04037 MOV R4,RB
2495 NOP *1
2495 02134 0 00000 + MOV AUX,AUX
2496 02135 6 17001 XMT CALCBLO,IVR GET LOW ORDER B-NODE VALUE -> R4
2497 02136 6 00003 XMT CALCTYPM,AUX *1 PREPARE TO EXTRACT NODE TYPE
2498 02137 0 37002 MOV RB,R2
2499 02140 6 17000 XMT CALCBHI,IVR GET HIGH ORDER B-NODE VALUE -> R3
2500 02141 2 01711 AND R1(7),R11 *1 NODE TYPE -> R11
2501 02142 0 37001 MOV RB,R1
2502 02143 6 17003 XMT CALCCL0,IVR GET LOW ORDER C-NODE VALUE -> R2
2503 NOP *1
2503 02144 0 00000 + MOV AUX,AUX
2504 02145 0 37004 MOV RB,R4
2505 02146 6 17002 XMT CALCCHI,IVR GET HI ORDER C-NODE VALUE -> R1
2506 NOP *1
2506 02147 0 00000 + MOV AUX,AUX
2507 02150 0 37003 MOV RB,R3
2508 02151 4 11152 XEC LOG22TAB(R11),4 VECTOR TO DO CALCULATE

```



```

2509
2510
2511
2512
2513
2514 02152 7 02217 LOG22TAB JMP LOG22100 ADDITION NODE
2515 02153 7 02241 JMP LOG22200 SUBTRACTION NODE
2516 02154 7 02277 JMP LOG22300 MULTIPLICATION NODE
2517 02155 7 02450 JMP LOG22400 DIVISION NODE
2518
2519
2520
2521 02156 6 07021 LOG22020 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
2522 02157 6 17004 XMT CALCDHI,IVR STORE RESULTS
2523 02160 0 05037 MOV R5,RB
2524 NOP *1
2524 02161 0 00000 + MOV AUX,AUX
2525 02162 6 17005 XMT CALCDLO,IVR
2526 02163 0 06037 MOV R6,RB STORE LOW ORDER RESULTS
2527 02164 6 00370 XMT 11111000B,AUX *1
2528 02165 6 17025 XMT SAVER3,IVR RETRIEVE POWER BITS
2529 NOP
2529 02166 0 00000 + MOV AUX,AUX
2530 02167 2 37000 AND RB,AUX POWER MASKED. -> AUX
2531 02170 6 17026 XMT SAVER4,IVR RETRIEVE ROW COUNT
2532 02171 3 11003 XOR R11,R3 OR IN NEW POWER BITS
2533 02172 0 37004 MOV RB,R4
2534 02173 6 17023 XMT SAVER1,IVR RETRIEVE 1ST BYTE OF NODE
2535 NOP *1
2535 02174 0 00000 + MOV AUX,AUX
2536 02175 0 37001 MOV RB,R1
2537 02176 6 17024 XMT SAVER2,IVR RETRIEVE 2ND BYTE OF NODE
2538 NOP *1
2538 02177 0 00000 + MOV AUX,AUX
2539 02200 0 37002 MOV RB,R2
2540 02201 6 11031 CALL REGVAL GET D-NODE REGISTER ADDR
2540 02202 7 05700
2541 02203 6 17004 XMT CALCDHI,IVR GET RESULT VALUES
2542 02204 6 07021 XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
2543 02205 0 37001 MOV RB,R1
2544 02206 6 17005 XMT CALCDLO,IVR GET LOW ORDER RESULT
2545 NOP
2545 02207 0 00000 + MOV AUX,AUX
2546 02210 0 37002 MOV RB,R2
2547 02211 6 11032 CALL STORE
2547 02212 7 05777
2548 02213 6 17023 XMT SAVER1,IVR GET FIRST BYTE OF NODE
2549 02214 6 07021 XMT IVOSPD+IVISPD,IVL
2550 02215 0 37001 MOV RB,R1
2551 02216 7 00761 JMP LOG1C020 SOLVE NEXT NODE
2552
2554
2555
2556
2557
2558
2559
2560 02217 0 04000 LOG22100 MOV R4,AUX ADD LOW ORDER
2561 02220 1 02006 ADD R2,R6 R4+R2 -> R6
2562 02221 0 10000 MOV OVFL,AUX GET OVERFLOW
2563 02222 1 03000 ADD R3,AUX ADD HI ORDER
2564 02223 1 01005 ADD R1,R5 OVFL+R3+R1 -> R5
2565
2566 02224 6 00030 LOG22110 XMT NEG1000L,AUX TEST FOR OVERFLOW
2567 02225 1 06002 ADD R6,R2 [R5,R6]-1000->[R1,R2]
2568 02226 0 10000 MOV OVFL,AUX
2569 02227 6 01374 XMT NEG1000H,R1
2570 02230 1 01000 ADD R1,AUX
2571 02231 1 05001 ADD R5,R1
2572 02232 6 00200 XMT 10000000B,AUX TEST [R1,R2] FOR NEGATIVITY
2573 02233 2 01000 AND R1,AUX
2574 02234 5 00240 NZT AUX,LOG22120 IF R1.LT.0 THEN NO OVERFLOW, EXIT
2575
2576 02235 6 11004 XMT 00001000B,R11 ELSE, OVERFLOW OCCURED, SET POWER
2577 02236 0 01005 MOV R1,R5 MAKE SUM MODULO 1000
2578 02237 0 02006 MOV R2,R6
2579 02240 7 02156 LOG22120 JMP LOG22020 EXIT
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591 02241 6 00377 LOG22200 XMT -1,AUX MFGATE SUBTRAHEND
2592 02242 3 03003 XOR R3,R3 [R3,R4]=-[R3,R4]
2593 02243 3 04004 XOR R4,R4
2594 02244 6 00001 XMT 1,AUX
2595 02245 1 04004 ADD R4,R4
2596 02246 0 10000 MOV OVFL,AUX
2597 02247 1 03003 ADD R3,R3

```


2598			*			
2599	02250	0	04000	MOV	R4,AUX	ADD NEGATED SUBTRAHEND TO MINUFND
2600	02251	1	02006	ADD	R2,R6	[R5,R6]=[R3,R4]+[R1,R2]
2601	02252	0	10000	MOV	OVF,AUX	
2602	02253	1	03000	ADD	R3,AUX	
2603	02254	1	01005	ADD	R1,R5	
2604	02255	6	00200	XMT	10000000B,AUX	TEST FOR NEGATIVE DIFFERENCE
2605	02256	2	05000	AND	R5,AUX	
2606	02257	5	00266	NZT	AUX,LOG22220	IF DIFF.LT.0 THEN GOTO LOG22220
2607	02260	5	05264	NZT	R5,LOG22210	ELSE, CHECK IF DIFF.EQ.0
2608	02261	5	06264	NZT	R6,LOG22210	IF DIFF.GT.0 THEN GOTO LOG22210
2609	02262	6	11002	XMT	0000010B,R11	ELSE, DIFF.EQ.0, SET POWER
2610	02263	7	02156	JMP	LOG22020	EXIT
2611	02264	6	11004	LOG22210 XMT	00000100B,R11	DIFF.GT.0, SET POWER
2612	02265	7	02156	JMP	LOG22020	EXIT
2613	02266	6	11001	LOG22220 XMT	00000001B,R11	DIFF.LT.0, SET POWER
2614	02267	6	00377	XMT	-1,AUX	MAKE DIFFERENCE ABSOLUTE VALUE
2615	02270	3	05005	XOR	R5,R5	[R5,R6]=-[R5,R6]
2616	02271	3	06006	XOR	R6,R6	
2617	02272	6	00001	XMT	1,AUX	
2618	02273	1	06006	ADD	R6,R6	
2619	02274	0	10000	MOV	OVF,AUX	
2620	02275	1	05005	ADD	R5,R5	
2621	02276	7	02156	JMP	LOG22020	EXIT
2623			*			
2624			***MULTIPLY NODE			
2625			*			
2626			*		[R1,R2] CONTAINS MULTIPLICAND	
2627			*		[R3,R4] CONTAINS MULTIPLIER	
2628			*			
2629			*		MULTIPLY, STORE LOW ORDER PRODUCT	
2630			*		EXIT TO CALCULATE MAIN WITH:	
2631			*		HI PRODUCT IN [R5,R6]	
2632			*		POWER R11 (2) = 1	
2633			*			
2634	02277	6	07021	LOG22300 XMT	IVOSPD+IVISPD,IVL	SELECT SCRATCH PAD READ/WRITE
2635	02300	6	17000	XMT	CALCBHI,IVR	
2636			*	CLR	R11	CLEAR MULTIPLICAND EXTENTION
2636	02301	6	11000	XMT	0,R11	
2637	02302	0	11037	MOV	R11,RB	
2638	02303	6	00001	XMT	1,AUX	
2639	02304	6	17010	XMT	CALCNT,IVR	
2640	02305	0	00037	MOV	AUX,RB	INIT MASK
2641			*	CLR	R5	CLEAR PRODUCT
2641	02306	6	05000	XMT	0,R5	
2642			*	CLR	R6	
2642	02307	6	06000	XMT	0,R6	
2643			*			
2644	02310	0	37000	LOG22310 MOV	RB,AUX	COMPARE LOW MULTIPLIER
2645	02311	2	04000	LOG22315 AND	R4,AUX	AGAINST MASK
2646	02312	5	00314	NZT	AUX,LOG22320	IF NO MATCH
2647	02313	7	02331	JMP	LOG22330	GOTO LOG22330
2648			*			
2649	02314	6	17000	LOG22320 XMT	CALCBHI,IVR	ELSE
2650	02315	0	02000	MOV	R2,AUX	ADD MULTIPLICAND TO PRODUCT
2651	02316	1	06006	ADD	R6,R6	
2652	02317	0	10000	MOV	OVF,AUX	
2653	02320	1	05005	ADD	R5,R5	
2654	02321	0	10000	MOV	OVF,AUX	
2655	02322	1	11011	ADD	R11,R11	
2656	02323	0	01000	MOV	R1,AUX	
2657	02324	1	05005	ADD	R5,R5	
2658	02325	0	10000	MOV	OVF,AUX	
2659	02326	1	37000	ADD	RB,AUX	
2660	02327	6	17010	XMT	CALCNT,IVR	
2661	02330	1	11011	ADD	R11,R11	
2662			*			
2663	02331	0	37000	LOG22330 MOV	RB,AUX	GETMASK
2664	02332	2	03000	AND	R3,AUX	COMPARE TO HI MULTIPLIER
2665	02333	5	00335	NZT	AUX,LOG22340	IF NO MATCH
2666	02334	7	02342	JMP	LOG22350	THEN GOTO LOG22350
2667			*			
2668	02335	0	02000	LOG22340 MOV	R2,AUX	ELSE, ADD LOW MULTIPLICAND
2669	02336	1	09005	ADD	R5,R5	TO HI PRODUCT
2670	02337	0	10000	MOV	OVF,AUX	
2671	02340	1	01000	ADD	R1,AUX	
2672	02341	1	11011	ADD	R11,R11	
2673			*			
2674	02342	6	17000	LOG22350 XMT	CALCBHI,IVR	MULTIPLY MULTIPLICAND BY 2
2675	02343	6	00177	XMT	01111111B,AUX	
2676	02344	2	37000	AND	RB,AUX	
2677	02345	0	00700	MOV	AUX(7),AUX	ROTATE HI ORDER
2678	02346	0	01000	MOV	R1,AUX	
2679	02347	1	01001	ADD	R1,R1	ROTATE MIDDLE ORDER
2680	02350	0	10000	MOV	OVF,AUX	
2681	02351	1	37037	ADD	RB,RB	MIDDLE OVERFLOW => HI
2682	02352	0	02000	MOV	R2,AUX	
2683	02353	1	02002	ADD	R2,R2	ROTATE LOW ORDER
2684	02354	0	10000	MOV	OVF,AUX	
2685	02355	6	17010	XMT	CALCNT,IVR	
2686	02356	1	01001	ADD	R1,R1	LOW OVERFLOW => MIDDLE
2687	02357	6	00200	XMT	10000000B,AUX	CHECK IF LOOP COMPLETED
2688	02360	2	37000	AND	RB,AUX	
2689	02361	5	00366	NZT	AUX,LOG22360	IF DONE LOOP, GOTO LOG22360

2690	02362	0	37000		MOV	RB,AUX	ELSE, ROTATE MASK
2691	02363	0	00700		MOV	AUX(7),AUX	
2692	02364	0	00037		MOV	AUX,RB	
2693	02365	7	02311		JMP	LOG22315	LOOP
2694				*			
2695	02366	0	05001	LOG22360	MOV	R5,R1	MULTIPLICATION DONE
2696	02367	0	06002		MOV	R6,R2	SET UP DIVIDE ROUTINE TO
2697	02370	0	11006		MOV	R11,R6	SPLIT HI PRODUCT FROM LO PRODUCT
2698	02371	6	03003		XMT	K1000HI,R3	
2699	02372	6	03350		XMT	K1000LO,R3	
2700				*			
2701	02373	6	11033		CALL	DIVIDE	
	02374	7	06605				
2702				*			
2703				*			
2704				*			
2705				*			
2706				*			
2707				*			
2708	02375	6	17005		XMT	CALCDLO,IVR	STORE HI PRODUCT
2709	02376	0	02037		MOV	R2,RB	
2710	02377	6	00003		XMT	011H,AUX	*1
2711	02400	6	17004		XMT	CALCDHI,IVR	
2712	02401	2	01037		AND	R1,RB	
2713	02402	2	06205		AND	R6(2),R5	LO PRODUCT => [R5,R6]
2714	02403	2	06006		AND	R6,R6	
2715	02404	6	00077		XMT	00111111B,AUX	
2716	02405	2	01200		AND	R1(2),AUX	
2717	02406	6	17000		XMT	CALCBHI,IVR	
2718	02407	0	05037		MOV	R5,RB	STORE LO PROD IN [CALCBHI,CALCBLO]
2719	02410	6	17001		XMT	CALCBLO,IVR	
2720	02411	3	06206		XOR	R6(2),R6	
2721	02412	0	06037		MOV	R6,RB	
2722	02413	6	00001		XMT	1,AUX	
2723	02414	6	17024		XMT	SAVER2,IVR	GET 2ND BYTE OF NODE
2724					NOP		
2724	02415	0	00000	+	MOV	AUX,AUX	
2725	02416	1	37002		ADD	RB,R2	INCREMENT REFERENCE
2726	02417	5	02023		NZT	R2,LOG22370	SCREEN OUT DUMMY REG REF
2727	02420	6	02377		XMT	-1,R2	DUMMY REF REF. R2 <= -1
2728	02421	6	01001		XMT	01B,R1	R1 <= HOLDING REG TYPE
2729	02422	7	02426		JMP	LOG22375	
2730	02423	6	17023	LOG22370	XMT	SAVER1,IVR	GET 1ST BYTE OF NODE
2731					NOP		
2731	02424	0	00000	+	MOV	AUX,AUX	
2732	02425	0	37001		MOV	RB,R1	
2733				*			
2734	02426	6	11034	LOG22375	CALL	REGVAL	
	02427	7	05700				
2735				*			
2736					RSP	CALCBHI,R1	LOW PRODUCT => [R1,R2]
2736	02430	6	17000	+	XMT	CALCBHI,IVR	LOAD ADDRESS
2736	02431	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
2736	02432	0	37001	+	MOV	RB,R1	READ DATA
2737					RSP	CALCBLO,R2	
2737	02433	6	17001	+	XMT	CALCBLO,IVR	LOAD ADDRESS
2737	02434	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
2737	02435	0	37002	+	MOV	RB,R2	READ DATA
2738	02436	6	11035		CALL	STORE	
	02437	7	05777				
2739				*			
2740					RSP	CALCDHI,R5	GET HI PRODUCT
2740	02440	6	17004	+	XMT	CALCDHI,IVR	LOAD ADDRESS
2740	02441	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
2740	02442	0	37005	+	MOV	RB,R5	READ DATA
2741					RSP	CALCDLO,R6	
2741	02443	6	17005	+	XMT	CALCDLO,IVR	LOAD ADDRESS
2741	02444	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
2741	02445	0	37006	+	MOV	RB,R6	READ DATA
2742	02446	6	11004		XMT	100B,R11	SET POWER
2743	02447	7	02156		JMP	LOG22020	CLOSE OUT NODE
2745				*			
2746				*			
2747				*			
2748				*			
2749				*			
2750				*			
2751				*			
2752				*			
2753				*			
2754				*			
2755				*			
2756				*			
2757				*			
2758				*			
2759				*			
2760				*			
2761				*			
2762				*			
2763				*			
2764				*			
2765				*			
2766	02450	5	03056	LOG22400	NZT	R3,LOG22405	CHECK FOR DIVISOR = 0
2767	02451	5	04056		NZT	R4,LOG22405	IF DIVR.NE.0 GOTO LOG22405
2768					CLR	R5	ELSE, CLEAR QUOTIENT


```

2768 02452 6 05000 + XMT D,R5
2769 CLR R6
2769 02453 6 06000 + XMT D,R6
2770 02454 6 11001 XMT D01B,R11 SET POWER
2771 02455 7 02156 JMP LOG22020 EXIT TO CLOSE OUT NODE
2772 *
2773 02456 6 17006 LOG22405 XMT CALBADRH,IVR CHECK FOR DOUBLE PRECISION DIVIDEND
2774 02457 6 07021 XMT IVOSPD+IVOSPD,IVL SELECT SCRATCH PAD READ/WRITE
2775 02460 6 00377 XMT -1,AUX
2776 02461 3 37006 XOR R8,R6
2777 02462 5 06064 NZT R6,LOG22410
2778 02463 7 02610 JMP LOG22460 IF DIVD IS DOUBLE PRECISION GOTO LOG22460
2779 *
2780 02464 3 03005 LOG22410 XOR R3,R5 ELSE, CHECK FOR DIVIDEND OVERFLOW
2781 02465 3 04006 XOR R4,R6 [R5,R6]<= -[R3,R4]
2782 02466 6 00001 XMT 1,AUX
2783 02467 1 06006 ADD R6,R6
2784 02470 0 10000 MOV OVF,AUX
2785 02471 1 05005 ADD R5,R5
2786 02472 0 06000 MOV R6,AUX
2787 02473 1 02006 ADD R2,R6
2788 02474 0 10000 MOV OVF,AUX
2789 02475 1 01000 ADD R1,AUX
2790 02476 1 05005 ADD R5,R5
2791 02477 5 10102 NZT OVF,LOG22415 IF DIVDHI.GE.DIVR GOTO LOG22415
2792 02500 6 11375 XMT -3,R11 SET COUNT FOR LOOP TO
2793 02501 7 02506 JMP LOG22420 MULTIPLY DIVDHI BY 1000
2795 *
2796 * HI DIVIDEND.GE.DIVISOR
2797 * CLEAR QUOTIENT, SET POWER, EXIT
2798 *
2799 LOG22415 CLR R5
2799 02502 6 05000 + XMT D,R5
2800 CLR R5
2800 02503 6 05000 + XMT D,R5
2801 02504 6 11002 XMT D10B,R11
2802 02505 7 02156 JMP LOG22020 EXIT TO CLOSE OUT NODE
2803 *
2804 *
2805 * LOOP TO MULTIPLY [R1,R2] BY 1000
2806 * RESULT => [R11,R1,R2]
2807 * COUNT FOR LOOP IS IN R11
2808 *
2809 02506 6 00374 LOG22420 XMT 1111100B,AUX MASK => AUX
2810 02507 2 01605 AND R1(6),R5
2811 02510 2 02606 AND R2(6),R6 [R5,R6]<=[R1,R2].TIMES.4
2812 02511 6 00003 XMT D11B,AUX
2813 02512 2 02600 AND R2(6),AUX
2814 02513 3 05005 XOR R5,R5
2815 *
2816 02514 6 00003 XMT D11B,AUX CHECK FOR OVERFLOW FROM R1
2817 02515 2 01600 AND R1(6),AUX
2818 02516 5 00131 NZT AUX,LOG22430 LOG22430 IF R1 OVERFLOW.TRUE. END LOOP
2819 *
2820 02517 0 06000 MOV R6,AUX [R1,R2]<=[R1,R2].TIMES.5
2821 02520 1 02002 ADD R2,R2 ([R1,R2]<=[R1,R2]+[R5,R6])
2822 02521 0 10000 MOV OVF,AUX
2823 02522 1 01000 ADD R1,AUX
2824 02523 1 05001 ADD R5,R1 CHECK FOR OVERFLOW
2825 02524 5 10142 NZT OVF,LOG22440 IF OVERFLOW.TRUE. GOTO LOG22440
2826 02525 6 00001 XMT 1,AUX ELSE, COUNT DOWN ON LOOP
2827 02526 1 11011 ADD R11,R11
2828 02527 5 11106 NZT R11,LOG22420 LOOP UNTIL R11 = 0
2829 02530 7 02543 JMP LOG22450
2830 *
2831 *
2832 *
2833 02531 0 00011 LOG22430 MOV AUX,R11 CLOSE OUT LOOP
2834 02532 0 06000 MOV R6,AUX
2835 02533 1 02002 ADD R2,R2 ADD 100.TIMES.ORIGINAL [R1,R2] TO
2836 02534 0 10000 MOV OVF,AUX 25 TIMES ORIGINAL [R1,R2]
2837 02535 1 01000 ADD R1,AUX
2838 02536 1 05001 ADD R5,R1
2839 02537 0 10000 MOV OVF,AUX ADD OVERFLOW TO R11
2840 02540 1 11011 ADD R11,R11
2841 02541 7 02543 JMP LOG22450
2842 *
2843 02542 0 10011 LOG22440 MOV OVF,R11 OVERFLOW => R11
2844 *
2845 * [R11,R1,R2]<= 125.TIMES.ORIGINAL [R1,R2]
2846 * MULTIPLY THAT BY 8 TO GIVE 1000.TIMES.ORIGINAL [R1,R2]
2847 *
2848 *
2849 02543 0 11511 LOG22450 MOV R1(5),R11 R11.TIMES.8 => R11
2850 02544 6 00007 XMT D111B,AUX
2851 02545 2 02506 AND R2(5),R6 R2 CARRYOUT => R6
2852 02546 2 01500 AND R1(5),AUX R1 CARRYOUT => AUX
2853 02547 3 11011 XOR R11,R11 R11.OR.AUX(R1 CARRYOUT) => R11
2854 02550 6 00370 XMT 11111000B,AUX
2855 02551 2 02502 AND R2(5),R2 R2.TIMES.8 => R2
2856 02552 2 01500 AND R1(5),AUX R1.TIMES.8 => R1
2857 02553 3 06001 XOR R6,R1 OR IN R2 CARRYOUT => R1
2858 *
2859 WSP DIVDX1KH,R11 SAVE HI DIVIDEND, HI PART
2859 02554 6 07021 + XMT IVOSPD+IVOSPD,IVL SELECT SPD READ/WRITE
2859 02555 6 17011 + XMT DIVDX1KH,IVR LOAD ADDRESS
2859 02556 0 11037 + MOV R11,RB WRITE DATA

```



```

2860          WSP      DIVDX1KM,R1      MIDDLE PART
2860 02557 6 07021 +      XMT      IVISPD+IVOSPD,IVL  SELECT SPD READ/WRITE
2860 02560 6 17012 +      XMT      DIVDX1KM,IVR      LOAD ADDRESS
2860 02561 0 01037 +      MOV      R1,RB              WRITE DATA
2861          WSP      DIVDX1KL,R2      LOW PART
2861 02562 6 07021 +      XMT      IVISPD+IVOSPD,IVL  SELECT SPD READ/WRITE
2861 02563 6 17013 +      XMT      DIVDX1KL,IVR      LOAD ADDRESS
2861 02564 0 02037 +      MOV      R2,RB              WRITE DATA
2862          *
2863 02565 6 17006          XMT      CALBADRH,IVR
2864 02566 6 07021          XMT      IVOSPD+IVISPD,IVL  SELECT SCRATCH PAD READ/WRITE
2865 02567 0 37001          MOV      RB,R1              GET B NODE DATA
2866 02570 6 17007          XMT      CALBADRL,IVR
2867 02571 6 00001          XMT      1,AUX              *1
2868 02572 1 37002          ADD      RB,R2              STEP REF # TO POINT AT NEXT REG
2869          *
2870 02573 6 11036          CALL     REGVAL
2870 02574 7 05700
2871          *
2873          *
2874          *
2875 02575 6 17013          XMT      DIVDX1KL,IVR
2876 02576 6 07021          XMT      IVOSPD+IVISPD,IVL  *1 SELECT SCRATCH PAD READ/WRITE
2877 02577 0 37000          MOV      RB,AUX              GET HI DIVIDEND
2878 02600 1 02002          ADD      R2,R2              ADD TO LOW DIVIDEND
2879 02601 0 10000          MOV      OVF,AUX
2880 02602 6 17012          XMT      DIVDX1KM,IVR
2881 02603 1 01000          ADD      R1,AUX              *1
2882 02604 1 37001          ADD      RB,R1
2883 02605 6 17011          XMT      DIVDX1KM,IVR
2884 02606 0 10000          MOV      OVF,AUX              *1
2885 02607 1 37006          ADD      RB,R6
2886          *
2887 02610 6 11037          LOG22460 CALL     DIVIDE          DIVIDE [R6,R1,R2] BY [R3,R4]
2887 02611 7 06605
2888          *
2889          *      RETURN WITH QUOTIENT IN [R1(1-0),R2]
2890          *
2891 02612 6 00003          XMT      011B,AUX
2892 02613 2 01005          AND      R1,R5              MOV QUOTIENT TO [R5,R6]
2893 02614 0 02006          MOV      R2,R6
2894 02615 6 11004          XMT      0100B,R11         SET POWER
2895 02616 7 02156          JMP      LOG22020          CLOSE OUT NODE
2896          ENDF
2897          ***END OF CONDITIONAL ASSEMBLY AREA FOR ENHANCED INST SET.
2898          *      IF THIS AREA DOES NOT ASSEMBLE, THE NODE VECTORS
2899          *      WILL BE SENT TO THE UNDEFINED NODE TYPE AREA
2900          *      WHICH IS ALSO CONTROLLED BY CONDITIONAL ASSEMBLY
2901          *
2902          *
2903          *      ***NULL NODE
2904          *
2905 02617 6 00001          LOG23000 XMT      1,AUX          ADD 1 TO ROW COUNT
2906 02620 1 04004          ADD      R4,R4              TO KEEP IT FROM DECREMENTING
2907 02621 7 00761          JMP      LOGIC020          SOLVE NEXT NODE
2908          *
2909          *
2910          *      ***UNDEFINED NODE TYPES
2911          *
2912          *      ***SET SYSTEM ERROR STATE AND EXIT TO EXEC TO CHANGE STATES
2913          *
2914          *      ***CONDITIONAL ASSEMBLY FOR NODE TYPES WHICH ARE
2915          *      ALLOWED IN THE ENHANCED SET BUT NOT IN THE BASIC SET
2916          *      IF      ENHANCE-1
2917          *      ENDF
2918          LOG24000 EQU      *
2919          LOG25000 EQU      *
2920          LOG26000 EQU      *
2921          LOG27000 EQU      *
2922          LOG28000 EQU      *
2923          LOG29000 EQU      *
2924          LOG30000 EQU      *
2925 02622 6 01003          LOG31000 XMT      SYSENODE,R1
2926          *
2927          *
2928          *      ***COMMON EXIT FROM LOGIC MODULE
2929          *
2930 02623 5 01225          LOGICX  NZT      R1,LOGICX10    R1.NE.0 => CHANGE STATES
2931 02624 7 02626          JMP      LOGICX20          R1.EQ.0 => GOOD EXIT
2932          *
2933 02625 7 00453          LOGICX10 JMP      EXEC          EXIT TO EXEC
2934          *
2935          LOGICX20 EQU      *      CONTINUE
2936          EXEC030 EQU      *
2937          *
2938          *
2939          *      ***COMMAND HANDLER FOR PERIPHERAL PORT
2940          *
2941          *
2942          *      CMDS000 CLR      R1          R1 <- ERROR EXIT CODE
2943          *      XMT      0,R1
2944          *      XMT      SAVSTATE,IVR      LOAD ADDRESS
2945          *      XMT      IVISPD+IVOSPD,IVL  SELECT PORTS
2946          *      MOV      R1,RB              LOAD ERROR EXIT CODE
2947          *      NOP
2948          *      MOV      AUX,AUX          *1 - WAIT
2949          *      XMT      CMDCONT,IVR      LOAD ADDRESS
2950          *      NOP
2951          *      MOV      AUX,AUX          *1 - EXECUTE VIA CONTINUATION CODE
2952          *      XEC      CMDSTAB1(RB),4

```

2950									
2951	02636	7	02643	CMDSTAB1	JMP	CMDS010		NO FUNCTION ACTIVE	
2952	02637	7	03522		JMP	CMDO3075		CONTINUE SEARCH	
2953	02640	7	04112		JMP	CMDO5900		CONTINUE INSERT	
2954	02641	7	04334		JMP	CMDO6400		CONTINUE DELETE	
2955	02642	7	03606		JMP	CMDO4010		CONTINUE POWER	
2957				*					
2958				***NO FUNCTIONS ACTIVE					
2959				*					
2960	02643	6	17074	CMDS010	XMT	MSGCOUNT,IVR		LOAD ADDRESS	
2961	02644	6	07021		XMT	IVISPD+IVOSPD,IVL		SELECT SPD READ AND WRITE	
2962	02645	6	00377		XMT	-1,AUX		AUX <- DECREMENT	
2963	02646	5	37010		NZT	RB,CMDS020		BRANCH OF COMMANDS PENDING	
2964	02647	7	04732		JMP	CMDSX		NO MESSAGES, GO TO EXIT	
2965				*					
2966	02650	1	37037	CMDS020	ADD	RB,RB		DECREMENT COMMAND COUNT	
2967	02651	6	02075		XMT	RCVRBLK,R2		R2 <- BUFFER BLOCK ADDR	
2968	02652	6	11040		CALL	UBFCH		GET COMMAND BYTE	
	02653	7	05565						
2969	02654	5	01256		NZT	R1,CMDS025		IF R1 = 0, SEND A NAK MESSAGE	
2970	02655	7	04651		JMP	CMDNAK00			
2971	02656	6	17233	CMDS025	XMT	CMDO2,IVR		SAVE COMMAND BYTE	
2972	02657	0	01037		MOV	R1,RB			
2973	02660	6	11041		CALL	UBFCH		GET LENGTH BYTE	
	02661	7	05565						
2974	02662	6	00374		XMT	-4,AUX		LENGTH LEFT = LENGTH-4	
2975	02663	1	01005		ADD	R1,R5		R5 <- LENGTH LEFT	
2976	02664	6	17234		XMT	CMDO3,IVR		SAVE LENGTH BYTE	
2977	02665	0	01037		MOV	R1,RB			
2978	02666	6	06235		XMT	CMDO4,R6		R6 <- COMMAND BUFFER POINTER	
2979	02667	7	02701		JMP	CMDS035		GO TO TEST	
2980				*					
2981	02670	6	02075	CMDS030	XMT	RCVRBLK,R2		R2 <- BUFFER BLOCK ADDR	
2982	02671	6	11042		CALL	UBFCH		GET NEXT BYTE	
	02672	7	05565						
2983	02673	0	06017		MOV	R6,IVR		LOAD COMMAND ADDR	
2984	02674	0	01037		MOV	R1,RB		WRITE TO BUFFER	
2985	02675	6	00001		XMT	1,AUX		AUX <- INCREMENT	
2986	02676	1	06006		ADD	R6,R6		BUMP BUFFER POINTER	
2987	02677	6	00377		XMT	-1,AUX		AUX <- DECREMENT	
2988	02700	1	05005		ADD	R5,R5		DECREMENT COUNT	
2989				*					
2990	02701	5	05270	CMDS035	NZT	R5,CMDS030		LOOP IF NOT DONE	
2991				*					
2992	02702	6	17233	CMDS040	XMT	CMDO2,IVR		LOAD ADDR	
2993	02703	6	07021		XMT	IVISPD+IVOSPD,IVL		*1 - DO SELECTS	
2994	02704	0	33401		MOV	34H,4,R1		READ COMMAND ONLY	
2995	02705	4	01306		XEC	CMDSTAB2(R1),16		BRANCH TO COMMAND HANDLER	
2996				*					
2997	02706	7	04645	CMDSTAB7	JMP	CMDO0000		NOT USED	
2998	02707	7	02726		JMP	CMDO1000		READ COMMAND	
2999	02710	7	03066		JMP	CMDO2000		WRITE COMMAND	
3000	02711	7	03350		JMP	CMDO3000		SEARCH COMMAND	
3001	02712	7	03562		JMP	CMDO4000		POWER COMMAND	
3002	02713	7	03625		JMP	CMDO5000		INSERT COMMAND	
3003	02714	7	04121		JMP	CMDO6000		DELETE COMMAND	
3004	02715	7	04362		JMP	CMDO7000		LED COMMAND	
3005	02716	7	04372		JMP	CMDO8000		STOP COMMAND	
3006	02717	7	04403		JMP	CMDO9000		GO COMMAND	
3007	02720	7	04442		JMP	CMDO10000		INITIALIZE COMMAND	
3008	02721	7	04550		JMP	CMDO11000		INSERT AT END-OF-COLUMN	
3009	02722	7	04601		JMP	CMDO12000		DELETE AT END-OF-COLUMN	
3010	02723	7	04645		JMP	CMDO13000		NOT USED	
3011	02724	7	04645		JMP	CMDO14000		NOT USED	
3012	02725	7	04645		JMP	CMDO15000		NOT USED	
3014				*					
3015				***READ COMMAND					
3016				*					
3017	02726	6	11043	CMDO1000	CALL	ADRVAL		VALIDATE ADDRESS	
	02727	7	06041						
3018				*					
3019				* ADRVAL RETURNS WITH THE READ ADDR IN R5,R6 AND THE - LENGTH TO					
				* READ IN R2					
3020	02730	6	17234		XMT	CMDO3,IVR		CALCULATE RESPONSE LENGTH	
3021	02731	6	00371		XMT	-7,AUX			
3022	02732	1	02003		ADD	R2,R3		R3 <- -(LENGTH+7)	
3023	02733	6	00377		XMT	-1,AUX			
3024	02734	3	03000		XOR	R3,AUX		AUX <- LENGTH+6 = RESPONSE LENGTH	
3025	02735	0	00037		MOV	AUX,RB		PUT IN RESPONSE	
3026	02736	6	11237		XMT	CMDO6,R11		R11 <- DATA START ADDR	
3027				*					
3028	02737	4	01340	CMDO1005	XEC	CMDO1TAB(R1),4		EXECUTE OFF FIELD TYPE	
3029				*					
3030	02740	7	02744	CMDO1TAB	JMP	CMDO1010		LOGIC SPACE	
3031	02741	7	02762		JMP	CMDO1020		I/O SPACE	
3032	02742	7	03000		JMP	CMDO1030		REGISTER SPACE	
3033	02743	7	03025		JMP	CMDO1040		SCRATCHPAD	
3034				*					
3035				***LOGIC RAM					
3036				*					
3037	02744	6	07004	CMDO1010	XMT	IVOLRHI,IVL		SELECT LOGIC ADDRHI	
3038	02745	0	05027		MOV	R5,LB		LOAD ADDRESS	
3039	02746	6	07003		XMT	IVOLRLO,IVL		SELECT LOGIC ADDRLO	
3040	02747	0	06027		MOV	R6,LB		LOAD ADDRESS	
3041	02750	6	07000		XMT	IVILRDAT+IVOCTRL,IVL		*1 - SELECT PORTS	
3042	02751	6	04000		XMT	CTRLINCL,R4		*2 - R4 <- INCREMENT	

3043	02752	0	00000	+	NOP		*3	
3044	02753	0	37003		MOV	AUX,AUX		
3045	02754	0	04027		MOV	RB,R3	R3 <- DATAHI	
3046					MOV	R4,LB	BUMP ADDRESS	
3046	02755	0	00000	+	NOP		*1 - WAIT	
3047					NOP		*2 - WAIT	
3048	02756	0	00000	+	MOV	AUX,AUX		
3048					NOP		*3 - WAIT	
3049	02757	0	00000	+	MOV	AUX,AUX		
3050	02760	0	37004		MOV	RB,R4	R4 <- DATALO	
3051	02761	7	03034		JMP	CMD01100	CONTINUE	
3052					*			
3053					***I/O SPACE			
3054	02762	6	07001		CMD01020	XMT	IVOCRHI,IVL	SELECT COIL ADDRHI
3055	02763	0	05027		MOV	R5,LB	LOAD ADDRESS	
3056	02764	6	07000		XMT	IVOCRLO,IVL	SELECT COIL	
3057	02765	0	06037		MOV	R6,RB	LOAD ADDRESS	
3058	02766	6	04001		XMT	CTRLINCC,R4	*1 - R4 <- INCREMENT	
3059	02767	6	07000		XMT	IVICRDAT+IVOCTRL,IVL	*2 - SELECT PORTS	
3060					NOP		*3 - WAIT	
3061	02770	0	00000	+	MOV	AUX,AUX		
3062	02771	0	27003		MOV	LR,R3	R3 <- DATAHI	
3063	02772	0	04027		MOV	R4,LB	INCREMENT ADDRESS	
3064					NOP		*1 - WAIT	
3065	02773	0	00000	+	MOV	AUX,AUX		
3066					NOP		*2 - WAIT	
3067	02774	0	00000	+	MOV	AUX,AUX		
3068					NOP		*3 - WAIT	
3069	02775	0	00000	+	MOV	AUX,AUX		
3070	02776	0	27004		MOV	LB,R4	R4 <- DATALO	
3071	02777	7	03034		JMP	CMD01100	CONTINUE	
3072					*			
3073					***REGISTER SPACE			
3074	03000	6	07000		CMD01030	XMT	IVOCRLO,IVL	SELECT COIL ADDRLO
3075	03001	0	06037		MOV	R6,RB	LOAD ADDRESS	
3076	03002	6	07001		XMT	IVOCRHI+IVICRDAT,IVL	SELECT PORTS	
3077	03003	6	00001		XMT	1,AUX	AUX <- INCREMENT	
3078	03004	1	05027		ADD	R5,LB	BUMP HI-ORDER ADDR	
3079	03005	6	00002		XMT	2,AUX	*1 - INCREMENT FOR ADDRHI	
3080					NOP		*2 - WAIT	
3081	03006	0	00000	+	MOV	AUX,AUX		
3082					NOP		*3 - WAIT	
3083	03007	0	00000	+	MOV	AUX,AUX		
3084	03010	0	27404		MOV	REGDATA,R4	R4 <- LOW-ORDER NIBBLE	
3085	03011	1	05027		ADD	R5,LB	LOAD ADDR MIDDLE NIBBLE	
3086	03012	6	00003		XMT	3,AUX	*1 - INC FOR NEXT ADDRHI	
3087					NOP		*2 - WAIT	
3088	03013	0	00000	+	MOV	AUX,AUX		
3089					NOP		*3 - WAIT	
3090	03014	0	00000	+	MOV	AUX,AUX		
3091	03015	0	27403		MOV	REGDATA,R3	R3 <- MIDDLE NIBBLE	
3092	03016	1	05027		ADD	R5,LB	LOAD ADDR - HI NIBBLE	
3093	03017	0	03400		MOV	R3(4),AUX	*1 - AUX <- MIDDLE NIBBLE	
3094	03018	3	04004		XOR	R4,R4	*2 - R4 <- LOW-ORDER BYTE	
3095	03021	6	00003		XMT	0000011B,AUX	*3 - AUX <- MASK	
3096	03022	0	27403		MOV	REGDATA,R3	R3 <- HIGH-ORDER NIBBLE	
3097	03023	2	03003		AND	R3,R3	ISOLATE DATA	
3098	03024	7	03034		JMP	CMD01100	CONTINUE	
3099					*			
3100					***SCRATCHPAD SPACE			
3101	03025	0	06017		CMD01040	MOV	R6,IVR	LOAD ADDRESS
3102	03026	6	07020		XMT	IVISPD,IVL	*1 - SELECT SCRATCHPAD READ	
3103	03027	0	37003		MOV	RB,R3	R3 <- DATAHI	
3104	03030	6	00001		XMT	1,AUX	AUX <- INCREMENT	
3105	03031	1	06017		ADD	R6,IVR	BUMP ADDRESS	
3106					NOP		*1 - WAIT	
3107	03032	0	00000	+	MOV	AUX,AUX		
3108	03033	0	37004		MOV	RB,R4	R4 <- DATALO	
3109					*			
3110	03034	0	11017		CMD01100	MOV	R11,IVR	LOAD ADDR OF RESPONSE
3111	03035	6	07021		XMT	IVISPD+IVOSPD,IVL	DO SELECTS	
3112	03036	0	03037		MOV	R3,RB	WRITE DATA HI	
3113	03037	6	00001		XMT	1,AUX	*1 - INC FOR ADDR	
3114	03040	1	11017		ADD	R11,IVR	LOAD ADDRESS	
3115	03041	0	04037		MOV	R4,RB	WRITE DATA LO	
3116	03042	6	00002		XMT	2,AUX	*1 - INC FOR ADDR	
3117	03043	1	11011		ADD	R11,R11	INC RESPONSE ADDR	
3118	03044	4	01045		XEC	READCONT(R1),R4	EXECUTE CONTINUE OFF FIELD	
3119					*			
3120	03045	7	03051		READCONT	JMP	CMD01200	LOGIC
3121	03046	7	03051		JMP	CMD01200	I/O SPACE	
3122	03047	7	03060		JMP	CMD01300	REGISTER SPACE	
3123	03050	7	03051		JMP	CMD01200	SPD	
3124					*			
3125	03051	1	02002		CMD01200	ADD	R2,R2	INC LENGTH LEFT (AUX = 2)
3126	03052	5	02054		NZT	R2,CMD01210		
3127	03053	7	04660		JMP	CMDRSP	IF NONE LEFT, BUILD RESPONSE	
3128	03054	1	06006		CMD01210	ADD	R6,R6	UPDATE ADDR LO
3129	03055	0	10000		MOV	OVF,AUX		
3130	03056	1	05005		ADD	R5,R5	UPDATE ADDR HI	
3131	03057	7	02737		JMP	CMD01005		


```

3125
3126 03060 1 02002 * CMDD01300 ADD R2,R2 INC LENGTH LEFT
3127 03061 5 02063 NZT R2,CMDD01310
3128 03062 7 04660 JMP CMDRSP IF NONE LEFT, BUILD RESPONSE
3129 03063 6 00001 CMDD01310 XMT 1,AUX COIL ADDRLO GETS BUMPED BY 1
3130 03064 1 06006 ADD R6,R6
3131 03065 7 03000 JMP CMDD01030 CONTINUE COIL READ

3133 *
3134 ***WRITE COMMAND
3135 *
3136 03066 6 11044 CMDD02000 CALL PROTECT
      03067 7 06541
3137 03070 6 11045 CALL ADRVAL
      03071 7 06041
3138 03072 6 00370 XMT -8,AUX WRITELNGTH = CMDLEN - 8, TELL LENVAL
3139 03073 6 11046 CALL LENVAL VALIDATE LENGTH
      03074 7 06171
3140 03075 6 03237 XMT CMDD06,R3 R3<- WRITE DATA ADDR
3141 03076 6 17234 XMT CMDD03,IVR CALCULATE MASK ADDR
3142 03077 6 00227 XMT CMDD01-3,AUX *1
3143 03100 1 37004 ADD RB,R4 R4<- MASK ADDR
3144 03101 4 01102 XEC CMDD02TAB(R1),4
3145 *
3146 03102 7 03106 CMDD02TAB JMP CMDD02010 LOGIC SPACE
3147 03103 7 03257 JMP CMDD02020 I/O SPACE
3148 03104 7 03314 JMP CMDD02030 REGISTER SPACE
3149 03105 7 03344 JMP CMDD02040 SCRATCHPAD

3150 *
3151 ***LOGIC SPACE
3152 *
3153 ORG 8,256
3154 03106 5 05116 CMDD02010 NZT R5,CMDD02012 CHECK IF ADDRESS = (0,0)
3155 03107 5 06116 NZT R6,CMDD02012
3156 03110 6 00002 XMT 2,AUX IF SO, DON'T DO WRITE, BUT DON'T COMPLAIN
3157 03111 1 06006 ADD R6,R6 BUMP LOGIC ADDR TO 2
3158 03112 1 03003 ADD R3,R3 BUMP COMMAND DATA ADDR
3159 03113 1 02002 ADD R2,R2 DEC LENGTH
3160 03114 5 02116 NZT R2,CMDD02012 SEE IF ANY WRITING TO DO
3161 03115 7 04660 JMP CMDRSP IF NOT, EXIT
3162 03116 6 17024 CMDD02012 XMT SAVER2,IVR SAVE LENGTH
3163 03117 6 07021 XMT IVISPD+IVOSPD,IVL
3164 03120 0 02037 MOV R2,RB
3165 03121 0 02001 MOV R2,R1 *1 - R1<- LENGTH TO VALIDATE
3166 03122 6 17023 XMT SAVER1,IVR SAVE IT TOO
3167 03123 0 01037 MOV R1,RB
3168 *
3168 03124 0 00000 * MOV AUX,AUX *1 - WAIT
3169 03125 6 17025 * XMT SAVER3,IVR SAVE DATA ADDRESS
3170 03126 0 03037 MOV R3,RB
3171 *
3171 03127 0 00000 * MOV AUX,AUX *1 - WAIT
3172 03130 6 17027 * XMT SAVER5,IVR SAVE ADDRESSES
3173 03131 0 05037 MOV R5,RB
3174 *
3174 03132 0 00000 * MOV AUX,AUX *1
3175 03133 6 17030 * XMT SAVER6,IVR
3176 03134 0 06037 MOV R6,RB
3177 03135 6 07003 XMT IVOLRLO,IVL SELECT LOGIC ADDRLO
3178 03136 0 06027 MOV R6,LB SET IT
3179 03137 6 07024 XMT IVOLRHI+IVISPD,IVL SELECT LOGIC ADDRHI AND SPD READ

3180 03140 0 05027 MOV R5,LB SET LOGIC ADDRHI
3181 03141 0 03017 CMDD02013 MOV R3,IVR SET COMMAND DATA ADDR
3182 03142 6 00001 XMT 1,AUX *1
3183 03143 0 37001 MOV RB,R1 R1<- DATAHI
3184 03144 0 04017 MOV R4,IVR SET MASK ADDR
3185 03145 1 03003 ADD R3,R3 *1 - INC DATA ADDR
3186 03146 0 37000 MOV RB,AUX AUX<- MASKHI
3187 03147 6 07000 XMT IVILRDAT,IVL SELECT LOGIC READ
3188 03150 2 37000 AND RB,AUX MASK BITS
3189 03151 3 01001 XOR R1,R1 R1<- REPLACEMENT DATAHI
3190 03152 6 07020 XMT IVOCTRL+IVISPD,IVL SELECT CONTROL REG AND SPD READ
3191 03153 6 27300 XMT CTRLINCL,CTRLREG INC LOGIC ADDR
3192 03154 0 03017 MOV R3,IVR SET DATA ADDR
3193 03155 6 00001 XMT 1,AUX *1
3194 03156 0 37002 MOV RB,R2 R2<- DATALO
3195 03157 1 04017 ADD R4,IVR SET MASKLO ADDR
3196 03160 1 03003 ADD R3,R3 *1 - INC DATA ADDR
3197 03161 0 37000 MOV RB,AUX AUX<- MASKLO
3198 03162 6 07000 XMT IVILRDAT,IVL SELECT LOGIC READ
3199 03163 2 37000 AND RB,AUX MASK BITS
3200 03164 3 02002 XOR R2,R2 R2<- REPLACEMENT DATALO
3201 03165 6 11047 CALL VALIDATE SEE IF VALID NODE
      03166 7 06702
3202 03167 6 00377 * XMT -1,AUX VALIDATE RETURNS WITH -1 IN R1
3203 03170 3 01000 XOR R1,AUX IF BAD NODE
3204 03171 5 00174 NZT AUX,CMDD02014
3205 03172 6 01012 XMT ERRNOD,R1 TAKE ERROR EXIT
3206 03173 7 04646 JMP CMDERR
3207 03174 6 17023 CMDD02014 XMT SAVER1,IVR CHCK IF MORE TO VALIDATE
3208 03175 6 07021 XMT IVISPD+IVOSPD,IVL
3209 03176 6 00002 XMT 2,AUX
3210 03177 1 37001 ADD RB,R1
3211 03200 0 01037 MOV R1,RB SAVE NEW VALUE
3212 03201 6 07000 XMT IVOCTRL,IVL BUMP LOGIC ADDR NOW
3213 03202 6 27300 XMT CTRLINCL,CTRLREG BUMP LOGIC ADDR NOW
3214 03203 5 01141 NZT R1,CMDD02013 LOPP IF MORE
3215 RSP SAVER3,R3 ELSE, RESTORE DATA AND LOGIC ADDRS

```

```

3215 03204 6 17025 + XMT SAVER3,IVR LOAD ADDRESS
3215 03205 6 07021 + XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
3215 03206 0 37003 + MOV RB,R3 READ DATA
3216 RSP SAVER5,R5
3216 03207 6 17027 + XMT SAVER5,IVR LOAD ADDRESS
3216 03210 6 07021 + XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
3216 03211 0 37005 + MOV RB,R5 READ DATA
3217 RSP SAVER6,R6
3217 03212 6 17030 + XMT SAVER6,IVR LOAD ADDRESS
3217 03213 6 07021 + XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
3217 03214 0 37006 + MOV RB,R6 READ DATA
3218 03215 0 03017 CMD02015 MOV R3,IVR LOAD ADDRESS
3219 03216 6 00001 XMT 1,AUX *1 - SET AUX TO INCREMENT
3220 03217 0 37003 MOV RB,R1 R1<- WRITE DATA
3221 03220 0 04017 MOV R4,IVR LOAD MASK ADDR
3222 03221 1 03003 ADD R3,R3 *1 - INC DATA ADDR
3223 03222 0 37000 MOV RB,AUX AUX<- MASK
3224 03223 6 07004 XMT IVOLRHI,IVL SELECT LOGIC ADDRHI
3225 03224 0 05027 MOV R5,LB LOAD ADDRHI
3226 03225 6 07003 XMT IVOLRLO,IVL SELECT LOGIC ADDRLO
3227 03226 0 06027 MOV R6,LB LOAD ADDRLO
3228 03227 6 07011 XMT IVILRDAT+IVOLRDAT,IVL *1 - SELECT LR READ/WRITE
3229 NOP *2 - WAIT
3229 03230 0 00000 + MOV AUX,AUX *3 - WAIT
3230 NOP
3230 03231 0 00000 + MOV AUX,AUX
3231 03232 2 37000 AND RB,AUX MASK BITS
3232 03233 3 01001 XOR R1,R1
3233 03234 6 11050 CALL WRTUP WRITE OUT DATA
3233 03235 7 05456
3234 03236 6 17024 XMT SAVER2,IVR LOAD ADDR OF -LENGTH
3235 03237 6 07021 XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ/WRITE
3236 03240 6 00001 XMT 1,AUX
3237 03241 1 37037 ADD RB,RB
3238 03242 1 06006 ADD R6,R6 * INC LOGIC ADDRLO (HAVE TO WAIT)
3239 03243 0 10000 MOV OVF,AUX *2 FOR INC'ING LOGIC ADDRHI
3240 03244 5 37010 NZT RB,CMD02016 IF MORE LEFT, CONTINUE
3241 03245 6 11051 CALL CLRDIAG ELSE,CLEAR DIAGNOSTIC
3241 03246 7 06214
3242 03247 7 04660 JMP CMDRSP EXIT
3243 03250 1 05005 CMD02016 ADD R5,R5 SET LOGIC ADDRHI
3244 03251 6 00001 XMT 1,AUX * K WHETHER TO USE MASKHI OR LO
3245 03252 2 06000 AND R6,AUX
3246 03253 5 00255 NZT AUX,CMD02017
3247 03254 6 00377 XMT -1,AUX IF R6 ODD, GO FROM LO TO HI
3248 03255 1 04004 CMD02017 ADD R4,R4 ELSE HI TO LO
3249 03256 7 03215 JMP CMD02015
3250 *
3251 ***I/O SPACE
3252 *
3253 EQU 003257
3254 03257 6 07021 CMD02020 EQU *
3255 03260 0 03017 CMD02025 XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
3256 03261 6 00001 MOV R3,IVR LOAD DATA ADDRESS
3257 03262 0 37001 XMT 1,AUX *1 - SET AUX FOR INCREMENT
3258 03263 0 04017 MOV RB,R1 R1<- WRITE DATA
3259 03264 1 03003 MOV R4,IVR LOAD MASK ADDR
3260 03265 0 37000 ADD R3,R3 *1 - INC DATA ADDR
3261 03266 6 07001 MOV RB,AUX AUX<- MASK
3262 03267 0 05027 XMT IVOCRHI,IVL SELECT COIL ADDRHI
3263 03270 6 07000 MOV R5,LB SET ADDRHI
3264 03271 0 06037 XMT IVOCRLO,IVL SELECT COIL ADDRLO
3265 03272 6 07002 MOV R6,RB SET ADDRLO
3266 03273 0 00000 + XMT IVICRDAT+IVOCRDA1,IVL *1 - SELECT COIL READ/WRITE
3267 NOP *2 - WAIT
3268 03274 0 00000 + MOV AUX,AUX *3 - WAIT
3269 03275 2 27000 AND LB,AUX MASK BITS
3270 03276 3 01027 XOR R1,LB
3271 03300 1 02002 XMT 1,AUX *1 - LENGTH LEFT
3272 03301 5 02303 ADD R2,R2
3273 03302 7 04660 NZT R2,CMD02026 IF MORE LEFT, CONTINUE
3274 03303 1 06006 JMP CMDHSP EXIT
3275 03304 0 10000 CMD02026 ADD R6,R6 INC COIL ADDRESS
3276 03305 1 05005 MOV OVF,AUX
3277 03306 6 00001 ADD R5,R5
3278 03307 2 06000 XMT 1,AUX CHECK WHICH MASK TO USE
3279 03310 5 00312 AND R6,AUX
3280 03311 6 00377 NZT AUX,CMD02027 IF R6 ODD, GO FROM LO TO HI
3281 03312 1 04004 XMT -1,AUX ELSE HI TO LO
3282 03313 7 03257 CMD02027 ADD R4,R4
3283 JMP CMD02025
3284 *
3285 ***REGISTER SPACE
3286 EQU 003314
3287 03314 6 07021 CMD02030 EQU *
3288 03315 6 17024 CMD02035 XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
3289 03316 0 02037 XMT SAVER2,IVR LOAD ADDRESS
3290 03317 6 00001 MOV R2,RB SAVE -LENGTH
3291 03320 0 03017 XMT 1,AUX *1
3292 03321 1 03003 MOV R3,IVR LOAD DATA ADDR
3293 03322 0 37001 ADD R3,R3 *1 - INC DATA ADDR
3294 03323 0 03017 MOV RB,R1 R1<- DATAHI
3295 03324 1 03003 MOV R3,IVR NEXT DATA WORD
3296 03325 0 37002 ADD R3,R3 *1 - INC DATA ADDR
3296 MOV RB,R2 R2<- DATALO

```


3297	03326	6	00005	XMT	00000011B,AUX	ISOLATE DATA	
3298	03327	2	01001	AND	R1,R1		
3299	03330	6	05001	XMT	1,R5	STORE WANTS ADDRHI SET TO 1	
3300	03331	6	11052	CALL	STORE	WRITE DATA	
	03332	7	05777				
3301	03333	6	17024	XMT	SAVER2,IVR	INC LENGTH LEFT	
3302	03334	6	07021	XMT	IVISPD+IVOSPD,IVL		
3303	03335	6	00002	XMT	2,AUX		
3304	03336	1	37037	ADD	RB,RB		
3305	03337	6	00001	XMT	1,AUX	*1	
3306	03340	1	06006	ADD	R6,R6	*2 - INC COIL ADDR	
3307	03341	5	37003	NZT	RB,CMD02036	LOOP IF MORE TO WRITE	
3308	03342	7	04660	JMP	CMDRSP	ELSE, EXIT	
3309	03343	7	03314	CMD02036 JMP	CMD02035		
3310				*			
3311				***SCRATCHPAD SPACE			
3312				*			
3313	03344	6	01005	CMD02040 XMT	ERRADI,R1	NO WRITE ALLOWED TO SPD	
3314	03345	7	04646	JMP	CMDERR	ERROR EXIT	
3316				*			
3317				***SEARCH COMMAND			
3318				*			
3319				* SEARCH MUST DO ITS OWN ADDRESS VALIDATION. IT CAN'T USE SUBROUTINE ADRVAL			
3320				*			
3321	03346	6	01005	CMD03005 XMT	ERRADI,R1	ERROR EXIT	
3322	03347	7	04646	JMP	CMDERR		
3323				*			
3324	03350	6	17235	CMD03000 XMT	CMD04,IVR	CHECK ADDRESS	
3325	03351	6	07021	XMT	IVOSPD+IVISPD,IVL	SELECT SPD READ/WRITE	
3326	03352	0	37005	MOV	RB,R5	R5<- ADDRHI	
3327	03353	0	32301	MOV	ADRFLD,R1	R1<- FIELD	
3328	03354	5	01346	NZT	R1,CMD03005	ERROR IF NOT LOGIC SPACE	
3329	03355	6	17236	XMT	CMD05,IVR	SET ADDRLO ADDR	
3330	03356	6	00001	XMT	1,AUX	*1	
3331	03357	0	37006	MOV	RB,R6	R6<- ADDRLO	
3332	03360	2	06000	AND	R6,AUX	R6 SHOULD BE EVEN	
3333	03361	5	00346	NZT	AUX,CMD03005		
3334	03362	6	11053	CALL	LENZERO	CHECK THAT CMD-LEN = 0	
	03363	7	06205				
3335				*			
3336	03364	5	05367	CMD03010 NZT	R5,CMD03011	ADDRESS SHOULDN'T BE (0,0)	
3337	03365	5	06367	NZT	R6,CMD03011		
3338	03366	7	03346	JMP	CMD03005		
3339	03367	6	11000	CMD03011 XMT	0,R11		
3340	03370	6	17243	XMT	CMD10,IVR	USE CMD10 AND CMD11 FOR NETHI	
3341	03371	6	07001	XMT	IVOSPD,IVL	AND NETLO FOR NOW	
3342	03372	0	11037	MOV	R11,RB	CLEAR NETHI AND NETLO	
3343				NOP		*1 - WAIT	
3343	03373	0	00000	MOV	AUX,AUX		
3344	03374	6	17244	XMT	CMD11,IVR		
3345	03375	0	11037	MOV	R11,RB		
3346	03376	6	00377	CMD03015 XMT	-1,AUX	*1	
3347	03377	6	17237	XMT	CMD06,IVR	LOAD ADDRESS	
3348	03400	6	07024	XMT	IVISPD+IVOLRHI,IVL	*1 - DO SELECTS	
3349	03401	0	37001	MOV	RB,R1	R1<- DATAHI	
3350	03402	6	17240	XMT	CMD07,IVR		
3351	03403	0	05027	MOV	R5,LB	*1 - LOAD LOGIC ADDRHI	
3352	03404	0	37002	MOV	RB,R2	R2<- DATALO	
3353	03405	6	17241	XMT	CMD08,IVR	LOAD ADDRESS	
3354	03406	6	07023	XMT	IVISPD+IVOLRLO,IVL	*1 - DO SELECTS	
3355	03407	0	37003	MOV	RB,R3	R3<- MASKHI	
3356	03410	6	17242	XMT	CMD09,IVR	LOAD ADDRESS	
3357	03411	0	06027	MOV	R6,LB	*1 - LOAD LOGIC ADDRLO	
3358	03412	0	37004	MOV	RB,R4	R4<- MASKLO	
3359	03413	3	03003	XOR	R3,R3	COMPLEMENT MASKHI	
3360	03414	3	04004	XOR	R4,R4	COMPLEMENT MASKLO	
3361				*			
3362	03415	6	07000	CMD03020 XMT	IVOCTRL+IVILRDAT,IVL	DO SELECTS	
3363	03416	6	00000	XMT	NODESON.L.2,AUX	AUX<- MASK	
3364	03417	3	37000	XOR	RB,AUX	AUX.EQ.0 => START-OF-NETWORK NODE	
3365	03420	5	00032	NZT	AUX,CMD03030	AUX.NE.0 => NOT A START NODE	
3366				*			
3367	03421	6	17244	XMT	CMD11,IVR	LOAD ADDRESS	
3368	03422	6	07021	XMT	IVISPD+IVOSPD,IVL	*1 - DO SELECTS	
3369	03423	6	00001	XMT	1,AUX	AUX<- INCREMENT	
3370	03424	1	37037	ADD	RB,RB	NETLO<- NETLO + 1	
3371	03425	0	10000	MOV	OVF,AUX	AUX<- OVERFLOW	
3372	03426	6	17243	XMT	CMD10,IVR	LOAD ADDRESS	
3373				NOP		*1 - WAIT	
3373	03427	0	00000	MOV	AUX,AUX		
3374	03430	1	37037	ADD	RB,RB	NETHI<- NETHI + OVF	
3375	03431	6	07000	XMT	IVOCTRL+IVILRDAT,IVL	DO SELECTS	
3376				*			
3377	03432	0	03000	CMD03030 MOV	R3,AUX	AUX<- MASKHI	
3378	03433	2	37000	AND	RB,AUX	ISOLATE NEEDED BITS	
3379	03434	6	27300	XMT	CTRLINCL,CTRLREG	INC LOGIC ADDR	
3380	03435	3	01000	XOR	R1,AUX	*1 - AUX.EQ.0 => MATCH	
3381	03436	5	00102	NZT	AUX,CMD03050	*2 - AUX.NE.0 => NO MATCH	
3382				*			
3383	03437	0	04000	MOV	R4,AUX	*3 - AUX<- MASKLO	
3384	03440	2	37000	AND	RB,AUX	ISOLATE NEEDED BITS	
3385	03441	3	02000	XOR	R2,AUX	AUX.EQ.0 => MATCH	

3386	03442	5	00102		NZT	AUX,CMD03050	AUX.NE.0 => NO MATCH
3387	03443	6	17235	CMD03040	XMT	CMD04,IVR	LOAD ADDRESS
3388	03444	6	07004		XMT	IVOLRHI,IVL	SELECT SPDOUT ALSO
3389	03445	0	05037		MOV	R5,RB	LOAD RESPONSE ADDRHI
3390	03446	0	05027		MOV	R5,LB	*1 - LOAD LOGIC ADDRHI
3391	03447	6	17236		XMT	CMD05,IVR	LOAD ADDRESS
3392	03450	6	07003		XMT	IVOLRLO,IVL	SELECT SPDOUT ALSO
3393	03451	0	06037		MOV	R6,RB	LOAD RESPONSE ADDRLO
3394	03452	0	06027		MOV	R6,LB	*1 - LOAD LOGIC ADDRLO
3395	03453	6	07001		XMT	IVILRDAT+IVOSPD,IVL	*1 - DO SELECTS
3396	03454	6	17237		XMT	CMD06,IVR	*2 - LOAD ADDRESS
3397					NOP		*3 - WAIT
3397	03455	0	00000	+	MOV	AUX,AUX	
3398	03456	0	37037		MOV	RB,RB	STORE DATAHI
3399	03457	6	07000		XMT	IVOCTRL,IVL	DO SELECT
3400	03460	6	27300		XMT	CTRLINCL,CTRLREG	INC LOGIC ADDR
3401	03461	6	07001		XMT	IVILRDAT+IVOSPD,IVL	*1 - DO SELECTS
3402	03462	6	17240		XMT	CMD07,IVR	*2 - LOAD SPD ADDR
3403					NOP		*3 - WAIT
3403	03463	0	00000	+	MOV	AUX,AUX	
3404	03464	0	37037		MOV	RB,RB	STORE DATALO
3405					NOP		*1 - WAIT
3405	03465	0	00000	+	MOV	AUX,AUX	
3406	03466	6	17243		XMT	CMD10,IVR	NOW PUT NETHI AND NETLO
3407	03467	6	07021		XMT	IVISPD+IVOSPD,IVL	*1 - WHERE THEY BELONG
3408	03470	0	37003		MOV	RB,R3	
3409	03471	6	17241		XMT	CMD08,IVR	
3410	03472	0	03037		MOV	R3,RB	
3411					NOP		*1 - WAIT
3411	03473	0	00000	+	MOV	AUX,AUX	
3412	03474	6	17244		XMT	CMD11,IVR	
3413					NOP		*1 - WAIT
3413	03475	0	00000	+	MOV	AUX,AUX	
3414	03476	0	37003		MOV	RB,R3	
3415	03477	6	17242		XMT	CMD09,IVR	
3416	03500	0	03037		MOV	R3,RB	
3417	03501	7	04660		JMP	CMDRSP	GO TO COMMON EXIT
3418					*		
3419	03502	6	27300	CMD03050	XMT	CTRLINCL,CTRLREG	INCREMENT ADDRESS
3420	03503	6	00002		XMT	Z,AUX	AUX <- DOUBLE INCREMENT
3421	03504	1	06006		ADD	R6,R6	ADDRLO <- ADDRLO + 2
3422	03505	5	10136		OVF,CMD03060	OVF.NE.0 => CHANGE FIELDS	
3423	03506	6	00177		XMT	01111111B,AUX	SEE IF WE HAVE CHANGED 128 BYTE PAGES
3424	03507	2	06000		AND	R6,AUX	
3425	03510	5	00015		NZT	AUX,CMD03020	NO, CONTINUE
3426					WSP	SADDRHI,R5	SAVE ADDRESSES
3426	03511	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
3426	03512	6	17263	+	XMT	SADDRHI,IVR	LOAD ADDRESS
3426	03513	0	05037	+	MOV	R5,RB	WRITE DATA
3427					WSP	SADDRLO,R6	
3427	03514	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
3427	03515	6	17264	+	XMT	SADDRLO,IVR	LOAD ADDRESS
3427	03516	0	06037	+	MOV	R6,RB	WRITE DATA
3428	03517	6	11054		CALL	INTRP	CALL INTERRUPT HANDLER
	03520	7	05103				
3429	03521	5	01131		NZT	R1,CMD03999	IF R1 .NE. 0, TROUBLES
3430					RSP	SADDRLO,R6	RESTORE ADDRESSES
3430	03522	6	17264	+	XMT	SADDRLO,IVR	LOAD ADDRESS
3430	03523	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
3430	03524	0	37006	+	MOV	RB,R6	READ DATA
3431					RSP	SADDRHI,R5	AND CONTINUE SEARCH
3431	03525	6	17263	+	XMT	SADDRHI,IVR	LOAD ADDRESS
3431	03526	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
3431	03527	0	37005	+	MOV	RB,R5	READ DATA
3432	03530	7	03376		JMP	CMD03015	
3433					*		
3434	03531	6	00001	CMD03999	XMT	SRCHCONT,AUX	SET CONTINUATION BIT
3435					WSP	CMDCONT,AUX	
3435	03532	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
3435	03533	6	17262	+	XMT	CMDCONT,IVR	LOAD ADDRESS
3435	03534	0	00037	+	MOV	AUX,RB	WRITE DATA
3436	03535	7	00453		JMP	EXEC	
3437					*		
3438	03536	0	10000	CMD03060	MOV	OVF,AUX	AUX <- OVF
3439	03537	1	05005		ADD	R5,R5	INCREMENT ADDRHI
3440	03540	6	17276		XMT	SPDCONF1,IVR	LOAD ADDRESS
3441	03541	6	07020		XMT	IVISPD,IVL	DO SELECT
3442	03542	0	34500		MOV	SYS0256B,5,AUX	AUX <- NUMBER OF LOGIC FIELDS
3443	03543	3	05000		XOR	R5,AUX	AUX.EQ.0 => SEARCH FAILED
3444	03544	5	00150		NZT	AUX,CMD03080	AUX.NE.0 => CONTINUE
3445	03545	6	05377		XMT	-1,R5	ADDRHI <- -1
3446	03546	6	06377		XMT	-1,R6	ADDRLO <- -1
3447	03547	7	03443		JMP	CMD03040	GO TO COMMON CODE
3448					*		
3449				CMD03080	WSP	SADDRHI,R5	SAVE ADDRESSES
3449	03550	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
3449	03551	6	17263	+	XMT	SADDRHI,IVR	LOAD ADDRESS
3449	03552	0	05037	+	MOV	R5,RB	WRITE DATA
3450					WSP	SADDRLO,R6	
3450	03553	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
3450	03554	6	17264	+	XMT	SADDRLO,IVR	LOAD ADDRESS
3450	03555	0	06037	+	MOV	R6,RB	WRITE DATA
3451	03556	6	00001		XMT	SRCHCONT,AUX	*1 - SET SEARCH TO CONTINUE
3452	03557	6	17262		XMT	CMDCONT,IVR	
3453	03560	0	00037		MOV	AUX,RB	
3454	03561	7	04732		JMP	CMD5X	

```

3456
3457
3458
3459 03562 6 11055  *
      03563 7 06205  ***POWER COMMAND
3460 03564 6 17235  *
      03565 6 07021  CMDD04000 CALL LENZERO CHECK THAT CMD-LEN = 0
3461 03566 0 37003  XMT CMD04,IVR LOAD ADDRESS
3462 03567 6 17040  XMT IVISPD+IVOSPD,IVL *1 - DO SELECTS
3463 03570 0 03037  MOV RB,R3 R3 <- POWERHI
3464 03571 6 00000  XMT POWERHI,IVR LOAD ADDRESS
3465 03572 6 17236  MOV R3,RB SET POWERHI
3466 03573 0 00000  *1 - WAIT
3467 03574 0 37003  *
      03575 6 17041  MOV AUX,AUX
3468 03576 0 03037  XMT CMD05,IVR LOAD ADDRESS
3469 03577 6 00000  *1 - WAIT
3470 03578 6 17063  MOV RB,R3 R3<- POWERLO
3471 03600 6 17063  XMT POWERLO,IVR
3472 03601 0 00037  MOV R3,RB SET POWERLO
3473 03602 6 00004  XMT D,AUX *1
3474 03603 6 17262  XMT LEDLOC,IVR SET FOR NO LED OUTPUT
3475 03604 0 00037  MOV AUX,RB
3476 03605 7 04732  XMT PWRCONT,AUX *1
3477 03606 6 00004  XMT CMDCONT,IVR SET POWER CONTINUATION
3478 03607 0 00037  MOV AUX,RB
3479 03608 7 04732  JMP CMDSX
3480
3481 03609 6 02237  *
3482 03610 6 03365  * POWER CONTINUES HERE
3483 03611 6 01045  *
3484 03612 6 07021  CMDD04010 XMT CMD06,R2 R2 <- DESTINATION ADDRESS
3485 03613 6 00001  XMT -11,R3 R3 <- COUNT
3486 03614 0 01017  XMT POWER1,R1 BEGINNING POWER ADDRESS
3487 03615 1 01001  XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
3488 03616 0 37004  XMT 1,AUX FOR INC'ING
3489 03617 0 02017  CMDD04020 MOV R1,IVR LOAD ADDRESS
3490 03618 1 02002  ADD R1,R1 *1 - BUMP POWER ADDR
3491 03619 0 04037  MOV RB,R4 R4<- POWER BYTE
3492 03620 1 03003  MOV R2,IVR LOAD RESPONSE ADDR
3493 03621 5 03213  MOV R4,RB LOAD RESPONSE BUFFER
3494 03622 7 04660  ADD R2,R2 BUMP RESPONSE ADDR
3495 03623 7 04660  ADD R3,R3 BUMP COUNT
3496 03624 5 03213  NZT R3,CMD04020 LOOP ON COUNT
3497 03625 7 04660  JMP CMDRSP BUILD RESPONSE
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548

***INSERT COMMAND
*
* PSEUDO-CODE DESCRIPTION
* DEFINITIONS:
* PAGE - 128 BYTES
* PAGADR - ADDRESS OF BEGINNING OF THE PAGE
* NOWPAGE - THE PAGE INSERT IS NOW WORKING ON
* EOLPAGE - THE PAGE THE FIRST EOL IS ON
* LASTPAGE - THE LAST PAGE IN PHYSICAL MEMORY
* INPAGE - THE PAGE THE DATA WILL BE INSERTED ON
* INNUM - THE NUMBER OF BYTES BEING INSERTED
* EOLAD - END OF LOGIC ADDRESS
* INSTAD - ADDRESS PASSED IN THE INSERT COMMAND (INSERT START ADDRESS)
* PASS1STB - FIRST PASS FLAG
* ENT1STB - FIRST ENTRY FLAG
* INSERT WILL MOVE ONE PAGE OF DATA, CALL INTRP, THEN MOVE ANOTHER
* PAGE OF DATA. IT THEN EXITS AND SETS THE COMMAND CONTINUATION WORD
* IF NECESSARY. INSERT MUST THEREFORE KNOW IF IT HAS MOVED ONE OR TWO
* PAGES (PASS1STB). IT MUST ALSO KNOW IF IT HAS JUST STARTED FOR THE
* FIRST TIME (ENT1STB).
*
*** INITIALIZE
*
* VALIDATE ADDRESS
* CHECK MEMORY PROTECT
* VALIDATE NODES TO INSERT
* VALIDATE ENOUGH ROOM IN MEMORY
*
* EOLAD = EOLAD + INNUM
* PASS1STB = 1
* ENT1STB = 1
*
*** CONTINUE HERE AFTER INTRP OR EXIT
*
* IF INPAGE .EQ. NOWPAGE
* IF NOWPAGE .EQ. LASTPAGE
* COUNT = EOLAD - INSTAD - INNUM
* FROMADDR = EOLAD - INNUM - 1
* TOADDR = FROMADDR + INNUM
* ELSE
* COUNT = 128 - INSTAD(MOD(128))
* FROMADDR = PAGADR + 127
* TOADDR = FROMADDR + INNUM
* DO WHILE COUNT .GT. 0
* C(TOADDR) = C(FROMADDR)
* TOADDR = TOADDR - 1
* FROMADDR = FROMADDR - 1
* COUNT = COUNT - 1
* COUNT = -INNUM
* TOADDR = INSTAD
* FROMADDR = CMDD04 (IN SPD)
* CHKPLUS = 0

```



```

3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3593
3594 03624 7 03773
3595
3596 03625 6 11056
      03626 7 06224
3597
3598
3599
3600 03627 6 07021
3601 03630 0 02000
3602 03631 3 01000
3603 03632 5 00224
3604 03633 6 17276
3605 03634 6 00377
3606 03635 1 35600
3607 03636 3 01000
3608 03637 5 00261
3609 03640 6 17066
3610 03641 6 00377
3611 03642 0 37006
3612 03643 6 17065
3613 03644 1 06004
3614 03645 0 37005
3615 03646 6 17265
3616 03647 0 05003
3617 03650 3 37000
3618 03651 1 06006
3619 03652 6 17236
3620 03653 6 00377
3621 03654 3 37000
3622 03655 1 06001
3623 03656 6 00002
3624 03657 1 01001
3625 03660 7 03702
3626
3627 03661 0 01105
3628 03662 6 00200
3629 03663 2 05006
3630 03664 6 00177
3631 03665 1 06006
3632 03666 6 17265
3633 03667 2 05005
3634 03670 0 37000
3635 03671 1 06004
3636 03672 0 10000
3637 03673 1 05003
3638 03674 6 17236
3639 03675 6 01177
3640 03676 0 37700
3641 03677 1 01001
3642 03700 6 00377
3643 03701 3 01001
3644 03702 6 11057

```

```

* DO WHILE COUNT .LT. 0
* C(TOADDR) = C(FROMADDR)
* CHKPLUS = CHKPLUS + C(FROMADDR)
* TOADDR = TOADDR + 1
* FROMADDR = FROMADDR + 1
* COUNT = COUNT + 1
* IF ENT1STB = 1
*   CHKMINUS = INNUM/2 * EOL
* ELSE
*   CHKMINUS = INNUM/2 * NULL
* LOGIC CHECKSUM = LOGIC CHECKSUM + CHKPLUS - CHKMINUS
* JMP CMDRSP - DONE, BUILD RESPONSE
* IF INPAGE .NE. NOWPAGE
*   IF NOWPAGE .EQ. LASTPAGE
*     COUNT = EOLAD(MOD(128)) - INNUM
*   ELSE
*     COUNT = 128
*   FROMADDR = PAGADR + COUNT - 1
*   TOADDR = FROMADDR + INNUM
* DO WHILE COUNT .GT. 0
*   C(TOADDR) = C(FROMADDR)
*   TOADDR = TOADDR - 1
*   FROMADDR = FROMADDR - 1
*   COUNT = COUNT - 1
* COUNT = -INNUM
* TOADDR = PAGADR
* DO WHILE COUNT .LT. 0
*   C(TOADDR,TOADDR+1) = NULL
*   TOADDR = TOADDR + 2
*   COUNT = COUNT + 2
* IF ENT1STB = 1
*   LRCHECKSUM = LRCHECKSUM + INNUM/2 * (NULL - EOL)
* ENT1STB = 0
* NOWPAGE = NOWPAGE - 1
* IF PASS1STB = 1
*   CALL INTRP
*   PASS1STB = 0
*   CONTINUE
* ELSE
*   PASS1STB = 1
*   SET INSERT CONTINUE
*   EXIT
*
CMD05101 JMP      CMD05500          SHORT BRANCH PROBLEM
*
CMD05000 CALL     INSTINIT         CALL INSERT INITIALIZE
*
* AT THIS POINT, R1 = NOWPAGE AND R2 = INPAGE
*
CMD05100 XMT      IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
      MOV        R2,AUX            AUX<- INPAGE
      XOR        R1,AUX            SEE IF NOWPAGE = INPAGE
      NZT        AUX,CMD05101
CMD05102 XMT      SPDCONF1,IVR     LOAD ADDRESS
      XMT        -1,AUX            *1
      ADD        32H,6,AUX         AUX<- LASTPAGE
      XOR        R1,AUX            SEE IF NOWPAGE = LASTPAGE IN MEM
      NZT        AUX,CMD05150     IF NOT, JUMP
      XMT        EOLLO,IVR        LOAD ADDRESS
      XMT        -1,AUX            *1 - WE WANT (R5,R6)<- FROMADDR
      MOV        R6,R6            WHERE FROMADDR = EOLAD - INNUM - 1
      XMT        EOLHI,IVR        WE WANT (R3,R4) = TOADDR
      ADD        R6,R4            *1 - WHERE TOADDR = EOLAD - 1
      MOV        R6,R5
      XMT        INNUM,IVR
      MOV        R5,R3            *1
      XOR        R6,AUX
      ADD        R6,R6
      XMT        CMD05,IVR        LOAD INSTAD ADDR
      XMT        -1,AUX            *1
      XOR        R6,AUX
      ADD        R6,R1            R1<- COUNT * EOLAD - INSTAD - INNUM
      XMT        2,AUX            IF COUNT = 0, INLOOP WILL TAKE CARE OF THINGS
      ADD        R1,R1
      JMP        CMD05200        DO THE DATA MOVE
*
CMD05150 MOV        R1(1),R5       HERE, NOWPAGE .NE. LASTPAGE
      XMT        10000000B,AUX     SO, FROMADDR = PAGEADDR+127
      AND        R5,R6            AND TOADDR = FROMADDR + INNUM
      XMT        01111111B,AUX     AND COUNT = 128 - INSTAD(LO7BITS)
      ADD        R6,R6            (R5,R6)<- FROMADDR
      XMT        INNUM,IVR        GET INNUM
      AND        R5,R5            *1
      MOV        R6,AUX            AUX<- INNUM
      ADD        R6,R4            (R3,R4)<- TOADDR
      MOV        0VF,AUX
      ADD        R5,R3
      XMT        CMD05,IVR        GET LO 7 BITS OF INSTAD
      XMT        -129,R1          *1
      MOV        30H,7,AUX
      ADD        R1,R1            R1<- COUNT
      XMT        -1,AUX
      XOR        R1,R1
CMD05200 CALL     INLOOP         MOVE THE DATA

```


3645
 3646
 3647
 3648
 3649
 3650 03704 6 04237
 3651 03705 6 17265
 3652 03706 6 07020
 3653 03707 6 00377
 3654 03710 3 37001
 3655 03711 6 00001
 3656 03712 1 01001
 3657 03713 6 07003
 3658 03714 1 06027
 3659 03715 6 07004
 3660 03716 0 05027
 3661 03717 6 02000
 3662 03720 0 04017
 3663 03721 6 07031
 3664 03722 0 37027
 3665 03723 0 37000
 3666 03724 1 02002
 3667 03725 6 00001
 3668 03726 1 01001
 3669 03727 6 07000
 3670 03730 6 27300
 3671 03731 1 04004
 3672 03732 5 01320
 3673
 3674
 3675
 3676
 3677
 3678 03733 6 17265
 3679 03734 6 07021
 3680 03735 0 36704
 3681 03736 6 17263
 3682
 3682 03737 0 00000
 3683
 3684 03740 5 30113
 3685 03741 4 04342
 3686 03742 7 03765
 3687
 3688
 3689
 3690 03743 6 00244
 3691 03744 6 00110
 3692 03745 6 00354
 3693 03746 6 00220
 3694 03747 6 00064
 3695 03750 6 00330
 3696 03751 6 00174
 3697 03752 6 00040
 3698
 3699 03753 4 04354
 3700 03754 7 03765
 3701
 3702
 3703
 3704 03755 6 00374
 3705 03756 6 00370
 3706 03757 6 00364
 3707 03760 6 00360
 3708 03761 6 00354
 3709 03762 6 00350
 3710 03763 6 00344
 3711 03764 6 00340
 3712
 3713 03765 1 02000
 3714 03766 6 11060
 03767 7 06503
 3715 03770 6 11061
 03771 7 06214
 3716 03772 7 04660
 3718
 3719
 3720
 3721 03773 6 17265
 3722 03774 6 00377
 3723 03775 3 37003
 3724 03776 6 17276
 3725 03777 6 00377
 3726 04000 1 35600
 3727 04001 3 01000
 3728 04002 5 00011
 3729 04003 6 17066
 3730 04004 6 00001
 3731 04005 1 37702
 3732 04006 0 03000
 3733 04007 1 02002
 3734 04010 7 04012
 3735 04011 6 02200
 3736 04012 0 01105
 3737 04013 6 00200
 3738 04014 2 05006

```

*
* AT THIS POINT, THE PAGE HAS BEEN MOVED
* (R5,R6+1) = INSTAD
* NOW WE WANT TO MOVE IN THE INSERT DATA
*
      XMT      CMD06,R4          R4<- FROMADDR = CMD06
      XMT      INNUM,IVR        LOAD SPD ADDR
      XMT      IVISPD,IVL      SELECT SPD READ
      XMT      -1,AUX          *1
      XOR      RB,R1           R1<- COUNT = -INNUM
      XMT      1,AUX
      ADD      R1,R1
      XMT      IVOLRLO,IVL     LOGIC ADDRLO
      ADD      R6,LB           TOADDR = INSTAD = (R5,R6) + 1
      XMT      IVOLRHI,IVL     SELECT LOGIC ADDRHI
      MOV      R5,LB
      XMT      0,R2            *1 - R2<- CHKPLUS = 0
      CMD05250 MOV      R4,IVR    *2 - LOAD SPD ADDRESS
      XMT      IVOLRDAT+IVISPD,IVL *3 - SELECT LR WRITE, SPD READ
      MOV      RB,LB           MOVE INSERT DATA
      MOV      RB,AUX
      ADD      R2,R2           UPDATE CHKPLUS
      XMT      1,AUX
      ADD      R1,R1           DEC COUNT
      XMT      IVCTRL,IVL     SELECT CONTROL PULSE
      XMT      CTRLINCL,CTRLREG INC LOGIC ADDR
      ADD      R4,R4           *1 - INC SPD ADDR
      NZT     R1,CMD05250     *2 - LOOP ON COUNT
*
* NOW WE HAVE TO UPDATE THE LOGIC CHECKSUM
* IF THIS IS THE FIRST ENTRY, LRCHKSUM = LRCHKSUM+CHKPLUS-INNUM/2+EOL
* IF NOT, LRCHKSUM = LRCHKSUM+CHKPLUS-INNUM/2=NULL
*
      XMT      INNUM,IVR        LOAD INNUM ADDRESS
      XMT      IVISPD+IVOSPD,IVL *1 - SELECT SPD READ/WRITE
      MOV      31H,7,R4        R4<- INNUM/2
      XMT      NOWPAGE,IVR     LOAD ADDRESS OF FLAGS
      NOP
      MOV      AUX,AUX
      ORG      11,32
      NZT     ENT1STB,CMD05300 JUMP IF THIS WAS THE FIRST ENTRY
      XEC     NULLTAB1-1(R4),8 ELSE WE WANT TO SUBTRACT NULLS
      JMP     CMD05400         FROM THE CHECKSUM
*
* THE FOLLOWING TABLE MUST BE CHANGED IF NULLNODE .NE. 23!!!!!!
*
NULLTAB1 XMT      244H,AUX     -(1*NULL)
          XMT      110H,AUX     -(2*NULL)
          XMT      354H,AUX     -(3*NULL)
          XMT      220H,AUX     -(4*NULL)
          XMT      064H,AUX     -(5*NULL)
          XMT      330H,AUX     -(6*NULL)
          XMT      174H,AUX     -(7*NULL)
          XMT      040H,AUX     -(8*NULL)
*
CMD05300 XEC     EOLTAB1-1(R4),8 SUBTRACT EOL'S
          JMP     CMD05400
*
* THE FOLLOWING TABLE MUST BE CHANGED IF EOLNODE .NE. 1!!!!!!
*
EOLTAB1  XMT      374H,AUX     -(1*EOL)
          XMT      370H,AUX     -(2*EOL)
          XMT      364H,AUX     -(3*EOL)
          XMT      360H,AUX     -(4*EOL)
          XMT      354H,AUX     -(5*EOL)
          XMT      350H,AUX     -(6*EOL)
          XMT      344H,AUX     -(7*EOL)
          XMT      340H,AUX     -(8*EOL)
*
CMD05400 ADD      R2,AUX       AUX<- CHKPLUS - CHKMINUS
          CALL     UPDTLCHK     UPDATE THE CHECKSUM
*
CMD05410 CALL     CLRDIAG     CLEAR DIAGNOSTIC
          JMP     CMDRSP       EXIT
*
* AT THIS POINT, INPAGE .NE. NOWPAGE, R1 = NOWPAGE
*
CMD05500 XMT      INNUM,IVR
          XMT      -1,AUX       * 1
          XOR      RB,R3       R3<- -INNUM-1
          XMT      SPDCONF1,IVR LOAD ADDRESS
          XMT      -1,AUX       *1 - CALCULATE LAST PAGE IN MEMORY
          ADD      32H,6,AUX
          XOR      R1,AUX
          NZT     AUX,CMD05510 SEE IF NOWPAGE = LASTPAGE
          XMT      EOLLO,IVR   IF NOT, BRANCH
          XMT      1,AUX       SET COUNT = EOLAD(L07BIT) - INNUM
          ADD      30H,7,R2    *1
          MOV      R3,AUX     R2<- EOLAD(L07BIT) + 1
          ADD      R2,R2     AUX<- -INNUM-1
          JMP     CMD05520    R2<- COUNT
*
CMD05510 XMT      128,R2      R2<- COUNT = 128
CMD05520 MOV      R1(1),R5    (R5,R6)<- PAGADR
          XMT      1000000B,AUX
          AND      R5,R6
    
```

```

3739 04015 6 00177
3740 04016 2 05005
3741 04017 6 00377
3742 04020 1 02000
3743 04021 1 06006
3744 04022 6 00377
3745 04023 3 03000
3746 04024 1 06004
3747 04025 0 10000
3748 04026 1 05003
3749 04027 0 02001
3750 04030 6 11062
      04031 7 06456
3751 04032 6 00001
3752 04033 1 06006
3753 04034 6 17265
3754 04035 6 07020
3755 04036 0 37001
3756 04037 0 01102
3757 04040 6 11063
      04041 7 06431
3758
3759
3760
3761
3762 04042 6 17263
3763 04043 6 07021
3764 04044 5 30106
3765 04045 7 04062
3766 04046 4 02047
3767 04047 7 04060
3768
3769
3770
3771
3772 04050 6 00130
3773 04051 6 00260
3774 04052 6 00010
3775 04053 6 00140
3776 04054 6 00270
3777 04055 6 00020
3778 04056 6 00150
3779 04057 6 00300
3780
3781 04060 6 11064
      04061 7 06503
3782
3783 04062 6 17263
3784 04063 6 07021
3785 04064 0 31101
3786 04065 5 01104
3787 04066 6 00100
3788 04067 1 37601
3789 04070 6 00377
3790 04071 1 01037
3791 04072 6 01000
3792 04073 6 00002
3793 04074 6 17262
3794 04075 6 07001
3795 04076 0 00037
3796 04077 6 11065
      04100 7 06214
3797 04101 5 01103
3798 04102 7 04732
3799
3800 04103 7 00453
3801
3802 04104 0 37601
3803 04105 6 00377
3804 04106 1 01037
3805 04107 6 11066
      04110 7 05103
3806 04111 5 01073
3807
3808 04112 6 17264
3809 04113 6 07020
3810 04114 0 37002
3811 04115 6 17263
3812
3812 04116 0 00000
3813 04117 0 37601
3814 04120 7 03627
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829

```

```

      XMT 01111111B,AUX
      AND R5,R5
      XMT -1,AUX
      ADD R2,AUX
      ADD R6,R6
      XMT -1,AUX
      XOR R3,AUX
      ADD R6,R4
      MOV OV1,AUX
      ADD R5,R3
      MOV R2,R1
      CALL INLOOP
      XMT 1,AUX
      ADD R6,R6
      XMT INNUM,IVR
      XMT IVISPD,IVL
      MOV RB,R1
      MOV R1(1),R2
      CALL NULLFILL
      ON RETURN,(R5,R6+1) = PAGADR
      SET (R5,R6) = PAGADR
      LOAD ADDRESS
      *1 - SELECT SPD READ
      R1 <- COUNT = INNUM
      R2 <- INNUM/2
      FILL WITH NULLS
*
* IF THIS IS THE FIRST ENTRY, LRCHKSUM = LRCHKSUM + INNUM/2*(NULL-EOL)
* IF NOT, THE CHECKSUM DOESN'T CHANGE
*
      XMT NOWPAGE,IVR
      XMT IVISPD+IVOSPD,IVL
      NZT ENT1STB,CMD0555D
      JMP CMD05700
      XMT CHKTABI-1(R2),8
      JMP CMD05650
      PICK UP THE FLAG
      *1 - SELECT SPD READ/WRITE
      JUMP IF NOT 1ST PAGE
      CALCULATE CHKPLUS
*
* THE FOLLOWING TABLE MUST BE CHANGED IF EOLNODE .NE. 1 OR IF
* NULLNODE .NE. 23!!!!!!!!!!!!
*
      XMT 130H,AUX
      XMT 260H,AUX
      XMT 010H,AUX
      XMT 140H,AUX
      XMT 270H,AUX
      XMT 020H,AUX
      XMT 150H,AUX
      XMT 300H,AUX
      1*(NULL-EOL)
      2*(NULL-EOL)
      3*(NULL-EOL)
      4*(NULL-EOL)
      5*(NULL-EOL)
      6*(NULL-EOL)
      7*(NULL-EOL)
      8*(NULL-EOL)
*
      XMT UPDTLCHK
      UPDATE THE CHECKSUM
*
      XMT NOWPAGE,IVR
      XMT IVISPD+IVOSPD,IVL
      MOV PASS1STB,R1
      NZT R1,CMD05750
      XMT PASS1STM,AUX
      ADD NOWPAGEB,R1
      XMT -1,AUX
      ADD R1,RB
      XMT 0,R1
      XMT INSTCONT,AUX
      XMT CMDCONT,IVR
      XMT IVOSPD,IVL
      MOV AUX,RB
      CALL CLRDIAG
      CHECK IF WE MOVED 2 PAGES THIS SWEEP
      *1 - SELECT SPD READ/WRITE
      READ 1ST PASS FLAG
      JUMP IF ANOTHER PAGE TO MOVE
      SET 1ST PASS FLAG, ZERO 1ST
      ENTRY FLAG,
      DECREMENT NOWPAGE
      CLEAR R1 TO DENOTE GOOD EXIT
      *1 AUX<- INSERT CONTINUE
      MUST BE HERE FOR COMMON ENTRY
      IF R1 .NE. 0, ERROR EXIT
      EXIT
*
      XMT EXEC
      ERROR EXIT
*
      XMT NOWPAGEB,R1
      XMT -1,AUX
      ADD R1,RB
      CALL INTRP
      READ NOWPAGE
      DEC NOWPAGE, ZERO BOTH FLAGS
      CALL INTERRUPT HANDLER
*
      XMT R1,CMD05705
      IF R1 .NE. 0, PROBLEMS
*
      XMT INPAGE,IVR
      XMT IVISPD,IVL
      MOV RB,R2
      XMT NOWPAGE,IVR
      NOP
      MOV AUX,AUX
      MOV NOWPAGEB,R1
      JMP CMD05100
      INSERT CONTINUES HERE
      *1 - SELECT SPD READ
      R2<- INPAGE
      *1 - WAIT
      R1<- NOWPAGE
      CONTINUE
*
***DELETE COMMAND
*
* PSEUDO-CODE DESCRIPTION
* SEE INSERT FOR DEFINITIONS
* ADDITIONAL DEFINITIONS:
* DLNUM - NUMBER OF BYTES TO DELETE
* DLSTAD - ADDRESS TO START DELETING
*
***INITIALIZE
*
* VALIDATE ADDRESS
* CHECK MEMORY PROTECT
*

```


3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885

```
* NOWPAGE = DLSTAD/128
* ENT1STB = 1
* PASS1STB = 1
* ADDR = DLSTAD
* CHKMINUS = 0
* COUNT = DLNUM
* DO WHILE COUNT .GT. 0
*   CHKMINUS = CHKMINUS + C(ADDR)
*   ADDR = ADDR + 1
*   COUNT = COUNT - 1
* TOADDR = DLSTAD
* FROMADDR = TOADDR + DLNUM
* COUNT = FROMADDR(MOD(128)) - 128
*
***CONTA - COME HERE AFTER DELETECONT
*
* DO WHILE COUNT .LT. 0
*   C(TOADDR) = C(FROMADDR)
*   TOADDR = TOADDR + 1
*   FROMADDR = FROMADDR + 1
*   COUNT = COUNT + 1
* TOADDR = PAGADR + 128 - DLNUM
* COUNT = -DLNUM
* IF NOWPAGE .EQ. EOLPAGE
*   EOLAD = EOLAD - DLNUM
*   VALUE = EOL
* ELSE
*   VALUE = NULL
* DO WHILE COUNT .LT. 0
*   C(TOADDR,TOADDR+1) = VALUE
*   TOADDR = TOADDR + 2
*   COUNT = COUNT + 2
* LRCHECKSUM = LRCHECKSUM + DLNUM/2 * VALUE - CHKMINUS
* IF NOWPAGE .EQ. EOLPAGE
*   JMP CMDRSP - DONE, BUILD RESPONSE
* ELSE
*   IF PASS1STB .EQ. 1
*     NOWPAGE = NOWPAGE + 1
*     PASS1STB = 0
*     ENT1STB = 0
*     CALL INTRP
*     GO TO DELETECONT
*   ELSE
*     PASS1STB = 1
*     ENT1STB = 0
*     NOWPAGE = NOWPAGE + 1
*     SET DFLETE CONTINUE
*     EXIT
*
***DELETECONT
*
* CHKMINUS = DLNUM/2 * NULL
* COUNT = -128
* FROMADDR = PAGADR
* TOADDR = FROMADDR - DLNUM
* GO TO CONTA
```

3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923

04121 6 11067
04122 7 06366
04123 6 07004
04124 0 03027
04125 6 07003
04126 0 04027
04127 6 05000
04130 6 11000
04131 0 02001
04132 0 37000
04133 6 07000
04134 0 11027
04135 1 05005
04136 6 00377
04137 1 01001
04140 5 01132
04141 0 02000
04142 6 17024
04143 6 07021
04144 0 05037
04145 1 04006
04146 0 10000
04147 1 03005
04150 6 00001
04151 2 06700
04152 1 05701
04153 6 17263
04154 6 00300
04155 1 01037
04156 6 00177
04157 2 06001
04160 6 00200
04161 1 01001

04162 6 00001
04163 6 07004

```
CMDD06000 CALL DLETINIT INITIALIZE FOR DELETE
CMDD06010 XMT IVDLRHI,IVL SELECT LOGIC ADDRHI
MOV R3,LB COMPUTE CHKMINUS, WHICH IS THE SUM
XMT IVDLRLO+IVILRDAT,IVL OF THE DATA WE ARE DELETING
MOV R4,LB
XMT 0,R5 *1 - R5<- CHKMINUS = 0
XMT CTRLINCL,R11 *2 - LOGIC INCREMENT VALUE -> R11
MOV R2,R1 *3 - R1<- COUNT = DLNUM
CMDD06050 MOV RB,AUX AUX<- DATA
XMT IVOCTRL+IVILRDAT,IVL SELECT CTRL AND LOGIC INPUT
MOV R11,LB INC LOGIC ADDR
ADD R5,R5 *1 - NEW CHKMINUS
XMT -1,AUX *2
ADD R1,R1 *3 - DECREMENT COUNT
NZT R1,CMDD06050
MOV R2,AUX AUX<- DLNUM
XMT SAVER2,IVR
XMT IVOSPD+IVISPD,IVL SELECT SPR READ/WRITE
MOV R5,RB SAVE CHKMINUS IN SAVER2
ADD R4,R6 (R3,R4) = TOADDR
MOV OV,AUX (R5,R6)<- FROMADDR = TOADDR + DLNUM
ADD R3,R5
XMT 0000001B,AUX CALCULATE NOWPAGE
AND R6(7),AUX
ADD R5(7),R1 R1<- NOWPAGE
XMT NOWPAGE,IVR SAVE NOWPAGE, AND SET 1ST ENTRY AND
XMT ENT1STM+PASS1STM,AUX 1ST PASS FLAGS
ADD R1,RB
XMT 01111111B,AUX COUNT = HOW MUCH PAGE IS LEFT
AND R6,R1 ABOVE DLEND (R6)
XMT -128,AUX
ADD R1,R1 R1<- COUNT
*
* MOVE DATA LOOP
*
CMDD06075 XMT 1,AUX SET AUX FOR INCING
DELOOP01 XMT IVDLRHI,IVL SELECT LOGIC ADDRHI
```


3924	04164	0	05027	MOV	R5, LB	FROMADDR HI	
3925	04165	6	07003	XMT	IVOLRLO+IVILRDAT, IVL	SELECT LOGIC ADDRLO	
3926	04166	0	06027	MOV	R6, LB	FROMADDR LO	
3927	04167	1	06006	ADD	R6, R6	*1 - INC FROMADDR (CAN'T OVERFLOW)	
3928	04170	1	01001	ADD	R1, R1	*2 - DEC LOOP COUNT	
3929				NOP		*3 - WAIT	
3929	04171	0	00000	+	MOV	AUX, AUX	
3930	04172	0	37002	MOV	RB, R2	R2<- DATA	
3931	04173	0	04027	MOV	R4, LB	TOADDR LO	
3932	04174	6	07004	XMT	IVOLRHI, IVL	SELECT LOGIC ADDRHI	
3933	04175	0	03027	MOV	R3, LB	TOADDR HI	
3934	04176	1	04004	ADD	R4, R4	*1 - INC TOADDR LO	
3935	04177	5	04201	-	NZT	R4, DELOOPD2	*2 - CHECK FOR OVERFLOW
3936	04200	1	03003	ADD	R3, R3	*3 - IF SO, INC TOADDR HI	
3937	04201	6	07011	DELOOPD2	XMT	IVOLRDAT, IVL	SELECT LOGIC WRITE
3938	04202	0	02027	MOV	R2, LB	WRITE OUT DATA	
3939	04203	5	01163	NZT	R1, DELOOPD1	LOOP ON COUNT	
3940				*			
3941	04204	6	17265	XMT	DLNUM, IVR	PICK UP DLNUM	
3942	04205	6	07021	XMT	IVISPD+IVOSPD, IVL	SELECT SPD READ/WRITE	
3943	04206	0	37001	MOV	RB, R1	R1<- DLNUM	
3944	04207	6	00377	XMT	-1, AUX		
3945	04210	1	01002	ADD	R1, R2	R2<- DLNUM - 1	
3946	04211	3	02000	XOR	R2, AUX	AUX<- -DLNUM	
3947	04212	1	06006	ADD	R6, R6	(R5, R6)<- ADDR FOR EOLFILL OR NULLFILL	
3948	04213	6	17065	XMT	EOLHI, IVR	LOAD ADDRESS	
3949	04214	0	01102	MOV	R1(1), R2	*1 - R2<- DLNUM/2	
3950	04215	0	37003	MOV	RB, R3	R3<- EOLLOCHI	
3951	04216	6	17066	XMT	EOLLO, IVR		
3952				NOP		*1 - WAIT	
3952	04217	0	00000	+	MOV	AUX, AUX	
3953	04220	0	30100	MOV	37H, 1, AUX	AUX<- HI BIT OF EOLLOCLO	
3954	04221	1	03703	ADD	R3(7), R3	R3<- EOLPAGE	
3955	04222	6	17263	XMT	NOWPAGE, IVR	PICK UP NOWPAGE	
3956				NOP		*1 - WAIT	
3956	04223	0	00000	+	MOV	AUX, AUX	
3957	04224	0	37600	MOV	NOWPAGEB, AUX	DON'T READ FLAGS	
3958	04225	3	03003	XOR	R3, R3	SEE IF NOWPAGE = EOLPAGE	
3959	04226	6	17025	XMT	SAVER3, IVR	SAVE ANSWER	
3960	04227	0	03037	MOV	R3, RB		
3961	04230	5	03260	NZT	R3, CMD06100	JUMP IF NOT EOLPAGE	
3962	04231	6	17066	XMT	EOLLO, IVR	UPDATE EOLAD	
3963	04232	0	02704	MOV	R2(7), R4	*1 - R4<- DLNUM	
3964	04233	6	00377	XMT	-1, AUX		
3965	04234	3	04004	XOR	R4, R4	R4<- -DLNUM - 1	
3966	04235	6	00001	XMT	1, AUX		
3967	04236	1	04000	ADD	R4, AUX	AUX<- -DLNUM	
3968	04237	1	37037	ADD	RB, RB	UPDATE EOLLOCLO	
3969	04240	5	10244	NZT	OVF, CMD06090	CHECK FOR UNDERFLOW	
3970	04241	6	17065	CMDD06080	XMT	EOLHI, IVR	
3971	04242	6	00377	XMT	-1, AUX	*1	
3972	04243	1	37037	ADD	RB, RB		
3973	04244	6	11070	CMDD06090	CALL	EOLFILL	FILL ENDOF PAGE WITH EOL'S
	04245	7	06433				
3974	04246	4	02247	XEC	EOLTABD-1(R2), 8	CALCULATE CHKPLUS + 1	
3975	04247	7	04274	JMP	CMDD6200		
3976				*			
3977				*		*THE FOLLOWING TABLE MUST BE CHANGED IF EOLNODE .NE. 1!!!!!!	
3978				*			
3979	04250	6	03005	EOLTABD	XMT	5H, R3	1*EOL + 1
3980	04251	6	03011	XMT	11H, R3	2*EOL + 1	
3981	04252	6	03015	XMT	15H, R3	3*EOL + 1	
3982	04253	6	03021	XMT	21H, R3	4*EOL + 1	
3983	04254	6	03025	XMT	25H, R3	5*EOL + 1	
3984	04255	6	03031	XMT	31H, R3	6*EOL + 1	
3985	04256	6	03035	XMT	35H, R3	7*EOL + 1	
3986	04257	6	03041	XMT	41H, R3	8*EOL + 1	
3987				*			
3988	04260	6	11071	CMDD06100	CALL	NULLFILL	FILL ENDOFPAGE WITH NULLS
	04261	7	06431				
3989	04262	4	02263	XEC	NULLTABD-1(R2), 8	CALCULATE CHKPLUS+1	
3990	04263	7	04274	JMP	CMDD6200		
3991				*			
3992				*		* THE FOLLOWING TABLE MUST BE CHANGED IF NULLNODE .NE. 23!!!!!!	
3993				*			
3994	04264	6	03135	NULLTABD	XMT	135H, R3	1*NULL + 1
3995	04265	6	03271	XMT	271H, R3	2*NULL + 1	
3996	04266	6	03025	XMT	025H, R3	3*NULL + 1	
3997	04267	6	03161	XMT	161H, R3	4*NULL + 1	
3998	04270	6	03315	XMT	315H, R3	5*NULL + 1	
3999	04271	6	03051	XMT	051H, R3	6*NULL + 1	
4000	04272	6	03205	XMT	205H, R3	7*NULL + 1	
4001	04273	6	03341	XMT	341H, R3	8*NULL + 1	
4002				*			
4003	04274	6	17024	CMDD06200	XMT	SAVER2, IVR	SAVER2 CONTAINS CHKMINUS
4004	04275	6	07021	XMT	IVISPD+IVOSPD, IVL	*1 - DO SELECTS	
4005	04276	6	00377	XMT	-1, AUX		
4006	04277	3	37000	XOR	RB, AUX	AUX<- -CHKMINUS - 1	
4007	04300	1	03000	ADD	R3, AUX	AUX<- CHKPLUS - CHKMINUS	
4008	04301	6	11072	CALL	UPDTLCHK	UPDATE THE CHECKSUM	
	04302	7	06503				
4009	04303	6	17025	XMT	SAVER3, IVR	IF SAVER3 = 0, WE ARE DONE	
4010	04304	6	07021	XMT	IVISPD+IVOSPD, IVL	*1	
4011	04305	5	37007	NZT	RB, CMD06250		
4012	04306	7	03770	JMP	CMDD05410	GO TO COMMON CODE	
4013				*			

4014	04307	6	17263	CMD06250	XMT	NOWPAGE,IVR	SEE IF WE HAVE MOVED 2 PAGES
4015	04310	6	00101		XMT	PASS1STM+1,AUX	*1 - THIS PASS
4016	04311	5	31126		NZT	PASS1STB,CMD06300	JUMP IF NOT
4017	04312	1	37037		ADD	RB,RB	INC NOWPAGE, SET 1ST PASS FLAG, CLR 1ST ENTRY
4018	04313	6	01000		XMT	0,R1	SHOW GOOD EXIT
4019	04314	6	00003	CMD06275	XMT	DLETCONT,AUX	AUX = DELCONT
4020	04315	7	04074		JMP	CMD05710	GO TO COMMON CODE
4021				*			
4022	04316	6	03134	CHKTABD2	XMT	134H,R3	1*NULL
4023	04317	6	03270		XMT	270H,R3	2*NULL
4024	04320	6	03024		XMT	024H,R3	3*NULL
4025	04321	6	03160		XMT	160H,R3	4*NULL
4026	04322	6	03314		XMT	314H,R3	5*NULL
4027	04323	6	03050		XMT	050H,R3	6*NULL
4028	04324	6	03204		XMT	204H,R3	7*NULL
4029	04325	6	03340		XMT	340H,R3	8*NULL
4030				*			
4031	04326	0	37601	CMD06300	MOV	NOWPAGEB,R1	READ NOWPAGE
4032	04327	6	00001		XMT	1,AUX	
4033	04330	1	01037		ADD	R1,RB	INC NOWPAGE, CLEAR FLAGS
4034	04331	6	11073		CALL	INTRP	CALL INTERRUPT HANDLER
	04332	7	05103				
4035	04333	5	01314		NZT	R1,CMD06275	ERROR IF R1 .NE. 0
4036				*			
4037				* DELETE CONTINUES HERE			
4038				*			
4039	04334	6	17265	CMD06400	XMT	DLNUM,IVR	
4040	04335	6	07021		XMT	IVISPD+IVOSPD,IVL	DO SELECTS
4041	04336	0	36702		MOV	31H,7,R2	R2<- DLNUM/2
4042	04337	4	02315		XEC	CHKTABD2-1(R2),8	FAKE OUT OTHER ROUTINE
4043	04340	6	17263		XMT	NOWPAGE,IVR	PICK UP NOWPAGE
4044	04341	6	00200		XMT	10000000B,AUX	*1 - FOR ANDING LATER
4045	04342	0	37601		MOV	30H,6,R1	R1<- NOWPAGE
4046	04343	6	17024		XMT	SAVER2,IVR	BY SETTING CHKMINUS
4047	04344	0	03037		MOV	R3,RB	
4048	04345	2	01706		AND	R1(7),R6	(R5,R6) <- FROMADDR = PAGADR
4049	04346	6	00177		XMT	01111111B,AUX	
4050	04347	2	01105		AND	R1(1),R5	
4051	04350	6	00377		XMT	-1,AUX	(R3,R4)<- TOADDR = FROMADDR - DLNUM
4052	04351	3	02703		XOR	R2(7),R3	
4053	04352	6	00001		XMT	1,AUX	
4054	04353	1	03000		ADD	R3,AUX	AUX<- -DLNUM
4055	04354	1	06004		ADD	R6,R4	
4056	04355	6	00377		XMT	-1,AUX	
4057	04356	1	10000		ADD	0VF,AUX	
4058	04357	1	05003		ADD	R5,R3	
4059	04360	6	01200		XMT	-128,R1	R1<- COUNT = -128
4060	04361	7	04162		JMP	CMD06075	
4061				*			
4063				*			
4064				***LED COMMAND			
4065				*			
4066	04362	6	11074	CMD07000	CALL	LENZERO	CHECK THAT CMD-LEN = 0
	04363	7	06205				
4067	04364	6	17235		XMT	CMD04,IVR	LOAD ROW/COL ADDR
4068	04365	6	07021		XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ/WRITE
4069	04366	0	37003		MOV	RB,R3	R3 <- ROWCOL FOR LED
4070	04367	6	17063		XMT	LEDLOC,IVR	
4071	04370	0	03037		MOV	R3,RB	SET LEDLOC
4072	04371	7	04660		JMP	CMDRSP	
4074				*			
4075				***STOP COMMAND			
4076				*			
4077	04372	6	11075	CMD08000	CALL	LENZERO	CHECK THAT CMD-LEN = 0
	04373	7	06205				
4078	04374	6	11076		CALL	PROTECT	
	04375	7	06541				
4079	04376	6	02020		XMT	SYSSTOPH,R2	R2 <- STOP STATE
4080				*			
4081				CMD08010	WSP	SAVSTATE,R2	SAVE STATE VECTOR
4081	04377	6	07021		XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
4081	04400	6	17033		XMT	SAVSTATE,IVR	LOAD ADDRESS
4081	04401	0	02037		MOV	R2,RB	WRITE DATA
4082	04402	7	04660		JMP	CMDRSP	DO RESPONSE
4084				*			
4085				***GO COMMAND			
4086				*			
4087	04403	6	11077	CMD09000	CALL	LENZERO	CHECK THAT CMD-LEN = 0
	04404	7	06205				
4088	04405	6	11100		CALL	PROTECT	
	04406	7	06541				
4089	04407	6	17275		XMT	SYSSTATE,IVR	LOAD SYSTEM STATE ADDRESS
4090	04410	6	07020		XMT	IVISPD,IVL	SELECT SCRATCHPAD READ
4091	04411	5	33115		NZT	SYSSTOPB,CMD09020	BRANCH ON STOP STATE
4092	04412	5	37415		NZT	SYSSTATEB,CMD09020	BRANCH ON ERROR STATE
4093				*			
4094	04413	6	01014	CMD09010	XMT	ERRSTP,R1	R1 <- ERROR CODE
4095	04414	7	04646		JMP	CMDERR	GO TO ERROR CODE
4096				*			
4097	04415	6	01002	CMD09020	XMT	ASCSTX,R1	
4098	04416	6	02104		XMT	XMITBLK,R2	BUILD RESPONSE
4099	04417	6	11101		CALL	BFCB	
	04420	7	05526				
4100	04421	6	01220		XMT	GOCMD,R1	COMMAND
4101	04422	6	11102		CALL	BFCB	
	04423	7	05526				

4102	04424	6	01004	XMT	4,R1	LENGTH OF RESPONSE	
4103	04425	6	11103	CALL	BFCH		
	04426	7	05526				
4104	04427	6	01225	XMT	ASCSTX+GOCMD+4-1,R1	CHECKSUM	
4105	04430	6	11104	CALL	BFCH		
	04431	7	05526				
4106				CMD09025	RSP	XMITCNT,R1	AS SOON AS RESPONSE IS DONE, GO TO PWRDN
4106	04432	6	17110	+	XMT	XMITCNT,IVR	LOAD ADDRESS
4106	04433	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
4106	04434	0	37001	+	MOV	RB,R1	READ DATA
4107	04435	5	01037		NZT	R1,CMD09030	
4108	04436	7	00430		JMP	PWRDN	
4109	04437	6	11105	CMD09030	CALL	INTRP	SEND ANOTHER RESPONSE CHAR
	04440	7	05103				
4110	04441	7	04432		JMP	CMD09025	
4112				*			
4113				***INITIALIZE	COMMAND		
4114				*			
4115	04442	6	11106	CMD10000	CALL	PROTECT	CHECK MEMORY PROTECT
	04443	7	06541				
4116	04444	6	11107		CALL	LENZERO	CHECK THAT CMD-LEN = 0
	04445	7	06205				
4117	04446	6	17275		XMT	SYSSTATE,IVR	LOAD STATE VECTOR ADDRESS
4118	04447	6	07020		XMT	IVISPD,IVL	SELECT SCRATCHPAD READ
4119				*			
4120					ORG	5,32	CONDITIONAL ORG FOR BRANCHES
4121				*			
4122	04450	5	33113		NZT	SYSSTOPB,CMD10010	BRANCH ON STOP STATE
4123	04451	5	37413		NZT	SYSCODEB,CMD10010	BRANCH ON ERROR STATE
4124	04452	7	04413		JMP	CMD09010	BRANCH TO ERROR HANDLER
4125				*			
4126	04453	6	17276	CMD10010	XMT	SPDCONF1,IVR	LOAD ADDRESS
4127	04454	6	07024		XMT	IVISPD+IVOLRHI,IVL	*1 - SELECT PORTS
4128	04455	0	37001		MOV	RB,R1	R1 <- LOGIC RAM CONFIGURATION
4129	04456	6	02000		XMT	SYSUSERH,R2	
4130	04457	0	02027		MOV	R2,LB	
4131	04460	6	02002		XMT	SYSUSERL,R2	
4132	04461	6	07003		XMT	IVOLRLO,IVL	
4133	04462	0	02027		MOV	R2,LB	
4134	04463	0	01301		MOV	R1(3),R1	
4135	04464	6	03004		XMT	NODEEOL.L.2,R3	
4136	04465	6	04000		XMT	0,R4	
4137	04466	6	11377		XMT	-1,R11	
4138				*			
4139	04467	6	00002	CMD10020	XMT	2,AUX	
4140	04470	6	07011		XMT	IVOLRDAT,IVL	
4141	04471	0	03027		MOV	R3,LB	
4142	04472	6	07000		XMT	IVOCTRL,IVL	
4143	04473	6	27300		XMT	CTRLINCL,CTRLREG	
4144	04474	6	07011		XMT	IVOLRDAT,IVL	
4145	04475	0	04027		MOV	R4,LB	
4146	04476	6	07000		XMT	IVOCTRL,IVL	
4147	04477	6	27300		XMT	CTRLINCL,CTRLREG	
4148	04500	1	02002		ADD	R2,R2	
4149	04501	0	11000		MOV	R11,AUX	
4150	04502	1	03011		ADD	R3,R11	
4151	04503	5	02067		NZT	R2,CMD10020	
4152	04504	6	00377		XMT	-1,AUX	
4153	04505	1	01001		ADD	R1,R1	
4154	04506	5	01067		NZT	R1,CMD10020	
4155	04507	6	01000		XMT	SYSLRCHH,R1	
4156	04510	6	07004		XMT	IVOLRHI,IVL	
4157	04511	0	01027		MOV	R1,LB	
4158	04512	6	01000		XMT	SYSLRCHL,R1	
4159	04513	6	07003		XMT	IVOLRLO,IVL	
4160	04514	0	01027		MOV	R1,LB	
4161	04515	6	07011		XMT	IVOLRDAT,IVL	
4162	04516	0	11027		MOV	R11,LB	
4163	04517	6	11110		CALL	CLPDIAG	CLEAR DIAGNOSTICS
	04520	7	06214				
4164	04521	6	02000		XMT	SYSUSERH,R2	SET EOL ADDRESS IN SPD
4165	04522	6	17065		XMT	EOLHI,IVR	
4166	04523	0	02037		MOV	R2,RB	
4167	04524	6	02002		XMT	SYSUSERL,R2	*1
4168	04525	6	17066		XMT	EOLLO,IVR	
4169	04526	0	02037		MOV	R2,RB	
4170				*			
4171				***CLEAR	COIL RAM		
4172				*			
4173					CLR	R1	R1 <- 0
4173	04527	6	01000	+	XMT	0,R1	
4174	04530	6	07001		XMT	IVOCRHI,IVL	SELECT COIL ADDRHI
4175	04531	0	01027		MOV	R1,LB	LOAD ADDRESS
4176	04532	6	07000		XMT	IVOCRLO,IVL	SELECT COIL ADDRLO
4177	04533	0	01037		MOV	R1,RB	LOAD ADDRESS
4178	04534	6	02004		XMT	4,R2	R2 <- COUNTER
4179					CLR	R3	R3 <- 0
4179	04535	6	03000	+	XMT	0,R3	
4180	04536	6	00377		XMT	-1,AUX	AUX <- DECREMENT
4181				*			
4182	04537	6	07002	CMD10030	XMT	IVOCRDAT,IVL	SELECT COIL DATA OUT
4183	04540	0	03027		MOV	R3,LB	CLEAR LOCATION
4184	04541	6	07000		XMT	IVOCTRL,IVL	SELECT CONTROL
4185	04542	6	27301		XMT	CTRLINCC,CTRLREG	
4186	04543	1	01001		ADD	R1,R1	DECREMENT COUNTER
4187	04544	5	01137		NZT	R1,CMD10030	LOOP UNTIL DONE

4188	04545	1	02002	ADD	R2,R2	DECREMENT LOOP COUNTER
4189	04546	5	02137	NZT	R2,CMD10030	LOOP UNTIL DONE
4190				*		
4191	04547	7	04660	JMP	CMDRSP	
4193				*		
4194				***INSERT AT END OF COLUMN		
4195				*		
4196	04550	6	11111	CMD11000	CALL INSTINIT	INITIALIZE FOR INSERT
	04551	7	06224			
4197	04552	6	17030	XMT	SAVER6,IVR	SAVER5,SAVER6 CONTAIN THE INSTAD
4198	04553	6	07020	XMT	IVISPD,IVL	*1 - SELECT SPD READ
4199	04554	6	00376	XMT	-2,AUX	BACK UP ONE NODE
4200	04555	1	37006	ADD	RB,R6	
4201	04556	6	17027	XMT	SAVER5,IVR	LOAD ADDRESS
4202	04557	6	00377	XMT	-1,AUX	*1
4203	04560	0	37005	MOV	RB,R5	
4204	04561	5	10163	NZT	OVF,CMD11010	
4205	04562	1	05005	ADD	R5,R5	
4206	04563	6	07004	CMD11010	XMT IVOLRHI,IVL	SELECT LOGIC ADDRHI
4207	04564	0	05027	MOV	R5,LB	LOAD ADDRESS
4208	04565	6	07003	XMT	IVOLRLO,IVL	SELECT LOGIC ADDRLO
4209	04566	0	06027	MOV	R6,LB	LOAD ADDRESS
4210	04567	6	00177	XMT	-1-NODEEOCM,AUX	*1 - AUX <- MASK
4211	04578	6	07011	XMT	IVILRDAT+IVOLROAT,IVL	*2 - SELECT PORTS
4212	04571	0	01003	MOV	R1,R3	*3 - SAVE R1
4213	04572	0	02004	MOV	R2,R4	SAVE R2
4214	04573	2	37001	AND	RB,R1	CLEAR EOC BIT
4215	04574	6	11112	CALL	WRTUP	WRITE IT OUT, UPDATE CHKSUM
	04575	7	05456			
4216	04576	0	03001	MOV	R3,R1	RESTORE R1
4217	04577	0	04002	MOV	R4,R2	RESTORE R2
4218	04600	7	03627	JMP	CMD05100	GO TO COMMON CODE
4220				*		
4221				***DELETE AT END-OF-COLUMN		
4222				*		
4223	04601	6	11113	CMD12000	CALL DLETINIT	INITIALIZE FOR DELETE
	04602	7	06366			
4224				*		
4225				* (R3,R4) = DLSTAD ON RETURN		
4226				*		
4227				CMD12010	WSP SAVER2,R2	
4227	04603	6	07021	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
4227	04604	6	17024	XMT	SAVER2,IVR	LOAD ADDRESS
4227	04605	0	02037	MOV	R2,RB	WRITE DATA
4228	04606	6	07003	XMT	IVOLRLO,IVL	READ IN THE NODE WE ARE GOING
4229	04607	6	00377	XMT	-1,AUX	TO TURN ON THE END OF COLUMN
4230	04610	1	04027	ADD	R4,LB	ON IN
4231	04611	1	10000	ADD	OVF,AUX	
4232	04612	6	07004	XMT	IVOLRHI,IVL	SELECT LOGIC ADDRHI
4233	04613	1	03027	ADD	R3,LB	
4234	04614	6	00376	XMT	-2,AUX	*1 - SET R6 = PREVIOUS NODE
4235	04615	1	04006	ADD	R4,R6	*2 - ADDRESS
4236				NOP		*3 - WAIT
4236	04616	0	00000	MOV	AUX,AUX	
4237	04617	0	37002	MOV	RB,R2	R2<- DATA0
4238	04620	6	07003	XMT	IVOLRLO,IVL	
4239	04621	0	06027	MOV	R6,LB	SET LOGIC ADDRLO
4240	04622	6	00177	XMT	-1-NODEEOCM,AUX	*1 - SET MASK
4241	04623	6	07000	XMT	IVILRDAT,IVL	*2 - SELECT LOGIC READ
4242				NOP		*3 - WAIT
4242	04624	0	00000	MOV	AUX,AUX	
4243	04625	2	37001	AND	RB,R1	
4244	04626	6	00200	XMT	NODEEOCM,AUX	
4245	04627	3	01001	XOR	R1,R1	R1<- REPLACEMENT DATAHI
4246	04630	6	11114	CALL	VALIDATE	VALIDATE NEW DATA
	04631	7	06702			
4247	04632	6	00377	XMT	-1,AUX	IF R1 .EQ. -1, ERROR
4248	04633	3	01000	XOR	R1,AUX	ELSE, R1 = DATAHI
4249	04634	5	00237	NZT	AUX,CMD12030	
4250	04635	6	01012	XMT	ERRNOD,R1	SET ERROR CODE
4251	04636	7	04646	JMP	CMDERR	EXIT
4252	04637	6	11115	CMD12030	CALL WRTUP	WRITE OUT DATA
	04640	7	05456			
4253				RSP	SAVER2,R2	RESTORE R2
4253	04641	6	17024	XMT	SAVER2,IVR	LOAD ADDRESS
4253	04642	6	07021	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ
4253	04643	0	37002	MOV	RB,R2	READ DATA
4254	04644	7	04123	JMP	CMD06010	GO TO COMMON CODE
4256				*		
4257				***UNIMPLEMENTED COMMANDS		
4258				*		
4259		004645		CMD00000	EQU *	
4260		004645		CMD13000	EQU *	
4261		004645		CMD14000	EQU *	
4262	04645	6	01006	CMD15000	XMT ERRCMD,R1	R1 <- ERROR CODE
4263				*		
4264		004646		CMDERR	EQU *	
4265	04646	6	17235	XMT	CMD04,IVR	ERROR ADDRESS
4266	04647	6	07001	XMT	IVOSPD,IVL	SELECT SPD WRITE
4267	04650	0	01037	MOV	R1,RB	
4268	04651	6	01320	CMDNAK00	XMT ASCNAK,R1	*1 - NAKCMD
4269	04652	6	07001	XMT	IVOSPD,IVL	FOR ENTRY AT CMD13000
4270	04653	6	17233	XMT	CMD02,IVR	FCN ADDR
4271	04654	0	01037	MOV	R1,RB	
4272	04655	6	01005	XMT	S,R1	*1 - NAKLEN
4273	04656	6	17234	XMT	CMD03,IVR	
4274	04657	0	01037	MOV	R1,RB	

```

4276
4277
4278
4279 04660 6 07021
4280 04661 6 17262
4281 04662 6 00000
4282 04663 0 00037
4283 04664 6 03002
4284 04665 6 17232
4285 04666 0 03037
4286 04667 6 11377
4287 04670 6 17234
4288 04671 6 00377
4289 04672 1 37002
4290 04673 0 02005
4291 04674 6 06232
4292
4293 04675 0 06017
4294 04676 0 11000
4295 04677 1 37011
4296 04700 6 00001
4297 04701 1 06006
4298 04702 6 00377
4299 04703 1 02002
4300 04704 5 02275
4301
4302 04705 0 06017
4303 04706 0 11037
4304
4305 04707 6 00001
4306 04710 1 05005
4307 04711 6 06232
4308 04712 0 06017
4309 04713 6 02104
4310 04714 0 37001
4311 04715 6 11116
      04716 7 05526
4312 04717 6 00377
4313 04720 1 05005
4314 04721 6 00001
4315 04722 1 06006
4316 04723 5 05312
4317
4318
4318 04724 6 17033
4318 04725 6 07021
4318 04726 0 37001
4319 04727 5 01331
4320 04730 7 04732
4321
4322 04731 7 00453
4323
4324      004732
4326
4327
4328
4329
4330
4331 04732 6 11117
      04733 7 05103
4332 04734 5 01364
4333 04735 6 17034
4334 04736 6 07024
4335 04737 0 37001
4336 04740 6 17035
4337 04741 0 01027
4338 04742 0 37002
4339 04743 6 07003
4340 04744 0 02027
4341 04745 5 01365
4342 04746 5 02365
4343 04747 6 07000
4344 04750 6 27300
4345 04751 6 03377
4346 04752 6 27300
4347 04753 6 04002
4348 04754 6 17276
4349 04755 6 07021
4350 04756 0 37001
4351 04757 6 00037
4352 04760 6 17037
4353 04761 2 01301
4354 04762 0 01037
4355 04763 7 04771
4356
4357 04764 7 00453
4358
4359 04765 6 17036
4360 04766 6 07020
4361 04767 0 37003
4362
4362 04770 6 04000
4363
4364 04771 6 17276
4365 04772 6 07020
4366 04773 6 05001

```

```

*
***BUILD AND TRANSMIT RESPONSE
*
CMDRSP XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
XMT CMDCONT,IVR CLEAR CONTINUE FLAGS
XMT 0,AUX
MOV AUX,RB
XMT ASCSTX,R3 *1 - R3<- STX CHAR
XMT CMD01,IVR
MOV R3,RB SET MESSAGE START
XMT -1,R11 *1 - R11<- CHECKSUM SEED
XMT CMD03,IVR LENGTH BYTE
XMT -1,AUX *1
ADD RB,R2 R2<- MESSAGE LENGTH (NOT CHECKSUM)
MOV R2,R5 SAVE IT IN R5
XMT CMD01,R6 R6 <- ADDRESS
*
CMDRSP10 MOV R6,IVR LOAD ADDRESS
MOV R11,AUX AUX <- CHKSUM
ADD RB,R11 R11 <- UPDATED CHKSUM
XMT 1,AUX AUX <- INCREMENT
ADD R6,R6 BUMP POINTER
XMT -1,AUX AUX <- DECREMENT
ADD R2,R2 DECREMENT COUNTER
NZT R2,CMDRSP10 LOOP UNTIL DONE
*
MOV R6,IVR LOAD ADDRESS
MOV R11,RB LOAD CHKSUM
*
XMT 1,AUX *1 - AUX <- INCREMENT
ADD R5,R5 BUMP COUNT
XMT CMD01,R6 *1 - R6 <- INITIAL ADDRESS
CMDRSP20 MOV R6,IVR LOAD ADDRESS
XMT XMITBLK,R2 R2 <- BLOCK ADDRESS
MOV RB,R1 R1 <- CHARACTER
CALL BFCN BUFFER CHARACTER
XMT -1,AUX AUX <- DECREMENT
ADD R5,R5 DECREMENT COUNTER
XMT 1,AUX
ADD R6,R6 INC ADDRESS
NZT R5,CMDRSP20 LOOP ON COUNT
*
CMDRSP40 RSP SAVSTATE,R1 R1 <- STATE CHANGE (IF ANY)
+ XMT SAVSTATE,IVR LOAD ADDRESS
+ XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
+ MOV RB,R1 READ DATA
NZT R1,CMDRSP50 BRANCH ON STATE CHANGE
JMP CMDSX GO TO EXIT
*
CMDRSP50 JMP EXEC CHANGE STATES
*
CMDSX EQU * EXIT COMMAND MODULE
*
*
DIAGNOSTICS MODULE
*
*
PERFORMS CHEXSUM ON LOGIC RAM
*
DIAGS000 CALL INTRP CALL INTERRUPT HANDLER
NZT R1,DIAGS070 IF R1 .NE. 0, PROBLEMS
XMT DIAGSHI,IVR GET STARTING CHEXSUM HI
XMT IVOLRHI+IVISPD,IVL SELECT LOGIC ADDR HI,
MOV RB,R1
XMT DIAGSLO,IVR GET CHEX ADDR LO
MOV R1,LB *1 LOAD LOGIC ADDR HI
MOV RB,R2
XMT IVOLRLO,IVL SELECT LOGIC RAM LO
MOV R2,LB
NZT R1,DIAGS010 *1
NZT R2,DIAGS010 *2IF [R1,R2].NE.0 THEN GOTO DIAGS010
DIAGS005 XMT IVOCTRL,IVL *3 ELSE, SET LOGIC ADDR TO BEGIN+1
XMT CTRLINCL,CTRLREG INC LOGIC ADDR
XMT 11111111B,R3 R3 <- CHECKSUM SEED
XMT CTRLINCL,CTRLREG BUMP ADDRESS BEYOND CHECKSUM NODE
XMT 2,R4 SKIP LOGIC CHECKSUM BYTE
XMT SPDCONF1,IVR LOAD SCRATCHPAD ADDRESS
XMT IVISPD+IVOSPD,IVL SELECT SCRATCHPAD READ/WRITE
MOV RB,R1 R1 <- SYSTEM CONFIGURATION
XMT 00011111B,AUX AUX <- MASK
XMT DIAGCTR,IVR LOAD SCRATCHPAD ADDRESS
AND R1(3),R1 R1 <- NUMBER OF 256-BYTE BLOCKS
MOV R1,RB LOAD COUNTER
JMP DIAGS020
*
DIAGS070 JMP EXEC EXIT TO EXECUTIVE
*
DIAGS010 XMT DIAGCHK,IVR GET PARTIAL CHEXSUM
XMT IVISPD,IVL SELECT SCRATCH PAD READ
MOV RB,R3
CLR R4
XMT 0,R4
*
DIAGS020 XMT SPDCONF1,IVR LOAD SCRATCHPAD ADDRESS
XMT IVISPD,IVL SELECT PORT
XMT 1,R5 R5 <- LOOP COUNTER

```



```

4367 04774 5 34136
4368 04775 6 05002
4369
4370 04776 6 07000
4372 04777 0 37000
4373 05000 6 27300
4374 05001 1 03003
4375 05002 6 00001
4376 05003 1 04004
4377 05004 5 04377
4378 05005 6 00377
4379 05006 6 17037
4380 05007 6 07021
4381 05010 1 37037
4382 05011 1 05005
4383 05012 5 05376
4384 05013 0 37000
4385 05014 5 00042
4386 05015 6 01000
4387 05016 6 07004
4388 05017 0 01027
4389 05020 6 07003
4390 05021 6 01000
4391 05022 0 01027
4392 05023 6 07000
4393
4393 05024 6 05000
4394 05025 0 03000
4395 05026 3 37000
4396 05027 6 01002
4397 05030 5 00032
4398
4398 05031 6 01000
4399 05032 6 07001
4400 05033 6 17034
4401 05034 0 05037
4402
4402 05035 0 00000
4403 05036 6 17035
4404 05037 0 05037
4405 05040 5 01364
4406 05041 7 05051
4407
4408
4409 05042 6 17034
4410 05043 6 00002
4411 05044 1 37037
4412
4412 05045 0 00000
4413 05046 6 17036
4414 05047 0 03037
4415
4415 05050 0 00000
4416
4417 005051
4419
4420
4421
4422
4422 05051 6 17015
4422 05052 6 07021
4422 05053 0 37001
4423
4423 05054 6 07021
4423 05055 6 17016
4423 05056 0 01037
4424
4424 05057 0 00000
4425
4425 05060 6 17021
4425 05061 6 07021
4425 05062 0 37002
4426 05063 6 04366
4427 05064 6 05017
4428 05065 6 11120
4428 05066 7 05627
4429
4429 05067 6 17022
4429 05070 6 07021
4429 05071 0 37002
4430 05072 6 04234
4431 05073 6 05020
4432 05074 6 11121
4432 05075 7 05627
4433
4433 05076 6 01000
4434
4434 05077 6 07021
4434 05100 6 17015
4434 05101 0 01037
4435 05102 7 00504
4437 05103
4438
4439
4440
4441

```

```

NMT SYS0256B,DIAGS030 256-BYTE MEMORIES => ONE LOOP
XMT Z,R5 ALL OTHERS NEED TWO LOOPS
*
DIAGS030 XMT IVOCTRL+IVILRDAT,IVL SELECT CONTROL PULSE,LOGIC DAA
DIAGS040 MOV RB,AUX GET LOGIC RAM DATA
XMT CTRLINCL,CTRLREG INCR ADDR
ADD R3,R3 *1 ADD TO CHEXSUM
XMT 1,AUX *2
ADD R4,R4 *3 CHECK IF DONE PORTION
NMT R4,DIAGS040 IF .NOT.DONE 256 BYTE PORTION, LOOP
XMT -1,AUX ELSE, TEST IF MORE TO DO
XMT DIAGCTR,IVR LOAD SCRATCHPAD ADDRESS
XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
ADD RB,RB DECREMENT COUNTER
ADD R5,R5 DECREMENT COUNT
NMT R5,DIAGS030 R5.NE.0 => CONTINUE LOOP
MOV RB,AUX AUX <- COUNT (SHORT BRANCH PROBLEM)
NMT AUX,DIAGS100 AUX.NE.0 => NOT DONE YET
XMT SYSLRCHM,R1 DIAGSCTR.EQ.0 => DONE, FETCH CHECKSUM
XMT IVOLRHI,IVL SELECT LOGIC ADDRESS HIGH
MOV R1,LB LOAD ADDRESS
XMT IVOLRLO,IVL SELECT LOGIC ADDRESS LOW
XMT SYSLRCHL,R1 R1 <- ADDRESS
MOV R1,LB LOAD ADDRESS
XMT IVILRDAT,IVL *1 SELECT LOGIC RAM READ
CLR R5 *2 - CLEAR R5 TO RESET ADDRESS
XMT 0,R5
MOV R3,AUX *3 - AUX <- COMPUTED CHECKSUM
XOR RB,AUX COMPARE WITH LATEST CHEXSUM
XMT SYSELCHK,R1
NMT AUX,DIAGS060 IF CHEXSUMS COMPARE
CLR R1 THEN R1=0, ELSE R1=STOPMEM
XMT 0,R1
DIAGS060 XMT IVOSPD,IVL RESET CHEXSUM ADDR
XMT DIAGSHI,IVR
MOV R5,RB
NOP *1 - WAIT
MOV AUX,AUX
XMT DIAGSLO,IVR
MOV R5,RB
NMT R1,DIAGS070 BRANCH ON ERROR
JMP DIAGSX NO ERROR, CONTINUE
*
*
DIAGS100 XMT DIAGSHI,IVR INCRIMENT CHEXSUM ADDR BY 512
XMT 2,AUX *1
ADD RB,RB
NOP *1
MOV AUX,AUX
DIAGS110 XMT DIAGCHK,IVR
MOV R3,RB *1
NOP
MOV AUX,AUX
DIAGSX ERU * EXIT
*
***UPDATE SYSTEM TIMERS
*
EXEC040 RSP MSTRCLK,R1 R1 <- SCAN TIMER
XMT MSTRCLK,IVR LOAD ADDRESS
XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
MOV RB,R1 READ DATA
WSP TIMER001,R1 SET 0.01 TIMER
XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
XMT TIMER001,IVR LOAD ADDRESS
MOV R1,RB WRITE DATA
NOP *1 - WAIT
RSP AUX,AUX
TMR010,R2 R2 <- 0.10 TIMER TICK COUNTER
XMT TMR010,IVR LOAD ADDRESS
XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
MOV RB,R2 READ DATA
XMT -10,R4 R4 <- -(TIME BASE)
XMT TMR010,R5 R5 <- DESTINATION ADDRESS
CALL UPTIMER UPDATE TIMER
RSP TMR100,R2 R2 <- 1.00 TIMER TICK COUNTER
XMT TMR100,IVR LOAD ADDRESS
XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
MOV RB,R2 READ DATA
XMT -100,R4 R4 <- -(TIME BASE)
XMT TMR100,R5 R5 <- DESTINATION ADDRESS
CALL UPTIMER UPDATE TIMER
CLR R1 RESET SCAN TIMER
XMT 0,R1
WSP MSTRCLK,R1 LOAD TO SCRATCHPAD
XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
XMT MSTRCLK,IVR LOAD ADDRESS
MOV R1,RB WRITE DATA
JMP EXEC010
PROC INTRP
*
***SUBROUTINE INTRP
*
***THIS SUBROUTINE IS THE INTERRUPT HANDLER

```



```

4442 *
4443 ***CALLING SEQUENCE:
4444 *
4445 *      CALL  INTRP
4446 *
4447 ***REGISTER USAGE:
4448 *
4449 *      R1  - R1.EQ.0 => NO ERROR
4450 *           R1.NE.0 => ERROR STATE
4451 *      R2  - SCRATCH
4452 *      R3  - SCRATCH
4453 *      R4  - SCRATCH
4454 *      R5  - SCRATCH
4455 *      R6  - SCRATCH
4456 *      R11 - SUBROUTINE LINAKGE
4457 *      AUX - SCRATCH
4458 *
4459 *
4460 INTRP000 WSP      SAVERET,R11      SAVE RETURN
4461 05103 6 07021 + XMT      IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
4461 05104 6 17032 + XMT      SAVERET,IVR      LOAD ADDRESS
4461 05105 0 11037 + MOV      R11,R8      WRITE DATA
4462 ORG      10,32
4463 INTRP001 XMT      IVIINTRP,IVL    SELECT INTERRUPT SENSE REGISTER
4464 05107 5 27114 NZT      INTRPWF8,INTRPD10 BRANCH ON POWER-FAILURE
4465 05110 5 26115 NZT      INTRRTC8,INTRPD20 BRANCH ON REAL-TIME CLOCK
4466 05111 5 25116 NZT      INTRRRYB,INTRPD30 BRANCH ON RECEIVER READY
4467 05112 5 24117 NZT      INTRTRYB,INTRPD40 BRANCH ON TRANSMITTER READY
4468 05113 7 05120 JMP      INTRP050      EXIT WHEN NO INTERRUPTS PENDING
4469 *
4470 05114 7 05125 INTRPD10 JMP      INTRP100      POWER-FAILURE
4471 *
4472 05115 7 05127 INTRPD20 JMP      INTRP200      REAL-TIME CLOCK
4473 *
4474 05116 7 05136 INTRPD30 JMP      INTRP300      RECEIVER READY
4475 *
4476 05117 7 05332 INTRPD40 JMP      INTRP400      TRANSMITTER READY
4477 *
4478 INTRPD50 CLR      R1      EXIT WITH NO INTERRUPT PENDING
4478 05120 6 01000 + XMT      0,R1
4479 *
4480 INTRPX  RSP      SAVERET,R11      RESTORE RETURN
4480 05121 6 17032 + XMT      SAVERET,IVR      LOAD ADDRESS
4480 05122 6 07021 + XMT      IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
4480 05123 0 37011 + MOV      RB,R11      READ DATA
4481 05124 7 07176 RTN      RETURN
4483 *
4484 ***POWER-FAILURE
4485 *
4486 05125 6 01040 INTRP100 XMT      SYSSPDNM,R1      R1 <- POWER-DOWN STATE
4487 05126 7 05121 JMP      INTRPX      EXIT IMMEDIATELY
4488 *
4489 ***REAL-TIME CLOCK
4490 *
4491 05127 6 07000 INTRP200 XMT      IVOCTRL,IVL    SELECT CONTROL PULSES
4492 05130 6 27306 XMT      CTRLRTC,CTRLREG    RESET RTC
4493 05131 6 07021 XMT      IVOSPD+IVOSPD,IVL    SELECT SPD READ AND WRITE
4494 05132 6 17015 XMT      MSTRCLK,IVR      LOAD MSTRCLK ADDRESS
4495 05133 6 00001 XMT      1,AUX      *1 - AUX <- INCREMENT
4496 05134 1 37037 ADD      RB,R8      UPDATE MSTRCLK
4497 05135 7 05106 JMP      INTRP001      RECHECK INTERRUPTS
4499 *
4500 ***PERIPHERAL PORT RECEIVER
4501 *
4502 05136 6 07040 INTRP300 XMT      IVISTAT,IVL    SELECT STATUS REGISTER
4503 05137 7 05140 ORG      7,32
4504 05140 5 23105 NZT      STATERRB,INTRP305 BRANCH ON PARITY/FRAMING ERROR
4505 05141 5 22104 NZT      STATOVRB,INTRP301 BRANCH ON NO OVERRUN ERROR
4506 05142 6 01001 XMT      SYSEOVR,R1      R1 <- ERROR STATE ON OVERRUN
4507 05143 7 05146 JMP      INTRP310      CONTINUE PROCESSING
4508 *
4509 05144 7 05165 INTRP301 JMP      INTRP315      TO GET OVER SHORT BRANCH
4510 *
4511 ***ERROR HANDLING
4512 *
4513 INTRP305 CLR      R1      INDICATE NO ERROR
4513 05145 6 01000 + XMT      0,R1
4514 *
4515 INTRP310 CLR      R2      RESET RECEIVER STATUS
4515 05146 6 02000 + XMT      0,R2
4516 05147 6 17074 INTRP312 XMT      MSGCOUNT,IVR    LOAD ADDRESS
4517 05150 6 07021 XMT      IVISPD+IVOSPD,IVL    DO SELECTS
4518 05151 0 02037 MOV      R2,R8      CLEAR MESSAGE COUNT
4519 05152 6 02112 XMT      RCVRBUFF,R2      R2 <- BUFFER ADDRESS
4520 05153 6 17033 XMT      SAVSTATE,IVR      LOAD ADDRESS
4521 05154 0 01037 MOV      R1,R8      SAVE RETURN CODE
4522 05155 6 01075 XMT      RCVRBLK,R1      R1 <- BLOCK ADDRESS
4523 05156 6 03050 XMT      RCVRBLN,R3      R3 <- BUFFER LENGTH
4524 05157 6 11122 CALL     BUFFINIT      INITIALIZE BUFFER
4524 05160 7 05500
4525 RSP      SAVSTATE,R1      GET EXIT CODE
4525 05161 6 17033 + XMT      SAVSTATE,IVR      LOAD ADDRESS
4525 05162 6 07021 + XMT      IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
4525 05163 0 37001 + MOV      RB,R1      READ DATA
4526 05164 7 05325 JMP      INTRP365      AND EXIT
4528 *
4529 ***READ CHARACTER

```


4616	05302	6	01000	+	XMT	D,R1		
4617	05303	6	17102		XMT	RCVRSTAT,IVR	LOAD ADDRESS	
4618					NOP		*1 - WAIT	
4618	05304	0	00000	+	MOV	AUX,AUX		
4619	05305	0	01737		MOV	R1,7,RB	LEAVE EIA BIT ALONE	
4620	05306	7	05325		JMP	INTRP365	GO TO COMMON EXIT	
4621				*				
4622	05307	6	01003		INTRP350 XMT	ERRCHK,R1	R1 <- ERROR CODE FOR BAD CHECKSUM	
4623	05310	6	11125		CALL	ERRMSG	LOAD ERROR MESSAGE TO BUFFER	
	05311	7	05621					
4624	05312	7	05147		JMP	INTRP312	FLUSH BUFFER	
4625				*				
4626	05313	6	00377		INTRP355 XMT	-1,AUX	AUX <- DECREMENT	
4627	05314	1	37037		ADD	RB,RB	DECREMENT BYTE COUNT	
4628	05315	0	01000		MOV	R1,AUX	*1 - AUX <- CHARACTER	
4629	05316	6	17073		XMT	MSGCHECK,IVR	LOAD ADDRESS	
4630					NOP		*1 - WAIT	
4630	05317	0	00000	+	MOV	AUX,AUX		
4631	05320	1	37037		ADD	RB,RB	UPDATE CHECKSUM	
4632				*				
4633	05321	6	02075		INTRP360 XMT	RCVRBLK,R2	R2 <- RECEIVER BUFFER BLOCK	
4634	05322	6	11126		CALL	BFCH	BUFFER CHARACTER	
	05323	7	05526					
4635					CLR	R1	CLEAR EXIT CODE	
4635	05324	6	01000	+	XMT	D,R1		
4636				*				
4637	05325	6	07000		INTRP365 XMT	IVCTRL,IVL	SELECT CONTROL PULSES	
4638	05326	6	27304		XMT	CTRLCLR,CTRLREG	CLEAR RECEIVER	
4639	05327	5	01331		NZT	R1,INTRP370	BRANCH ON SYSTEM ERROR	
4640	05330	7	05106		JMP	INTRP001	ELSE, CONTINUE	
4641				*				
4642	05331	7	05121		INTRP370 JMP	INTRPX	GO TO EXIT	
4644				*				
4645				*	***PERIPHERAL PORT TRANSMITTER			
4646				*				
4647				*	INTRP400 RSP	XMITCNT,R1	R1 <- BUFFER COUNT	
4647	05332	6	17110	+	XMT	XMITCNT,IVR	LOAD ADDRESS	
4647	05333	6	07021	+	XMT	IVISPD+IVOSPD,IVL	*1 - SELECT SPD READ	
4647	05334	0	37001	+	MOV	RB,R1	READ DATA	
4648	05335	5	01337		NZT	R1,INTRP410	BRANCH IF BUFFER NOT EMPTY	
4649	05336	7	05344		JMP	INTRP420	BRANCH IF BUFFER EMPTY	
4650				*				
4651	05337	6	02104		INTRP410 XMT	XMITBLK,R2	R2 <- BUFFER BLOCK ADDRESS	
4652	05340	6	11127		CALL	UBFCH	GET NEXT CHAR	
	05341	7	05565					
4653	05342	6	07010		XMT	IVOPPDAT,IVL	SELECT TRANSMITTER PORT	
4654	05343	0	01027		MOV	R1,LB	WRITE OUT DATA	
4655				*				
4656	05344	6	07060		INTRP420 XMT	IVIINTRP,IVL	SELECT INTERRUPT SENSE	
4657	05345	5	27107		NZT	INTRPWF,INTRP430	BRANCH ON POWER-FAIL	
4658	05346	7	05120		JMP	INTRP050	GO TO SUCCESS EXIT	
4659				*				
4660	05347	7	05125		INTRP430 JMP	INTRP100	POWER-FAILURE	
4661				*				
4662				*	END	INTRP		
4664				*				
4665				*	***SYSTEM SUBROUTINE MODULES			
4666				*				
4668	05350				PROC	CRCHK		
4669				*				
4670				*	***SUBROUTINE CRCHK			
4671				*				
4672				*	***THIS SUBROUTINE COMPUTES THE CHECKSUM FOR THE COIL RAM			
4673				*				
4674				*	***CALLING SEQUENCE:			
4675				*				
4676				*	CALL	CRCHK		
4677				*				
4678				*	***PARAMETERS:			
4679				*				
4680				*	[SYSICON2H,SYSICON2L] - COIL RAM CONFIGURATION			
4681				*				
4682				*	***REGISTER USAGE:			
4683				*				
4684				*	R1 - COUNTER			
4685				*	R2 - COUNTER			
4686				*	R3 - NOT USED (PRESERVED)			
4687				*	R4 - NOT USED (PRESERVED)			
4688				*	R5 - NOT USED (PRESERVED)			
4689				*	R6 - CHECKSUM			
4690				*	R11 - SUBROUTINE LINKAGE			
4691				*	AUX - SCRATCH			
4692				*				
4693				*	***CHECKSUM SEED IS B'11111111'			
4694				*				
4695		000377			CRCKSEED EQU	11111111B	COIL RAM CHECKSUM SEED	
4697					CRCHK000 WSP	SAVERET,R11	SAVE RETURN ADDRESS	
4697	05350	6	07021	+	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE	
4697	05351	6	17032	+	XMT	SAVERET,IVR	LOAD ADDRESS	
4697	05352	0	11037	+	MOV	R11,RB	WRITE DATA	
4698	05353	6	07001		XMT	IVOCRHI,IVL	SELECT COIL ADDRESS HIGH	
4699					CLR	R2	FIRST CHECKSUM COIL DATA	
4699	05354	6	02000	+	XMT	0,R2		
4700	05355	0	02027		MOV	R2,LB	LOAD ADDRESS	
4701	05356	6	07000		XMT	IVOCRHI+IVICRDAT,IVL	SELECT PORTS	
4702	05357	0	02037		MOV	R2,RB	LOAD ADDRESS	
4703	05360	6	06377		XMT	CRCKSEED,R6	*1 - R6 <- CHECKSUM SEED	
4704	05361	6	11130		CALL	CRCHKSUB	*2,*3 - USE SUBROUTINE	


```

05362 7 05404
4705
4706
4707
4708 05363 6 04001 XMT SYSSTATH,R4 R4 <- HIGH-ORDER ADDRESS
4709
4710 05364 6 02001 CRCHK010 XMT SYSSTATL,R2 R2 <- START OF REGISTER SPACE
4711 05365 0 02037 MOV R2,RB LOAD ADDRESS
4712 05366 6 07001 XMT IVOCRHI,IVL SELECT COIL ADDRESS HIGH
4713 05367 0 04027 MOV R4,LB LOAD ADDRESS
4714 05370 6 03077 XMT 63,R3 ALTERNATE COUNT FOR FIRST CHIP
4715 05371 6 11131 CALL CRCHKENT USE ALTERNATE ENTRY POINT
05372 7 05405
4716 05373 6 00001 XMT 1,AUX AUX <- INCREMENT
4717 05374 1 04004 ADD R4,R4 BUMP COUNTER
4718 05375 6 00004 XMT 00000100B,AUX AUX <- MASK
4719 05376 3 04000 XOR R4,AUX AUX.EQ.0 => DONE
4720 05377 5 00364 NZT AUX,CRCHK010 AUX.NE.0 => CONTINUE
4721
4722 RSP SAVERET,R11 RESTORE RETURN ADDRESS
4722 05400 6 17032 + XMT SAVERET,IVR LOAD ADDRESS
4722 05401 6 07021 + XMT IVISPD+IVOSPD,IVL +1 - SELECT SPD READ
4722 05402 0 37011 + MOV RB,R11 READ DATA
4723 05403 7 07176 RTN EXIT
4724 END CRCHK
4726 05404 PROC CRCHKSUB
4727
4728 ***SUBROUTINE CRCHKSUB
4729
4730 05404 6 03100 CRCHKSDO XMT 64,R3 R3 <- COUNTER
4731
4732 000360 CRCHKMSK ERU SYSC256M+SYSC192M+SYSC128M+SYSC064M MASK FOR COIL CNFIGURATION
4733
4734 ENTRY CRCHKENT ALTERNATE ENTRY POINT
4735
4736 RSP SPDCONF2,R1 R1 <- COIL RAM CONFIGURATION
4736 05405 6 17277 + XMT SPDCONF2,IVR LOAD ADDRESS
4736 05406 6 07021 + XMT IVISPD+IVOSPD,IVL +1 - SELECT SPD READ
4736 05407 0 37001 + MOV RB,R1 READ DATA
4737
4738 05410 0 03002 CRCHKMS10 MOV R3,R2 SET UP COUNTER
4739 05411 6 00360 XMT CRCHKMSK,AUX AUX <- MASK
4740 05412 2 01001 AND R1,R1 R1 <- COUNTER
4741 05413 6 07000 XMT IVOCRTL+IVICRDAT,IVL SELECT PORTS
4742
4743 05414 0 27000 CRCHKMS20 MOV LB,AUX AUX <- COIL DATA
4744 05415 1 06006 ADD R6,R6 UPDATE CHECKSUM
4745 05416 6 27301 XMT CTRLINCC,CTRLREG BUMP ADDR
4746 05417 6 00377 XMT -1,AUX +1 - AUX <- DECREMENT
4747 05420 1 02002 ADD R2,R2 +2 - DECREMENT COUNTER
4748 05421 5 02014 NZT R2,CRCHKMS20 LOOP UNTIL DONE
4749 05422 6 00360 XMT CRCHKMSK,AUX AUX <- MASK
4750 05423 2 01101 AND R1(1),R1 SHIFT COUNTER
4751 05424 5 01010 NZT R1,CRCHKMS10 CONTINUE UNTIL COMPLETED
4752 05425 7 07176 RTN EXIT
4753 END CRCHKSUB
4755 05426 PROC LRCHK
4756
4757 ***SUBROUTINE LRCHK
4758
4759 ***THIS SUBROUTINE COMPUTES THE LOGIC RAM CHECKSUM
4760
4761 ***CALLING SEQUENCE:
4762
4763 CALL LRCHK
4764
4765 ***REGISTER USAGE:
4766
4767 R1 - SCRATCH
4768 R2 - SCRATCH
4769 R3 - SCRATCH
4770 R4 - NOT USED
4771 R5 - NOT USED
4772 R6 - CHECKSUM
4773 R11 - LINKAGE
4774 AUX - SCRATCH
4775
4776
4777
4778 LRCHK000 RSP SPDCONF1,R1 R1 <- LOGIC RAM CONFIGURATION
4778 05426 6 17276 + XMT SPDCONF1,IVR LOAD ADDRESS
4778 05427 6 07021 + XMT IVISPD+IVOSPD,IVL +1 - SELECT SPD READ
4778 05430 0 37001 + MOV RB,R1 READ DATA
4779 05431 6 00037 XMT 00011111B,AUX AUX <- MASK
4780 05432 2 01301 AND R1(3),R1 R1 <- NUMBER OF PAGES
4781 05433 6 07004 XMT IVOLRHI,IVL SELECT LOGIC ADDRHI
4782 05434 6 02000 XMT SYSUSERH,R2 R2 <- START OF LOGIC ADDRHI
4783 05435 0 02027 MOV R2,LB LOAD ADDRESS
4784 05436 6 02002 XMT SYSUSERL,R2 R2 <- START OF LOGIC ADDRLO
4785 05437 6 07003 XMT IVOLRLO,IVL SELECT PORT
4786 05440 0 02027 MOV R2,LB LOAD ADDRESS
4787 05441 6 03000 XMT CTRLINCL,R3 +1 - R3 <- INCREMENT PULSE
4788 05442 6 06377 XMT 11111111B,R6 +2 - R6 <- CHKSUM SEED
4789 05443 6 07000 XMT IVILRDAT+IVOCRTL,IVL +3 - SELECT PORTS
4790
4791 05444 0 37000 LRCHK010 MOV RB,AUX AUX <- DATA
4792 05445 0 03027 MOV R3,LB INCREMENT ADDRESS
4793 05446 1 06006 ADD R6,R6 +1 - UPDATE CHECKSUM

```

```

4794 05447 6 00001 XMT 1,AUX *2 - AUX <- INCREMENT
4795 05450 1 02002 ADD R2,R2 *3 - BUMP POINTER
4796 05451 5 02044 NZT R2,LRCHK010 R2.NE.0 =. CONTINUE
4797 05452 6 00377 XMT -1,AUX AUX <- DECREMENT
4798 05453 1 01001 ADD R1,R1 DECREMENT FIELD COUNTER
4799 05454 5 01044 NZT R1,LRCHK010 R1.NE.0 => CONTINUE
4800 05455 7 07176 RTN EXIT
4801 END LRCHK
4803 05456 PROC WRTUP
4804 *
4805 ***SUBROUTINE WRTUP
4806 *
4807 ***THIS SUBROUTINE WRITES ONE BYTE TO THE LOGIC RAM AND UPDATES
4808 ***THE LOGIC RAM CHECKSUM.
4809 *
4810 ***CALLING SEQUENCE:
4811 *
4812 * CALL WRTUP
4813 *
4814 * ON ENTRY, THE LRAM ADDRESSES ARE SET
4815 ***PARAMETERS:
4816 *
4817 * R1 - DATA BYTE (PRESERVED)
4818 * R2 - SCRATCH
4819 * R3 - R6 - NOT USED (PRESERVED)
4820 * R11 - SUBROUTINE LINKAGE
4821 * AUX - SCRATCH
4822 *
4823 ***
4825 05456 6 02000 XMT SYSLRCHL,R2 *1 - R2 <- LOGIC CHECKSUM ADDR LOW
4826 05457 6 07011 XMT IVILRDAT+IVOLRDAT,IVL *2 - SELECT PORTS
4827 NOP *3 - WAIT
4827 05460 0 00000 + MOV AUX,AUX
4828 05461 0 37000 MOV RB,AUX AUX <- OLD DATA
4829 05462 0 01027 MOV R1,LB WRITE OUT NEW DATA
4830 05463 6 07003 XMT IVOLRLO,IVL *1 - SELECT LOGIC ADDR LOW
4831 05464 0 02027 MOV R2,LB LOAD ADDRESS
4832 05465 6 07004 XMT IVOLRHI,IVL SELECT LOGIC ADDR HIGH
4833 05466 6 02000 XMT SYSLRCHH,R2 R2 <- LOGIC CHECKSUM ADDR HIGH
4834 05467 0 02027 MOV R2,LB LOAD ADDRESS
4835 05470 6 02377 XMT -1,R2 *1 - R2 <- -1
4836 05471 3 02000 XOR R2,AUX *2 - AUX <- ONE'S COMP OF OLD DATA
4837 05472 6 07011 XMT IVILRDAT+IVOLRDAT,IVL *3 - SELECT PORTS
4838 05473 1 37000 ADD RB,AUX AUX <- CHECKSUM - OLD DATA - 1
4839 05474 1 01000 ADD R1,AUX AUX <- UPDATED CHECKSUM
4840 05475 6 02001 XMT 1,R2 R2 <- INCREMENT
4841 05476 1 02027 ADD R2,LB WRITE OUT NEW CHECKSUM
4842 *
4843 05477 7 07176 RTN EXIT
4844 END WRTUP
4846 05500 PR C BUFFINIT
4847 *
4848 ***SUBROUTINE BUFFINIT
4849 *
4850 ***THIS SUBROUTINE IS USED TO INITIALIZE A CIRCULAR BUFFER.
4851 *
4852 ***CALLING SEQUENCE:
4853 *
4854 * CALL BUFFINIT
4855 *
4856 ***PARAMETERS:
4857 *
4858 * R1 - BUFFER BLOCK
4859 * R2 - BUFFER BASE ADDRESS
4860 * R3 - BUFFER LENGTH
4861 *
4862 ***REGISTER USAGE:
4863 *
4864 * R1 - BUFFER BLOCK ADDRESS
4865 * R2 - BUFFER BASE ADDRESS
4866 * R3 - BUFFER LENGTH
4867 * R4 - SCRATCH
4868 * R5 - NOT USED
4869 * R6 - NOT USED
4870 * R11 - LINKAGE
4871 * AUX - SCRATCH
4872 *
4874 05500 6 00001 BFIND000 XMT 1,AUX AUX <- INCREMENT
4875 CLR R4 R4 <- 0
4875 05501 6 04000 + XMT 0,R4
4876 05502 6 07001 XMT IVOSPD,IVL SELECT SCRATCHPAD WRITE
4877 *
4878 05503 0 01017 MOV R1,IVR LOAD ADDRESS
4879 05504 0 02037 MOV R2,RB LOAD BUFFER BASE
4880 05505 1 01001 ADD R1,R1 *1 - INCREMENT ADDRESS
4881 05506 0 01017 MOV R1,IVR LOAD ADDRESS
4882 05507 0 04037 MOV R4,RB INITIALIZE IPTR
4883 05510 1 01001 ADD R1,R1 *1 - INCREMENT ADDRESS
4884 05511 0 01017 MOV R1,IVR LOAD ADDRESS
4885 05512 0 04037 MOV R4,RB INITIALIZE OPTR
4886 05513 1 01001 ADD R1,R1 *1 - INCREMENT ADDRESS
4887 05514 0 01017 MOV R1,IVR LOAD ADDRESS
4888 05515 0 03037 MOV R3,RB LOAD BUFFER LENGTH
4889 05516 1 01001 ADD R1,R1 *1 - INCREMENT ADDRESS
4890 05517 0 01017 MOV R1,IVR LOAD ADDRESS
4891 05520 0 04037 MOV R4,RB INITIALIZE USAGE COUNT

```



```

4892 05521 1 01001 ADD R1,R1 *1 - INCREMENT ADDRESS
4893 05522 0 01017 MOV R1,IVR LOAD ADDRESS
4894 NOP *1 - WAIT (FOR 7 BIT WRITE)
4894 05523 0 00000 + MOV AUX,AUX
4895 05524 0 04737 MOV R4,7,RB INITIALIZE STATUS
4896 05525 7 07176 RTN EXIT
4897 END BUFFINIT
4899 05526 PROC BFCH
4900 *
4901 ***SUBROUTINE BFCH - BUFFER CHARACTER
4902 *
4903 ***THIS SUBROUTINE BUFFERS A CHARACTER TO A CIRCULAR BUFFER
4904 *
4905 ***CALLING SEQUENCE:
4906 *
4907 * CALL BFCH
4908 *
4909 ***PARAMETERS:
4910 *
4911 * R1 - CHARACTER TO BE BUFFERED
4912 * R2 - BUFFER DATA BLOCK
4913 *
4914 ***REGISTER USAGE:
4915 *
4916 * R1 - CHARACTER (PRESERVED)
4917 * R2 - BUFFER DATA BLOCK ADDRESS (PRESERVED)
4918 * R3 - NOT USED
4919 * R4 - SCRATCH
4920 * R5 - NOT USED
4921 * R6 - NOT USED
4922 * R11 - SUBROUTINE LINKAGE
4923 * AUX - SCRATCH
4924 *
4925 ***
4927 05526 6 00001 BFCH000 XMT BFIPTR,AUX AUX <- OFFSET
4928 05527 1 02017 ADD R2,IVR LOAD IPTR ADDRESS
4929 05530 6 07021 XMT IVISPD+IVOSPD,IVL *1 - SELECR SPD READ/WRITE
4930 05531 0 37003 MOV RB,R3 R3<- IPTR
4931 05532 1 37004 ADD RB,R4 R4 <- NEW IPTR
4932 05533 6 00003 XMT BFLEN,AUX AUX <- OFFSET
4933 05534 1 02017 ADD R2,IVR LOAD ADDRESS
4934 05535 0 04000 MOV R4,AUX *1 - AUX <- NEW IPTR
4935 05536 3 37000 XOR RB,AUX AUX.EQ.0 => WRAP-AROUND
4936 05537 5 00141 NZT AUX,BFCH010 AUX.NE.0 => NO WRAP-AROUND
4937 05540 6 04000 + CLR R4 RESET IPTR ON WRAP-AROUND
4938 *
4939 05541 6 00002 BFCH010 XMT BFOPTR,AUX AUX <- OFFSET
4940 05542 1 02017 ADD R2,IVR LOAD ADDRESS
4941 05543 0 04000 MOV R4,AUX AUX <- NEW IPTR
4942 05544 3 37000 XOR RB,AUX AUX.EQ.0 => BUFFER FULL
4943 05545 5 00141 NZT AUX,BFCH020 AUX.NE.0 => BUFFER NOT FULL
4944 05546 6 00377 XMT -1,AUX AUX <- INVERT
4945 05547 3 01001 XOR R1,R1 COMPLEMENT CHARACTER
4946 05550 7 05564 JMP BFCHX AND EXIT
4947 *
4948 05551 0 02017 BFCH020 MOV R2,IVR LOAD BASE ADDRESS
4949 05552 0 03000 MOV R3,AUX *1 - AUX <- IPTR
4950 05553 1 37017 + ADD RB,IVR LOAD NEW BUFFER ADDRESS
4951 05554 0 01037 MOV R1,RB WRITE BYTE TO BUFFER
4952 05555 6 00001 XMT BFIPTR,AUX *1 - AUX <- OFFSET
4953 05556 1 02017 ADD R2,IVR LOAD IPTR ADDRESS
4954 05557 0 04037 MOV R4,RB LOAD NEW IPTR
4955 05560 6 00004 XMT BFUSE,AUX *1 - AUX <- OFFSET
4956 05561 1 02017 ADD R2,IVR LOAD ADDRESS
4957 05562 6 00001 XMT 1,AUX *1 - AUX <- INCREMENT
4958 05563 1 37037 ADD RB,RB USAGE <- USAGE + 1
4959 *
4960 05564 7 07176 BFCHX RTN EXIT
4961 END BFCH
4963 05565 PROC UBFCH
4964 *
4965 ***SUBROUTINE UBFCH
4966 *
4967 ***THIS SUBROUTINE UNBUFFERS CHARACTERS FROM A CIRCULAR BUFFER.
4968 *
4969 ***CALLING SEQUENCE:
4970 *
4971 * CALL UBFCH
4972 *
4973 ***PARAMETERS:
4974 *
4975 * R1 - CHARACTER ON RETURN
4976 * R2 - BUFFER BLOCK POINTER
4977 *
4978 ***REGISTER USAGE:
4979 *
4980 * R1 - CHARACTER
4981 * R2 - BUFFER BLOCK ADDRESS (DESTROYED)
4982 * R3 - NOT USED
4983 * R4 - SCRATCH
4984 * R5 - NOT USED
4985 * R6 - NOT USED
4986 * R11 - SUBROUTINE LINKAGE
4987 * AUX - SCRATCH
4988 *
4989 ***

```



```

4991 05565 6 00004  UBFC000 XMT  BFUSE,AUX      AUX <- OFFSET
4992 05566 1 02017      ADD  R2,IVR      LOAD ADDRESS
4993 05567 6 07021      XMT  IVISPD+IVOSPD,IVL *1 - SELECT SPD READ/WRITE
4994 05570 5 37033      NZT  RB,UBFC010  BRANCH IF BUFFER NOT EMPTY
4995      CLR  R1        R1 <- 0
4995 05571 6 01000  +  XMT  0,R1
4996 05572 7 05620      JMP  UBFC010    AND EXIT
4997      *
4998 05573 6 00377  UBFC010 XMT  -1,AUX      AUX <- DECREMENT
4999 05574 1 37037      ADD  RB,RB      USAGE <- USAGE - 1
5000 05575 6 00002      XMT  BFOPTR,AUX *1 - AUX <- OFFSET
5001 05576 1 02017      ADD  R2,IVR      LOAD ADDRESS
5002      NOP        *1 - WAIT
5002 05577 0 00000  +  MOV  AUX,AUX
5003 05600 0 37004      MOV  RB,R4      R4 <- OPTR
5004 05601 0 02017      MOV  R2,IVR      LOAD ADDRESS
5005 05602 0 04000      MOV  R4,AUX      *1 - AUX <- OFFSET
5006 05603 1 37017      ADD  RB,IVR      LOAD BUFFER ADDRESS
5007 05604 6 00003      XMT  BFLEN,AUX   AUX <- OFFSET
5008 05605 0 37001      MOV  RB,R1      R1 <- DATA BYTE
5009 05606 1 02017      ADD  R2,IVR      LOAD LENGTH ADDRESS
5010 05607 6 00001      XMT  1,AUX      *1 - AUX <- INCREMENT
5011 05610 1 04004      ADD  R4,R4      OPTR <- OPTR + 1
5012 05611 0 04000      MOV  R4,AUX      AUX <- NEW IPTR
5013 05612 3 37000      XOR  RB,AUX      AUX.EQ.0 => WRAP-AROUND
5014 05613 5 00215      NZT  AUX,UBFC020 AUX.NE.0 => NO WRAP-AROUND
5015      CLR  R4      RESET OPTR
5015 05614 6 04000  +  XMT  0,R4
5016      *
5017 05615 6 00002  UBFC020 XMT  BFOPTR,AUX   AUX <- OFFSET
5018 05616 1 02017      ADD  R2,IVR      LOAD OPTR ADDRESS
5019 05617 0 04037      MOV  R4,RB      LOAD NEW OPTR
5020      *
5021 05620 7 07176  UBFC010 RTN
5022      END  UBFC010  EXIT
5024 05621      PROC  ERRMSG
5025      *
5026      ***SUBROUTINE ERRMSG
5027      *
5028      ***THIS SUBROUTINE BUFFERS AN ERROR MESSAGE FOR THE TRANSMIT BUFFER.
5029      *
5030  EMSG000 WSP  CM004,R1  WRITE OUT ERROR CODE IN MESSAGE
5030 05621 6 07021  +  XMT  IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
5030 05622 6 17235  +  XMT  CM004,IVR    LOAD ADDRESS
5030 05623 0 01037  +  MOV  R1,RB      WRITE DATA
5031 05624 6 01000      XMT  0,R1      CLEAR R1 FOR EXEC
5032 05625 6 02001      XMT  1,R2      R2 <- MSGCOUNT
5033 05626 7 07176      RTN
5034      END  ERRMSG  EXIT
5036 05627      PROC  OPTIMER
5037      *
5038      ***SUBROUTINE OPTIMER
5039      *
5040      ***THIS SUBROUTINE IS USED TO UPDATE TIMERS AT END-OF-SWEEP.
5041      *
5042      ***CALLING SEQUENCE:
5043      *
5044      CALL  OPTIMER
5045      *
5046      ***PARAMETERS:
5047      *
5048      R1 - MSTRCLK
5049      R2 - TICK COUNTER
5050      R4 - TIMER BASE (TWO'S COMPLEMENT)
5051      R5 - DESTINATION ADDRESS
5052      IVR - TICK COUNTER ADDRESS
5053      *
5054      ***REGISTER USAGE:
5055      *
5056      R1 - MSTRCLK (PRESERVED)
5057      R2 - TICK COUNTER (UPDATED)
5058      R3 - UPDATED CLOCK
5059      R4 - TIMER BASE (TWO'S COMPLEMENT)
5060      R5 - DESTINATION ADDRESS (PRESERVED)
5061      R6 - NOT USED (PRESERVED)
5062      R11 - SUBROUTINE LINKAGE
5063      AUX - SCRATCH
5064      *
5065      ***
5067 05627 0 01000  UPTIM000 MOV  R1,AUX      AUX <- SWEEP TIMER
5068      CLR  R3      R3 TO HOLD UPDATED TIMER
5068 05630 6 03000  +  XMT  0,R3
5069 05631 1 02002      ADD  R2,R2      UPDATE TICK COUNTER
5070 05632 0 04000      MOV  R4,AUX      AUX <- TIMER BASE
5071 05633 1 02000      ADD  R2,AUX      AUX <- TICK COUNTER - BASE
5072 05634 5 10236      NZT  OV,OPTIM010 OV.NE.0 => TICKCOUNTER.GE.BASE
5073 05635 7 05640      JMP  OPTIM020  OV.EQ.0 => TICKCOUNTER.LT.BASE
5074      *
5075 05636 6 03001  UPTIM010 XMT  1,R3      SET TIMER VALUE
5076 05637 0 00002      MOV  AUX,R2      R2 <- NEW TICK COUNTER VALUE
5077      *
5078 05640 6 07001  UPTIM020 XMT  IVOSPD,IVL  SELECT SCRATCHPAD WRITE
5079 05641 0 02037      MOV  R2,RB      WRITE OUT NEW TICK COUNTER
5080      NOP        *1 - WAIT

```

```

5080 05642 0 00000 + MOV AUX,AUX
5081 05643 0 05017 MOV R5,IVR LOAD ADDRESS
5082 05644 0 03037 MOV R3,RB WRITE OUT TIMER VALUE
5083 05645 7 07176 RTN RETURN
5084 END UPTIMER
5086 05646 PROC OUTPUT
5087 *
5088 ***SUBROUTINE OUTPUT
5089 *
5090 ***THIS SUBROUTINE BUILDS A NIBBLE OF DATA FROM THE COIL RAM
5091 *
5092 ***CALLING SEQUENCE:
5093 *
5094 CALL OUTPUT
5095 *
5096 ***PARAMETERS:
5097 *
5098 R3 - OUTPUT BYTE
5099 *
5100 ***REGISTER USAGE:
5101 *
5102 R1 - I/O ADDRESS
5103 R2 - SCRATCH
5104 R3 - OUTPUT BYTE
5105 R4 - INPUT BYTE
5106 R5 - SCRATCH
5107 R6 - NOT USED
5108 R11 - LINKAGE
5109 AUX - SCRATCH
5110 *
5111 ***
5113 05646 6 07000 OUTPUT00 XMT IVICRDAT+IVOCCTRL,IVL SELECT PORTS
5114 05647 6 02004 XMT 4,R2 R2 <- COUNTER
5115 *
5116 05650 0 26100 OUTPUT10 MOV CROUTPUT,AUX AUX <- COIL STAT
5117 05651 6 27301 XMT CTRLINCC,CTRLREG INC ADDR
5118 05652 3 03003 XOR R3,R3 *1 - LOAD COIL STATE
5119 05653 0 03103 MOV R3(1),R3 *2 - ROTATE OUTPUT VECTOR
5120 05654 6 00377 XMT -1,AUX *3 - AUX <- DECREMENT
5121 05655 7 02002 ADD R2,R2 DECREMENT COUNTER
5122 05656 5 02250 NZT R2,OUTPUT10 LOOP UNTIL DONE
5123 *
5124 05657 7 07176 RTN EXIT
5125 END OUTPUT
5127 05660 PROC INPUT
5128 *
5129 ***SUBROUTINE INPUT
5130 *
5131 ***THIS SUBROUTINE UNPACKS I/O INPUT DATA
5132 *
5133 ***CALLING SEQUENCE:
5134 *
5135 CALL INPUT
5136 *
5137 ***PARAMETERS:
5138 *
5139 R4 - INPUT BYTE
5140 *
5141 ***REGISTER USAGE:
5142 *
5143 R1 - I/O ADDRESS
5144 R2 - SCRATCH
5145 R3 - OUTPUT BYTE
5146 R4 - INPUT BYTE
5147 R5 - NOT USED
5148 R6 - NOT USED
5149 R11 - LINKAGE
5150 AUX - SCRATCH
5151 *
5152 ***
5154 05660 6 02004 INPUT00 XMT 4,R2 R2 <- LOOP COUNT R
5155 *
5156 05661 6 07002 INPUT010 XMT IVICRDAT+IVOCRDAT,IVL SELECT COIL READ AND WRITE
5157 05662 0 25100 MOV CRINPUT,AUX AUX <- CURRENT INPUT STATE
5158 05663 0 00121 MOV AUX,CRINHIS LOAD HISTORY HIT
5159 *
5159 05664 0 00000 + MOV AUX,AUX *1 - WAIT FOR WRITE
5160 *
5160 05665 0 00000 + MOV AUX,AUX *2 - TO OCCUR BEFORE NEXT READ CYCLE
5161 *
5161 05666 0 00000 + MOV AUX,AUX *3 - BEFORE TESTING STATUS
5162 05667 5 24131 NZT CRINDISB,INPUT020 BRANCH IF DISABLED
5163 05670 0 04125 MOV R4,CRINPUT LOAD INPUT STATE
5164 *
5165 05671 0 04104 INPUT020 MOV R4(1),R4 ROTATE INPUT BYTE
5166 05672 6 00377 XMT -1,AUX AUX <- DECREMENT
5167 05673 6 07000 XMT IVOCCTRL,IVL SELECT CONTROL
5168 05674 6 27301 XMT CTRLINCC,CTRLREG BUMP ADDR
5169 05675 1 02002 ADD R2,R2 *1 - DECREMENT COUNTER
5170 05676 5 02261 NZT R2,INPUT010 *2 - BRANCH UNTIL DONE
5171 *
5172 05677 7 07176 RTN EXIT
5173 *
5174 END INPUT

```

```

5176 05700          PROC   REGVAL
5177
5178          *
5179          *   ROUTINE: REGVAL
5180          *   ENTER WITH NODE DATA IN [R1,R2]
5181          *   COMPUTE REGISTER ADDR, AND DECIDE IF HOLDING
5182          *   REGISTER OR INPUT REGISTER
5183          *   EXIT WITH REGISTER CONTENTS IN [R1,R2] AND
5184          *   WITH REGISTER ADDR IN [R5,R6]
5185 05700 6 00003    REGVAL00 XMT   3,AUX           CHECK FOR HOLDING REGISTER OR
5186 05701 2 01000    AND     R1,AUX           INPUT REGISTER
5187 05702 4 00303    XEC     REGVALTB(AUX),4    VECTOR TO REGISTER TYPE
5188
5189 05703 7 05707    REGVALTB JMP   REGVAL10        HOLDING REGISTER
5190 05704 7 05746    JMP   REGVAL50        INPUT REGISTER
5191 05705 7 05761    JMP   REGVAL60        DUMMY REGISTER
5192 05706 7 05775    JMP   REGVAL99        INVALID REGISTER TYPE
5193
5194          *
5195 05707 6 07000    REGVAL10 XMT   IVOCtrl,IVL        SELECT COIL RAM ADDR LO
5196 05710 6 00002    XMT   REGBASEL,AUX      GET BASE ADDR OF REG TABLE
5197 05711 1 02006    ADD     R2,R6           ADD REFERENCE NUMBER -> R6
5198 05712 0 06037    MOV     R6,RB           SET COIL ADDR LO
5199 05713 0 10005    MOV     OVf,R5
5200 05714 6 00001    XMT   REGBASEH,AUX
5201 05715 1 05005    ADD     R5,R5           GET HI REG ADDR -> R5
5202 05716 6 07001    XMT   IVOCRHI+IVICRDAT,IVL SELECT COIL HI ADDR, COIL READ
5203 05717 0 05027    MOV     R5,LB           REG ADDR HI -> COIL ADDR HI
5204
5204 05720 6 02000    + XMT   0,R2            *1
5205 05721 6 00001    XMT   1,AUX            *2
5206 05722 1 05000    ADD     R5,AUX          *3
5207 05723 0 27402    MOV     REGDATA,R2      GET LOW NIBBLE OF REGISTER DATA
5208 05724 0 00027    MOV     AUX,LB          ADDRESS MIDDLE NIBBLE
5209 05725 6 00002    XMT   2,AUX            *1
5210 05726 1 05000    ADD     R5,AUX          *2
5211
5211 05727 6 01000    + XMT   0,R1            *3
5212 05730 0 27401    MOV     REGDATA,R1      MOVE MIDDLE NIBBLE -> R1 TEMP'ARLY
5213 05731 0 00027    MOV     AUX,LB          ADDRESS HIGH ORDER NIBBLE
5214 05732 0 02000    MOV     R2,AUX          *1 COMBINE LOW AND MIDDLE NIBBLE
5215 05733 3 01402    XOR     R1(4),R2        *2
5216
5216 05734 6 00000    + XMT   0,AUX            *3
5217 05735 0 27401    MOV     REGDATA,R1      GET HIGH NIBBLE
5218 05736 0 00027    MOV     AUX,LB          000->COIL ADDR HI
5219 05737 6 07001    XMT   IVOSPD,IVL        SELECT SCRATCH PAD WRITE
5220 05740 6 17067    XMT   CNTRPWR,IVR        PUT COUNTER POWER HISTORY INTO SCRATCH PAD
5221 05741 6 00003    XMT   00000011B,AUX
5222 05742 0 01201    MOV     R1(2),R1
5223 05743 2 01037    AND     R1,RB           WRITE COUNTER POWER
5224 05744 2 01601    AND     R1(6),R1        MASK COUNTER POWER OUT OF REG DATA
5225
5226 05745 7 05776    JMP     REGVALX          TAKE COMMON EXIT
5228
5229          *
5230          *   INPUT REGISTERS
5231
5231 05746 6 00300    REGVAL50 XMT   REG3001H,AUX    INPUT REGISTERS, GET BASE ADDR
5232 05747 1 02000    ADD     R2,AUX
5233 05750 1 02006    ADD     R2,R6           ADD REFERENCE NUMBER -> R6
5234 05751 6 07021    XMT   IVOSPD+IVISPD,IVL SELECT SCRATCHPAD READ WRITE
5235 05752 0 06017    MOV     R6,IVR
5236 05753 6 00001    XMT   1,AUX            *
5237 05754 0 37001    MOV     RB,R1           GET HI ORDER REGISTER DATA -> R1
5238 05755 1 06017    ADD     R6,IVR          ADDR LOW REG
5239
5239 05756 6 05000    + XMT   0,R5            *
5240 05757 0 37002    MOV     RB,R2           GET LO ORDER REGISTER DATA -> R2
5241 05760 7 05776    JMP     REGVALX
5242
5243          *
5243 05761 6 17070    REGVAL60 RSP   REG4000H,R1    GET DUMMY REGISTER HI
5243 05762 6 07021    + XMT   REG4000H,IVR      LOAD ADDRESS
5243 05763 0 37001    + XMT   IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
5244
5244 05764 6 17071    + MOV     RB,R1           READ DATA
5244 05765 6 07021    + RSP   REG4000L,R2      GET DUMMY REGISTER LO
5244 05766 0 37002    + XMT   REG4000L,IVR      LOAD ADDRESS
5245
5245 05767 6 00000    + XMT   IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
5246
5246 05770 6 07021    + MOV     RB,R2           READ DATA
5246 05771 6 17067    + CLR     AUX
5246 05772 0 00037    + XMT   0,AUX
5247 05773 6 05377    + WSP   CNTRPWR,AUX      CLEAR COUNTER POWER HISTORY
5248 05774 7 05776    + XMT   IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
5249
5249 05775 6 01377    + XMT   CNTRPWR,IVR      LOAD ADDRESS
5250
5250 05776 7 07176    + MOV     AUX,RB          WRITE DATA
5251
5251 05777          + XMT   -1,R5           SET DUMMY REG FLAG
5252
5252 05778          + JMP     REGVALX
5253
5253 05779          *
5254 05780          *   REGVAL99 XMT   -1,R1           ERROR RETURN
5255
5255 05781          *
5256 05782          *   REGVALX RTN
5257
5258 05783          *   END   REGVAL
5259 05784          *   PROC  STORE
5260
5261          *
5262          *   ROUTINE: STORE
5263          *   ENTER WITH REGISTER (HOLDING) ADDR IN [R5,R6] AND
5264          *   DATA IN [R1,R2]

```



```

5260
5261 05777 6 00377 STORE000 XMT -1,AUX CHECK FOR DUMMY REGISTER
5262 06000 3 05000 XDR R5,AUX
5263 06001 5 00003 NZT AUX,STORE010 NOT DUMMY REG, GOTO STOREG10
5264 06002 7 06040 JMP STORE020 DUMMY REG, GO TO STORE020
5265 STORE010 RSP CNTRPWR,AUX GET COUNTER POWER
5265 06003 6 17067 + XMT CNTRPWR,IVR LOAD ADDRESS
5265 06004 6 07021 + XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
5265 06005 0 37000 + MOV RB,AUX READ DATA
5266 06006 0 00600 MOV AUX(6),AUX ROTATE INTO PLACE
5267 06007 3 01001 XDR R1,R1 COMBINE WITH REGISTER DATA
5268 06010 6 07000 XMT IVOCRLO,IVL SELECT COIL ADDR LO
5269 06011 0 06037 MOV R6,RB LOAD LO ADDR
5270 06012 6 07001 XMT IVOCRHI,IVL SELECT COIL ADDR HI
5271 06013 0 05027 MOV R5,LB
5272 06014 6 07002 XMT IVOCRDAT,IVL *1 SELECT COIL WRITE
5273 06015 6 00017 XMT 00001111B,AUX GET MASK
5274 06016 2 02027 AND R2,LB WRITE LO ORDER NIBBLE
5275 06017 6 00001 XMT 1,AUX
5276 06020 1 05000 ADD R5,AUX STEP TO NEXT NIBBLE
5277 06021 6 07001 XMT IVOCRHI,IVL SELECT COIL RAM ADDR HI
5278 06022 0 00027 MOV AUX,LB
5279 06023 6 00017 XMT 00001111B,AUX *1 GET MASK
5280 06024 2 02400 AND R2(4),AUX PREPARE MIDDLE NIBBLE
5281 06025 6 07002 XMT IVOCRDAT,IVL SELECT COIL RAM WRITE
5282 06026 0 00027 MOV AUX,LB WRITE MIDDLE NIBBLE
5283 06027 6 00002 XMT 2,AUX
5284 06030 1 05000 ADD R5,AUX STEP TO HI ORDER NIBBLE
5285 06031 6 07001 XMT IVOCRHI,IVL SELECT COIL RAM ADDR HI
5286 06032 0 00027 MOV AUX,LB
5287 06033 6 07002 XMT IVOCRDAT,IVL SELECT COIL RAM WRITE
5288 06034 0 01027 MOV R1,LB WRITE HI ORDER NIBBLE
5289 06035 6 07001 XMT IVOCRHI,IVL SELECT COIL ADDR HI
5290 CLR AUX
5290 06036 6 00000 + XMT 0,AUX
5291 06037 0 00027 MOV AUX,LB 0-> COIL RAM ADDR HI
5292 06040 7 07176 STORE020 RTN EXIT
5293 END STORE
5295 06041 PROC ADRVAL
5296
5297 ***SUBROUTINE ADRVAL
5298
5299 ***THIS SUBROUTINE VALIDATES AN ADDRESS FOR A COMMAND FUNCTION
5300 * THE ONLY COMMANDS TO CALL ADRVAL ARE ONES WITH VARIABLE LENGTH
5301 * FIELDS. ADRVAL ENSURES THAT THE ADDRESS+LENGTH-1 IS IN BOUNDS.
5302
5303 ***CALLING SEQUENCE:
5304
5305 * CALL ADRVAL
5306
5307 ***RETURNS:
5308
5309 * TO CALLER IF NO ERROR
5310 * TO CMDS1610 ON ERROR
5311
5312 ***REGISTER USAGE:
5313
5314 * R1 - ADDRESS FIELD
5315 * R2 - -(DATA LENGTH)
5316 * R3 - SCRATCH
5317 * R4 - SCRATCH
5318 * R5 - ADDRHI
5319 * R6 - ADDRLO
5320 * R11 - LINKAGE
5321 * AUX - SCRATCH
5322
5323 ***
5325 06041 6 17233 ADRVAL00 XMT CMD02,IVR LOAD ADDRESS
5326 06042 6 07021 XMT IVOSPD+IVISPD,IVL DO SELECTS
5327 06043 0 37404 MOV LENFLD,R4 PICK UP LENGTH FIELD IN COMMAND BYTE
5328 06044 6 00367 XMT -9,AUX LENGTH SHOULD BE < 9
5329 06045 1 04002 ADD R4,R2 SEE IF OK
5330 06046 6 00200 XMT 10000000B,AUX R2 SHOULD BE -
5331 06047 2 02002 AND R2,R2
5332 06050 5 02052 NZT R2,ADRVAL05
5333 06051 7 06147 JMP ADRVAL55 IF NOT, ERROR
5334 06052 6 17235 ADRVAL05 XMT CMD04,IVR LOAD ADDRESS
5335 NOP *1 - WAIT
5335 06053 0 00000 + MOV AUX,AUX
5336 06054 0 37005 MOV RB,R5 R5<- ADDRHI
5337 06055 0 32301 MOV ADRFLD,R1 PICK UP FIELD
5338 06056 6 17236 XMT CMD05,IVR LOAD ADDRESS
5339 06057 6 00377 XMT -1,AUX *1
5340 06060 0 37006 MOV RB,R6 R6<- ADDRLO
5341 06061 6 17030 XMT SAVER6,IVR SAVE ADDR
5342 06062 0 06037 MOV R6,RB
5343 NOP *1 - WAIT
5343 06063 0 00000 + MOV AUX,AUX
5344 06064 6 17027 XMT SAVER5,IVR
5345 06065 0 05037 MOV R5,RB
5346 06066 4 01077 XEC ADRVALTB(R1),8 AUX<- LENGTH - 1 IN MEMORY
5347 06067 1 06006 ADD R6,R6 ADD IN LENGTH
5348 06070 0 10000 MOV OV,AUX
5349 06071 1 05005 ADD R5,R5
5350 06072 6 00007 XMT 7,AUX MAKE SURE DIDN'T CHANGE FIELDS
5351 06073 2 05500 AND R5(5),AUX AUX<- NEW FIELD

```

5352	06074	3	01002	XOR	R1,R2	SHOULD = OLD FIELD
5353	06075	5	02145	NZT	R2,ADRVAL50	IF NOT, ERROR
5354	06076	7	06111	JMP	ADRVAL20	
5355				*		
5356	06077	1	04700	ADRVALTB	ADD R4(7),AUX	LOGIC SPACE
5357	06100	1	04700	ADD	R4(7),AUX	I/O SPACE
5358	06101	1	04000	ADD	R4,AUX	REGISTER SPACE
5359	06102	1	04700	ADD	R4(7),AUX	SCRATCHPAD SPACE
5360	06103	7	06107	JMP	ADRVAL10	ILLEGAL
5361	06104	7	06107	JMP	ADRVAL10	ILLEGAL
5362	06105	7	06107	JMP	ADRVAL10	ILLEGAL
5363	06106	7	06107	JMP	ADRVAL10	ILLEGAL
5364				*		
5365	06107	6	01005	ADRVAL10	XMT ERRADI,R1	R1 <- ERROR CODE
5366	06110	7	04646	JMP	CMDERR	GO TO ERROR HANDLER
5367				*		
5368	06111	6	17276	ADRVAL20	XMT SPDCONF1,IVR	LOAD ADDRESS
5369	06112	6	00037	XMT	-1-ADRMSK,AUX	AUX <- MASK
5370	06113	0	37003	MOV	R8,R3	R3 <- CONF1
5371	06114	6	17277	XMT	SPDCONF2,IVR	LOAD ADDRESS
5372	06115	2	05005	AND	R5,R5	MASK OUT FIELD DESIGNATOR
5373	06116	0	37004	MOV	R8,R4	R4 <- CONF2
5374	06117	4	01120	-XEC	ADRVALT2(R1),4	EXECUTE VIA TABLE OF FIELD TYPES
5375				*		
5376	06120	7	06124	ADRVALT2	JMP ADRVAL30	LOGIC SPACE
5377	06121	7	06151	JMP	ADRVAL60	I/O SPACE
5378	06122	7	06151	JMP	ADRVAL60	REGISTER SPACE
5379	06123	7	06154	JMP	ADRVAL80	SCRATCHPAD SPACE
5380				*		
5381	06124	6	00001	ADRVAL30	XMT 1,AUX	AUX <- MASK
5382	06125	2	06000	AND	R6,AUX	R6 SHOULD BE ODD
5383	06126	5	00130	NZT	AUX,ADRVAL31	
5384	06127	7	06107	JMP	ADRVAL10	
5385	06130	6	00037	ADRVAL31	XMT 00011111B,AUX	AUX <- MASK
5386	06131	2	03303	AND	R3(3),R3	R3 <- NUMBER OF LOGIC FIELDS
5387				*		
5388	06132	6	00377	ADRVAL35	XMT -1,AUX	AUX <- MASK
5389	06133	3	05002	XOR	R5,R2	R2 <- COMPLEMENT OF ADDRHI
5390	06134	6	00001	XMT	1,AUX	AUX <- INCREMENT
5391	06135	1	02000	ADD	R2,AUX	AUX <- -(ADDRHI)
5392	06136	1	03000	ADD	R3,AUX	AUX <- FIELDS - ADDRHI
5393	06137	5	00141	NZT	AUX,ADRVAL40	AUX.NE.0 => CHECK OVF
5394	06140	7	06145	JMP	ADRVAL50	AUX.EQ.0 => ERROR
5395				*		
5396	06141	6	02200	ADRVAL40	XMT 10000000B,R2	R2 <- MASK
5397	06142	2	02002	AND	R2,R2	R2 <- MSB
5398	06143	5	02145	NZT	R2,ADRVAL50	R2.NE.0 => ERROR
5399	06144	7	06155	JMP	ADRVALX	R2.EQ.0 => SUCCESS
5400				*		
5401	06145	6	01004	ADRVAL50	XMT ERRADR,R1	R1 <- ERROR CODE
5402	06146	7	04646	JMP	CMDERR	ERROR EXIT
5403				*		
5404	06147	6	01015	ADRVAL55	XMT ERRLEN,R1	R1 <- ERROR CODE
5405	06150	7	04646	JMP	CMDERR	
5406				*		
5407	06151	6	00017	ADRVAL60	XMT 00001111B,AUX	AUX <- MASK
5408	06152	2	04403	AND	R4(4),R3	R3 <- COIL RAM CONFIGURATION
5409	06153	7	06132	JMP	ADRVAL35	
5410				*		
5411	06154	5	05145	ADRVAL80	NZT R5,ADRVAL50	SCRATCHPAD ADDRHI.EQ.0
5412				*		
5413	06155	6	17233	ADRVALX	XMT CMD02,IVR	SEND BACK -DATA LENGTH IN R2
5414	06156	6	00377	XMT	-1,AUX	+1
5415	06157	0	37402	MOV	LENFLD,R2	
5416	06160	6	17027	XMT	SAVER5,IVR	SEND BACK ADDRESS IN (R5,R6)
5417	06161	0	02702	MOV	R2(7),R2	+1 - HAVE TO SHIFT LENFLD
5418	06162	0	37505	MOV	30H,S,R5	DON'T READ FIELD
5419	06163	6	17030	XMT	SAVER6,IVR	
5420	06164	3	02002	XOR	R2,R2	*1 R2<- -LENGTH - 1
5421	06165	0	37006	MOV	R8,R6	
5422	06166	6	00001	XMT	1,AUX	
5423	06167	1	02002	ADD	R2,R2	R2<- -DATA LENGTH
5424	06170	7	07176	RTN		
5425				END	ADRVAL	
5427	06171			PROC	LENVAL	
5428				*		
5429				***SUBROUTINE LENVAL		
5430				*		
5431				***THIS SUBROUTINE VALIDATES THE LENGTH FIELD AND THE LENGTH BYTE		
5432				* IN A COMMAND WITH A VARIABLE LENGTH FIELD.		
5433				*		
5434				***CALLING SEQUENCE:		
5435				*		
5436				* CALL LENVAL		
5437				* ON ENTRY, AUX = WHAT TO ADD TO LENGTH BYTE TO GET		
5438				* DATA LENGTH		
5439				* R2 CONTAINS -DATA LENGTH		
5440				* IVL = IVISPD+IVOSPD		
5441				*		
5442				***RETURNS:		
5443				*		
5444				* TO CALLER IF NO ERROR		
5445				* TO CMDERR IF ERROR		
5446				*		
5447				* IN ADDITION, LENVAL PUTS THE DATA LENGTH INTO INNUM		
5448				*		


```

5449 06171 6 17234 *XMT CMD03,IVR ADDR OF LENGTH BYTE
5450 06172 1 02000 ADD R2,AUX
5451 06173 1 37000 ADD RB,AUX
5452 06174 5 00203 NZT AUX,LENVAL10 IF AUX .NE. 0, ERROR
5453 06175 6 00377 XMT -1,AUX NOW WRITE OUT INNUM
5454 06176 3 02003 XOR R2,R3 WITHOUT DISTURBING R1,R2,R5 OR R6
5455 06177 6 00001 XMT 1,AUX
5456 06200 6 17265 XMT INNUM,IVR
5457 06201 1 03037 ADD R3,RB
5458 06202 7 07176 RTN
5459 06203 6 01015 LENVAL10 XMT ERRLEN,R1 LENGTH ERROR CODE
5460 06204 7 04646 JMP CMDERR
5461 END LENVAL
5463 06205 PROC LENZERO
5464 *
5465 ***CHECK THAT THE LENGTH NIBBLE IN THE COMMAND BYTE = 0
5466 *
5467 06205 6 17233 XMT CMD02,IVR ADDRESS OF COMMAND BYTE
5468 06206 6 07020 XMT IVISPD,IVL SELECT SPD READ
5469 06207 0 37401 MOV 30H,4,R1 READ LENGTH
5470 06210 5 01212 NZT R1,LENZERO1
5471 06211 7 07176 RTN
5472 06212 6 01006 LENZERO1 XMT ERRCMD,R1
5473 06213 7 04646 JMP CMDERR
5474 END LENZERO
5476 06214 PROC CLRDIAG
5477 *
5478 * THIS PROC CLEARS THE FLAGS FOR THE CHECKSUM DIAGNOSTIC
5479 *
5480 06214 6 17034 XMT DIAGSHI,IVR LOAD ADDRESS
5481 06215 6 07021 XMT IVOSPD+IVISPD,IVL
5482 06216 6 02000 XMT 0,R2
5483 06217 0 02037 MOV R2,RB
5484 NOP *1 - WAIT
5485 06220 0 00000 + MOV AUX,AUX
5486 06221 6 17035 XMT DIAGSLO,IVR
5487 06222 0 02037 MOV R2,RB
5488 RTN
5489 END CLRDIAG
5490 06224 PROC INSTINIT
5491 *
5492 * PERFORM INSERT INITIALIZATION. THIS PROC EXISTS ONLY BECAUSE
5493 * THERE ARE TWO INSERT COMMANDS
5494 * ON EXIT, R1 = NOWPAGE, R2 = INPAGE, AND THE INSTAD IS IN SAVER5 AND SAVER6
5495 *
5496 WSP SAVERET,R11 SAVE RETURN ADDR
5496 06224 6 07021 + XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
5496 06225 6 17032 + XMT SAVERET,IVR LOAD ADDRESS
5496 06226 0 11037 + MOV R11,RB WRITE DATA
5497 06227 6 11132 CALL ADRVAL VALIDATE ADDRESS
5498 06230 7 06041
5498 06231 6 00372 XMT -6,AUX INSERT LENGTH SHOULD =
5499 06232 6 11133 CALL LENVAL CMDLEN - 6
5499 06233 7 06171
5500 06234 6 11134 CALL PROTECT CHECK MEMORY PROTECT
5500 06235 7 06541
5501 06236 4 01240 * XEC INSTITAB(R1),4 EXECUTE OFF FIELD TYPE
5502 06237 7 04646 JMP CMDERR GO TO ERROR EXIT
5503 *
5504 06240 7 06245 INSTITAB JMP INSTI010 LOGIC SPACE
5505 06241 6 01005 XMT ERRADI,R1 I/O SPACE - ILLEGAL
5506 06242 6 01005 XMT ERRADI,R1 REGISTER SPACE - ILLEGAL
5507 06243 6 01005 INSTI009 XMT ERRADI,R1 SCRATCHPAD SPACE - ILLEGAL
5508 06244 7 04646 * JMP CMDERR
5509 *
5510 06245 5 05250 INSTI010 NZT R5,INSTI011 IF ADDRESS = (0,0), ERROR
5511 06246 5 06250 NZT R6,INSTI011
5512 06247 7 06243 JMP INSTI009
5513 06250 6 17030 INSTI011 XMT SAVER6,IVR SAVE ADDRESSES
5514 06251 6 07021 XMT IVISPD+IVOSPD,IVL
5515 06252 0 06037 MOV R6,RB
5516 06253 6 03237 XMT CMD06,R3 *1 - SET COMMAND DATA ADDR
5517 06254 6 17027 XMT SAVER5,IVR
5518 06255 0 05037 MOV R5,RB
5519 06256 0 02004 MOV R2,R4 *1 - R4<- LOOP COUNT FOR VALIDATE
5520 06257 6 17024 XMT SAVER2,IVR SAVE LENGTH
5521 06260 0 02037 MOV R2,RB
5522 06261 6 07020 INSTI015 XMT IVISPD,IVL SELECT SPD READ
5523 06262 6 00001 XMT 1,AUX
5524 06263 0 03017 MOV R3,IVR LOAD DATA ADDR
5525 06264 1 03003 ADD R3,R3 *1 - INC ADDR
5526 06265 0 37001 MOV RB,R1 R1<- DATAHI
5527 06266 0 03017 MOV R3,IVR LOAD ADDRESS
5528 06267 1 03003 ADD R3,R3 *1 - INC DATA ADDR
5529 06270 0 37002 MOV RB,R2 R2<- DATALO
5530 06271 6 11135 CALL VALIDATE SEE IF VALID NODE
5530 06272 7 06702
5531 06273 6 00377 XMT -1,AUX IF NOT, R1 = -1
5532 06274 3 01000 XOR R1,AUX
5533 06275 5 00300 NZT AUX,INSTI017
5534 06276 6 01012 XMT ERRNOD,R1
5535 06277 7 04646 JMP CMDERR

```



```

5536 06300 6 00002 INSTI(17 XMT 2,AUX
5537 06301 1 04004 ADD R4,R4 INC LOOP COUNT
5538 06302 5 04261 NZT R4,INSTI015 LOOP UNTIL R4 = 0
5539 RSP SAVER5,R5 RESTORE ADDRESSES
5539 06303 6 17027 + XMT SAVER5,IVR LOAD ADDRESS
5539 06304 6 07021 + XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
5539 06305 0 37005 + MOV RB,R5 READ DATA
5540 RSP SAVER6,R6
5540 06306 6 17030 + XMT SAVER6,IVR LOAD ADDRESS
5540 06307 6 07021 + XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
5540 06310 0 37006 + MOV RB,R6 READ DATA
5541 06311 6 11136 CALL EOLCHECK CHECK THAT INSTAD .LE. EOLAD
5541 06312 7 06516
5542 06313 6 00001 XMT 00000001B,AUX CALCULATE INPAGE = ADDR/128
5543 06314 2 06700 AND R6(7),AUX
5544 06315 1 05702 ADD R5(7),R2 R2<- INPAGE
5545 WSP INPAGE,R2 SAVE
5545 06316 6 07021 + XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
5545 06317 6 17264 + XMT INPAGE,IVR LOAD ADDRESS
5545 06320 0 02037 + MOV R2,RB WRITE DATA
5546 NOP *1 - WAIT
5546 06321 0 00000 + MOV AUX,AUX
5547 06322 6 17065 XMT EOLHI,IVR GET EOL ADDR
5548 06323 6 07021 XMT IVOSPD+IVISPD,IVL *1 - SELECT SPD READ/WRITE
5549 06324 0 37003 MOV RB,R3 R3<- EOLLOCHI
5550 06325 6 17066 XMT EOLLO,IVR
5551 06326 6 00001 XMT 00000001B,AUX *1 - FOR ANDING LATER
5552 06327 0 37004 MOV RB,R4 R4<- EOLLOCLO
5553 06330 2 04700 AND R4(7),AUX CALCULATE EOLPAGE
5554 06331 1 03701 ADD R3(7),R1 R1<- EOLPAGE = NOWPAGE
5555 06332 6 00300 XMT ENT1STM+PASS1STM,AUX SET 1ST ENTRY AND 1ST PASS
5556 06333 6 17263 XMT NOWPAGE,IVR FLAGS
5557 06334 1 01037 ADD R1,RB SAVE FLAGS AND NOWPAGE
5558 NOP *1 - WAIT
5558 06335 0 00000 + MOV AUX,AUX
5559 06336 6 17265 XMT INNUM,IVR GET INNUM (INSERT LENGTH)
5560 NOP *1 - WAIT
5560 06337 0 00000 + MOV AUX,AUX
5561 06340 0 37000 MOV RB,AUX AUX<- INNUM
5562 06341 1 04004 ADD R4,R4 EOLAD <- EOLAD + INNUM
5563 06342 0 10000 MOV OVF,AUX
5564 06343 1 03003 ADD R3,R3
5565 06344 6 17276 XMT SPDCONF1,IVR CHECK CONFIG
5566 06345 6 00037 XMT 00011111B,AUX *1
5567 06346 2 34506 AND 33H,S,R6 R6<- NUMBER OF LOGIC 256 BYTE PAGES
5568 06347 0 03000 MOV R3,AUX CHECK THAT EOLAD < MAX MEM
5569 06350 3 06006 XOR R6,R6 IF NOT, R3=R6
5570 06351 5 06354 NZT R6,INITI020
5571 06352 6 01021 XMT ERRFUL,R1 TAKE ERROR EXIT
5572 06353 7 04646 JMP CMDERR
5573 06354 6 17066 INITI020 XMT EOLLO,IVR UPDATE EOLAD IN SPD
5574 06355 0 04037 MOV R4,RB
5575 NOP *1 - WAIT
5575 06356 0 00000 + MOV AUX,AUX
5576 06357 6 17065 XMT EOLHI,IVR
5577 06360 0 03037 MOV R3,RB
5578 NOP *1 - WAIT
5578 06361 0 00000 + MOV AUX,AUX
5579 RSP SAVERET,R11 GET RETURN ADDR
5579 06362 6 17032 + XMT SAVERET,IVR LOAD ADDRESS
5579 06363 6 07021 + XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
5579 06364 0 37011 + MOV RB,R11 READ DATA
5580 06365 7 07176 RTN RETURN
5581 END INSTINIT
5583 06366 PROC DLETINIT
5584 *
5585 * INITIALIZE FOR DELETE. THE ONLY REASON THIS PROC EXISTS IS THAT THERE
5586 * ARE TWO DELETE COMMANDS
5587 * ON RETURN, R2 = DLNUM, (R3,R4) = DLSTAD
5588 *
5589 WSP SAVERET,R11 SAVE RETURN ADDR
5589 06366 6 07021 + XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
5589 06367 6 17032 + XMT SAVERET,IVR LOAD ADDRESS
5589 06370 0 11037 + MOV R11,RB WRITE DATA
5590 06371 6 11137 CALL ADRVAL VALIDATE ADDRESS
5590 06372 7 06041
5591 06373 6 11140 CALL PROTECT CHECK MEMORY PROTECT
5591 06374 7 06541
5592 06375 4 01377 XEC DLETITAR(R1),4 EXECUTE OFF FIELD TYPE
5593 06376 7 04646 JMP CMDERR TAKE ERROR EXIT
5594 *
5595 06377 7 06403 DLETITAR JMP DLETI010 LOGIC SPACE
5596 06400 6 01005 XMT ERRADI,R1 I/O SPACE - ILLEGAL
5597 06401 6 01005 XMT ERRADI,R1 REGISTER SPACE - ILLEGAL
5598 06402 6 01005 XMT ERRADI,R1 SCRATCHPAD SPACE - ILLEGAL
5599 *
5600 06403 6 07021 DLETI010 XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
5601 06404 6 00377 XMT -1,AUX SAVE +DLNUM
5602 06405 6 17265 XMT DLNUM,IVR LOAD ADDRESS
5603 06406 3 02002 XOR R2,R2 R2<- DLNUM -1
5604 06407 6 00001 XMT 1,AUX
5605 06410 1 02002 ADD R2,R2
5606 06411 0 02037 MOV R2,RB
5607 06412 0 06004 MOV R6,R4 (R3,R4)<- DLSTAD
5608 06413 0 05003 MOV R5,R3
5609 06414 0 02000 MOV R2,AUX MAKE SURE DLSTAD + DLNUM .LE. EOLAD

```

```

5610 06415 1 06006      ADD    R6,R6          (R5,R6)= DLSTAD+DLNUM
5611 06416 0 10000      MOV    OV,F,AUX
5612 06417 1 05005      ADD    R5,R5
5613 06420 6 11141      CALL   EOLCHECK
      06421 7 06516
5614                                RSP    DLNUM,R2          EOLCHECK DESTROYED R2
5614 06422 6 17265      XMT    DLNUM,IVR      LOAD ADDRESS
5614 06423 6 07021      XMT    IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
5614 06424 0 37002      MOV    RB,R2          READ DATA
5615                                RSP    SAVERET,R11      GET RETURN ADDR
5615 06425 6 17032      XMT    SAVERET,IVR   LOAD ADDRESS
5615 06426 6 07021      XMT    IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
5615 06427 0 37011      MOV    RB,R11        READ DATA
5616 06430 7 07176      RTN
5617                                END    DLETINIT
5619 06431                                PROC  NULLFILL
5620                                *
5621                                * THIS PROCEDURE FILLS A REGION OF LOGIC WITH EITHER NULLS OR EOLS
5622                                * DEPENDING ON WHICH ENTRY POINT IS CALLED
5623                                * ON ENTRY, (R5,R6) = START ADDR
5624                                *          R1 = COUNT
5625                                * R3, R4 ARE DESTROYED
5626                                *
5627 06431 6 03134      FILL00 XMT    NODENULL.L.2,R3  R3<- NULLNODE HI
5628 06432 7 06434      JMP    FILL01
5629                                ENTRY  EOLFILL
5630 06433 6 03004      XMT    NODEEOL.L.2,R3  R3<- FOL HI
5631 06434 6 07003      FILL01 XMT    IVOLRLO,IVL   SELECT LOGIC ADDRLO
5632 06435 0 06027      MOV    R6,LB
5633 06436 6 07004      XMT    IVOLRHI,IVL   SELECT LOGIC ADDRHI
5634 06437 0 05027      MOV    R5,LB
5635 06440 6 00376      XMT    -2,AUX         *1 - TO DEC COUNT
5636 06441 6 04000      FILL02 XMT    0,R4          *2, LOOP*2, R4<- EOL/NULL LO
5637 06442 6 07011      XMT    IVOLRDAT,IVL  *3
5638 06443 0 03027      MOV    R3,LB         EOL/NULL HI -> LOGIC
5639 06444 6 07000      XMT    IVOCTRL,IVL  *1 - SELECT CTRL
5640 06445 6 27300      XMT    CTRLINCL,CTRLREG
5641 06446 6 07011      XMT    IVOLRDAT,IVL  *1 - SELECT LOGIC WRITE
5642 06447 1 01001      ADD    R1,R1         *2 - DEC COUNT
5643                                NOP          *3 - WAIT
5643 06450 0 00000      +     MOV    AUX,AUX
5644 06451 0 04027      MOV    R4,LB         EOL/NULL LO
5645 06452 6 07000      XMT    IVOCTRL,IVL  *1
5646 06453 6 27300      XMT    CTRLINCL,CTRLREG  INC ADDR
5647 06454 5 01041      NZT   R1,FILL02     *1 - LOOP ON COUNT
5648 06455 7 07176      RTN
5649                                END    NULLFILL
5651 06456                                PROC  INLOOP
5652                                *
5653                                * THIS PROCEDURE PERFORMS A MOVE OF DATA FOR THE INSERT COMMAND
5654                                * ON ENTRY, (R3,R4) = TOADDR
5655                                *          (R5,R6) = FROMADDR
5656                                *          R1 = COUNT
5657                                *
5658 06456 5 01060      INLOOP00 NZT   R1,INLOOP01  IF R1 = 0, DO NOTHING
5659 06457 7 06502      JMP    INLOOP04
5660 06460 6 00377      INLOOP01 XMT    -1,AUX        SET AUX FOR DECREMENTING
5661 06461 6 07004      INLOOP02 XMT    IVOLRHI,IVL  SELECT LOGIC ADDRHI
5662 06462 0 05027      MOV    R5,LB         FROMADDR HI
5663 06463 6 07003      XMT    IVOLRLO,IVL  LOGIC ADDRLO
5664 06464 0 06027      MOV    R6,LB         FROMADDR LO
5665 06465 1 06006      ADD    R6,R6         *1 - DEC ADDRLO (CAN'T UNDERFLOW)
5666 06466 1 01001      ADD    R1,R1         *2 - DEC COUNT
5667                                NOP          *3 - WAIT
5667 06467 0 00000      +     MOV    AUX,AUX
5668 06470 0 37002      MOV    RB,R2          READ DATA
5669 06471 0 04027      MOV    R4,LB         TOADDR LO
5670 06472 6 07004      XMT    IVOLRHI,IVL  SELECT LOGIC ADDRHI
5671 06473 0 03027      MOV    R3,LB         SET TOADDR HI
5672 06474 5 04076      NZT   R4,INLOOP03  *1 - CHECK IF R4 WILL UNDERFLOW
5673 06475 1 03003      ADD    R3,R3         IF SO, DEC R3
5674 06476 1 04004      INLOOP03 ADD    R4,R4         *2 - DEC TOADDR
5675 06477 6 07011      XMT    IVOLRDAT,IVL  *3 - SELECT LOGIC WRITE
5676 06500 0 02027      MOV    R2,LB         WRITE DATA
5677 06501 5 01061      NZT   R1,INLOOP02  LOOP ON COUNT
5678 06502 7 07176      INLOOP04 RTN    RETURN
5679                                END    INLOOP
5681 06503                                PROC  UPDTLCHK
5682                                *
5683                                * THIS PROCEDURE UPDATES THE LOGIC CHECKSUM.
5684                                * ON ENTRY, AUX = VALUE TO ADD TO THE CHECKSUM
5685                                * R4 IS DESTROYED
5686                                *
5687 06503 6 04000      XMT    SYSLRCHL,R4   LOGIC CHECKSUM LO
5688 06504 6 07003      XMT    IVOLRLO,IVL  LOGIC ADDRLO
5689 06505 0 04027      MOV    R4,LB
5690 06506 6 04000      XMT    SYSLRCHH,R4   CHECKSUM ADDRHI
5691 06507 6 07004      XMT    IVOLRHI,IVL  LOGIC ADDRHI
5692 06510 0 04027      MOV    R4,LB
5693 06511 6 07011      XMT    IVILRDAT+IVOLRDAT,IVL *1 - SELECT LOGIC READ/WRITE
5694                                NOP          *2 - WAIT
5694 06512 0 00000      +     MOV    AUX,AUX
5695                                NOP          *3 - WAIT
5695 06513 0 00000      +     MOV    AUX,AUX
5696 06514 1 37027      ADD    RB,LB         DO THE UPDATE
5697 06515 7 07176      RTN
5698                                END    UPDTLCHK

```



```

5700 06516          PROC   EOLCHECK
5701
5702      ***THIS SUBROUTINE CHECKS THAT (R5,R6) .LE. EOLADDR
5703
5704      *
5705      *RETURNS:
5706      *
5707      *   TO CALLER IF NO ERROR
5708      *   TO CMDERR IF ERROR
5709      *   R2 IS DESTROYED
5710      *
5711      *
5712      *
5713      *
5714      *
5715      *
5716      *
5717      *
5718      *
5719      *
5720      *
5721      *
5722      *
5723      *
5724      *
5725      *
5726      *
5727      *
5728      *
5729      *
5730      *
5731      *
5732      *
5733      *
5734      *
5735      *
5736      *
5737      *
5738      *
5739      *
5740      *
5741      *
5742      *
5743      *
5744      *
5745      *
5746      *
5747      *
5748      *
5749      *
5750      *
5751      *
5752      *
5753      *
5754      *
5755      *
5756      *
5757      *
5758      *
5759      *
5760      *
5761      *
5762      *
5763      *
5764      *
5765      *
5766      *
5767      *
5768      *
5769      *
5770      *
5771      *
5772      *
5773      *
5774      *
5775      *
5776      *
5777      *
5778      *
5779      *
5780      *
5781      *
5782      *
5783      *
5784      *
5785      *
5786      *
5787      *
5788      *
5789      *
5790      *
5791      *
5792      *
5793      *
5794      *
5795      *
5796      *
5797      *
5798      *
5799      *

```

PROC EOLCHECK
 ***THIS SUBROUTINE CHECKS THAT (R5,R6) .LE. EOLADDR
 *RETURNS:
 * TO CALLER IF NO ERROR
 * TO CMDERR IF ERROR
 * R2 IS DESTROYED
 *
 XMT EOLHI,IVR EOL ADDRHI
 XMT IVISPD+IVOSPD,IVL *1 - DO SELECTS
 XMT -1,AUX
 XOR RB,R2 R2<- -EOLLOCHI-1
 XMT 1,AUX
 ADD R2,AUX AUX<- -EOLLOCHI
 ADD R5,R2
 XMT 10000000B,AUX CHECK IF R2 NEGATIVE
 AND R2,AUX
 NZT AUX,EOLCHK01 IF SO, OK
 NZT R2,EOLERR IF R2 .NE. 0, ERROR
 XMT EOLLO,IVR LOAD ADDRESS
 XMT -1,AUX
 XOR RB,AUX AUX<- -EOLLOCLO - 1
 ADD R6,R2
 NZT OVF,EOLERR IF OVF SET, ERROR
 EOLCHK01 RTN ELSE, RETURN
 EOLERR XMT ERRADI,R1
 JMP CMDERR
 END EOLCHECK
 PROC PROTECT

*****SUBROUTINE PROTECT
 *
 ***CHECKS FOR MEMORY PROTECT FAULTS
 *
 ***RETURNS:
 *
 * TO CALLER IF MEMORY PROTECT CLEAR
 * TO CMDERR IF MEMORY PROTECT SET
 *
 ***REGISTER USAGE:
 *
 * R1 - NOT USED
 * R2 - NOT USED
 * R3 - NOT USED
 * R4 - NOT USED
 * R5 - NOT USED
 * R6 - NOT USED
 * R11 - LINKAGE
 * AUX - NOT USED
 *

 PROTECT0 XMT IVISTAT,IVL SELECT PORT
 NZT STATEMB,PROTECT1 BRANCH ON MEMORY PROTECT
 RTN EXIT
 *
 PROTECT1 XMT ERRMEM,R1 R1 <- ERROR FLAG
 JMP CMDERR EXIT ON ERROR
 END PROTECT
 PROC PWRotate

*****SUBROUTINE PWRotate
 *
 ***ROTATES POWER OUTPUT AND STORE POWER BYTES
 *
 ***REGISTER USAGE:
 *
 * R1 - PRESERVED
 * R2 - PRESERVED
 * R3 - POWER BYTE
 * R4 - COUNTER
 * R5 - SCRATCH
 * R6 - SCRATCH
 * R11 - LINKAGE
 * AUX - SCRATCH
 *

 PWRotate EQU *
 XEC PWRCTAB1-1(R4),8 SET UP MASK
 XEC PWRCTAB2-1(R4),8 MASK AND ROTATE POWER
 *
 XMT POWER,IVR LOAD SPD ADDRESS
 XMT IVISPD+IVOSPD,IVL *1 - SELECT SPD READ/WRITE
 NZT RB,PWRotate0 BRANCH IF POWER FOR THIS NETWORK
 JMP PWRotateX ELSE, EXIT

***TABLE 1 - SET UP MASK
 *
 PWRotate1 XMT 11111111B,AUX 8 NODES PER COLUMN
 XMT 11111110B,AUX 7 NODES PER COLUMN
 XMT 11111100B,AUX 6 NODES PER COLUMN
 XMT 11111000B,AUX 5 NODES PER COLUMN
 XMT 11110000B,AUX 4 NODES PER COLUMN
 XMT 11100000B,AUX 3 NODES PER COLUMN
 XMT 11000000B,AUX 2 NODES PER COLUMN
 XMT 10000000B,AUX 1 NODE PER COLUMN


```

5800
5801
5802
5803 06564 2 03003
5804 06565 2 03703
5805 06566 2 03603
5806 06567 2 03503
5807 06570 2 03403
5808 06571 2 03303
5809 06572 2 03203
5810 06573 2 03103
5811
5812 06574 6 17060
5813 06575 6 00001
5814 06576 0 37005
5815 06577 1 37037
5816
5816 06600 0 00000
5817 06601 0 05017
5818 06602 0 03037
5819
5820 06603 6 04010
5821 06604 7 07176
5822
5824
5825 06605
5826
5827
5828
5829
5830
5831
5832
5833
5834
5835
5836
5837
5838
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5850
5851
5851 06605 6 07021
5851 06606 6 17031
5851 06607 0 11037
5852 06610 0 06011
5853 06611 6 07021
5854 06612 6 00365
5855 06613 6 17010
5856 06614 0 00037
5857
5858 06615 0 03603
5859 06616 6 00003
5860 06617 2 04600
5861 06620 3 03003
5862 06621 6 00374
5863 06622 2 04604
5864
5865 06623 6 00377
5866 06624 3 04006
5867 06625 3 03005
5868 06626 6 00001
5869 06627 1 06006
5870 06630 0 10000
5871 06631 1 05005
5872
5873
5874 06632 6 00001
5875 06633 1 37000
5876 06634 5 00236
5877 06635 7 06666
5878
5879 06636 0 00037
5880 06637 0 11000
5881 06640 1 11011
5882 06641 0 01000
5883 06642 1 01001
5884 06643 0 10000
5885 06644 1 11011
5886 06645 0 02000
5887 06646 1 02002
5888 06647 0 10000
5889 06650 1 01000
5890
5891 06651 1 06001
5892 06652 0 10000

```

```

*
***TABLE 2 - ROTATE AND MASK POWER
*
PWR0TAB2 AND R3,R3 8 NODES PER COLUMN
AND R3(7),R3 7 NODES PER COLUMN
AND R3(6),R3 6 NODES PER COLUMN
AND R3(5),R3 5 NODES PER COLUMN
AND R3(4),R3 4 NODES PER COLUMN
AND R3(3),R3 3 NODES PER COLUMN
AND R3(2),R3 2 NODES PER COLUMN
AND R3(1),R3 1 NODE PER COLUMN
*
PWR0T010 XMT POWERPTR,IVR LOAD ADDRESS
XMT 1,AUX *1 - AUX <- INCREMENT
MOV RB,R5 R5 <- POINTER
ADD RB,RB UPDATE POINTER
NOP *1 - WAIT
MOV AUX,AUX
MOV R5,IVR LOAD ADDRESS
MOV R3,RB WRITE POWER BYTE TO BUFFER
*
PWR0TX XMT 8,R4 INITIALIZE ROTATE COUNTER
RTN EXIT
END PWR0TATE
*
PROC DIVIDE
IF ENHANCE
*
DIVIDE SUBROUTINE
*
REGISTER USE:
R1 - DIVIDEND ON ENTRY, REMAINDER/QUOTIENT ON EXIT
R2 - DIVIDEND ON ENTRY, QUOTIENT ON EXIT
R3 - DIVISOR
R4 - DIVISOR
R5 - SCRATCH
R6 - DIVIDEND ON ENTRY, REMAINDER ON EXIT
AUX - SCRATCH
R11 - SUBROUTINE LINKAGE
*
SCRATCH PAD USE: CALCNT, SAVER11
*
ON ENTRY:
DIVIDEND [R6,R1,R2]
DIVISOR [R3,R4]
*
ON EXIT:
REMAINDER IN [R6,R1(7-2)]
QUOTIENT IN [R1(1-0),R2]
*
DIVIDEND0 WSP SAVER11,R11 SAVE SUBROUTINE LINK
XMT IVISPD+IVOSPD,IVL SELECT SPD READ/WRITE
XMT SAVER11,IVR LOAD ADDRESS
MOV R11,RB WRITE DATA
MOV R6,R11 HI DIVIDEND => R11
XMT IVOSPD+IVISPD,IVL SELECT SCRATCH PAD READ/WRITE
XMT -11,AUX
XMT CALCNT,IVR
MOV AUX,RB INIT COUNT
*
MOV R3(6),R3 NORMALIZE DIVISOR
XMT 011B,AUX
AND R4(6),AUX [R3,R4] <= [R3,R4].ROTATED LEFT.2
XOR R3,R3
XMT 1111100B,AUX
AND R4(6),R4
*
DIVIDE10 XMT -1,AUX DIVISOR HOLD <= -DIVISOR
XOR R4,R6
XOR R3,R5
XMT 1,AUX
ADD R6,R6
MOV OV,F,AUX
ADD R5,R5
*
DIVIDE20 XMT 1,AUX COUNT DOWN ON LOOP
ADD RB,AUX
NZT AUX,DIVIDE30
JMP DIVIDE90
*
DIVIDE30 MOV AUX,RB ELSE, UPDATE COUNT
MOV R11,AUX MULTIPLY DIVIDEND BY 2
ADD R11,R11
MOV R1,AUX [R11,R1,R2] <= [R11,R1,R2].ROTATE LEFT
ADD R1,R1
MOV OV,F,AUX
ADD R11,R11
MOV R2,AUX
ADD R2,R2
MOV OV,F,AUX
ADD R1,AUX
*
ADD R6,R1 COMPARE DIVIDEND TO DIVISOR HOLD
MOV OV,F,AUX

```

```

5893 06653 1 05000      ADD    R5,AUX          [R11,R1] <= [R11,R1] + [R5,R6]
5894 06654 1 11011      ADD    R11,R11
5895 06655 6 00200      XMT    10000000B,AUX  CHECK SIGN OF RESULT
5896 06656 2 11000      AND    R11,AUX
5897 06657 5 00263      NZT    AUX,DIVIDE40   IF RESULT.LT.0 GOTO DIVIDE40
5898
5899 06660 6 00001      XMT    1,AUX          ELSE, CONTINUE
5900 06661 1 02002      ADD    R2,R2          SET QUOTIENT BIT TO '1'
5901 06662 7 06623      JMP    DIVIDE10       LOOP, SET DIVR HOLD = -DIVR
5902
5903 06663 0 03005  DIVIDE40 MOV    R3,R5          SET DIVR HOLD = DIVR
5904 06664 0 04006      MOV    R4,R6          (QUOTIENT BIT = 0)
5905 06665 7 06632      JMP    DIVIDE20       LOOP
5906
5907
5908 06666 6 00200  DIVIDE90 XMT    10000000B,AUX  CHECK FOR NEGATIVE REMAINDER
5909 06667 2 11000      AND    R11,AUX
5910 06670 5 00276      NZT    AUX,DIVIDE95
5911
5912 06671 0 04000      MOV    R4,AUX          FIX REMAINDER
5913 06672 1 01001      ADD    R1,R1          HI DIVIDEND = HI DIVIDEND + DIVISOR
5914 06673 0 10000      MOV    0VF,AUX        [R11,R1] <= [R11,R1] + [R3,R4]
5915 06674 1 03000      ADD    R3,AUX
5916 06675 1 11011      ADD    R11,R11
5917
5918 06676 6 17031  DIVIDE95 XMT    SAVER11,IVR  GET SUBROUTINE LINK
5919 06677 0 11006      MOV    R11,R6
5920 06700 0 37011      MOV    RB,R11
5921
5922 06701 7 07176      ENDIF
5923      RTN
5924
5926 06702      END    DIVIDE
5927      PROC  VALIDATE
5928
5929 ***VALIDATE NODE SUBROUTINE
5930
5931      REGISTER USE:
5932      R1 = <ON ENTRY> FIRST BYTE OF NODE.
5933      <ON EXIT> ENTRY VALUE, IF VALID.  -1 IF INVALID.
5934      R2 = 2ND BYTE OF NODE
5935      R3 = UNUSED
5936      R4 = UNUSED
5937      R5 = SCRATCH
5938      R6 = SCRATCH
5939      R11 = SUBROUTINE LINK
5940      AUX = SCRATCH
5941
5942      SCRATCH PAD USE:  DIVDFLAG,SPDCONF2
5943
5944
5945      NOTE:
5946      IF NODE IS INVALID (EITHER NODE TYPE OR REFERENCE VALUE)
5947      R1 IS SET TO '-1' ON EXIT.
5948      IF NODETYPE '11111B' IS EXECUTED AS A VALID NODE,
5949      THEN MODIFICATIONS MAY BE NEEDED (BECAUSE THE FIRST BYTE
5950      OF THE NODE MAY POSSIBLY BE '-1')
5951
5952
5953 06702 6 00037  VALID000 XMT    NODETYPM,AUX  VECTOR OFF NODE TYPE
5954 06703 2 01200      AND    R1(2),AUX
5955 06704 4 00305      XEC    VALIDTAB(AUX),32
5956
5957
5958
5959 06705 7 06745  VALIDTAB JMP    VAL00000      START OF NETWORK
5960 06706 7 06745      JMP    VAL01000      END OF LOGIC
5961 06707 7 06754      JMP    VAL02000      END OF COLUMN
5962 06710 7 06764      JMP    VAL03000      NORMALLY-OPEN RELAY
5963 06711 7 06764      JMP    VAL04000      NORMALLY-CLOSED RELAY
5964 06712 7 07006      JMP    VAL05000      POSITIVE-GOING TRANSITIONAL
5965 06713 7 07006      JMP    VAL06000      NEGATIVE-GOING TRANSITIONAL
5966 06714 7 07012      JMP    VAL07000      COIL
5967 06715 7 07012      JMP    VAL08000      LATCHED COIL
5968 06716 7 07012      JMP    VAL09000      DISABLED COIL
5969 06717 7 07012      JMP    VAL10000     DISABLED LATCHED COIL
5970 06720 7 06747      JMP    VAL11000     HORIZONTAL OPEN
5971 06721 7 06747      JMP    VAL12000     HORIZONTAL CLOSED
5972 06722 7 07021      JMP    VAL13000     PRESET/CALCULATE-B-NODE CONSTANT
5973 06723 7 07036      JMP    VAL14000     PRESET/CALCULATE-B-NODE REGISTER
5974 06724 7 07112      JMP    VAL15000     COUNTER
5975 06725 7 07112      JMP    VAL16000     TIMER 1.00
5976 06726 7 07112      JMP    VAL17000     TIMER 0.10
5977 06727 7 07112      JMP    VAL18000     TIMER 0.01
5978 06730 7 07121      JMP    VAL19000     CONVERT NODE
5979 06731 7 07025      JMP    VAL20000     CALCULATE-C-NODE CONSTANT
5980 06732 7 07133      JMP    VAL21000     CALCULATE-C-NODE REGISTER
5981 06733 7 07154      JMP    VAL22000     CALCULATE - D NODE
5982 06734 7 06745      JMP    VAL23000     NULL NODE
5983 06735 7 07174      JMP    VAL24000     UNASSIGNED - ERROR
5984 06736 7 07174      JMP    VAL25000     UNASSIGNED - ERROR
5985 06737 7 07174      JMP    VAL26000     UNASSIGNED - ERROR
5986 06740 7 07174      JMP    VAL27000     UNASSIGNED - ERROR
5987 06741 7 07174      JMP    VAL28000     UNASSIGNED - ERROR
5988 06742 7 07174      JMP    VAL29000     UNASSIGNED - ERROR
5989 06743 7 07174      JMP    VAL30000     UNASSIGNED - ERROR
5990 06744 7 07174      JMP    VAL31000     UNASSIGNED - ERROR

```


5992		006745	VAL00000 EQU	*	START OF NETWORK NODE
5993		006745	VAL01000 EQU	*	END OF LOGIC NODE
5994		006745	VAL23000 EQU	*	NULL NODE
5995			*		
5996	06745	6 00203	XMT	10000011B,AUX	R1(7,1-0) AND R2 MUST BE .EQ. 0
5997	06746	7 06750	JMP	VAL11010	
5998			*		
5999		006747	VAL11000 EQU	*	HORIZONTAL OPEN NODE
6000		006747	VAL12000 EQU	*	HORIZONTAL CLOSED NODE
6001			*		REFERENCE MUST BE '0'
6002	06747	6 00003	XMT	011B,AUX	CHECK R1 (1-0)
6003	06750	2 01000	VAL11010 AND	R1,AUX	
6004	06751	5 00363	NZT	AUX,VAL02500	INVALID EXIT
6005	06752	5 02363	NZT	R2,VAL02500	CHECK R2
6006	06753	7 07175	JMP	VALIDOK	VALID EXIT
6007			*		
6008			*		
6009	06754	6 00203	VAL02000 XMT	10000011B,AUX	END OF CLOUD NODE
6010	06755	2 01000	AND	R1,AUX	R1(7,1-0) AND R2 (0) MUST BE '0'
6011	06756	5 00363	NZT	AUX,VAL02500	
6012	06757	6 00001	XMT	01B,AUX	
6013	06760	2 02000	AND	R2,AUX	
6014	06761	5 00363	NZT	AUX,VAL02500	
6015	06762	7 07175	JMP	VALIDOK	VALID EXIT
6016			*		
6017	06763	7 07174	VAL02500 JMP	VALIDERR	
6018			*		
6019			*		
6021			*		
6022		006764	VAL03000 EQU	*	NORMALLY OPEN NODE
6023		006764	VAL04000 EQU	*	NORMALLY CLOSED NODE
6024	06764	6 00003	XMT	011B,AUX	
6025	06765	2 01005	AND	R1,R5	REFERENCE TYPE => R5
6026	06766	3 05005	XOR	R5,R5	CHECK FOR SEQUENCER REFERENCE
6027	06767	5 05371	NZT	R5,VAL03010	
6028	06770	7 07175	JMP	VALIDOK	SEQUENCER REF, ANY R2 VALUE VALID
6029			*		
6030	06771	0 02006	VAL03010 MOV	R2,R6	SETUP FOR GENERAL COIL RAM VALIDATE
6031	06772	6 17277	VAL03020 XMT	SPDCONF2,IVR	GET CONFIGURATION FOR COIL RAM
6032	06773	6 07021	XMT	IVOSPD+IVISPD,IVL	SELECT SCRATCH PAD READ/WRITE
6033	06774	0 33305	MOV	34H,3,R5	READ COIL CONFIG
6034	06775	4 05001	NEC	VAL03TAB(R5),5	GET MAX VALUE
6035	06776	1 06006	ADD	R6,R6	
6036	06777	5 10174	NZT	OVF,VALIDERR	REFERENCE IS INVALID
6037	07000	7 07175	JMP	VALIDOK	
6038			*		
6039	07001	6 00000	VAL03TAB XMT	0,AUX	256 COILS
6040	07002	6 00300	XMT	-64,AUX	64 COILS
6041	07003	6 00200	XMT	-128,AUX	128 COILS
6042	07004	6 00377	XMT	-1,AUX	INVALID
6043	07005	6 00100	XMT	-192,AUX	192 COILS
6044			*		
6046			*		
6047			*		
6048			*	TRANSITIONAL NODES, VALIDATE IF ENHANCED SET.	
6049		007006	IF	ENHANCE	
6050		007006	VAL05000 EQU	*	POSITIVE TRANSITIONAL NODE
6051	07006	2 01005	VAL06000 EQU	*	NEGATIVE TRANSITIONAL NODE
6052	07007	3 05005	AND	R1,R5	
6053	07010	5 05371	XOR	R5,R5	
6054	07011	7 07174	NZT	R5,VAL03010	NOT SEQ REF, VALIDATE R2
6055			JMP	VALIDERR	SEQ REF, INVALID
6056			*		
6057			*	ENDIF	
6058			*	END OF TRANSITIONAL NODES VALIDATION	
6059			*		
6060		007012	VAL07000 EQU	*	COIL
6061		007012	VAL08000 EQU	*	LATCHED COIL
6062		007012	VAL09000 EQU	*	DISABLED COIL
6063		007012	VAL10000 EQU	*	DISABLED LATCHED COIL
6064	07012	6 00003	XMT	011B,AUX	
6065	07013	2 01005	AND	R1,R5	REF TYPE CAN BE ONLY OUTPUT AND
6066	07014	4 05015	NEC	VAL07TAB(R5),4	INTERNAL COILS
6067			*		
6068	07015	7 07174	VAL07TAB JMP	VALIDERR	INPUT REF - INVALID
6069	07016	7 06771	JMP	VAL03010	OUTPUT COIL - VALIDATE R2
6070	07017	7 06771	JMP	VAL03010	INTERNAL COIL - VALIDATE R2
6071	07020	7 07174	JMP	VALIDERR	SEQ REF - INVALID
6072			**		
6074			*		
6075		007021	VAL13000 EQU	*	TIMER/COUNTR PRESET CONSTANT NODE
6076			*		AND CALCULATE-B-NODE IF ENHANCED SET.
6077			IF	ENHANCE	
6078			*		
6079			*	ON ENHANCED SET, SET DIVIDEND 'OK' FLAG	
6079	07021	6 00000	CLR	AUX	TIMER/COUNTR PRESET-CALC B NODE CONSTANT
6080			XMT	0,AUX	
6080	07022	6 07021	WSP	DIVDFLAG,AUX	SET DIVIDEND 'OK' FLAG
6080	07023	6 17014	XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
6080	07024	0 00037	XMT	DIVDFLAG,IVR	LOAD ADDRESS
6081		007025	MOV	AUX,R8	WRITE DATA
6082			VAL20000 EQU	*	CALC C NODE CONSTANT NODE
6083			*	END OF	CONDITIONAL ASSEMBLY AREA
6083			ENDIF		
6084	07025	6 00003	XMT	011B,AUX	
6085	07026	2 01005	AND	R1,R5	CONSTANT MUST BE .LT.1000
6086	07027	6 00030	XMT	NEG1000L,AUX	

6087	07030	1	02000		ADD	R2,AUX	
6088	07031	6	0037*		XMT	NEG1000H,AUX	
6089	07032	1	10000		ADD	OVF,AUX	
6090	07033	1	05000		ADD	R5,AUX	
6091	07034	5	10174		NZT	OVF,VALIDERR	CONSTANT.GE.1000, INVALID
6092	07035	7	07175		JMP	VALIDOK	CONSTANT.LT.1000, VALID
6093				*			
6095				*			
6096	07036	6	00003	VAL14000	XMT	011R,AUX	VECTOR TO REG TYPE
6097	07037	2	01000		AND	R1,AUX	
6098	07040	4	00041		XEC	VAL14TAB(AUX),4	
6099				*			
6100	07041	7	07045	VAL14TAB	JMP	VAL14100	HOLDING REGS
6101	07042	7	07070		JMP	VAL14200	INPUT REGS
6102	07043	7	07077		JMP	VAL14300	DUMMY REGS
6103	07044	7	07174		JMP	VALIDERR	INVALID
6104				*			
6105					IF	ENHANCE-1	
6106					ENDIF		
6107					IF	ENHANCE	
6108				*			
6109				*			ENHANCED SET, VALIDATE HOLDING REG,
6110	07045	6	17277	VAL14100	XMT	SPDCONF2,IVR	THEN SET DIVIDEND 'OK' FLAG.
6111	07046	0	02006		MOV	R2,R6	GET SYSTEM CONFIGURATION
6112	07047	6	07021		XMT	IVOSPD+IVOSPD,IVL	
6113	07050	0	33305		MOV	34H,3,R5	
6114				*			
6115	07051	4	05063		XEC	VAL141TB(R5),5	GET VALIDATE VALUE
6116	07052	1	06000		ADD	R6,AUX	
6117	07053	5	10174		NZT	OVF,VALIDERR	INVALID
6118				*			
6119				*			REGISTER IS VALID,
6120				*			CHECK REF AND SET
6121				*			DIVIDEND OK FLAG
6122	07054	6	00001		XMT	1,AUX	
6123	07055	1	06006		ADD	R6,R6	
6124	07056	5	10105		NZT	OVF,VAL14BAD	DIVIDEND BAD
6125	07057	4	05063		XEC	VAL141TB(R5),5	
6126	07060	1	06000		ADD	R6,AUX	
6127	07061	5	10105		NZT	OVF,VAL14BAD	DIVIDEND BAD
6128	07062	7	07100		JMP	VAL14OK	DIVIDEND GOOD
6129				*			
6130	07063	6	00002	VAL141TB	XMT	2,AUX	256 REGS
6131	07064	6	00302		XMT	-62,AUX	62 REGS
6132	07065	6	00202		XMT	-126,AUX	126 REGS
6133	07066	6	00377		XMT	-1,AUX	INVALID
6134	07067	6	00102		XMT	-190,AUX	190 REGS
6135				*			
6136				*			END OF ENHANCED SET HOLDING REG VALIDATION
6137					ENDIF		
6138				*			
6139	07070	6	00160	VAL14200	XMT	1110000B,AUX	VALIDATE INPUT REG REF
6140	07071	2	02000		AND	R2,AUX	
6141	07072	5	00174		NZT	AUX,VALIDERR	REF > 32, ERROR
6142					IF	ENHANCE-1	
6143					ENDIF		
6144					IF	ENHANCE	
6145				*			ENHANCED SET B NODE INPUT REG VALIDATION
6146				*			MUST TEST FOR VALID DIVIDEND
6147	07073	6	00341		XMT	-31,AUX	CHECK DIVIDEND
6148	07074	1	02000		ADD	R2,AUX	
6149	07075	5	10105		NZT	OVF,VAL14BAD	DIVIDEND BAD
6150	07076	7	07100		JMP	VAL14OK	DIVIDEND OK
6151				*			END OF B NODE INPUT REG ENHANCED VALIDATION
6152					ENDIF		
6153				*			
6154				*			
6155	07077	5	02174	VAL14300	NZT	R2,VALIDERR	DUMMY REG, R2 MUST BE 0
6156				*			
6157				*			
6158					IF	ENHANCE	
6159				*			ENHANCED SET B NODE REGISTER VALIDATION
6160				*			SET DIVIDEND 'OK' FLAG,
6161				VAL140K	CLR	R5	
6161	07100	6	05000		XMT	0,R5	
6162					WSP	DIVDFLAG,R5	SET DIVIDEND 'OK' FLAG
6162	07101	6	07021		XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
6162	07102	6	17014		XMT	DIVDFLAG,IVR	LOAD ADDRESS
6162	07103	0	05037		MOV	R5,RB	WRITE DATA
6163	07104	7	07175		JMP	VALIDOK	
6164				*			
6165	07105	6	05377	VAL14BAD	XMT	-1,R5	
6166					WSP	DIVDFLAG,R5	SPOIL DIVIDEND 'OK' FLAG
6166	07106	6	07021		XMT	IVISPD+IVOSPD,IVL	SELECT SPD READ/WRITE
6166	07107	6	17014		XMT	DIVDFLAG,IVR	LOAD ADDRESS
6166	07110	0	05037		MOV	R5,RB	WRITE DATA
6167	07111	7	07175		JMP	VALIDOK	
6168				*			END OF ENHANCED SET B NODE REG VALIDATION
6169					ENDIF		
6171				*			
6172		007112		VAL15000	EQU	*	COUNTER
6173		007112		VAL16000	EQU	*	TIMER 1.00
6174		007112		VAL17000	EQU	*	TIMER 0.10
6175		007112		VAL18000	EQU	*	TIMER 0.01
6176	07112	6	00003		XMT	011R,AUX	
6177	07113	2	01000		AND	R1,AUX	

```

6178 07114 4 00115      XEC   VAL15TAB(AUX),4  BRANCH TO REFERENCE VALIDATION
6179      *
6180      IF   ENHANCE-1
6181      ENDIF
6182      IF   ENHANCE
6183      *
6184      *
6185 07115 7 07142      VAL15TAB JMP   VAL21100  ENHANCED SET VECTOR TABLE FOR TIMER/COUNTER
6186 07116 7 07174      JMP   VALIDERR  REFERENCE TYPE.
6187 07117 7 07152      JMP   VAL21300  HOLDING REG
6188 07120 7 07174      JMP   VALIDERR  INPUT REG - INVALID
6189      *
6190      *
6191      *
6192      *
6193      *
6194      *
6195      *
6196      *
6197      *
6198 07121 6 00001      VAL19000 XMT   1,AUX  CONVERT NODE
6199 07122 1 02000      ADD   R2,AUX  CHECK FOR DUMMY REG
6200 07123 5 10175      NZT  OVF,VALIDOK DUMMY REF, VALID
6201      *
6202 07124 2 01000      AND   R1,AUX  GET REF TYPE, REGOR DISCRETE
6203 07125 4 00131      XEC   VAL19TAB(AUX),2 GET OFFSET
6204 07126 1 02006      ADD   R2,R6  ADD OFFSET TO REF
6205 07127 5 10174      NZT  OVF,VALIDERR
6206 07130 7 06772      JMP   VAL03020 COMPLETE VALIDATION
6207      *
6208 07131 6 00014      VAL19TAB XMT   12,AUX  DISCRETE OFFSET
6209 07132 6 00002      XMT   2,AUX  REGISTER OFFSET
6210      *
6211 07133 6 00003      VAL21000 XMT   011B,AUX  C NODE REGISTER
6212 07134 2 01000      AND   R1,AUX
6213 07135 4 00136      XEC   VAL21TAB(AUX),4 VECTOR TO REF VALIDATION
6214      *
6215 07136 7 07142      VAL21TAB JMP   VAL21100  HOLDING REG REF
6216 07137 7 07146      JMP   VAL21200  INPUT REG REF
6217 07140 7 07152      JMP   VAL21300  DUMMY REG REF
6218 07141 7 07174      JMP   VALIDERR  INVALID
6219      *
6220 07142 6 00002      VAL21100 XMT   2,AUX  ADD OFFSET TO REF
6221 07143 1 02006      ADD   R2,R6
6222 07144 5 10174      NZT  OVF,VALIDERP  REF.GT.254, INVALID
6223 07145 7 06772      JMP   VAL03020
6224      *
6225 07146 6 00340      VAL21200 XMT   11100000B,AUX INPUT REG REF
6226 07147 2 02000      AND   R2,AUX  MUST BE .LT.32
6227 07150 5 00174      NZT  AUX,VALIDERR
6228 07151 7 07175      JMP   VALIDOK
6229      *
6230 07152 5 02174      VAL21300 NZT  R2,VALIDERR  DUMMY REF, R2.MUST.BE 0
6231 07153 7 07175      JMP   VALIDOK
6232      *
6233      *
6234      *
6235      *
6236 07154 6 00003      VAL22000 XMT   011B,AUX  CALCULATE NODES
6237 07155 2 01000      AND   R1,AUX  VECTOR TO HANDLER
6238 07156 4 00157      XEC   VAL22TAB(AUX),5
6239      *
6240 07157 7 07142      VAL22TAB JMP   VAL21100  CALC-ADD
6241 07160 7 07142      JMP   VAL21100  CALC-SUB
6242 07161 7 07163      JMP   VAL22100  CALC-MPX
6243 07162 7 07167      JMP   VAL22200  CALC-DIV
6244      *
6245 07163 6 00003      VAL22100 XMT   3,AUX  GET OFFSET FOR REF #
6246 07164 1 02006      ADD   R2,R6
6247 07165 5 10174      NZT  OVF,VALIDERR
6248 07166 7 06772      JMP   VAL03020
6249      *
6250      *
6250 07167 6 17014      VAL22200 RSP  DIVDFLAG,AUX  CHECK IF DIVIDEND IS OK
6250 07170 6 07021      + XMT  DIVDFLAG,IVR  LOAD ADDRESS
6250 07171 0 37000      + XMT  IVISPD+IVOSPD,IVL *1 - SELECT SPD READ
6251 07172 5 00174      + MOV  RB,AUX  READ DATA
6252 07173 7 07142      + NZT  AUX,VALIDERR
6253      *
6254      *
6255      *
6256      *
6257      *
6258      *
6259      *
6260      *
6261      *
6262      *
6263      *
6264      *
6265      *
6266      *
6267      *
6268      *
6269      *
6270      *
6271      *
6272      *
6273      *
6274      *

```

```

6275 07174 6 01377 VALIDERR XMT -1,R1 SET ERROR INDICATOR
6276 *
6277 *
6278 ***COMMON EXIT
6279 07175 7 07176 VALIDOK RTN
6280 *
6281 END VALIDATE
6282 END MAIN

```

RETURN TABLE

```

07176 4 11177
07177 7 00122
07200 7 00127
07201 7 00133
07202 7 00150
07203 7 00153
07204 7 00255
07205 7 00432
07206 7 00506
07207 7 00610
07210 7 00617
07211 7 00626
07212 7 00644
07213 7 00663
07214 7 00775
07215 7 01044
07216 7 01150
07217 7 01204
07220 7 01251
07221 7 01375
07222 7 01414
07223 7 01474
07224 7 01505
07225 7 02047
07226 7 02057
07227 7 02101
07230 7 02203
07231 7 02213
07232 7 02375
07233 7 02430
07234 7 02440
07235 7 02575
07236 7 02612
07237 7 02654
07240 7 02662
07241 7 02673
07242 7 02730
07243 7 03070
07244 7 03072
07245 7 03075
07246 7 03167
07247 7 03236
07250 7 03247
07251 7 03333
07252 7 03364
07253 7 03521
07254 7 03564
07255 7 03627
07256 7 03704
07257 7 03770
07260 7 03772
07261 7 04032
07262 7 04042
07263 7 04062
07264 7 04101
07265 7 04111
07266 7 04123
07267 7 04246
07270 7 04262
07271 7 04303
07272 7 04333
07273 7 04364
07274 7 04374
07275 7 04376
07276 7 04405
07277 7 04407
07300 7 04421
07301 7 04424
07302 7 04427
07303 7 04432
07304 7 04441
07305 7 04444
07306 7 04446
07307 7 04521
07310 7 04552
07311 7 04576
07312 7 04603
07313 7 04632
07314 7 04641
07315 7 04717
07316 7 04734
07317 7 05067
07320 7 05076
07321 7 05161

```



```

07322 7 05200
07323 7 05240
07324 7 05312
07325 7 05324
07326 7 05342
07327 7 05363
07330 7 05373
07331 7 06231
07332 7 06234
07333 7 06236
07334 7 06273
07335 7 06313
07336 7 06373
07337 7 06375
07340 7 06422

```

TOTAL ASSEMBLY ERRORS = 10

```

001 JOB P180 : 001 : MOD 01 : REV AX23
002

```

```

\ *****
\ NOTE! EQUATES FOR REVISION LEVEL
\ FOR DEVELOPMENT AND RELEASE; PLEASE

```

```

006
007 / THESE EQUATES DEFINE THE DEVELOPMENT AND RELEASE
008 / DISPLAY OF THE REVISION WHEN THE P180 IS
009 / INITIALIZED.
010
011 / MAJREV IS THE MAJOR REVISION, A,B,C, ETC
012 0041 MAJREV="A"
013
014 / DVR1-3 DEFINE THE DEVELOPMENT REVISIONS.
015 / THESE MUST BE UPDATED
016 / DURING DEVELOPMENT FOR EACH EDIT!
017 / THESE MUST BE SET TO "SPACE"
018 / ON EACH EQUATE WHEN RELEASED!
019 0058 DVR1="X"
020 0032 DVR2="2"
021 0033 DVR3="3"
022

```

```

\ *****

```

```

024 /
025 /
026 / COPYRIGHT, (C) 1978, GOULD-MODICON DIV., ALL
027 / RIGHTS RESERVED. NO PART OF THIS PROGRAM
028 / MAY BE REPRODUCED IN ANY FORM WITHOUT THE
029 / EXPRESS WRITTEN PERMISSION OF GOULD-MODICON DIV.
030 /
031 /
032 /
033 EJECT

```

SUBJOB GLOBAL DEFINITIONS

```

001 /
002 /
003 / ***I/O ADDRESSES
004 /
005 0020 DMA0AD=: 20 / DMA CHN 0 - ADDRESS
006 /DMA0TC=: 21 / DMA CHN 0 - TERM COUNT
007 /DMA1AD=: 22 / DMA CHN 1 - ADDRESS
008 /DMA1TC=: 23 / DMA CHN 1 - TERM COUNT
009 0024 DMA2AD=: 24 / DMA CHN 2 - ADDRESS
010 0025 DMA2TC=: 25 / DMA CHN 2 - TERM COUNT
011 0026 DMA3AD=: 26 / DMA CHN 3 - ADDRESS
012 0027 DMA3TC=: 27 / DMA CHN 3 - TERM COUNT
013 /DMASTA=: 28 / DMA STATUS REGISTER (IN)
014 0028 DMAMOD=: 28 / DMA MODE REGISTER (OUT)
015 /
016 0038 CRTSTA=: 38 / CRT STATUS REGISTER (IN)
017 0038 CRTCTL=: 38 / CRT CONTROL REG (OUT)
018 0039 CRTDAT=: 39 / CRT DATA REGISTER
019 003A SP1STA=: 3A / PORT #1 - STATUS (IN)
020 003A SP1CTL=: 3A / PORT #1 - CONTROL (OUT)
021 003B SP1IN=: 3B / PORT #1 - DATA IN
022 003B SP1OUT=: 3B / PORT #1 - DATA OUT
023 003C SP2STA=: 3C / PORT #2 - STATUS (IN)
024 003C SP2CTL=: 3C / PORT #2 - CONTROL (OUT)
025 003D SP2IN=: 3D / PORT #2 - DATA IN
026 003D SP2OUT=: 3D / PORT #2 - DATA OUT
027 003E PARIN=: 3E / PARALLEL INPUT PORT
028 003E PAROUT=: 3E / PARALLEL OUTPUT PORT
029 /IOTEST=: 3F / SPARE (TEST I/O)
030 EJECT

```

```

001 /
002 /***PARALLEL PORT DEFINITIONS
003 /
004 /***PAROUT
005 /
006 0080 POFWR=: 80 / CRT DISPLAY POWER
007 0040 PDBEEP=: 40 / BEEPER
008 /PORENB=: 20 / PORT 2 RCVR - INT ENB
009 /PDTENB=: 10 / PORT 2 XMIT - INT ENB
010 /POSEL=: 08 / KEYBOARD/PORT 2
011 /POCODE=: 07 / OUTPUT CODE
012 /
013 /***POCODE VALUES, POSEL = 0 (KEYBOARD)
014 /
015 /POROW0=: 00 / KEYBOARD ROW 0
016 /POROW1=: 01 / KEYBOARD ROW 1
017 /POROW2=: 02 / KEYBOARD ROW 2
018 /POROW3=: 03 / KEYBOARD ROW 3
019 /POROW4=: 04 / KEYBOARD ROW 4
020 /POROW5=: 05 / KEYBOARD ROW 5
021 /POROW6=: 06 / KEYBOARD ROW 6
022 /POROW7=: 07 / KEYBOARD ROW 7
023 /
024 /***POCODE VALUES, POSEL = 1 (SERIAL PORT #2 CONTROLS)
025 /
026 0000 POPENB=: 00 / PARITY ENABLE
027 0001 POPEVN=: 01 / EVEN PARITY
028 0002 POZSTP=: 02 / TWO STOP BITS
029 / :03 / NOT USED
030 / :04 / NOT USED
031 / :05 / NOT USED
032 / :06 / NOT USED
033 / :07 / NOT USED
034 /
035 /***PARIN
036 /
037 /***POSEL = 1 (SERIAL PORT TWO STATUS)
038 /
039 0080 PISTAT=: 80
040 EJECT

001 /
002 /***KEYBOARD DATA
003 /
004 /KBDROW=: 08 / NUMBER OF ROWS
005 /KBD CNT=: 07 / CHARACTERS PER ROW
006 /KBDMSK=: 01 / STARTING MASK
007 EJECT

001 /
002 /***CRT CONTROLLER DEFINITIONS
003 /
004 /****COMMANDS
005 /
006 0000 CMDRST=: 00 / RESET AND STOP DISPLAY
007 0020 CMDST=: 20 / START DISPLAY
008 /CMPSTP=: 40 / STOP DISPLAY
009 /CMDRLP=: 60 / READ LIGHT PEN POSITIC
010 0080 CMDCUR=: 80 / LOAD CURSOR POSITION
011 /CMDEI=: A0 / ENABLE INTERRUPT
012 /CMDDI=: C0 / DISABLE INTERRUPT
013 00E0 CMDPRE=: E0 / PRESET COUNTERS
014 /
015 /****SCREEN COMPOSITION BYTES
016 /
017 0043 COMFB1=: 43 / VERTICALLY SPACED ROWS
018 / 68 CHARS PER ROW
019 0014 COMFB2=: 14 / CHARACTER ROWS
020 / ROWS PER SCREEN
021 007B COMFB3=: 7B / UNDERLINE ROW POSITION
022 / LINES PER ROW
023 0036 COMFB4=: 36 / CURSOR FORMAT
024 / 14 CHARS PER HOZ RETRA
025 /
026 /****CRT STATUS PORT DEFINITIONS
027 /
028 / :80 / NOT USED

```

```

029          CRISTE= 40          / INTERRUPT ENABLED
030    0020    CRISIR= 20          / INTERRUPT REQUEST
031          /CRISLE= 10          / LIGHT PEN ACTIVE
032          /CRISCL= 08          / COMMAND INCOMPLETE
033    0004    CRISVE= 04          / VIDEO ENABLED
034          /CRISDU= 02          / DMA UNDERFLOW
035          /CRISDO= 01          / STACK OVERFLOW
036
037          ****CONSOLE DEFINITION - DISABLED
038
039    0000    CONVER= 00          / VERTICAL
040    0000    CONHIZ= 0F          / HORIZONTAL
041
042          /
043          ****BURST MODE
044    000B    BURST= 0B          /
045          EJECT
001
002          /
003          /****DMA CONTROLLER DEFINITIONS
004          /
005          /DMAVER= 00          / DMA VERIFY
006    0080    /DMAWRT= 40          / DMA WRITE
007          /DMARED= 80          / DMA READ
008
009    0080    /DMAMAL= 80          / ENABLE AUTO-LOAD
010          /DMAMTC= 40          / ENABLE TC STOP
011          /DMAMEW= 20          / ENABLE EXTENDED WRITE
012          /DMAMRP= 10          / ENABLE ROTATING PRIORITY
013    0008    /DMAME3= 08          / ENABLE DMA CHANNEL 3
014    0004    /DMAME2= 04          / ENABLE DMA CHANNEL 2
015          /DMAME1= 02          / ENABLE DMA CHANNEL 1
016          /DMAME0= 01          / ENABLE DMA CHANNEL 0
017
018    008C    /DMACMD= DMAMAL+DMAME3+DMAME2 / DMA COMMAND
          EJECT
001
002          /
003          /****SERIAL PORT DEFINITIONS
004          /
005          /****STATUS REGISTER
006          /
007    0080    SPDSR= 80          / DATA SET READY
008    0040    SPSSYN= 40          / SYNC DETECT
009    0020    SPSFE= 20          / FRAMING ERROR
010    0010    SPSOE= 10          / OVERRUN ERROR
011    0008    SPSPE= 08          / PARITY ERROR
012    0004    SPSTE= 04          / TRANSMITTER EMPTY
013    0002    SPSRRY= 02          / RECEIVER READY
014    0001    SPSTRY= 01          / TRANSMITTER READY
015
016          /
017          /****COMMAND DEFINITION
018          /
019    0080    SPCEH= 80          / ENTER HUNT MODE
020    0040    SPCIR= 40          / INTERNAL RESET
021    0020    SPCRTS= 20          / REQUEST TO SEND
022    0010    SPCER= 10          / RESET ERROR FLAGS
023    0008    SPCBRK= 08          / SEND BREAK CHARACTER
024    0004    SPCRE= 04          / RECEIVER ENABLED
025    0002    SPCDTR= 02          / DATA TERMINAL READY
026    0001    SPCTE= 01          / TRANSMIT ENABLE
027
028          /
029          /****MODE DEFINITION - ASYNCH MODE
030          /
031    00C0    SPMSTP= C0          / 2 STOP BITS
032    0020    SPMEVN= 20          / EVEN PARITY
033    0010    SPMPAR= 10          / PARITY ENABLED
034    000C    SPMLLEN= 0C          / 8-BIT CHARACTERS
035    0002    SPMBRF= 02          / BAUD RATE X 16
036
037          /
038          /****COMMAND DEFINITIONS
039          /
040    00FE    PFMODE=SPMSTP+SPMEVN+SPMPAR+SPMLLEN+SPMBRF
041    0079    P2MODE= 79          / 1 STOP BIT, EVEN PARITY, 7 DATA BITS
042
043          /
044    0025    PFCMD=SPCRTS+SPCRE+SPCTE / ENABLE RECEIVER, TRANS

```



```

040 / / MITTER, AND RECEIVER
041 / / INTERRUPT
042 0081 PFNULL=SFCEH+SFCTE / NULL COMMAND
043 EJECT

001 /
002 /***NODE TYPES
003 /
004 0000 NOSDN=00 / START OF NETWORK
005 0001 NOEDL=01 / END OF LOGIC
006 0002 NOEDC=02 / END OF COLUMN
007 0003 NOOREL=03 / NORMALLY OPEN RELAY
008 0004 NOCREL=04 / NORMALLY CLOSED RELAY
009 0005 NOPOST=05 / POSITIVE TRANSITIONAL
010 0006 NONEGT=06 / NEGATIVE TRANSITIONAL
011 0007 NOCOIL=07 / COIL
012 0008 NOLATC=10 / LATCH
013 0009 NODCOL=11 / DISABLED COIL
014 000A NODLAT=12 / DISABLED LATCH
015 000B NOHDZO=13 / HORIZONTAL OPEN
016 000C NOHDZS=14 / HORIZONTAL SHORT
017 000D NOCPRE=15 / CONSTANT PRESET/B-NODE
018 000E NORPRE=16 / REGISTER PRESET/B-NODE
019 000F NOCTR=17 / COUNTER
020 0010 NOT100=20 / TIMER 1.00 SECS
021 0011 NOT010=21 / TIMER 0.10 SECS
022 0012 NOT001=22 / TIMER 0.01 SECS
023 0013 NOCON=23 / CONVERT
024 0014 NOCCON=24 / C-NODE CONSTANT
025 0015 NOCREG=25 / C-NODE REGISTER
026 0016 NOCALC=26 / CALCULATE
027 0017 NONULL=27 / NULL NODE
028 / 30 / NOT USED
029 / 31 / NOT USED
030 / 32 / NOT USED
031 / 33 / NOT USED
032 / 34 / NOT USED
033 / 35 / NOT USED
034 / 36 / NOT USED
035 / 37 / NOT USED
036 EJECT

001 SUBJOB MEMORY SPACE DEFINITION
002 /
003 /***MEMORY SPACE DEFINITION
004 /
005 0000 ROMLO=:0000 / START OF ROM
006 37FF ROMHI=:37FF / END OF ROM
007 000E NUM1K=ROMHI+1-ROMLO%1024 / # OF 1K AREAS
008 /
009 F800 RAMLO=:F800 / START OF RAM
010 FFFF RAMHI=:FFFF / END OF RAM
011 /
012 3800 ROMSIZ=ROMHI-ROMLO+1 / ROM SIZE
013 0800 RAMSIZ=RAMHI-RAMLO+1 / RAM SIZE
014 EJECT

001 SUBJOB DIAGNOSTIC INTERRUPT VECTOR
002 /
003 /***DIAGNOSTIC INTERRUPT VECTOR
004 /
005 /***LOADED DURING POWER-UP WITH A RET
006 / INSTRUCTION (109)
007 /
008 /***DIAGNOSTIC WILL LOAD A JUMP TO ITS INTERRUPT
009 /***HANDLER.
010 /
011 F800 INTVEC=RAMLO / INTERRUPT VECTOR, 3 BYTES
012 /
013 /***THE NEXT LOCATION IS RESERVED FOR DIAGNOSTICS ALSO.
014 /
015 0004 INTVCL=4 / NUMBER OF RESERVED BYTES
016
017

```

```

018 / THE INDEPENDENT ON-BOARD DIAGNOSTIC SYSTEM
019 / IS JUMPED-TO @ POWER UP TIME IF THE STAR-
020 / LINK CASSETTE MACHINE IS CONNECTED. THE
021 / JUMP ADDRESS IS.
022
023 3300 PDIA= 3300
024 EJECT

001
002 SUBJOB REFRESH MEMORY ALLOCATION
003 /
004 /***CRT REFRESH
005 /
006 /***FORMAT.
007 /
008 / ROW 0 - PAD ROW (50 HZ)
009 / ROW 1 - PAD ROW
010 / ROW 2 - DISPLAY LOGIC ROW
011 / ROW 3 - DISPLAY LOGIC ROW
012 / ROW 4 - DISPLAY LOGIC ROW
013 / ROW 5 - DISPLAY LOGIC ROW
014 / ROW 6 - DISPLAY LOGIC ROW
015 / ROW 7 - DISPLAY LOGIC ROW
016 / ROW 8 - DISPLAY LOGIC ROW
017 / ROW 9 - DISPLAY LOGIC ROW
018 / ROW 10 - DISPLAY LOGIC ROW
019 / ROW 11 - DISPLAY LOGIC ROW
020 / ROW 12 - DISPLAY LOGIC ROW
021 / ROW 13 - DISPLAY LOGIC ROW
022 / ROW 14 - DISPLAY LOGIC ROW
023 / ROW 15 - DISPLAY LOGIC ROW
024 / ROW 16 - DISPLAY LOGIC ROW
025 / ROW 17 - DISPLAY LOGIC ROW
026 / ROW 18 - SEPARATION ROW
027 / ROW 19 - ASSEMBLY/STATUS ROW 1
028 / ROW 20 - ASSEMBLY/STATUS ROW 2
029 / ROW 21 - PAD ROW
030 / ROW 22 - PAD ROW (50 HZ)
031 /
032 EJECT

001 /
002 /***REFRESH MEMORY ALLOCATION
003 /
004 /***ROW TYPES
005 /
006 0002 ROWA=: 02 / END-OF-ROW + PAD
007 0050 ROWB=: 50 / LOGIC ROW
008 0045 ROWC=: 45 / SEPARATION ROW
009 004D ROWD=: 4D / ASM/STATUS - ROW 1
010 004E ROWE=: 4E / ASM/STATUS - ROW 2
011 /
012 F804 CRTRFH=INTVEC+INTVCL / START OF REFRESH
013 /
014 / FULL DATA AREA
015 0544 CRTTMP=16!ROWB+ROWC+ROWD+ROWE+ROWA+ROWA
016 /
017 F804 CRT50S=CRTRFH / START OF 50HZ DISPLAY
018 /
019 F806 CRT60S=CRTRFH+ROWA / START OF 60HZ DISPLAY
020 /
021 FD4E CRT50E=CRT60S+CRTTMP+ROWA+ROWA / END OF 50HZ DISPLAY
022 /
023 FD4C CRT60E=CRT60S+CRTTMP+ROWA / END OF 60HZ DISPLAY
024 /
025 FD4E CRTRFX=CRT50E / END-OF-REFRESH
026 /
027 /***SPECIAL DMA CHARACTERS
028 /
029 000E ROWCNT=: 0E / 7X2 ROW OF LOGIC
030 0001 PADCNT=: 01 / 1 PAD AT END (60 HZ)
031 00F1 DMAEOR=: F1 / END-OF-ROW, STOP DMA
032 0020 DMABLK=: 20 / SPACE
033 0080 DMAFAN=: 80 / FIELD ATTRIBUTE
034 EJECT

```

```

035 /
036 /***DMA ADDRESS AND TERMINAL COUNTS
037 /
038 0549 DM50TL=CRT50E-CRTRFH-1 / 50HZ TERMINAL COUNT
039 0547 DM60TL=CRT60E-CRTRFH-1 / 60HZ TERMINAL COUNT
040 /
041 0005 DM50TH=DM50TL*.100 / 50HZ TC - HIGH
042 0049 DM50TL=-.100!DM50TH+DM50TL / 50HZ TC - LOW
043 /
044 0005 DM60TH=DM60TC*.100 / 60HZ TC - HIGH
045 0047 DM60TL=-.100!DM60TH+DM60TC / 60HZ TC - LOW
046 /
047 00F8 DM50AH=CRT50S*.100 / 50HZ ADDRESS - HIGH
048 0004 DM50AL=-.100!DM50AH+CRT50S / 50HZ ADDRESS - LOW
049 /
050 00F8 DM60AH=CRT60S*.100 / 60HZ ADDRESS - HIGH
051 0006 DM60AL=-.100!DM60AH+CRT60S / 60HZ ADDRESS - LOW
052 EJECT

001 /
002 /***LOGIC SPACE DEFINITIONS
003 /
004 F808 DSPLOG=CRTRFH+ROWA+ROWA / START OF LOGIC
005 /
006 0007 DSPNOD=7 / NODE LENGTH
007 /
008 0003 DSPPOW=3 / POWER-RAIL LENGTH
009 / (# OF CHARS ON
010 / THE HOR FOR
011 / EACH SCN LINE)
012 /
013 000B ADVFLD=. B / LENGTH OF ADVISORY FIELD
014 /
015 000D ERRFLD=. D / LENGTH OF ERROR FIELD
016 /
017 FC6C DSPBSY=16!ROWB+DSPLOG+ROWA+ROWA / BUSY LINE
018 /
019 0004 REFLN=. 4 / LENGTH OF REF FIELDS
020 /
021 0006 ASMCOL=. 6 / FIRST REF COLUMN
022 EJECT

SUBJOB ASSEMBLY/STATUS AREA DEFINITION
001 /
002 /
003 /***DEFINE ASSEMBLY/STATUS AREA
004 /
005 / START OF AREA
006 FCB1 DSPASM=2!ROWA+CRTRFH+CRTTMP-ROWD-ROWE
007 /
008 FCB3 DSPCON=DSPASM+2 / CONTACT FIELD (ASSEMBLY)
009 /
010 FCB8 DSPVER=DSPCON+@5 / VERTICAL FIELD (ASSEMBLY)
011 /
012 FCBB DSPERR=DSPVER+3 / ERROR FIELD
013 /
014 FCDA DSPREF=DSPERR+@31 / FIRST REFERENCE NUMBER
015 /
016 FCFE DSPNUM=DSPASM+ROWD / NUMERIC FIELD (ASSEMBLY)
017 /
018 FD08 DSPSHT=DSPNUM+@10 / SHIFT FIELD
019 /
020 FD0A DSPADV=DSPSHT+2 / ADVISORY FIELD
021 /
022 FD18 DSPSTP=DSPADV+@14 / STEP FIELD
023 /
024 FD1D DSPUSE=DSPSTP+. 5 / USAGE FIELD
025 /
026 FD28 DSPVAL=DSPUSE+@11 / FIRST VALUE FIELD
027 EJECT

001 /
002 /***LOCATION OF EACH NODE IN REFRESH
003 /
004 F80B L1C01U=DSPLOG+DSPPOW / COL 1
005 F85B L1C01L=L1C01U+ROWB
006 /L1C02U=L1C01U+DSPNOD / COL 2
007 /L1C02L=L1C02U+ROWB

```


008		/L1C03U=L1C02U+DSPNOD	/ COL 3
009		/L1C03L=L1C03U+ROWB	
010		/L1C04U=L1C03U+DSPNOD	/ COL 4
011		/L1C04L=L1C04U+ROWB	
012		/L1C05U=L1C04U+DSPNOD	/ COL 5
013		/L1C05L=L1C05U+ROWB	
014		/L1C06U=L1C05U+DSPNOD	/ COL 6
015		/L1C06L=L1C06U+ROWB	
016		/L1C07U=L1C06U+DSPNOD	/ COL 7
017		/L1C07L=L1C07U+ROWB	
018		/L1C08U=L1C07U+DSPNOD	/ COL 8
019		/L1C08L=L1C08U+ROWB	
020		/L1C09U=L1C08U+DSPNOD	/ COL 9
021		/L1C09L=L1C09U+ROWB	
022		/L1C10U=L1C09U+DSPNOD	/ COL 10
023		/L1C10L=L1C10U+ROWB	
024		/L1C11U=L1C10U+DSPNOD	/ COL 11
025		/L1C11L=L1C11U+ROWB	
026		EJECT	

001		/	
002		/***LINE 2	
003		/	
004	F8AB	L2C01U=L1C01L+ROWB	/ COL 1
005	F8FB	L2C01L=L2C01U+ROWB	
006		/L2C02U=L2C01U+DSPNOD	/ COL 2
007		/L2C02L=L2C02U+ROWB	
008		/L2C03U=L2C02U+DSPNOD	/ COL 3
009		/L2C03L=L2C03U+ROWB	
010		/L2C04U=L2C03U+DSPNOD	/ COL 4
011		/L2C04L=L2C04U+ROWB	
012		/L2C05U=L2C04U+DSPNOD	/ COL 5
013		/L2C05L=L2C05U+ROWB	
014		/L2C06U=L2C05U+DSPNOD	/ COL 6
015		/L2C06L=L2C06U+ROWB	
016		/L2C07U=L2C06U+DSPNOD	/ COL 7
017		/L2C07L=L2C07U+ROWB	
018		/L2C08U=L2C07U+DSPNOD	/ COL 8
019		/L2C08L=L2C08U+ROWB	
020		/L2C09U=L2C08U+DSPNOD	/ COL 9
021		/L2C09L=L2C09U+ROWB	
022		/L2C10U=L2C09U+DSPNOD	/ COL 10
023		/L2C10L=L2C10U+ROWB	
024		/L2C11U=L2C10U+DSPNOD	/ COL 11
025		/L2C11L=L2C11U+ROWB	
026		EJECT	

001		/	
002		/***LINE 3	
003		/	
004		/L3C01U=L2C01L+ROWB	/ COL 1
005		/L3C01L=L3C01U+ROWB	
006		/L3C02U=L3C01U+DSPNOD	/ COL 2
007		/L3C02L=L3C02U+ROWB	
008		/L3C03U=L3C02U+DSPNOD	/ COL 3
009		/L3C03L=L3C03U+ROWB	
010		/L3C04U=L3C03U+DSPNOD	/ COL 4
011		/L3C04L=L3C04U+ROWB	
012		/L3C05U=L3C04U+DSPNOD	/ COL 5
013		/L3C05L=L3C05U+ROWB	
014		/L3C06U=L3C05U+DSPNOD	/ COL 6
015		/L3C06L=L3C06U+ROWB	
016		/L3C07U=L3C06U+DSPNOD	/ COL 7
017		/L3C07L=L3C07U+ROWB	
018		/L3C08U=L3C07U+DSPNOD	/ COL 8
019		/L3C08L=L3C08U+ROWB	
020		/L3C09U=L3C08U+DSPNOD	/ COL 9
021		/L3C09L=L3C09U+ROWB	
022		/L3C10U=L3C09U+DSPNOD	/ COL 10
023		/L3C10L=L3C10U+ROWB	
024		/L3C11U=L3C10U+DSPNOD	/ COL 11
025		/L3C11L=L3C11U+ROWB	
026		EJECT	

```

001 /
002 /***LINE 4
003 /
004 /L4C01U=L3C01L+ROWB / COL 1
005 /L4C01L=L4C01U+ROWB
006 /L4C02U=L4C01U+DSPNOD / COL 2
007 /L4C02L=L4C02U+ROWB
008 /L4C03U=L4C02U+DSPNOD / COL 3
009 /L4C03L=L4C03U+ROWB
010 /L4C04U=L4C03U+DSPNOD / COL 4
011 /L4C04L=L4C04U+ROWB
012 /L4C05U=L4C04U+DSPNOD / COL 5
013 /L4C05L=L4C05U+ROWB
014 /L4C06U=L4C05U+DSPNOD / COL 6
015 /L4C06L=L4C06U+ROWB
016 /L4C07U=L4C06U+DSPNOD / COL 7
017 /L4C07L=L4C07U+ROWB
018 /L4C08U=L4C07U+DSPNOD / COL 8
019 /L4C08L=L4C08U+ROWB
020 /L4C09U=L4C08U+DSPNOD / COL 9
021 /L4C09L=L4C09U+ROWB
022 /L4C10U=L4C09U+DSPNOD / COL 10
023 /L4C10L=L4C10U+ROWB
024 /L4C11U=L4C10U+DSPNOD / COL 11
025 /L4C11L=L4C11U+ROWB
026 EJECT

001 /
002 /***LINE 5
003 /
004 /L5C01U=L4C01L+ROWB / COL 1
005 /L5C01L=L5C01U+ROWB
006 /L5C02U=L5C01U+DSPNOD / COL 2
007 /L5C02L=L5C02U+ROWB
008 /L5C03U=L5C02U+DSPNOD / COL 3
009 /L5C03L=L5C03U+ROWB
010 /L5C04U=L5C03U+DSPNOD / COL 4
011 /L5C04L=L5C04U+ROWB
012 /L5C05U=L5C04U+DSPNOD / COL 5
013 /L5C05L=L5C05U+ROWB
014 /L5C06U=L5C05U+DSPNOD / COL 6
015 /L5C06L=L5C06U+ROWB
016 /L5C07U=L5C06U+DSPNOD / COL 7
017 /L5C07L=L5C07U+ROWB
018 /L5C08U=L5C07U+DSPNOD / COL 8
019 /L5C08L=L5C08U+ROWB
020 /L5C09U=L5C08U+DSPNOD / COL 9
021 /L5C09L=L5C09U+ROWB
022 /L5C10U=L5C09U+DSPNOD / COL 10
023 /L5C10L=L5C10U+ROWB
024 /L5C11U=L5C10U+DSPNOD / COL 11
025 /L5C11L=L5C11U+ROWB
026 EJECT

001 /
002 /***LINE 6
003 /
004 /L6C01U=L5C01L+ROWB / COL 1
005 /L6C01L=L6C01U+ROWB
006 /L6C02U=L6C01U+DSPNOD / COL 2
007 /L6C02L=L6C02U+ROWB
008 /L6C03U=L6C02U+DSPNOD / COL 3
009 /L6C03L=L6C03U+ROWB
010 /L6C04U=L6C03U+DSPNOD / COL 4
011 /L6C04L=L6C04U+ROWB
012 /L6C05U=L6C04U+DSPNOD / COL 5
013 /L6C05L=L6C05U+ROWB
014 /L6C06U=L6C05U+DSPNOD / COL 6
015 /L6C06L=L6C06U+ROWB
016 /L6C07U=L6C06U+DSPNOD / COL 7
017 /L6C07L=L6C07U+ROWB
018 /L6C08U=L6C07U+DSPNOD / COL 8
019 /L6C08L=L6C08U+ROWB
020 /L6C09U=L6C08U+DSPNOD / COL 9
021 /L6C09L=L6C09U+ROWB
022 /L6C10U=L6C09U+DSPNOD / COL 10
023 /L6C10L=L6C10U+ROWB
024 /L6C11U=L6C10U+DSPNOD / COL 11
025 /L6C11L=L6C11U+ROWB
026 EJECT

```

```

001 /
002 /***LINE 7
003 /
004 /L7C01U=L6C01L+ROWB / COL 1
005 /L7C01L=L7C01U+ROWB /
006 /L7C02U=L7C01U+DSPNOD / COL 2
007 /L7C02L=L7C02U+ROWB /
008 /L7C03U=L7C02U+DSPNOD / COL 3
009 /L7C03L=L7C03U+ROWB /
010 /L7C04U=L7C03U+DSPNOD / COL 4
011 /L7C04L=L7C04U+ROWB /
012 /L7C05U=L7C04U+DSPNOD / COL 5
013 /L7C05L=L7C05U+ROWB /
014 /L7C06U=L7C05U+DSPNOD / COL 6
015 /L7C06L=L7C06U+ROWB /
016 /L7C07U=L7C06U+DSPNOD / COL 7
017 /L7C07L=L7C07U+ROWB /
018 /L7C08U=L7C07U+DSPNOD / COL 8
019 /L7C08L=L7C08U+ROWB /
020 /L7C09U=L7C08U+DSPNOD / COL 9
021 /L7C09L=L7C09U+ROWB /
022 /L7C10U=L7C09U+DSPNOD / COL 10
023 /L7C10L=L7C10U+ROWB /
024 /L7C11U=L7C10U+DSPNOD / COL 11
025 /L7C11L=L7C11U+ROWB /
026 EJECT

SUBJOB STACK ALLOCATION
001 /
002 /
003 /***STACK ALLOCATION
004 /
005 0040 STACKL=100 / 64 BYTE STACK
006 /
007 FD8F STACK=CRTRFX+STACKL+1 / STACK BASE
008 EJECT

SUBJOB SYSTEM TIMERS
001 /
002 /
003 /***SYSTEM TIMERS ALLOCATION
004 /
005 FD8F TMRTAB=STACK / START OF TABLE
006 FD8F TMRBEP=TMRTAB / BEEP TIMER
007 FD90 TMRACK=TMRBEP+1 / ACK TIMER
008 FD91 TMRLED=TMRACK+1 / LED TIMER
009 FD92 TMRPWR=TMRLED+1 / POWER TIMER
010 FD93 TMRERR=TMRPWR+1 / ERROR BLINK TIMER
011 FD94 TMRDIS=TMRERR+1 / DISCRETE UPDATE TIMER
012 / BOTTOM OF SCREEN
013 /
014 FD95 TMRTBX=TMRDIS+1 / END OF TABLE
015 /
016 0006 TMRCNT=TMRDIS-TMRTAB+1 / NUMBER OF TIMERS
017 /
018 /***NOTE:
019 /
020 / THE TIMER MUST BE IN THE SAME SEQUENCE AS THE
021 / DISPATCH TABLE TMRDSP IN CLKINT..
022 /
023 /***TIMER VALUES
024 /
025 001E ERRTMR=@30 / ERROR FLASH RATE (2HZ)
026 /
027 001E LEDTMR=@30 / LED REQUEST RATE (2HZ)
028 /
029 0002 PWRTMR=@2 / POWER REQUEST RATE (30HZ)
030 /
031 003C ACKTMR=@60 / ACK TIMER (1 SEC)
032 /
033 0001 DISTMR=@1 / DISCRETE UPDATE (60HZ)
034 EJECT

SUBJOB BUFFER BLOCKS
001 /
002 /
003 /
004 /****CIRCULAR BUFFER BLOCK DEFINITION,
005 /
006 /****EACH BUFFER IS DEFINED BY A DATA BLOCK
007 /

```



```

008      0000      BFBASE=0                / ADDRESS OF BUFFER BASE
009      0002      BFIPTR=BFBASE+2          / INPUT POINTER
010      0003      BFOPTR=BFIPTR+1         / OUTPUT POINTER
011      0004      BFLEN=BFOPTR+1          / BUFFER LENGTH
012      0005      BFUSE=BFLEN+1           / USAGE COUNT
013      0006      BFBLKL=BFUSE+1          / BLOCK LENGTH
014      /
015      /***BUFFER BLOCKS:
016      /
017      FD96      SPLBLK=TMRTBX+1          / SPOOLER BLOCK
018      /
019      FD9C      PFIBLK=SPLBLK+BFBLKL    / PERIPHERAL RECEIVER BLOCK
020      /
021      FDA2      PPOBLK=PFIBLK+BFBLKL    / PERIPHERAL TRANSMIT BLOCK
022      /
023      FDA8      KBDBLK=PPOBLK+BFBLKL    / KEYBOARD BLOCK
024      /
025      /***BUFFER LENGTHS (MAX IS 255 BYTES)
026      /
027      0040      SPLBFL=100              / SPOOLER LENGTH
028      0020      PFIBFL= 40              / RECEIVER BUFFER LENGTH
029      0020      PPOBFL= 40              / TRANSMIT BUFFER LENGTH
030      0010      KBDBFL= 20              / KBD BUFFER LENGTH
031      /
032      /*** ACTUAL BUFFER LOCATIONS - TOP OF RAM SPACE
033      /
034      FFC0      SPLBUF= RAMHI-SPLBFL+1   / SPOOLER
035      FFA0      PFIBUF= SPLBUF-PFIBFL    / RECEIVER
036      FF80      PPOBUF= PFIBUF-PPOBFL    / TRANSMITTER
037      FF70      KBDBUF= PPOBUF-KBDBFL    / KEYBOARD
038      /
039      FF70      BUFFER= KBDBUF           / START OF BUFFERS
040      EJECT

001      SUBJOB PERIPHERAL PORT SCRATCHPAD ALLOCATIONS
002      /
003      FDAE      PPISTA= KBDBLK+BFBLKL    / RECEIVER STATUS
004      FDAF      PPOSTA= PPISTA+1         / TRANSMIT STATUS
005      FDB0      MSGLEN= PPOSTA+1         / INPUT MESSAGE LENGTH CTR
006      FDB1      PFICLK=MSGLEN+1         / INPUT MESSAGE CHECKSUM
007      FDB2      RCOUNT=PFICLK+1        / RETRY COUNT
008      FDB3      PPOCHK=RCOUNT+1         / OUTPUT MESSAGE CHECKSUM
009      /
010      FDB4      POSAVE= PPOCHK+1         / PARALLEL PORT IMAGE
011      /
012      0004      MAXTRY=: 4               / 4 RETRIES ALLOWED
013      /
014      /***PERIPHERAL PORT RECEIVER STATUS
015      / SOFTWARE BITS: SET AND RESET IN "PPISTA"
016      /
017      0080      PFIMSG=: 80              / MESSAGE IN PROCESS
018      0040      PFIFCN=: 40              / NEXT CHAR IS FUNCTION
019      0020      PFIDON=: 20              / MESSAGE COMPLETED
020      0010      PFIPAR=: 10              / PARITY/FRAMING ERROR
021      0008      PFIOVR=: 08              / OVERRUN ERROR
022      0004      PFIRET=: 04              / RETRANSMIT
023      0002      PFICER=: 02              / CHECKSUM ERROR
024      0001      PFICNT=: 01              / NEXT CHAR IS COUNT
025      EJECT

001      SUBJOB LOGIC DATA TABLE
002      /
003      / NOTE: THIS TABLE IS MAINTAINED TO DESCRIBE
004      / THE LAYOUT OF A NETWORK ON A ROW-BY-ROW BASIS. IT
005      / IS DESIGNED TO KEEP TRACK OF MORE THAN 1 NETWORK
006      / PER SCREEN, WHICH IS NOT IMPLEMENTED AT THIS TIME
007      /
008      /***BLOCK DEFINITION
009      /
010      0000      ROWFLG=0                 / FLAG CELL
011      0001      ROWFMA=ROWFLG+1         / FIRST MEMORY ADDRESS
012      0003      ROWLMA=ROWFMA+2         / LAST MEMORY ADDRESS
013      0005      ROWCUR=ROWLMA+2         / CURSOR POSITION
014      0006      ROWSEQ=ROWCUR+1        / SEQUENCE NUMBER
015      0008      ROWBKL=ROWSEQ+2         / LENGTH OF BLOCK
016      /
017      /***FLAG DEFINITION

```

```

018 /
019 0080 ROWFSN=: 80 / START OF NETWORK
020 0040 ROWFEN=: 40 / END OF NETWORK
021 0020 ROWFMN=: 20 / MIDDLE OF NETWORK
022 0010 ROWFBK=: 10 / BLANK ROW
023 / :08 / NOT USED
024 / :04 / NOT USED
025 / :02 / NOT USED
026 / :01 / NOT USED
027 /
028 /***STORAGE ALLOCATION
029 /
030 FDB5 ROWTAB=POSAVE+1
031 /
032 FDB5 ROWTB1=ROWTAB / LINE 1
033 FDBD ROWTB2=ROWTB1+ROWBKL / LINE 2
034 FDC5 ROWTB3=ROWTB2+ROWBKL / LINE 3
035 FDCD ROWTB4=ROWTB3+ROWBKL / LINE 4
036 FDD5 ROWTB5=ROWTB4+ROWBKL / LINE 5
037 FDDD ROWTB6=ROWTB5+ROWBKL / LINE 6
038 FDE5 ROWTB7=ROWTB6+ROWBKL / LINE 7
039 FDED ROWTBX=ROWTB7+ROWBKL / END OF TABLE
040 /
041 0038 ROWTBL=ROWTB7+ROWBKL-ROWTAB / LENGTH OF BLOCK
042 EJECT

001 SUBJOB COLUMN DATA TABLE
002
003
004 / THIS TABLE MAINTAINS, FOR EACH COLUMN, THE LOWEST
005 / AND HIGHEST CONTROLLER ADDRESSES FOR THE NODES
006 / IN THIS COLUMN. IT ALSO MAINTAINS THE ACTUAL
007 / "END-OF-COLUMN" NODE, IF THERE IS ONE.
008
009 /
010 /***BLOCK DEFINITION
011 /
012 0000 COLSHI=0 / START OF COLUMN - ADDR I
013 0001 COLSLO=COLSHI+1 / START OF COLUMN - ADDRLO
014 0002 COLEHI=COLSLO+1 / END OF COLUMN - ADDRHI
015 0003 COLELO=COLEHI+1 / END OF COLUMN - ADDRLO
016 0004 EOCHI=COLELO+1 / END-OF-COL - BYTE 0
017 0005 EOCLO=EOCHI+1 / END-OF-COL - BYTE 1
018 /
019 0006 COLBKL=EOCLO+1 / BLOCK LENGTH
020 /
021 FDED COLTAB=ROWTBX / START OF TABLE
022 /
023 FDED COLTB1=COLTAB / COLUMN 1
024 FDF3 COLTB2=COLTB1+COLBKL / COLUMN 2
025 FDF9 COLTB3=COLTB2+COLBKL / COLUMN 3
026 FDFE COLTB4=COLTB3+COLBKL / COLUMN 4
027 FE05 COLTB5=COLTB4+COLBKL / COLUMN 5
028 FE0B COLTB6=COLTB5+COLBKL / COLUMN 6
029 FE11 COLTB7=COLTB6+COLBKL / COLUMN 7
030 FE17 COLTB8=COLTB7+COLBKL / COLUMN 8
031 FE1D COLTB9=COLTB8+COLBKL / COLUMN 9
032 FE23 COLTBA=COLTB9+COLBKL / COLUMN A
033 FE29 COLTBB=COLTBA+COLBKL / COLUMN B
034 /
035 FE2F COLTBX=COLTBB+COLBKL / END OF TABLE
036 /
037 0042 COLTBL=COLTBX-COLTAB / TABLE SIZE
038 EJECT

001 /
002 /***NODE TYPE MATRIX
003 /
004 /***7 X 11 MATRIX WITH NODE TYPE FOR EACH POSITION
005 /
006 FE2F MATROW=COLTBX / START OF TABLE
007 /
008 FE2F MATRW1=MATROW / ROW 1
009 FE3A MATRW2=MATRW1+@11 / ROW 2
010 FE45 MATRW3=MATRW2+@11 / ROW 3
011 FE50 MATRW4=MATRW3+@11 / ROW 4

```

```

012 FE5B MATRW5=MATRW4+@11 / ROW 5
013 FE66 MATRW6=MATRW5+@11 / ROW 6
014 FE71 MATRW7=MATRW6+@11 / ROW 7
015 /
016 FE7C MATROX=MATRW7+@11 / END OF TABLE
017 /
018 004D MATROL=MATROX-MATROW / TABLE SIZE
019 EJECT

001
002 /
003 /***KEYBOARD STATE VECTOR
004 /
005 FE7C KSTATE=MATROX / STORAGE ALLOCATION
006 /
007 /***BIT DEFINITION
008 /
009 0080 KSHIFT=: 80 / SHIFT FLAG
010 0040 KERROR=: 40 / I/O ERROR STATE
011 0020 KRESET=: 20 / RESET REQUIRED
012 0010 KCLEAR=: 10 / CLEAR NUMERIC FIELD
013 0008 KNET=: 08 / NETWORK IN PROGRESS
014 0004 KCLADV=: 04 / CLEAR ADVISORY FIELD
015 0002 KSUPER=: 02 / SUPERVISORY STATE
016 / .01 / NOT USED
017 /
018 FE7D CURDSP=KSTATE+1 / DISPLAY CURSOR
019 /
020 FE7E CURACT=CURDSP+1 / ACTUAL CURSOR
021 /
022 000B MAXCOL=@11 / NUMBER OF LOGIC COLUMNS
023 /
024 0007 MAXROW=@7 / NUMBER OF LOGIC ROWS
025 /
026 0080 ASMROW=: 80 / ASSEMBLY ROW
027 /
028 000F COLMSK=: 0F / COLUMN MASK
029 /
030 00F0 ROWMSK=: F0 / ROW MASK
031 /
032 0006 ASMNUM=: 6 / NUMBER OF REF FIELDS
033 /
034 FE7F ASMDON=CURACT+1 / CONTACT TYPE - ASSEMBLY
035 /
036 FE80 CURDON=ASMDON+1 / CONTACT TYPE - CURSOR
037 /
038 FE81 DISPTR=CURDON+1 / PTR FOR DISCRETE DISPLAY
039 /
040 FE82 NEWKEY=DISPTR+1 / LATEST KEYSTROKE
041 /
042 FE83 LASTKY=NEWKEY+1 / LAST KEYSTROKE
043 EJECT

--

001 SUBJOB MAINFRAME DATA BASE
002 /
003 FE84 SCONF1=LASTKY+1 / CONFIGURATION - BYTE 1
004 FE85 SCONF2=SCONF1+1 / CONFIGURATION - BYTE 2
005 /
006 FE86 MEMSIZ=SCONF2+1 / MEMORY SIZE (2 BYTES)
007 FE88 MEMUSE=MEMSIZ+2 / MEMORY USAGE (2 BYTES)
008 /
009 FE8A STPNUM=MEMUSE+2 / STEP NUMBER (2 BYTES)
010 /
011 FE8C ADRSON=STPNUM+2 / ADDRESS - START-OF-NET
012 /
013 FE8E ADREON=ADRSON+2 / ADDRESS - END-OF-NET
014 /
015 /***MAINFRAME DATA BASE ADDRESSES
016 /
017 00C0 INFBAS=300 / BASE OF INPUT REG SPACE
018 /
019 60BD ADRSYS=: 6000+@189 / SYSTEM STATE BYTE
020 /
021 60BE ADRCON=: 6000+@190 / CONFIGURATION ADDRESS
022 /

```


		217	218
023	0002	ADRUSE=.0002	/ START OF USER LOGIC
024		/	
025		/***/ADDRESS FIELDS	
026		/	
027	0000	LOGFLD=.00	/ LOGIC FIELD
028	0020	IOFLD=.20	/ I/O FIELD
029	0040	REGFLD=.40	/ REGISTER FIELD
030	0060	SPDFLD=.60	/ SCRATCHPAD FIELD
031		/	
032		/***/NODE INFORMATION	
033		/	
034	0080	EOCFLG=.80	/ END-OF-COLUMN FLAG
035	0001	OUTFLG=.01	/ OUTPUT COIL
036	0002	INTFLG=.02	/ INTERNAL COIL
037	0003	SEQFLG=.03	/ SEQUENCER
038		/	
039	007C	NODMSK=.7C	/ NODE TYPE MASK
040		/	
041	0000	HLDFLG=.00	/ HOLDING REGISTER
042	0001	INPFLG=.01	/ INPUT REGISTER
043	0002	DUMFLG=.02	/ DUMMY REGISTER
044		/	
045	0000	ADDFLG=.00	/ CALCULATE - ADD
046	0001	SUBFLG=.01	/ CALCULATE - SUBTRACT
047	0002	MPXFLG=.02	/ CALCULATE - MULTIPLY
048	0003	DIVFLG=.03	/ CALCULATE - DIVIDE
049		EJECT	
050		/	
051	0001	INTSTA=.01	/ STATE - INTERNAL COIL
052	0002	OUTSTA=.02	/ STATE - OUTPUT COIL
053	0004	INPSTA=.04	/ STATE - INPUT
054	0008	INPDIS=.08	/ STATE - INPUT DISABLE
055		/	
056		/INTHIS=.10	/ HISTORY - INTERNAL COIL
057	0020	OUTHIS=.20	/ HISTORY - OUTPUT COIL
058		/INPHIS=.40	/ HISTORY - INPUT
059		/ :80	/ NOT USED
060		/	
061	F7FF	DISMSK=-INPDIS!:100-1	/ MASK INPUT DISABLE
062		/	
063		/CTRFLG=.00	/ COUNTER FLAG
064		/T10FLG=.01	/ TIMER 1.00 FLAG
065		/T01FLG=.02	/ TIMER 0.10 FLAG
066		/T00FLG=.03	/ TIMER 0.01 FLAG
067		/	
068	0000	SINFLG=.00	/ DISCRETE SOURCE FLAG
069	0001	SRGFLG=.01	/ REGISTER SOURCE FLAG
070	0002	DINFLG=.02	/ DISCRETE DESTINATION FLAG
071	0003	DRGFLG=.03	/ REGISTER DESTINATION FLAG
072		/	
073		EJECT	
001		SUBJOB SYSTEM CONFIGURATION BYTE DEFINITION	
002		/	
003		/***/SYSCONF1	
004		/	
005	0080	SY4096=.80	/ 4096 BYTE LOGIC RAM
006	0040	SY2048=.40	/ 2048 BYTE LOGIC RAM
007	0020	SY1024=.20	/ 1024 BYTE LOGIC RAM
008	0010	SY0512=.10	/ 512 BYTE LOGIC RAM
009	0008	SY0256=.08	/ 256 BYTE LOGIC RAM
010		/ :04	/ NOT USED
011		/ :02	/ NOT USED
012		/ :01	/ NOT USED
013		/	
014		/***/SYSCONF2	
015		/	
016	0080	SYS256=.80	/ 256 I/O POINTS
017	0040	SYS192=.40	/ 192 I/O POINTS
018	0020	SYS128=.20	/ 128 I/O POINTS
019	0010	SYS064=.10	/ 64 I/O POINTS
020		/ :08	/ NOT USED
021		/SYSTRN=:04	/ TRANSITIONAL OPTION
022	0002	SYSENH=:02	/ ENHANCED EXEC
023		/ :01	/ NOT USED
024		/	

```

025      /***SYSTEM STATE BYTE
026      /
027      0080      SYSRUN=: 80      / RUN STATE
028      /SYSFUP=: 40      / POWER-UP STATE
029      /SYSFDN=: 20      / POWER-DOWN STATE
030      0010      SYSSTP=: 10      / STOP STATE
031      /SYSERR=: 0F      / ERROR CODES
032      /
033      /***ERROR CODE/S
034      /
035      /SYSOVR=: 01      / COMMUNICATIONS OVERRUN
036      /SYSLCK=: 02      / LOGIC RAM CHECKSUM ERR R
037      /SYSNOD=: 03      / INVALID NODE TYPE
038      /SYSIO=: 04      / I/O PORT ERROR
039      /SYSSPD=: 05      / SCRATCHPAD DIAGS FAILE
040      /SYSOCK=: 06      / COIL RAM CHECKSUM ERROR
041      /SYSBIA=: 07      / CPU DIAGNOSTIC FAILED
042      /SYSTEMEM=: 08      / ILLEGAL MEM CONFIGURAT DN
043      /SYSRTL=: 09      / REAL-TIME CLOCK ERROR
044      /SYSWDT=: 0A      / WATCH-DOG TIMER EXPIRED
045      /SYSCOL=: 0B      / ILLEGAL COLUMN
046      000C      SYSEOL=: 0C      / NO END-OF-LOGIC NODE
047      /
048      /***SEQUENCER DATA
049      /
050      0033      SEQBAS=: 33      / BASE SEQUENCER REG - 4051
051      00E0      REGMSK=: E0      / REGISTER MASK
052      001F      STPMSK=: 1F      / STEP MASK
053      EJECT

      SUBJOB COMMUNICATIONS SCRATCHPAD ALLOCATIONS
001      /
002      /
003      /***COMMAND BUFFER
004      /
005      FE90      CMDBUF=ADREON+2      / START OF COMMAND BUFFER
006      /
007      0018      CMDBFL=@24      / BUFFER LENGTH (19 BYTES)
008      /
009      FEAB      RSPBUF=CMDBUF+CMDBFL      / RESPONSE BUFFER
010      /
011      001E      RSPBFL=@24      / BUFFER LENGTH (19 BYTES)
012      EJECT
013      SUBJOB \      RAM STORAGE FOR LOAD, DUMP, VER
014      /
015      FECC      *RSPBUF+RSPBFL
016      /
017      /      "EUSEG" IS A 2 BYTE WORD USED TO STORE
018      /      THE HIGHEST VALID 484 ADDR IN
019      /      A SEGMENT AS "HI-LO". I.E. LOGIC RAM, COIL
020      /      RAM, OR REG RAM.
021      /
022      FECD      EUSEG, DS      2
023      /
024      /
025      /      "TEMP" IS A 2 BYTE TEMPORARY LOC USED BY L-D-V
026      /
027      FECE      TEMP, DS      2
028      /
029      /
030      /      "CASBUF" IS USED BY L-D-V TO TRANSFER
031      /      DATA IN/OUT OF PORT 2; IT MUST CONTAIN 1
032      /      COMPLETE RECORD.
033      /
034      FECD      CASBUF, DS      @47
035      /
036      /
037      /      "VERBUF" IS USED BY THE "VERIFY" FUNCTION
038      /      OF L-D-V TO COMPARE A TAPE BUFFER TO A
039      /      BUFFER READ FROM THE 484.
040      /
041      FEF3      VERBUF, DS      @24
042      /
043      /
044      EJECT

```

```

045          SUBJOB A          SEARCH RAM LOCATIONS
046
047          BRCHST IS A 2 BYTE HOLDING CELL FOR
048          # OF START-OF-NETWORK NODES ENCOUNTERED
049          DURING SEARCH
050
051 FF0B          BRCHST D8          /
052
053          BRCHST
054
055          SUBJOB ASCII CHARACTER SET
056
057          /
058          /***ASCII CHARACTER***/
059          /
060          /ASCII0=.00          / 1.
061          /ASCII1=.01          / .1
062          /ASCII2=.02          / TOP LEFT BOARDER
063          /ASCII3=.03          / UPPER BOARDER
064          /ASCII4=.04          / TOP RIGHT BOARDER
065          /ASCII5=.05          / LEFT BOARDER
066          /ASCII6=.06          / COUNTER LEFT
067          /ASCII7=.07          / TIMER BOARDER T
068          /ASCII8=.08          / TIMER BOARDER T.
069          /ASCII9=.09          / RIGHT BOARDER
070          /ASCIIA=.0A          / DIVIDE
071          /ASCIIB=.0B          / MULTIPLY
072          /ASCII10=.0C          / OPEN INDICATOR
073          /ASCII11=.0D          / NON-ASCII BLANK
074          /ASCII12=.0E          / NOT USED
075          /ASCII13=.0F          / NOT USED
076          /
077          /ASCII14=.10          / 0 UNDERLINED
078          /ASCII15=.11          / 1 UNDERLINED
079          /ASCII16=.12          / 2 UNDERLINED
080          /ASCII17=.13          / 3 UNDERLINED
081          /ASCII18=.14          / 4 UNDERLINED
082          /ASCII19=.15          / 5 UNDERLINED
083          /ASCII20=.16          / 6 UNDERLINED
084          /ASCII21=.17          / 7 UNDERLINED
085          /ASCII22=.18          / 8 UNDERLINED
086          /ASCII23=.19          / 9 UNDERLINED
087          /ASCII24=.1A          / UP ARROW
088          /ASCII25=.1B          / DOWN ARROW
089          /ASCII26=.1C          / BACK SLASH
090          /ASCII27=.1D          / NUMERIC FIELD BLANK
091          /ASCII28=.1E          / VERTICAL FIELD BLANK
092          /ASCII29=.1F          / CONTACT FIELD BLANK
093          /ASCII30=.20          / SPACE
094          /
095          /ASCII31=.21          / !
096          /ASCII32=.22          / "
097          /ASCII33=.23          / #
098          /ASCII34=.24          / $
099          /ASCII35=.25          / %
100          /ASCII36=.26          / &
101          /ASCII37=.27          / '
102          /ASCII38=.28          / (
103          /ASCII39=.29          / )
104          /ASCII40=.2A          / *
105          /ASCII41=.2B          / +
106          /ASCII42=.2C          / ,
107          /ASCII43=.2D          / -
108          /ASCII44=.2E          / .
109          /ASCII45=.2F          / /
110          /
111          /ASCII46=.30          / 0
112          /ASCII47=.31          / 1
113          /ASCII48=.32          / 2
114          /ASCII49=.33          / 3
115          /ASCII50=.34          / 4
116          /ASCII51=.35          / 5
117          /ASCII52=.36          / 6
118          /ASCII53=.37          / 7
119          /ASCII54=.38          / 8
120          /ASCII55=.39          / 9
121          /
122          /ASCII56=.3A          / :
123          /ASCII57=.3B          / ;
124          /ASCII58=.3C          / <
125          /ASCII59=.3D          / =
126          /ASCII60=.3E          / >

```


068		/ASDCMK=: 3F	/ ?
069		/ASDCAT=: 40	/ @
070		/ASDCA=: 41	/ A
071		/ASDCB=: 42	/ B
072	0013	ASDC=: 43	/ C
073	0044	ASDCD=: 44	/ D
074		/ASDCE=: 45	/ E
075		/ASDCF=: 46	/ F
076		/ASDCG=: 47	/ G
077		/ASDCH=: 48	/ H
078		/ASDCI=: 49	/ I
079		/ASDCJ=: 4A	/ J
080		/ASDCK=: 4B	/ K
081		/ASDCL=: 4C	/ L
082		/ASDCM=: 4D	/ M
083	004E	ASDCN=: 4E	/ N
084	004F	ASDCO=: 4F	/ O
085		/ASDCP=: 50	/ P
086		/ASDCQ=: 51	/ Q
087	0052	ASDCR=: 52	/ R
088	0053	ASDCS=: 53	/ S
089	0054	ASDCT=: 54	/ T
090		/ASDCU=: 55	/ U
091		/ASDCV=: 56	/ V
092		/ASDCW=: 57	/ W
093		/ASDCX=: 58	/ X
094		/ASDCY=: 59	/ Y
095		/ASDCZ=: 5A	/ Z
096		/ASCLBK=: 5B	/ [
097		/ASCLSL=: 5C	/ \
098		/ASCRBL=: 5D	/]
099		/ASCUF=: 5E	/ UP ARROW
100		/ASCLF=: 5F	/ <-
101	0060	ASCLRE=: 60	/ -]
102		/ : 61	/ -] HIGH LIGHT
103		/ASCRRE=: 62	/ [-
104		/ : 63	/ [- HIGHLIGHT
105		/ASCOL=: 64	/ --
106		/ : 65	/ -- HIGHLIGHT
107	0066	ASCDIS=: 66	/ DISABLE
108		/ : 67	/ DISABLE HIGH LIGHT
109		/ASCOL=: 68	/ -(
110		/ : 69	/ -(HIGHLIGHT
111		/ASCCR=: 6A	/)-
112		/ : 6B	/)- HIGHLIGHT
113		/ASCCM=: 6C	/ ::
114		/ : 6D	/ :: HIGHLIGHT
115		/ASCLM=: 6E	/ :L:
116		/ : 6F	/ :L: HIGHLIGHT
117		/ASCOR=: 70	/ +-
118		/ : 71	/ +- HIGHLIGHT
119	0072	ASCDSH=: 72	/ -
120		/ : 73	/ - HIGHLIGHT
121	0074	ASCBAR=: 74	/ --
122		/ : 75	/ -- HIGHLIGHT
123		/ASCN5P=: 76	/ SPACE
124		/ : 77	/ SPACE - HILITE
125		/ASCNUP=: 78	/ UP ARROW
126		/ : 79	/ UP ARROW - HIGHLIGHT
127		/ASCNDA=: 7A	/ DOWN ARROW
128		/ : 7B	/ DOWN ARROW - HIGHLIGHT
129		/ASCNBS=: 7C	/ BACK SLASH
130		/ : 7D	/ BACK SLASH - HIGHLIGHT
131		/ : 7E	/ SPARE
132		/ : 7F	/ SPARE
133		/	
134		***FIELD ATTRIBUTES	
135			
136	0090	CDTTR=: 90	/ NORMAL
137	0090	CDTRV=: 91	/ REVERSE VIDEO
138		CDTUD=: 92	/ UNDERLINE
139		CDTRVU=: 93	/ REVERSE + UNDERLINE
140		/	
141		/	
142		***CHARACTER ATTRIBUTES	
143		/ THESE CHARS DRAW HOR AND VERT LINES	
144		/	

145	00D0	CA0101=.C0	/ L=0, R=1, U=0, D=1
146		/CA1001=.C4	/ L=1, R=0, U=0, D=1
147		/CA0110=.C8	/ L=0, R=1, U=1, D=0
148	00DC	CA1010=.CC	/ L=1, R=0, U=1, D=0
149	00D0	CA1101=.D0	/ L=1, R=1, U=0, D=1
150		/CA1011=.D4	/ L=1, R=0, U=1, D=1
151	00D8	CA0111=.D8	/ L=0, R=1, U=1, D=1
152	00DC	CA1110=.DC	/ L=1, R=1, U=1, D=0
153	00E0	CA1100=.E0	/ L=1, R=1, U=0, D=0
154	00E4	CA0011=.E4	/ L=0, R=0, U=1, D=1
155	00E8	CA1111=.E8	/ L=1, R=1, U=1, D=1
156		/ .EC	/ NOT USED
157		/CADMA=.F0	/ DMA CONTROL
158		/ .F4	/ NOT USED
159		/ .F8	/ NOT USED
160		/ .FC	/ NOT USED
161		/	
162		/**OFFSETS	
163		/	
164		/CATNDR=.0	/ NORMAL
165	0001	CATHI=.1	/ HIGHLIGHT
166		/CATBLK=.2	/ BLINK
167		/CATHBL=.3	/ HIGHLIGHT + BLINK
168		/	
169	0002	ASCSTX=.02	/ ASCII START-OF-TEXT
170	00D0	ASCNAK=.D0	/ ASCII NOT ACKNOWLEDGE
171		EJECT	
001		SUBJOB MAINFRAME COMMUNICATIONS DEFINITIONS	
002		/	
003		/**COMMANDS	
004		/	
005	0010	CMDRED=.10	/ READ
006	0020	CMDWRT=.20	/ WRITE
007	0030	CMDSCH=.30	/ SEARCH
008	0040	CMPFWR=.40	/ POWER
009	0050	CMDINS=.50	/ INSERT
010	0060	CMDDEL=.60	/ DELETE
011	0070	CMDLED=.70	/ LED
012	0080	CMDSTP=.80	/ STOP
013	0090	CMDGO=.90	/ GO
014	00A0	CMDINI=.A0	/ INITIALIZE
015	00B0	CMDINC=.B0	/ INSERT @EOC
016	00C0	CMDDEC=.C0	/ DELETE @EOC
017	00D0	CMDNAK=.D0	/ NAK
018		/ .E0	/ NOT USED
019		/ .F0	/ NOT USED
020		/	
021		/**VARIABLE LENGTH CODES	
022		/	
023	0001	CMD02=.01	/ 2 BYTES
024		/CMD04=.02	/ 4 BYTES
025		/CMD06=.03	/ 6 BYTES
026		/CMD08=.04	/ 8 BYTES
027		/CMD10=.05	/ 10 BYTES
028		/CMD12=.06	/ 12 BYTES
029		/CMD14=.07	/ 14 BYTES
030		/CMD16=.08	/ 16 BYTES
031		/	
032		/**COMMAND LENGTHS	
033		/	
034	0006	LENRED=.06	/ READ
035	000A	LENWRT=.0A	/ WRITE
036	000A	LENSCH=.0A	/ SEARCH
037	0006	LENFWR=.06	/ POWER
038	0008	LENINS=.08	/ INSERT
039	0006	LENDEL=.06	/ DELETE
040	0005	LENLED=.05	/ LED
041	0004	LENSTP=.04	/ STOP
042	0004	LENGO=.04	/ GO
043	0004	LENNI=.04	/ INITIALIZE
044	0005	LENNC=.05	/ INSERT @EOC
045	0006	LENNDE=.06	/ DELETE @EOC
046	0005	LENNAK=.05	/ NAK
047		/	
048		/**SMAPLES	

```

049 /
050 /CMDMSK=.F0 / FUNCTION CODE
051 /CNTMSK=.0F / COUNT CODE
052 /
053 /***ERROR CODES
054 /
055 0001 ERRPAR=.01 / PARITY/FRAMING ERROR
056 0002 ERROVR=.02 / OVERRUN ERROR
057 0003 ERRCHK=.03 / BAD CHECKSUM
058 0004 ERRADR=.04 / ADDRESS OUT-OF-RANGE
059 0005 ERRADI=.05 / ILLEGAL ADDRESS
060 0006 ERRCMD=.06 / ILLEGAL COMMAND
061 0007 ERRTIM=.07 / TIME OUT
062 0008 ERRMSK=.08 / INVALID MASK
063 0009 ERRSEQ=.09 / INVALID SEQUENCE NUMBER
064 000A ERRNOD=.0A / INVALID NODE
065 000B ERRMEM=.0B / MEMORY PROTECT FAULT
066 000C ERRSTP=.0C / SYSTEM NOT IN STOP STATE
067 000D ERRLEN=.0D / BAD LENGTH
068 000E ERRCON=.0E / NODE NOT A CONTACT
069 000F ERRNPD=.0F / NODE NOT IN POWER DISPLAY
070 0010 ERRSUP=.10 / NODE NOT SUPPORTED
071 0011 ERRFUL=.11 / MEMORY FULL
072 EJECT

001 0000 *.0000
002 /
003 /***LOCATION : X'0000' : POWER-UP RESTART
004 /
005 0000 F3 D1 / DISABLE INTERRUPTS
006 0001 C33F00 JMP PWRUP / VECTOR TO THE POWER-UP
007 0004 76 HLT / SHOULD NEVER REACH HERE
008 0005 76 HLT / SHOULD NEVER REACH HERE
009 0006 76 HLT / SHOULD NEVER REACH HERE
010 0007 76 HLT / SHOULD NEVER REACH HERE
011 EJECT

001 0008 *.0008
002 /
003 /***LOCATION : X'0008' : RESTART 1
004 /
005 / RST 1 IS DEFINED AS A "NIBBLE SWAP" ROUTINE.
006 / IT PUTS THE LEAST SIGNIFICANT 4 BITS TO
007 / MOST SIGNIFICANT 4 BITS AND VISA VERSA.
008 / EXAMPLE.
009 / A-REG ON CALL = AAAABBBB
010 /
011 / NSWP .
012 /
013 / A-REG ON EXIT = BBBBAAAA

001 SUBJOB LOW ROM ALLOCATION
002 /
003 /***LOW ROM ALLOCATION
004 /
005 /***THIS MODULE DEFINES THE ALLOCATION OF THE
006 /***FIRST 64 BYTES OF ROM. THESE ADDRESSES ARE
007 /***USED FOR THE RESTART INSTRUCTIONS
008 /
009 /
010 EJECT

001 SUBJOB LOWROM ALLOCATION
002 /
003 /***LOW ROM IS ALLOCATED FOR THE RESTART INSTRUCTIONS.
004 /
005 / RST 0 - POWER-UP
006 / RST 1 - NSWP (SWAP 4 BITS HI-LO IN A-REG)
007 / RST 2 - MEM[H,L] <- [B,C]
008 / RST 3 - FC <- MEM[H,L]
009 / RST 4 - [H,L] <- MEM[H,L]
010 / RST 5 - MEM[H,L] <- [D,E]
011 / RST 6 - DBL PREC. COMPARE D/E & H/L
012 / RST 7 - INTERRUPT HANDLER
013 /
014 EJECT

```



```

014
015      00CF      / NSWP=:CF
016      /
017 0008 0F      / RRC      / ROTATE 4 TIMES
018 0009 0F      / RRC      / X
019 000A 0F      / RRC      / X
020 000B 0F      / RRC      / X
021 000C 09      / RET      / DONE
022
023 000D 76      / HLT      / NEVER REACH HERE!
024 000E 76      / HLT      / X
025 000F 76      / HLT      / X
026
EJECT

001      0010      *:0010
002      /
003      /***LOCATION : X'0010' : RESTART 2
004      /
005      /***STORE [B,C] INTO MEMORY([H,L])
006      /
007      /***[B,C] PRESERVED
008      /***[H,L] ← [H,L] + 2
009      /
010      00D7      MOVBC=:D7      / OPCODE FOR RST 2
011      /
012 0010 70      MOV      M,B      / STORE HI-ORDER VALUE
013 0011 23      INX      H      / BUMP ADDRESS
014 0012 71      MOV      M,C      / STORE LOW-ORDER VALUE
015 0013 23      INX      H      / BUMP ADDRESS
016 0014 09      RET      / EXIT
017 0015 76      HLT      / SHOULD NEVER REACH HERE
018 0016 76      HLT      / SHOULD NEVER REACH HER
019 0017 76      HLT      / SHOULD NEVER REACH HERE
020
EJECT

001      0018      *:0018
002      /
003      /***LOCATION : X'001A' : RESTART 3
004      /
005      /***DISPATCH OFF A TABLE
006      /
007      /***PC ← MEM([H,L])
008      /
009      /***A DESTROYED
010      /***[H,L] DESTROYED
011      /
012      00DF      DSPTAB=:DF      / OPCODE FOR RST 3
013      /
014 0018 F1      POP      PSW      / POP RETURN FROM STACK
015 0019 7E      MOV      A,M      / A ← LOW-ORDER ADDRESS
016 001A 23      INX      H      / BUMP ADDRESS
017 001B 66      MOV      H;M      / L ← HIGH-ORDER ADDRESS
018 001C 6F      MOV      L;A      / H ← LOW-ORDER ADDRESS
019 001D E9      PCHL      / DISPATCH
020 001E 76      HLT      / SHOULD NEVER REACH HERE
021 001F 76      HLT      / SHOULD NEVER REACH HER
022
EJECT

001      0020      *:0020
002      /
003      /***LOCATION : X'0020' : RESTART 4
004      /
005      /***[H,L] ← MEM([H,L])
006      /
007      /***A DESTROYED
008      /
009      00E7      GETHL=:E7      / OPCODE FOR RST 4
010      /
011 0020 7E      MOV      A;M      / A ← HIGH-ORDER BYTE
012 0021 23      INX      H      / BUMP ADDRESS
013 0022 6E      MOV      L;M      / L ← LOW-ORDER ADDRESS
014 0023 67      MOV      H;A      / H ← HIGH-ORDER ADDRESS
015 0024 09      RET      / EXIT
016 0025 76      HLT      / SHOULD NEVER REACH HERE
017 0026 76      HLT      / SHOULD NEVER REACH HERE
018 0027 76      HLT      / SHOULD NEVER REACH HER
019
EJECT

```

```

001      0028      * 0028
002      /
003      /***LOCATION : X'0028' : RESTART 5
004      /
005      /***MEM[H, L] ← [D, E]
006      /***[H, L] ← [H, L] + 2
007      /
008      00EF      MOVDE = EF      / OP CODE FOR RST 5
009      /
010      0028 72      MOV      M, D      / STORE D
011      0029 23      INX      H      / BUMP ADDRESS
012      002A 73      MOV      M, E      / STORE E
013      002B 23      INX      H      / BUMP ADDRESS
014      002C C9      RET      / EXIT
015      002D 76      HLT      / SHOULD NEVER REACH HERE
016      002E 76      HLT      / SHOULD NEVER REACH HERE
017      002F 76      HLT      / SHOULD NEVER REACH HERE
018      EJECT

001      0030      * 0030
002      /
003      /***LOCATION : X'0030' : RESTART 6
004      /
005      / RST 6 IS DEFINED AS A ROUTINE TO DO A
006      / "DOUBLE PRECISION COMPARE" OF THE
007      / CONTENTS OF D/E TO THE CONTENTS OF H/L
008      /
009      / CALLED BY
010      /
011      / DCOMP      / COMPARE ETC.
012      /
NOTE: THE A-REG IS DESTROYED!!!
014      /
015      / EXIT:
016      / FLAGS ARE SET AS FOLLOWS:
017      /
018      / CARRY SET IF D/E < H/L
019      / CARRY RESET IF D/E >= H/L
020      / ZERO SET IF D/E = H/L, RESET IF NOT
021      /
022      00F7      DCOMP = F7      / "RST 6"
023      /
024      0030 7A      MOV      A, D      / GET MS BYTE OF D/E
025      0031 BC      CMP      H      / COMPARE AND SET FLAGS
026      0032 C0      RNZ      / NOT EQUAL, ALL DONE
027      0033 7B      MOV      A, E      / GET LS BYTE OF D/E
028      0034 BD      CMP      L      / COMPARE AND SET FLAGS
029      0035 C9      RET      / DONE
030      /
031      0036 76      HLT      / SHOULD NEVER REACH HERE
032      0037 76      HLT      / SHOULD NEVER REACH HERE
033      EJECT

001      0038      * 0038
002      /
003      /***LOCATION : X'0038' : RESTART 7
004      /
005      /***INTERRUPT HANDLER
006      /
007      0038 CD00F8      CALL     INTVEC / CHECK FOR DIAGNOSTIC LOAD
008      003B C34502      JMP      INTRP / BRANCH TO HANDLER
009      003E 76      HLT      / SHOULD NEVER REACH HERE
010      EJECT

001      /
002      / SUBJOB POWER-UP ROUTINE
003      /
004      /***POWER-UP ROUTINE
005      /
006      003F 318FFD      PWRUP, LXI     SP, STACK      / SET STACK POINTER
007      /
008      /***INITIALIZE PARALLEL PORT AND STOP CRT
009      /
010      0042 3E80      MVI     A, PDPWR      / A ← VIDEO POWER MASK
011      0044 D33E      OUT     PAROUT      / STOP BEEPER; CRT POWER ON
012      0046 AF      CLA      / A ← 0

```

```

013 0047 D338          OUT      CRTCTL          / STOP CRT CONTROLLER
014                    /
015                    /***WAIT FOR ANY HARDWARE RESETS TO SETTLE
016                    /
017 0049 010002        LXI      B,1000          / [B,C] ← 1000
018 004C 0B           PWR010, DCX      B          / WAIT LOOP FOR HARDWARE
019 004D 78           MOV      A,B          / [B,C].EQ. [0,0]?
020 004E B1           ORA      C          / X
021 004F C24C00        JNZ     PWR010          / BRANCH IF [B,C].NE. [0, ]
022                    EJECT

```

```

001                    /
002                    /***ROM CHECKSUM
003                    /
004

```

PATCH 3 NOP'S (00, 00, 00) TO
SKIP THE ROM CHECKSUM!

```

007
008 0052 C0BE00        CALL     ROMCHK          / VALIDATE CHECKSUM
009                    / RETURN ONLY IF GOOD!
010                    EJECT

```

```

001                    /
002                    /***RAM DIAGNOSTIC
003                    /
004 0055 2100F8        PWR020, LXI      H,RAMLO          / [H,L] ← START OF RAM
005 0058 010008        LXI      B,RAMSIZ          / [B,C] ← RAM SIZE
006 005B 1601          MVI      D,01             / D ← PATTERN
007 005D AF           CLA                      / CLEAR A FOR TESTING
008                    /
009 005E 72           PWR030, MOV      M,D          / STORE PATTERN
010 005F 23           INX      H          / INCREMENT POINTER
011 0060 0B           DCX      B          / DECREMENT COUNTER
012 0061 B8           CMP      B          / TEST B.EQ. 0
013 0062 C25E00        JNZ     PWR030          / CONTINUE LOOP
014 0065 B9           CMP      C          / C.EQ. 0?
015 0066 C25E00        JNZ     PWR030          / NO, CONTINUE
016 0069 1E08          MVI      E,08           / E ← COUNTER
017                    /
018 006B 2100F8        PWR040, LXI      H,RAMLO          / [H,L] ← START OF RAM
019 006E 010008        LXI      B,RAMSIZ          / [B,C] ← RAM SIZE
020 0071 7E           PWR050, MOV      A,M          / A ← CURRENT CONTENTS
021 0072 BA           CMP      D          / PATTERNS MATCH?
022 0073 C2AF00        JNZ     PWR020          / Z.EQ. 0 => ERROR
023 0076 87           ADD      A          / SHIFT LEFT
024 0077 77           MOV      M,A          / STORE INTO RAM
025 0078 23           INX      H          / INCREMENT POINTER
026 0079 0B           DCX      B          / DECREMENT COUNTER
027 007A AF           CLA                      / CLEAR A
028 007B B8           CMP      B          / B.EQ. 0?
029 007C C27100        JNZ     PWR050          / NO, CONTINUE
030 007F B9           CMP      C          / C.EQ. 0?
031 0080 C27100        JNZ     PWR050          / BRANCH IF NOT ZERO
032 0083 7A           MOV      A,D          / A ← PATTERN
033 0084 87           ADD      A          / SHIFT LEFT
034 0085 57           MOV      D,A          / D ← NEW PATTERN
035 0086 1D           DCR      E          / DECREMENT LOOP COUNTER
036 0087 C26B00        JNZ     PWR040          / E.NE. 0 => CONTINUE
037                    EJECT

```

```

001                    SUBJOB DECIDE IF DIAGNOSTIC OR P180
002
003                    / THIS AREA WILL DETERMINE IF A DIAGNOSTIC
004                    / IS TO BE RUN. IF, ON POWER UP, A DEVICE IS
005                    / CONNECTED TO THE PERIPHERAL PORT 2, AND
006                    / THAT DEVICE SETS 'DATA SET READY', WE WILL
007                    / UNCONDITIONALLY JUMP TO THE DIAGNOSTIC
008                    / LOADER SYSTEM, AND NOT START THE P180.
009                    / IF 'DATA SET READY' IS NOT
010                    / PRESENT, WE CONTINUE ON WITH NORMAL P180.
011
012 008A DE3C          IN      SP2CTL          / LOOK FOR 'DSR' ON PORT 2
013 008C E680          ANI     SP2DSR          / IS IT SET?
014 008E C20033        JNZ     PDIA           / YES! GO TO DIAG LOADER SYSTEM
015
016                    / NOT THERE, DO P180 COLD START
017                    EJECT

```



```

001      SUBJOB INITIALIZE THE P180
002      /
003      / I/O DEVICE INITIALIZATION
004      /
005 0091 CD0002      CALL      SPLINI      / INITIALIZE SPOOLER
006 0094 CD9E03      CALL      CLKINI      / INITIALIZE CLOCKS
007 0097 CD6B02      CALL      CRTINI      / INITIALIZE CRT CONTROLLER
008 009A CD2004      CALL      PFINIT      / INITIALIZE MAINFRAME PORT
009      /
010      / ENABLE INTERRUPTS
011      /
012 009D 3EC9      MVI      A,C9      / A ← RET INSTRUCTION
013 009F 3200F8      STA      INIVEC      / STORE INTO VECTOR
014 00A2 FB      EI      / ENABLE INTERRUPTS
015      /
016 00A3 CD9F05      CALL      KBDINI      / INITIALIZE KEYBOARD
017      /
018      / EXIT TO EXEC
019      /
020 00A6 C3EF00      JMP      EXEC
021      EJECT      20

```

```

001      SUBJOB ERRORS IN POWER-UP
002      /
003      / ERROR HANDLERS
004      /
005 00A9 CD0F01      PWRE10, CALL      BEEP      / ROM CHECKSUM FAILED
006 00AC C3A900      JMP      PWRE10      / STAY HERE
007      /
008 00AF CD0F01      PWRE20, CALL      BEEP      / TURN ON BEEPER
009 00B2 010004      LXI      B,2000      / [B,C] ← COUNTER
010      PWRE21,
011 00B5 0B      DCX      B      / COUNT DOWN
012 00B6 78      MOV      A,B      / SEE IF ZERO
013 00B7 B1      ORA      C      / X
014 00B8 C2B500      JNZ      PWRE21      / BRANCH UNTIL ZERO
015 00BB C3AF00      JMP      PWRE20      / TURN ON BEEP AGAIN
016      EJECT

```

```

001      SUBJOB ROM CHECKSUM DIAGNOSTIC
002      /
003      /***ROM CHECKSUM
004      /
005      / EACH 1K ROM HAS A CHECKSUM WHICH IS STORED IN THE
006      / TOP OF THE LAST 1K ROM. THIS ROUTINE VALIDATES
007      / THE CHECKSUM.
008      /
009      /***CALLING SEQUENCE
010      /
011      /      CALL      ROMCHK
012      /
013      /***PARAMETERS :
014      /
015      /      CHECKSUMS STORED AT TOP OF MEMORY
016      /
017      /***REGISTER USAGE:
018      /
019      /      [B,C] : SCRATCH
020      /      [D,E] : SCRATCH
021      /      [H,L] : SCRATCH
022      /
023      /***RETURN
024      /
025      /      RETURN ONLY IF OKAY; ELSE
026      /      UNCONDITIONAL JUMP TO ERROR BEEPER
027      /
028      EJECT

```

```

001      / INITIALIZE POINTER, BYTE COUNT TO COMPUTE CHECKSUM.
002      / A CHECKSUM IS COMPUTED ON ALL 1024 BYTES FOR EACH 1K
003      / AREA OF PROM EXCEPT THE LAST. THESE CHECKSUMS ARE
004      / STORED IN CONSECUTIVE LOCATIONS AT THE TOP OF THE
005      / LAST PROM. FOR THE LAST PROM,
006      / THE CHECKSUM IS COMPUTED
007      / FOR THE FIRST 1023 BYTES (WHICH INCLUDES THE LOWER-
008      / PROM CHECKSUMS) AND COMPARED WITH

```

```

009      / A CHECKSUM STORED IN THE
010      / LAST BYTE OF LAST PROM.
011
012      ROMCHK,
013 00BE 21F237      LXI      H,ROMHI-NUM1K+1 / ADDRESS OF CHECKSUMS
014 00C1 E5          PUSH     H          / SAVE ON STACK
015
016 00C2 210000      LXI      H,ROMLO / START OF PROGRAM PROMS
017 00C5 0E0E      MVI      D,NUM1K / SET # OF PROMS TO PASS OVER
018
019      /
020      / HERE TO TEST A 1K AREA OF PROM
021
022      ROMTES,
022 00C7 AF          CLA          / SET CARRY AND A TO 0
023 00C8 F5          PUSH     PSW    / SET STACK
024 00C9 11FF03      LXI      D,400-1 / # OF BYTES TO DO (-1)
025
026      /
027      / COMPUTE THE CHECKSUM
028
029      ROMTS1,
029 00CC F1          POP      PSW    / GET CURRENT SUM
030 00CD 8E          ADC      M      / ADD IN NEXT BYTE
031 00CE CE00      ACI      0      / ADD IN CARRY
032 00D0 F5          PUSH     PSW    / SAVE CURRENT SUM
033
034 00D1 23          INX      H      / INDEX THE BYTE POINTER
035 00D2 1B          DCX      D      / DECR COUNT
036 00D3 7B          MOV      A,E    / TEST D/E FOR 0
037 00D4 B2          DRA      D      / X
038 00D5 C2CC00      JNZ     ROMTS1 / NOT DONE, LOOP
039
040      /
041      / ARE WE IN THE LAST 1K?
042 00D8 F1          POP      PSW    / GET SUM
043 00D9 0D          DCR      C      / IN LAST 1K?
044 00DA CAE100      JZ      ROMTS2 / YES, MAKE FINAL CHECK
045
046      /
047      / HERE WHEN NOT IN LAST PROM; ADD LAST BYTE OF THIS
048 00DD 8E          ADD      M      / ADD LAST BYTE
049 00DE CE00      ACI      0      / ADD LAST CARRY
050 00E0 23          INX      H      / TO NEXT PROM
051
052      /
053      / HERE TO TEST CHECKSUM
054
055      ROMTS2,
055 00E1 E3          XTHL           / GET PTR TO SUM
056 00E2 96          SUB      M      / SUM-STORED = 0?
057 00E3 C2A900      JNZ     FWRE10 / NO! ERROR
058
059      /
060      / CHECKSUM OK!
061 00E6 23          INX      H      / STEP TO NEXT SUM IN MEMORY
062 00E7 E3          XTHL           / SAVE SUM PTR; GET PROM PTR
063
064 00E8 79          MOV      A,C    / GET COUNT OF AREAS TO DO
065 00E9 B7          TST           / ANY MORE?
066 00EA C2C700      JNZ     ROMTES / YES, LOOP
067
068      /
069      / DONE!
070 00ED E1          POP      H      / CLEAN STACK
071 00EE C9          RET
072      EJECT

001      SUBJOB  SYSTEM EXECUTIVE
002
003      /
004      / ***SYSTEM EXEC
005 00EF 318FFD      EXEC,  LXI      SP,STACK / INITIALIZE STACK
006
007 00F2 3A9BFD      EXEC10, LDA     SPLBLK+BFUSE / A ← SPOOLER USAGE COUNT
008 00F5 B7          TST           / A.EQ. 0 => SPOOLER EMPTY
009 00F6 C42F02      CNZ     SPOOLR / CALL SPOOLER
010 00F9 3AADFD      LDA     KBDBLK+BFUSE / A ← KEYBOARD USAGE COUNT
011 00FC B7          TST           / A.EQ. 0 => NO KEYSTROKES
012 00FD C4CD06      CNZ     KBD CMD / CALL KEYBOARD HANDLER
013 0100 C3F200      JMP     EXEC10 / CONTINUE ON
014      EJECT

```

```

001          SUBJOB  MOVE STRING UTILITY
002          /
003          /***SUBROUTINE MOVSTR
004          /
005          /***CALLING SEQUENCE:
006          /
007          /          CALL    MOVSTR  - STANDARD STRING FORMAT
008          /          CALL    MOVSI0  - B LOADED WITH COUNT
009          /
010          /***PARAMETERS:
011          /
012          /          [D,E] : ADDRESS OF STRING TO BE MOVED
013          /          [H,L] : ADDRESS OF DESTINATION
014          /
015          /***REGISTER USAGE
016          /
017          /          A      : SCRATCH
018          /          [B,C] : SCRATCH
019          /          [D,E] : SOURCE ADDRESS (DESTROYED)
020          /          [H,L] : DESTINATION ADDRESS (DESTROYED)
021          /
022          /***STRING FORMAT:
023          /
024          /          BYTE 0 : LENGTH OF STRING
025          /          BYTE 1 : DATA
026          /          BYTE 2 : DATA
027          /          ETC     ETC
028          /
029          /
030 0103 1A      MOVSTR, LDAX    D          / A ← BYTE COUNT
031 0104 47      MOV     B,A          / B ← BYTE COUNT
032 0105 13      INX     D          / BUMP POINTER
033          /
034 0106 1A      MOVSI0, LDAX   D          / A ← DATA BYTE
035 0107 77      MOV     M,A          / STORE IT
036 0108 13      INX     D          / BUMP SOURCE POINTER
037 0109 23      INX     H          / BUMP DESTINATION POINTER
038 010A 05      DCR     B          / DECREMENT COUNT
039 010B C20601  JNZ     MOVSI0       / COUNT.NE.0 => CONTINUE
040 010E C9      RET                    / EXIT
041          EJECT

```

```

001          SUBJOB  BEEPER SUBROUTINE
002          /
003          /***SUBROUTINE BEEP
004          /
005          /***TURN ON BEEPER
006          /
007          /***CALLING SEQUENCE:
008          /
009          /          CALL    BEEP    - ONE SECOND BEEP
010          /          CALL    BEEP10  - ONE TENTH SECOND BEEP
011          /
012          /***PARAMETERS:
013          /
014          /          NONE
015          /
016          /***REGISTER USAGE:
017          /
018          /          A      : SCRATCH
019          /          [B,C] : NOT USED
020          /          [D,E] : NOT USED
021          /          [H,L] : NOT USED
022          /
023          /
024 010F 3E3C     BEEP,   MVI     A,@60       / A ← 60HZ COUNT
025 0111 C31601   JMP     BEEP20      / GO TO COMMON CODE
026          /
027 0114 3E06     BEEP10, MVI     A,6         / A ← 0.1 SEC BEEP
028          /
029 0116 328FFD   BEEP20, STA     TMRBEP      / LOAD TIMER
030 0119 3AB4FD   LDA     POSAVE      / GET STATUS OF PAROUT
031 011C F640     ORI     POBEEP      / SET BEEPER FLAG
032 011E D33E     OUT     PAROUT      / TURN ON BEEPER
033 0120 32B4FD   STA     POSAVE      / SAVE PORT STATUS
034 0123 C9      RET                    / EXIT
035          EJECT

```



```

001          SUBJOB  BUFFER HANDLERS
002          /
003          /***CIRCULAR BUFFER HANDLERS
004          /
005          /***ROUTINES:
006          /
007          /          BFINIT : INITIALIZE BUFFER
008          /          BFCH   : BUFFER A CHARACTER (BYTE)
009          /          UBFCB  : UNBUFFER A CHARACTER (BYTE)
010          /
011          /          EJECT
012          /
013          /
014          /
015          /
016          /
017          /
018          /
019          /
020          /
021          /
022          /
023          /
024          /
025          /
026          /
027          /
028          /
029          /
030          /
031          /
032          /
033          /
034          /
035          /
036          /
037          /
038          /
039          /
040          /
041          /
042          /
043          /
044          /
045          /
046          /
047          /
048          /
049          /
050          /
051          /
052          /
053          /
054          /
055          /
056          /
057          /
058          /
059          /
060          /
061          /
062          /
063          /
064          /
065          /
066          /
067          /
068          /
069          /
070          /
071          /
072          /
073          /
074          /
075          /
076          /
077          /
078          /
079          /
080          /
081          /
082          /
083          /
084          /
085          /
086          /
087          /
088          /
089          /
090          /
091          /
092          /
093          /
094          /
095          /
096          /
097          /
098          /
099          /
100          /
101          /
102          /
103          /
104          /
105          /
106          /
107          /
108          /
109          /
110          /
111          /
112          /
113          /
114          /
115          /
116          /
117          /
118          /
119          /
120          /
121          /
122          /
123          /
124          /
125          /
126          /
127          /
128          /
129          /
130          /
131          /
132          /
133          /
134          /
135          /
136          /
137          /
138          /
139          /
140          /
141          /
142          /
143          /
144          /
145          /
146          /
147          /
148          /
149          /
150          /
151          /
152          /
153          /
154          /
155          /
156          /
157          /
158          /
159          /
160          /
161          /
162          /
163          /
164          /
165          /
166          /
167          /
168          /
169          /
170          /
171          /
172          /
173          /
174          /
175          /
176          /
177          /
178          /
179          /
180          /
181          /
182          /
183          /
184          /
185          /
186          /
187          /
188          /
189          /
190          /
191          /
192          /
193          /
194          /
195          /
196          /
197          /
198          /
199          /
200          /
201          /
202          /
203          /
204          /
205          /
206          /
207          /
208          /
209          /
210          /
211          /
212          /
213          /
214          /
215          /
216          /
217          /
218          /
219          /
220          /
221          /
222          /
223          /
224          /
225          /
226          /
227          /
228          /
229          /
230          /
231          /
232          /
233          /
234          /
235          /
236          /
237          /
238          /
239          /
240          /
241          /
242          /
243          /
244          /
245          /
246          /
247          /
248          /
249          /
250          /
251          /
252          /
253          /
254          /
255          /
256          /
257          /
258          /
259          /
260          /
261          /
262          /
263          /
264          /
265          /
266          /
267          /
268          /
269          /
270          /
271          /
272          /
273          /
274          /
275          /
276          /
277          /
278          /
279          /
280          /
281          /
282          /
283          /
284          /
285          /
286          /
287          /
288          /
289          /
290          /
291          /
292          /
293          /
294          /
295          /
296          /
297          /
298          /
299          /
300          /
301          /
302          /
303          /
304          /
305          /
306          /
307          /
308          /
309          /
310          /
311          /
312          /
313          /
314          /
315          /
316          /
317          /
318          /
319          /
320          /
321          /
322          /
323          /
324          /
325          /
326          /
327          /
328          /
329          /
330          /
331          /
332          /
333          /
334          /
335          /
336          /
337          /
338          /
339          /
340          /
341          /
342          /
343          /
344          /
345          /
346          /
347          /
348          /
349          /
350          /
351          /
352          /
353          /
354          /
355          /
356          /
357          /
358          /
359          /
360          /
361          /
362          /
363          /
364          /
365          /
366          /
367          /
368          /
369          /
370          /
371          /
372          /
373          /
374          /
375          /
376          /
377          /
378          /
379          /
380          /
381          /
382          /
383          /
384          /
385          /
386          /
387          /
388          /
389          /
390          /
391          /
392          /
393          /
394          /
395          /
396          /
397          /
398          /
399          /
400          /
401          /
402          /
403          /
404          /
405          /
406          /
407          /
408          /
409          /
410          /
411          /
412          /
413          /
414          /
415          /
416          /
417          /
418          /
419          /
420          /
421          /
422          /
423          /
424          /
425          /
426          /
427          /
428          /
429          /
430          /
431          /
432          /
433          /
434          /
435          /
436          /
437          /
438          /
439          /
440          /
441          /
442          /
443          /
444          /
445          /
446          /
447          /
448          /
449          /
450          /
451          /
452          /
453          /
454          /
455          /
456          /
457          /
458          /
459          /
460          /
461          /
462          /
463          /
464          /
465          /
466          /
467          /
468          /
469          /
470          /
471          /
472          /
473          /
474          /
475          /
476          /
477          /
478          /
479          /
480          /
481          /
482          /
483          /
484          /
485          /
486          /
487          /
488          /
489          /
490          /
491          /
492          /
493          /
494          /
495          /
496          /
497          /
498          /
499          /
500          /
501          /
502          /
503          /
504          /
505          /
506          /
507          /
508          /
509          /
510          /
511          /
512          /
513          /
514          /
515          /
516          /
517          /
518          /
519          /
520          /
521          /
522          /
523          /
524          /
525          /
526          /
527          /
528          /
529          /
530          /
531          /
532          /
533          /
534          /
535          /
536          /
537          /
538          /
539          /
540          /
541          /
542          /
543          /
544          /
545          /
546          /
547          /
548          /
549          /
550          /
551          /
552          /
553          /
554          /
555          /
556          /
557          /
558          /
559          /
560          /
561          /
562          /
563          /
564          /
565          /
566          /
567          /
568          /
569          /
570          /
571          /
572          /
573          /
574          /
575          /
576          /
577          /
578          /
579          /
580          /
581          /
582          /
583          /
584          /
585          /
586          /
587          /
588          /
589          /
590          /
591          /
592          /
593          /
594          /
595          /
596          /
597          /
598          /
599          /
600          /
601          /
602          /
603          /
604          /
605          /
606          /
607          /
608          /
609          /
610          /
611          /
612          /
613          /
614          /
615          /
616          /
617          /
618          /
619          /
620          /
621          /
622          /
623          /
624          /
625          /
626          /
627          /
628          /
629          /
630          /
631          /
632          /
633          /
634          /
635          /
636          /
637          /
638          /
639          /
640          /
641          /
642          /
643          /
644          /
645          /
646          /
647          /
648          /
649          /
650          /
651          /
652          /
653          /
654          /
655          /
656          /
657          /
658          /
659          /
660          /
661          /
662          /
663          /
664          /
665          /
666          /
667          /
668          /
669          /
670          /
671          /
672          /
673          /
674          /
675          /
676          /
677          /
678          /
679          /
680          /
681          /
682          /
683          /
684          /
685          /
686          /
687          /
688          /
689          /
690          /
691          /
692          /
693          /
694          /
695          /
696          /
697          /
698          /
699          /
700          /
701          /
702          /
703          /
704          /
705          /
706          /
707          /
708          /
709          /
710          /
711          /
712          /
713          /
714          /
715          /
716          /
717          /
718          /
719          /
720          /
721          /
722          /
723          /
724          /
725          /
726          /
727          /
728          /
729          /
730          /
731          /
732          /
733          /
734          /
735          /
736          /
737          /
738          /
739          /
740          /
741          /
742          /
743          /
744          /
745          /
746          /
747          /
748          /
749          /
750          /
751          /
752          /
753          /
754          /
755          /
756          /
757          /
758          /
759          /
760          /
761          /
762          /
763          /
764          /
765          /
766          /
767          /
768          /
769          /
770          /
771          /
772          /
773          /
774          /
775          /
776          /
777          /
778          /
779          /
780          /
781          /
782          /
783          /
784          /
785          /
786          /
787          /
788          /
789          /
790          /
791          /
792          /
793          /
794          /
795          /
796          /
797          /
798          /
799          /
800          /
801          /
802          /
803          /
804          /
805          /
806          /
807          /
808          /
809          /
810          /
811          /
812          /
813          /
814          /
815          /
816          /
817          /
818          /
819          /
820          /
821          /
822          /
823          /
824          /
825          /
826          /
827          /
828          /
829          /
830          /
831          /
832          /
833          /
834          /
835          /
836          /
837          /
838          /
839          /
840          /
841          /
842          /
843          /
844          /
845          /
846          /
847          /
848          /
849          /
850          /
851          /
852          /
853          /
854          /
855          /
856          /
857          /
858          /
859          /
860          /
861          /
862          /
863          /
864          /
865          /
866          /
867          /
868          /
869          /
870          /
871          /
872          /
873          /
874          /
875          /
876          /
877          /
878          /
879          /
880          /
881          /
882          /
883          /
884          /
885          /
886          /
887          /
888          /
889          /
890          /
891          /
892          /
893          /
894          /
895          /
896          /
897          /
898          /
899          /
900          /
901          /
902          /
903          /
904          /
905          /
906          /
907          /
908          /
909          /
910          /
911          /
912          /
913          /
914          /
915          /
916          /
917          /
918          /
919          /
920          /
921          /
922          /
923          /
924          /
925          /
926          /
927          /
928          /
929          /
930          /
931          /
932          /
933          /
934          /
935          /
936          /
937          /
938          /
939          /
940          /
941          /
942          /
943          /
944          /
945          /
946          /
947          /
948          /
949          /
950          /
951          /
952          /
953          /
954          /
955          /
956          /
957          /
958          /
959          /
960          /
961          /
962          /
963          /
964          /
965          /
966          /
967          /
968          /
969          /
970          /
971          /
972          /
973          /
974          /
975          /
976          /
977          /
978          /
979          /
980          /
981          /
982          /
983          /
984          /
985          /
986          /
987          /
988          /
989          /
990          /
991          /
992          /
993          /
994          /
995          /
996          /
997          /
998          /
999          /
1000         /

```

```

024 / Z-BIT. EQ. 0 => CHARACTER NOT BUFFERED, BUFFER FULL
025 / Z-BIT. EQ. 1 => CHARACTER BUFFERED
026 /
027 / EJECT

001 012E F5 BFCH, PUSH PSW / SAVE CHAR
002 012F 210400 LXI H, BFLEN / [H,L] <- OFFSET
003 0132 09 DAD B / [H,L] <- BFLEN ADDRESS
004 0133 7E MOV A, M / A <- LENGTH
005 0134 23 INX H / [H,L] <- USAGE ADDRESS
006 0135 BE CMP M / CHECK FOR BUFFER FULL
007 0136 C24001 JNZ BFCH10 / BRANCH IF SPACE AVAILABLE
008 0139 F1 POP PSW / RESTORE CHAR
009 013A 67 MOV H, A / H <- CHAR
010 013B 24 INR H / SET UP TEST
011 013C BC CMP H / TO CLEAR Z-BIT
012 013D C35501 JMP BFCHX / RIGHT HERE
013 /
014 0140 C5 BFCH10, PUSH B / SAVE [B,C]
015 0141 34 INR M / INCREMENT USAGE COUNT
016 0142 2B DCX H / [H,L] <- BFLEN ADDRESS
017 0143 7E MOV A, M / A <- LENGTH
018 0144 2B DCX H / [H,L] <- OPTR
019 0145 2B DCX H / [H,L] <- IPTR ADDRESS
020 0146 4E MOV C, M / GET CURRENT INPUT POINTER
021 0147 34 INR M / IPTR <- IPTR + 1
022 0148 3D DCR A / A <- MASK
023 0149 A6 ANA M / KEEP IPTR MODULO BASE
024 014A 77 MOV M, A / UPDATE POINTER
025 014B 2B DCX H / [H,L] <- BFBASE LOW
026 014C 2B DCX H / [H,L] <- BFBASE HIGH
027 014D 0600 MVI B, 0 / B <- 0
028 014F E7 GETHL / [H,L] <- BUFFER BASE
029 0150 09 DAD B / [H,L] <- BUFFER ADDRESS
030 0151 C1 POP B / RESTORE [B,C]
031 0152 F1 POP PSW / GET CHARACTER
032 0153 77 MOV M, A / STORE INTO BUFFER
033 0154 BF CMP A / SET Z-BIT
034 /
035 0155 C9 BFCHX, RET / EXIT
036 / EJECT

001 /
002 / ***SUBROUTINE UBFCH
003 /
004 / ***UNBUFFER A CHARACTER (BYTE) ((GET A BYTE))
005 /
006 / ***CALLING SEQUENCE:
007 /
008 / CALL UBFCH
009 /
010 / ***PARAMETERS:
011 /
012 / A : CHARACTER ON EXIT
013 / [B,C] : BUFFER BLOCK ADDRESS
014 /
015 / ***REGISTER USAGE:
016 /
017 / A : CHARACTER
018 / [B,C] : BUFFER BLOCK ADDRESS (PRESERVED)
019 / [D,E] : PRESERVED
020 / [H,L] : SCRATCH
021 /
022 / ***EXIT:
023 /
024 / Z-BIT. EQ. 0 => BUFFER EMPTY
025 / Z-BIT. EQ. 1 => A HAS CHARACTER
026 /
027 / EJECT

001 0156 C5 UBFCH, PUSH B / SAVE [B,C]
002 0157 210500 LXI H, BFUSE / [H,L] <- OFFSET
003 015A 09 DAD B / [H,L] <- USAGE COUNT
004 015B AF CLA / A <- 0
005 015C BE CMP M / BFUSE. EQ. 0 => BUFF EMPTY
006 015D C26601 JNZ UBFCH1 / BRANCH IF NOT EMPTY
007 0160 C1 POP B / POP STACK

```

```

008 0161 FEFF          CPI      -1          / CLEAR Z-BIT
009 0163 C38001       JMP      UBFCHX      / GO TO EXIT
010                   /
011 0166 35          UBFCH1, DCR      M          / USAGE <- USAGE - 1
012 0167 2B          DCX      H          / [H,L] <- BLEN
013 0168 2B          DCX      H          / [H,L] <- OPTR
014 0169 4E          MOV      C,M        / C <- OFFSET TO BUFFER
015 016A 0600       MVI      B,0        / B <- 0
016 016C 2B          DCX      H          / [H,L] <- IPTR
017 016D 2B          DCX      H          / [H,L] <- BFBASE ADDR LOW
018 016E 2B          DCX      H          / [H,L] <- BFBASE ADDR H 3H
019 016F E7          GETHL          / [H,L] <- BUFFER BASE
020 0170 09          DAD      B          / [H,L] <- CHARACTER ADDR
021 0171 7E          MOV      A,M        / A <- CHARACTER
022 0172 C1          POP      B          / RESTORE [B,C]
023 0173 F5          PUSH     PSW        / SAVE CHARACTER
024 0174 210400     LXI      H,BFLEN    / [H,L] <- OFFSET
025 0177 09          DAD      B          / [H,L] <- LENGTH ADDRESS
026 0178 7E          MOV      A,M        / A <- BUFFER LENGTH
027 0179 3D          DCR      A          / CREATE MASK
028 017A 2B          DCX      H          / [H,L] <- OPTR ADDRESS
029 017B 34          INR      M          / OPTR <- OPTR + 1
030 017C A6          ANA      M          / MASK OUT OVERFLOW
031 017D 77          MOV      M,A        / SET UP POINTER
032 017E F1          POP      PSW        / A <- CHARACTER
033 017F BF          CMP      A          / SET Z-BIT
034                   /
035 0180 C9          UBFCHX, RET        / EXIT
036                   EJECT

001                   SUBJOB  BCD-TO-BINARY CONVERSION
002                   /
003                   /***BCD-TO-BINARY CONVERSION
004                   /
005                   /***ENTRY POINTS:
006                   /
007                   /          BCDBN4 : 4-DIGIT BCD
008                   /          BCDBN3 : 3-DIGIT BCD
009                   /          BCDBN2 : 2-DIGIT BCD
010                   /          BCDBN1 : 1-DIGIT BCD
011                   /
012                   /***REGISTER USAGE:
013                   /
014                   /          A      - SCRATCH
015                   /          [B,C] - SCRATCH
016                   /          [D,E] - POINTER TO BCD NUMBER (DESTROYED)
017                   /          [H,L] - BINARY VALUE ON EXIT
018                   /
019                   /***ILLEGAL BCD NUMBER YIELDS RESULT OF ZERO
020                   /
021                   /***EXITS:
022                   /
023                   /          Z-BIT.EQ.0 => ILLEGAL BCD NUMBER
024                   /          Z-BIT.EQ.1 => VALID RESULT
025                   /
026 0181 210000     BCDBN4, LXI      H,0          / INITIALIZE RESULT
027 0184 01E803     LXI      B,@1000        / [B,C] <- 1000
028 0187 CDA901     CALL     BCDSUB        / GET AND VALIDATE DIGIT
029 018A C2A801     JNZ      BCDX         / BRANCH ON ERROR
030 018D 13          INX      D          / MOVE PTR TO NEXT DIGIT
031                   /
032 018E 016400     BCDBN3, LXI      B,@100        / [B,C] <- 100
033 0191 CDA901     CALL     BCDSUB        / GET AND VALIDATE DIGIT
034 0194 C2A801     JNZ      BCDX         / BRANCH ON ERROR
035 0197 13          INX      D          / MOVE PTR TO NEXT DIGIT
036                   /
037 0198 010A00     BCDBN2, LXI      B,@10         / [B,C] <- 10
038 019B CDA901     CALL     BCDSUB        / GET AND VALIDATE DIGIT
039 019E C2A801     JNZ      BCDX         / BRANCH ON ERROR
040 01A1 13          INX      D          / MOVE PTR TO UNIT'S DIGIT
041                   /
042 01A2 010100     BCDBN1, LXI      B,1          / [B,C] <- 1
043 01A5 CDA901     CALL     BCDSUB        / GET AND VALIDATE DIGIT
044                   /
045 01A8 C9          BCDX,  RET          / EXIT
046                   EJECT

```



```

001 01A9 1A          BCDSUB, LDAX    D          / A ← BCD DIGIT
002 01AA D630          SUI    ASC0          / MAKE IT BINARY
003 01AC FABD01        JM     BCDS20        / BRANCH ON ERROR
004 01AF FE0A          CPI    A             / CHECK FOR VALID BCD DIGIT
005 01B1 F2BD01        JP     BCDS20        / BRANCH ON ERROR
006 01B4 B7           TST
007 /
008 01B5 CAC101        BCDS10, JZ     BCDSX    / GO TO EXIT
009 01B8 09           DAD    B             / ADD TO BINARY VALUE
010 01B9 3D           DCR    A             / DECREMENT POINTER
011 01BA C3B501        JMP    BCDS10        / AND CONTINUE
012 /
013 01BD 210000        BCDS20, LXI    H,0    / CLEAR RESULT ON ERROR
014 01C0 BC           CMP    H             / CLEAR Z-BIT
015 /
016 01C1 C9           BCDSX, RET          / EXIT
017 EJECT

001 /
002 /
003 /
004 /
005 /
006 /
007 /
008 /
009 /
010 /
011 /
012 /
013 /
014 /
015 /
016 /
017 /
018 /
019 01C2 3E30          BNBCD4, MVI    A,ASC0    / SET A
020 01C4 0118FC        LXI    B,-@1000    / [B,C] ← DECREMENT
021 /
022 01C7 09           BN010, DAD    B          / COMPUTE THOUSAND'S DIGIT
023 01C8 D2CF01        JNC    BN020        / BRANCH ON BORROW
024 01CB 3C           INR    A             / BUMP DIGIT
025 01CC C3C701        JMP    BN010        / CONTINUE
026 /
027 01CF 12           BN020, STAX   D          / STORE DIGIT
028 01D0 13           INX    D             / BUMP POINTER
029 01D1 01E803        LXI    B,@1000    / [B,C] ← 1000
030 01D4 09           DAD    B             / RESET BINARY VALUE
031 /
032 01D5 3E30          BNBCD3, MVI    A,ASC0    / RESET A
033 01D7 019CFF        LXI    B,-@100    / [B,C] ← DECREMENT
034 /
035 01DA 09           BN030, DAD    B          / COMPUTE'S HUNDRED'S DIGIT
036 01DB D2E201        JNC    BN040        / BRANCH ON BORROW
037 01DE 3C           INR    A             / BUMP DIGIT
038 01DF C3DA01        JMP    BN030        / AND CONTINUE
039 /
040 01E2 12           BN040, STAX   D          / STORE DIGIT
041 01E3 13           INX    D             / BUMP POINTER
042 01E4 016400        LXI    B,@100    / RESET BINARY VALUE
043 01E7 09           DAD    B             / FOR TEN'S
044 /
045 01E8 3E30          BNBCD2, MVI    A,ASC0    / SET A
046 01EA 01F6FF        LXI    B,-@10    / [B,C] ← DECREMENT
047 /
048 01ED 09           BN050, DAD    B          / COMPUTE TEN'S DIGIT
049 01EE D2F501        JNC    BN050        / BRANCH ON BORROW
050 01F1 3C           INR    A             / INCREMENT RESULT
051 01F2 C3ED01        JMP    BN050        / AND CONTINUE
052 /
053 01F5 12           BN060, STAX   D          / STORE DIGIT
054 01F6 13           INX    D             / BUMP POINTER
055 01F7 010A00        LXI    B,@10    / RESET BINARY VALUE
056 01FA 09           DAD    B             / FOR UNIT'S DIGIT
057 /
058 01FB 3E30          BNBCD1, MVI    A,ASC0    / SET A
059 01FD 85           ADD    L             / COMPUTE UNIT'S DIGIT

```

```

060 01FE 12          STAX   D          / STORE IT
061 01FF C9          RET     / EXIT
062                  EJECT

001                  SUBJOB  SPOOLER HANDLERS
002                  /
003                  / SPOOLER FUNCTIONS
004                  /
005                  / SPLINI - INITIALIZE SPOOLER QUEUE
006                  /
007 0200 01COFF      SPLINI, LXI    B; SPLBUF   / [B,C] ← START OF QUEUE
008 0203 114000      LXI    D; SPLBFL   / [D,E] ← LENGTH OF QUEUE
009 0206 2196FD      LXI    H; SPLBLK   / [H,L] ← SPOOLER BLOCK
010 0209 CD2401      CALL   BFINIT   / INITIALIZE SPOOLER
011 020C C9          RET     / RETURN
012                  EJECT

001                  /
002                  / SPOOLI - BUFFER A SPOOLER COMMAND
003                  /
004                  / COMMAND FORMAT.
005                  /
006                  /          BYTES    USE
007                  /          -----
008                  /          0      NUMBER OF BYTES IN SPOOLER COMMAND
009                  /          1-7    ADDRESS OF START OF COMMAND
010                  /          DATA AS REQUIRED
011                  /
012                  / ***CALLING SEQUENCE:
013                  /
014                  /          CALL   SPOOLI
015                  /
016                  / ***PARAMETERS.
017                  /
018                  /          [B,C] → COMMAND PACKET
019                  /
020                  / ***REGISTER USAGE:
021                  /
022                  /          A      SCRATCH
023                  /          [B,C]  COMMAND PACKET ADDRESS (DESTROYED)
024                  /          [D,E]  PRESERVED
025                  /          [H,L]  SCRATCH
026                  /
027                  / ***EXIT:
028                  /
029                  /          Z. EQ. 0 ⇒ COMMAND NOT QUEUED. SPACE NOT AVAILABLE
030                  /          Z. EQ. 1 ⇒ COMMAND QUEUED
031                  /
032                  EJECT

001 020D 0A          SPOOLI, LDAX   B          / A ← BYTE COUNT
002 020E 2F          CMA          / MAKE IT NEGATIVE
003 020F 3C          INR          A          / TWO'S COMP
004 0210 219BFD      LXI    H, SPLBLK+BFUSE / [H,L] → WORDS IN USE
005 0213 96          SUB     M          / A ← SPACE LEFT
006 0214 C640        ADI    SPLBFL   / A.LT. 0 ⇒ NOT ENOUGH R 3M
007 0216 F21D02      JP     SPLI10   / A.GE. 0 ⇒ SPACE AVAILABLE
008 0219 B7          TST     / CLEAR Z-BIT
009 021A C32E02      JMP    SPLIX    / GO TO EXIT
010                  /
011 021D 0A          SPLI10, LDAX  B          / A ← BYTE COUNT
012                  /
013 021E F5          SPLI15, PUSH  PSW       / SAVE COUNT
014 021F 03          INX     B          / BUMP POINTER
015 0220 0A          LDAX   B          / A ← COMMAND BYTE
016 0221 C5          PUSH  B          / SAVE POINTER
017 0222 0196FD      LXI    B; SPLBLK / [B,C] ← SPOOLER BLOCK
018 0225 CD2E01      CALL   BFCH     / BUFFER BYTE
019 0228 C1          POP    B          / RESTORE COMMAND POINTER
020 0229 F1          POP    PSW      / RESTORE COUNT
021 022A 3D          DCR    A          / COUNT. EQ. 0 ⇒ QUIT
022 022B C21E02      JNZ   SPLI15   / COUNT. NE. 0 ⇒ CONTINUE
023                  /
024 022E C9          SPLIX, RET     / EXIT
025                  EJECT

```

```

001
002 /
003 /
004 /
005 /
006 /
007 /
008 /
009 /
010 /
011 /
012 /
013 /
014 /
015 /
016 /
017 /
018 /
019 /
020 /
021 /
    EJECT

001 022F 0196FD    SPOOLR, LXI    B, SPLBLK    / [B,C] ← SPOOLER BLOCK
002 0232 CD5601    CALL    UBFCH    / GET LOW-ORDER COMMAND
003 0235 C24402    JNZ     SPLRX    / BRANCH ON ERROR
004 0238 F5        PUSH   PSW       / SAVE BYTE
005 0239 CD5601    CALL    UBFCH    / GET HIGH-ORDER ADDRESS
006 023C C24302    JNZ     SPLR10   / BRANCH ON ERROR
007 023F 67        MOV    H, A      / H ← HIGH-ORDER
008 0240 F1        POP    PSW       / POP STACK
009 0241 6F        MOV    L, A      / L ← LOW-ORDER
010 0242 E9        PCHL             / EXECUTE
011
012 0243 F1        SPLR10, POP     PSW       / CLEAN STACK
013
014 0244 C9        SPLRX,  RET      / RETURN
015

001
002 /
003 /
004 /
005 /
006 /
007 /
008 /
009 /
010 /
011 /
012 /
013 /
014 /
015 /
016 /
017 /
018 /
019 /
020 /
021 /
022 /
    SUBJOB INTERRUPT HANDLER

001
002 /
003 /
004 /
005 /
006 /
007 /
008 /
009 /
010 /
011 /
012 /
013 /
014 /
015 /
016 /
017 /
018 /
019 /
020 /
021 /
022 /
    AN RST 7 INSTRUCTION IS GENERATED WHEN:
    INTERRUPT ENABLE IS TRUE AND
    A DEVICE ISSUES AN INTERRUPT REQUEST

001
002 /
003 /
004 /
005 /
006 /
007 /
008 /
009 /
010 /
011 /
012 /
013 /
014 /
015 /
016 /
017 /
018 /
019 /
020 /
021 /
022 /
    SAVES ALL REGISTERS AND RESTORES THEM

001
002 /
003 /
004 /
005 /
006 /
007 /
008 /
009 /
010 /
011 /
012 /
013 /
014 /
015 /
016 /
017 /
018 /
019 /
020 /
021 /
022 /
    CHECK FOR PERIPHERAL PORT. IF ACTIVE,
    ONLY DO IT!

001 0245 F5        INTRP,  PUSH   PSW       / SAVE A
002 0246 C5        PUSH   B         / SAVE [B,C]
003 0247 D5        PUSH   D         / SAVE [D,E]
004 0248 E5        PUSH   H         / SAVE [H,L]
005
006 /
007 /
008 /
009 0249 DB3A     IN      SP15IA    / READ STATUS OF PORT
010 024B 47        MOV    B, A      / SAVE IN B FOR CALL
011
012 024C E603     ANI    SP3RRY+SPSTRY / CHECK RECVR + XMIT
013 024E CA5702    JZ     INTR10    / NO ACTION, CHECK OTHERS
014

```



```

015 0251 CD5304      CALL    PPRINT    / ACTIVE! GO PROCESS
016
017 0254 C36502      JMP     INTREX    / EXIT TO REGULAR
018
019                /      HERE WHEN PERIPHERAL PORT NOT ACTIVE;
020                /      SEE IF CRT INTERRUPT IS UP
021
022                /      INTR10,
023 0257 DB38         IN      CRTSTA    / GET STATUS
024 0259 E620         ANI     CRTSIR    / CHECK FOR INTERRUPT
025 025B CA6502      JZ      INTREX    /   NONE, EXIT
026
027                /      HERE WHEN CRT INTERRUPTED. IT IS USED AS A
028                /      "CLOCK" FOR TIMERS AND KEYBOARD POLLING
029
030 025E CD6406      CALL    KBDINT    / POLL KEYBOARD
031 0261 FB          EI          / ALLOW RE-ENTRANCY ON TIMERS
032 0262 CDAB03      CALL    CLKINT    / HANDLE TIMERS
033
034                /      EXIT
035
036                /      INTREX,
037 0265 E1          POP     H          / RESTORE AND EXIT
038 0266 D1          POP     D          / X
039 0267 C1          POP     B          / X
040 0268 F1          POP     PSW         / X
041 0269 FB          EI          / ALLOW ALL
042 026A C9          RET
043
044                /      EJECT

SUBJOB CRT CONTROLLER FUNCTIONS
001                /
002                /
003                /      ***CRT CONTROLLERS FUNCTIONS
004                /
005                /      ***ROUTINES.
006                /
007                /      CRTINI - INITIALIZE CRT CONROLLER
008                /
009                /      EJECT

001                /
002                /      ***SUBROUTINE CRTINI
003                /
004                /      ***CALLING SEQUENCE:
005                /
006                /      CALL    CRTINI
007                /
008                /      ***PARAMETERS:
009                /
010                /      NONE
011                /
012                /      ***REGISTER USAGE.
013                /
014                /      A      : SCRATCH
015                /      [B,C] : SCRATCH
016                /      [D,E] : SCRATCH
017                /      [H,L] : SCRATCH
018                /
019                /      ***SUBROUTINES USED
020                /
021                /      DMAINT . DMA INITIALIZATION
022                /
023                /      EJECT

001 026B 3E00      CRTINI, MVI    A, CMDRST    / A ← RESET AND STOP CMD
002 026D D338      OUT    CRTCTL    / WRITE TO CONTROLLER
003                /
004 026F 3E43      MVI    A, COMPB1    / A ← COMPOSITION BYTE 1
005 0271 D339      OUT    CRTDAT    / LOAD IT
006 0273 3E14      MVI    A, COMPB2    / A ← COMPOSITION BYTE
007 0275 D339      OUT    CRTDAT    / LOAD IT
008 0277 3E7B      MVI    A, COMPB3    / A ← COMPOSITION BYTE 3
009 0279 D339      OUT    CRTDAT    / LOAD IT
010 027B 3E36      MVI    A, COMPB4    / A ← COMPOSITION BYTE 4
011 027D D339      OUT    CRTDAT    / LOAD IT
012                /

```

```

013 027F 3E80          MVI    A,DMDLUR          / A ← CURSOR COMMAND
014 0281 D338          OUT    CRTCTL            / ISSUE COMMAND
015 0283 3E7F          MVI    A,CURHDZ          / GET CURSOR COLUMN
016 0285 D338          OUT    CRTDAT            / LOAD IT
017 0287 3E00          MVI    A,CURVER          / GET CURSOR ROW
018 0289 D338          OUT    CRTDAT            / LOAD IT
019
020 028B CDB002        CALL   DMAIN1            / INITIALIZE DMA + FORMAT
021
022
023
024 028E 3EE0          MVI    A,DMDFRE          / A ← PRESET COMMAND
025 0290 D338          OUT    CRTCTL            / PRESET COUNTERS
026
027 0292 3E2F          MVI    A,DMDFR+BURST     / A ← START PARAMETERS
028 0294 D338          OUT    CRTCTL            / START DISPLAY
029
030 /***SYNC LOOP TO START DMA ON VERTICAL RETRACE
031 /
032 0296 D838          CRT010, IN    CRTSTA     / READ STATUS
033 0298 F670          ANI    CRTSIR            / CHECK FOR VERTICAL RET AC
034 029A D29602        JNZ    CRT010           / BRANCH IF ACTIVE.
035 /
036 029D D838          CRT020, IN    CRTSTA     / READ STATUS
037 029F E670          ANI    CRTSIR            / CHECK VERTICAL RETRACE
038 02A1 CA9D02        JZ     CRT020           / BRANCH IF NOT ACTIVE
039 /
040 02A4 3E80          MVI    A,DMACMD          / START DMA ON LEADING EDGE
041 02A6 D328          OUT    DMAIND            / OF VERTICAL RETRACE
042 02A8 3E80          MVI    A,POFWR           / A ← CRT POWER ENABLE
043 02AA D33E          OUT    PAROUT            / TURN ON TUBE
044 02AC 32B4FD        STA    POSAVE            / SAVE STATE
045 02AE C9             RET                      / EXIT
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

001 /
002 /***INITIALIZE REFRESH MEMORY
003 /
004 02D3 2104F8 LXI H: CRTFH / [H, L] ← START OF REFRESH
005 02D6 CDFE02 CALL ROWPAD / DO FIRST ROW
006 02D9 CDFE02 CALL ROWPAD / DO SECOND ROW
007 02DC 060E MVI B: ROWLNI / B ← COUNTER
008 /
009 02DE CD0503 DMA010, CALL ROWLOG / DO A LOGIC ROW
010 02E1 05 DCR B / DONE?
011 02E2 C2DE02 JNZ DMA010 / NO, CONTINUE
012 /
013 02E5 CDFE02 CALL ROWPAD / DO A PAD ROW
014 02E8 CDFE02 CALL ROWPAD / DO A PAD ROW
015 02EB CD2403 CALL ROWBLK / DO BLANK ROW
016 02EE CD2903 CALL ROWST1 / DO FIRST STATUS ROW
017 02F1 CD4003 CALL ROWST2 / DO SECOND STATUS ROW
018 /
019 02F4 0601 MVI B: PADCNT / B ← COUNT
020 02F6 CDFE02 DMA020, CALL ROWPAD / DO A PAD ROW
021 02F9 05 DCR B / DONE?
022 02FA C2F602 JNZ DMA020 / NO, CONTINUE
023 /
024 02FD C9 RET / EXIT
025 EJECT

001 /
002 /***SUBROUTINES TO BUILD REFRESH MEMORY
003 /
004 02FE 36F1 ROWPAD, MVI M: DMAEOR / STORE END-OF-ROW
005 0300 23 INX H / BUMP POINTER
006 0301 3620 MVI M: DMABLK / STORE A PAD CHARACTER
007 0303 23 INX H / BUMP POINTER
008 0304 C9 RET / EXIT
009 /
010 /***DO A LOGIC ROW
011 /
012 0305 0E0B ROWLOG, MVI C: MAXCOL / 11 NODES PER ROW
013 0307 CD1203 CALL ROWBEG / START ROW
014 /
015 030A CD1703 ROWL10, CALL ROWNOD / DO A NODE
016 030D 0D DCR C / DONE?
017 030E C20A03 JNZ ROWL10 / NO, LOOP AGAIN
018 0311 C9 RET / EXIT
019 /
020 0312 1602 ROWBEG, MVI D: 2 / START ROW WITH A FIELD
021 0314 C31903 JMP ROWN10 / ATTRIBUTE AND 2 BLANKS
022 /
023 0317 1606 ROWNOD, MVI D: DSPNOD-1 / CHARACTERS PER NODE
024 /
025 0319 3680 ROWN10, MVI M: DMAFAN / STORE A FIELD ATTRIBUTE
026 /
027 031B 23 ROWN20, INX H / BUMP POINTER
028 031C 3620 MVI M: DMABLK / STORE A BLANK
029 031E 15 DCR D / DONE?
030 031F C21B03 JNZ ROWN20 / NO, LOOP
031 0322 23 INX H / BUMP POINTER
032 0323 C9 RET / EXIT
033 EJECT

001 /
002 /***DO A BLANK ROW
003 /
004 0324 1644 ROWBLK, MVI D: ROWC-1 / 64 BLANKS
005 0326 C31903 JMP ROWN10 / PUT IN DISPLAY
006 /
007 /***DO ASSEMBLY/STATUS ROW 1
008 /
009 0329 118803 ROWST1, LXI D: DMAST3 / [D, E] ← STRING ADDRESS
010 032C CD0301 CALL MOVSTR / FORMAT CONTACT AREA
011 032F 1A0C MVI D: ERRFLD-1 / D ← COUNTER
012 0331 CD1903 CALL ROWN10 / FORMAT ERROR MESSAGE A' EA
013 0334 3680 MVI M: DMAFAN / STORE AN ATTRIBUTE
014 0336 23 INX H / BUMP POINTER
015 0337 116E03 LXI D: DMAST1 / D ← STRING ADDRESS
016 033A CD0301 CALL MOVSTR / LOAD TEXT
017 033D C35B03 JMP ROWST3 / FORMAT REF SECTION

```



```

018 /
019 /***ID0 ASSEMBLY/STATUS ROW2 /
020 /
021 0340 1193037 ROWST2, LX1 D, DMAS14 / [D,E] ← STRING ADDRESS
022 0343 CD0301 CALL MOVSTR / FORMAT NUMERIC FIELD
023 0346 1A01 MVI D, 1 / D ← COUNT
024 0348 CD1903 CALL ROWN10 / FORMAT SHIFT FIELD
025 034B 1A0B MVI D, ADVFLD / D ← COUNTER
026 034D CD1903 CALL ROWN10 / FORMAT ADVISORY AREA
027 0350 1A0A MVI D, @10 / D ← COUNTER
028 0352 CD1903 CALL ROWN10 / FORMAT STEP + USED VALUES
029 0355 119003 LXI D, DMAS12 / D ← STRING ADDRESS
030 0358 CD0301 CALL MOVSTR / LOAD TEXT
031 /
032 035B 0E0A ROWST3, MVI C, 6 / C ← COUNTER
033 /
034 035D 1A05 ROWST4, MVI D, 5 / D ← COUNTER
035 035F 3A80 MVI M, DMAFAN / STORE FIELD ATTRIBUTE
036 0361 23 INX H / BUMP POINTER
037 /
038 0362 3A20 ROWST5, MVI M, DMABLK / STORE A BLANK
039 0364 23 INX H / BUMP POINTER
040 0365 15 DCR D / DECREMENT COUNTER
041 0366 C26203 JNZ ROWST5 / LOOP UNTIL DONE
042 0369 0D DCR C / DECREMENT COUNTER
043 036A C25D03 JNZ ROWST4 / LOOP UNTIL DONE
044 036D 09 HLT / EXIT
045 EJECT

```

```

001 /
002 /***DATA STRINGS /
003 /
004 036E 11 DMAS11, DB DMAS1X
005 036F 20535445 DA < STEP# USED REF > /
006 0373 50232055
007 0377 53454420
008 037B 5245463E
009 037F 20
010 0011 DMAS1X = -DMAS11-1 / LENGTH OF STRING
011 /
012 0380 07 DMAS12, DB DMAS1X
013 0381 20205641 DA < VAL > /
014 0385 403E20
015 0007 DMAS12 = -DMAS12-1 / LENGTH OF STRING
016 /
017 0388 0A DMAS13, DB DMAS3X
018 0389 80201F DB DMAFAN; ASCBLK; ASCCBK
019 038C 1F1F1F1F DB ASCCBK; ASCCBK; ASCCBK; ASCCBK; ASCVBK
020 0390 1E
021 0391 2020 DB ASCBLK; ASCBLK
022 000A DMAS13 = -DMAS13-1 / LENGTH OF FIELD
023 /
024 0393 0A DMAS14, DB DMAS4X
025 0394 80201D DB DMAFAN; DMABLK; ASCNBK
026 0397 1D1D1D1D DB ASCNBK; ASCNBK; ASCNBK; ASCNBK; ASCVBK
027 039B 1E
028 039C 2020 DB ASCBLK; ASCBLK
029 000A DMAS14 = -DMAS14-1
030 EJECT

```

```

001 SUBJOB CLOCK FUNCTIONS
002 /
003 /***CLOCK FUNCTIONS /
004 /
005 /***ROUTINES. /
006 /
007 / CLKINI : CLOCK INITIALIZER
008 / CLKINT : CLOCK INTERRUPT HANDLER
009 /
010 EJECT

```

```

001 /
002 /***SUBROUTINE CLKINI /
003 /
004 /***THIS SUBROUTINE INITIALIZES ALL TIMER VALUES /
005 /
006 /***CALLING SEQUENCE: /
007 /

```

```

008 /          CALL    CLKINI
009 /
010 /***PARAMETERS:
011 /
012 /          NONE
013 /
014 /***REGISTER USAGE.
015 /
016 /          A      : SCRATCH
017 /          B      : COUNTER
018 /          C      : NOT USED
019 /          [D, E] : NOT USED
020 /          [H, L] : SCRATCH
021 /
022 /          EJECT

001 039E AF          CLKINI, CLA          / CLEAR A
002 039F 0606        MVI          B, TMRQNT / B ← CLOCK TABLE LENGTH
003 03A1 218FFD      LXI          H, TMRTAB / [H, L] ← CLOCK TABLE
004 /
005 03A4 77          CLKI10, MOV        M, A / RESET TIMER
006 03A5 23          INX          H / BUMP ADDRESS
007 03A6 05          DCR          B / DECREMENT COUNT
008 03A7 C2A403      JNZ          CLKI10 / BRANCH IF COUNT NOT ZERO
009 03AA C9          RET / EXIT
010 /          EJECT

001 /
002 /***SUBROUTINE CLKINT
003 /
004 /***CLOCK INTERRUPT ROUTINE
005 /
006 /***CALLING SEQUENCE.
007 /
008 /          CALL    CLKINT
009 /
010 /***PARAMETERS:
011 /
012 /          NONE
013 /
014 /***REGISTER USAGE.
015 /
016 /          A      : SCRATCH
017 /          [B, C] : SCRATCH
018 /          [D, E] : SCRATCH
019 /          [H, L] : SCRATCH
020 /
021 /          EJECT

001 03AB 218FFD      CLKINT, LXI          H, TMRTAB / [H, L] ← TIMERS ADDRESS
002 03AE AF          CLA / CLEAR A
003 /
004 03AF F5          CLK010, PUSH        PSW / SAVE COUNTER
005 03B0 F3          DI / PREVENT DURING TMR UPDATE
006 03B1 7E          MOV          A, M / A ← TIMER
007 03B2 B7          TST / TIMER.NE. 0 => RUNNING
008 03B3 CAC803      JZ          CLK030 / TIMER.EQ. 0 => NOT RUNNING
009 03B6 35          DCR          M / DECREMENT TIMER
010 03B7 C2C803      JNZ          CLK030 / BRANCH IF NOT EXPIRED
011 03BA FB          EI / NOW ALLOW HIGHER INTS.
012 03BB F1          POP          PSW / GET COUNTER
013 03BC F5          PUSH        PSW / SAVE IT AGAIN
014 03BD E5          PUSH        H / SAVE POINTER
015 03BE 87          ADD          A / COUNTER ← COUNTER*2
016 03BF 21D203      LXI          H, TMRDSP / [H, L] ← DISPATCH TABLE
017 03C2 5F          MOV          E, A / E ← OFFSET
018 03C3 1600        MVI          D, 0 / D ← 0
019 03C5 19          DAD          D / [H, L] ← ROUTINE ADDRESS
020 03C6 DF          DSPTAB / EXECUTE ROUTINE
021 /
022 03C7 E1          CLK020, POP          H / RESTORE [H, L]
023 /
024 /          CLK030,
025 03C8 FB          EI / ALLOW HIGHER RUPTS
026 03C9 23          INX          H / BUMP ADDRESS
027 03CA F1          POP          PSW / GET COUNTER

```

```

028 03CB 3C          INR      A          / BUMP COUNTER
029 03CC FE06      CPI      TMRCNT    / A. EQ. TMRCNT => DONE
030 03CE C2AF03    JNZ     CLK010    / A. NE. TMRCNT => CONTINU
031 03D1 C9        REI          / EXIT
032                EJECT

001                /
002                /***TIMER DISPATCH TABLE
003                /
004                / NOTE: IF TIMERS ARE ADDED/SUBTRACTED FROM SYSTEM,
005                / THIS TABLE MUST BE IN SAME ORDER AND SIZE
006                / AS THE TIMERS ARE. SEE:
007                / "SYSTEM TIMERS ALLOCATION"
008                /
009                /
010 03D2 DE03      TMRDSP, DW      CLK100    / BEEP TIMER
011 03D4 EB03      DW      CLK200    / ACK TIMER
012 03D6 EE03      DW      CLK300    / LED TIMER
013 03D8 F403      DW      CLK400    / POWER TIMER
014 03DA FD03      DW      CLK500    / ERROR TIMER
015 03DC 1104      DW      CLK600    / DISCRETE REFRESH
016                EJECT

001                /
002                /***BEEP TIMER
003                /
004 03DE 3AB4FD    CLK100, LDA     POSAVE    / GET CURRENT STATE OF PORT
005 03E1 E6BF      ANI     -1-POBEEP    / MASK OUT BEEPER
006 03E3 D33E      OUT     PAROUT      / OUTPUT DATA
007 03E5 32B4FD    STA     POSAVE      / STORE NEW STATE OF PORT
008 03E8 C3C703    JMP     CLK020      / GO BACK TO LOOP
009                /
010                /***ACK TIMER
011                /
012 03EB C3C703    CLK200, JMP     CLK020    / NO ACTION HERE
013                /
014                /***LED TIMER
015                /
016 03EE 011704    CLK300, LXI     B,SPLLED    / [B,C] <- COMMAND BLOCK
017 03F1 C3F703    JMP     CLK410      / GO TO COMMON CODE
018                /
019                /***POWER TIMER
020                /
021 03F4 011A04    CLK400, LXI     B,SPLPWR    / [B,C] <- COMMAND BLOCK
022                /
023 03F7 CD0D02    CLK410, CALL    SPOOLI     / SPOOL COMMAND
024 03FA C3C703    JMP     CLK020      / GO BACK TO LOOP
025                /
026                /***ERROR TIMER
027                /
028 03FD 21BBFC    CLK500, LXI     H, DSPERR    / [H,L] <- FIELD ATTRIBUTE
029 0400 3E80      MVI     A,FACNOR    / A <- NORMAL ATTRIBUTE
030 0402 BE        CMP     M          / CHECK FIELD ATTRIBUTE
031 0403 C20804    JNZ     CLK510      / BRANCH ON REVERSE VIDEO
032 0406 3E90      MVI     A,FACREV    / SET REVERSE VIDEO
033                /
034 0408 77        CLK510, MOV     M,A      / SET NORMAL VIDED
035                /
036 0409 3E1E      CLK520, MVI     A,ERRTMR    / A <- TIMER VALUE
037 040B 3293FD    STA     TMRERR      / STORE IT
038 040E C3C703    JMP     CLK020      / CONTINUE
039                /
040                /***DISCRETE REFRESH TIMER
041                /
042 0411 011D04    CLK600, LXI     B,SPLDIS    / [B,C] <- COMMAND BLOCK
043 0414 C3F703    JMP     CLK410      / SPOOL COMMAND
044                EJECT

001                /
002                /***LED COMMAND SPOOL BLOCK
003                /
004 0417 02        SPLLED, DB      SPLLEX      / NUMBER OF BYTES
005 0418 F11E      DW      KF21          / COMMAND ADDRESS
006                SPLLEX = -SPLLED-1 / COMMAND LENGTH
007                /
008                /***POWER COMMAND SPOOL BLOCK
009                /

```


265

266

```

010 041A 02      SPLPWR, DB      SPLPWX      / NUMBER OF BYTES
011 041B FA1D    DW      KF20      / COMMAND ADDRESS
012          0002  SPLPWX=, -SPLPWR-1 / COMMAND LENGTH
013          /
014          /***DISCRETE REFRESH SPOOL BLOCK
015          /
016 041D 02      SPLDIS, DB      SPLDIX      / NUMBER OF BYTES
017 041E B41B    DW      KF18      / COMMAND ADDRESS
018          0002  SPLDIX=, -SPLDIS-1 / COMMAND LENGTH
019          EJECT

                                SUBJOB PERIPHERAL PORT HANDLERS

001          /
002          /
003          /***PERIPHERAL PORT ROUTINES
004          /
005          /***ROUTINES:
006          /
007          /      PPINIT - INITIALIZATION
008          /      PPINT  - INTERRUPT HANDLER
009          /
010          /      EJECT

001          /
002          /***SUBROUTINE PPINIT
003          /
004          /***INITIALIZE PERIPHERAL PORT
005          /
006          /***CALLING SEQUENCE:
007          /
008          /      CALL    PPINIT
009          /
010          /***PARAMETERS:
011          /
012          /      NONE
013          /
014          /***REGISTER USAGE:
015          /
016          /      A      - SCRATCH
017          /      [B,C] - SCRATCH
018          /      [D,E] - SCRATCH
019          /      [H,L] - SCRATCH
020          /
021          /      EJECT

001 0420 01A0FF  PPINIT, LXI    B; PPIBUF      / [B,C] ← RECEIVER BUFB S
002 0423 112000  LXI    D; PPIBFL      / [D,E] ← RECEIVER BFLEN
003 0426 219CFD  LXI    H; PPIBLK      / [H,L] ← RECEIVER BUFBK
004 0429 CD2401  CALL   BFINIT          / INITIALIZE RCVR BUFFER
005          /
006 042C 0180FF  LXI    B; PPOBUF      / [B,C] ← TRANSMIT BUFBAS
007 042F 112000  LXI    D; PPOBFL      / [D,E] ← TRANSMIT BFLE
008 0432 21A2FD  LXI    H; PPOBLK      / [H,L] ← TRANSMIT BUFBK
009 0435 CD2401  CALL   BFINIT          / INITIALIZE XMIT BUFFER
010          /
011 0438 AF      CLA          / A ← 0
012 0439 32AEFD  STA    PPISTA         / CLEAR RECEIVER FLAGS
013 043C 32AFFD  STA    PPOSTA        / CLEAR TRANSMIT FLAGS
014          /
015 043F 3E81    MVI    A; PPNUL      / A ← NULL CHARACTER
016 0441 D33A    OUT   SP1CTL        / LOAD NULL INSTRUCTION
017 0443 00      NOP          / PRECAUTIONARY WAIT
018 0444 D33A    OUT   SP1CTL        / LOAD SECOND NULL
019          /
020 0446 3E40    MVI    A; SPCIR      / A ← RESET COMMAND
021 0448 D33A    OUT   SP1CTL        / RESET INTERFACE
022 044A 3EFE    MVI    A; PPMODE     / A ← INTERFACE MODE
023 044C D33A    OUT   SP1CTL        / SET INTERFACE MODE
024 044E 3E25    MVI    A; PPCMD      / A ← INTERFACE STATE
025 0450 D33A    OUT   SP1CTL        / LOAD STATE
026          /
027 0452 C9      RET          / EXIT
028          EJECT

001          /
002          /***SUBROUTINE PPINT
003          /
004          /***PERIPHERAL PORT INTERRUPT HANDLER

```

```

005 /
006 /****CALLING SEQUENCE:
007 /
008 /      CALL    PPINT
009 /
010 /****PARAMETERS:
011 /
012 /      NONE
013 /
014 /****REGISTER USAGE:
015 /
016 /      A      SCRATCH
017 /      B = STATUS OF PORT
018 /      C = SCRATCH
019 /      [D,E] : SCRATCH
020 /      [H,L] : SCRATCH
021 /
022 /      EJECT

001 PPINT,
002 0453 78      MOV     A;B      / GET PORT STATUS
003 0454 F5      PUSH    PSW        / SAVE IT
004 0455 E602    ANI     SPSRRY     / CHECK FOR RECEIVER READY
005 0457 CA0E05  JZ      PPI100      / BRANCH IF NOT READY
006 /
007 /****RECEIVER INTERRUPT
008 /
009 045A 78      MOV     A;B      / A ← STATUS
010 045B E628    ANI     SPSFE+SPSPE  / PARITY/FRAMING ERROR
011 /
012 045D CA6504  JZ      PPI010      / CHECK
013 /
014 /****PARITY/FRAMING ERROR
015 /
016 0460 3E10    MVI     A;PPIPAR    / A ← STATUS
017 0462 C3FB04  JMP     PPI090      / GO TO ERROR HANDLER
018 /
019 0465 78      PPI010, MOV    A;B      / A ← STATUS
020 0466 E610    ANI     SPSOE       / CHECK FOR OVERRUN
021 0468 CA7004  JZ      PPI030      / BRANCH IF NO ERROR
022 /
023 /****OVERRUN ERROR
024 /
025 046B 3E08    MVI     A;PPIOVR    / A ← STATUS
026 046D C3FB04  JMP     PPI090      / GO TO ERROR HANDLER
027 /

001 0470 DB3B   PPI030, IN     SP1IN   / READ DATA
002 0472 47     MOV     B;A      / SAVE CHARACTER
003 0473 3AAEFD LDA     PPISTA   / A ← RECEIVER STATUS
004 0476 4F     MOV     C;A      / SAVE STATUS
005 0477 E680    ANI     PPIMSG   / CHECK FOR MSG IN PROGRESS
006 0479 C2D004  JNZ    PPI050   / BRANCH IF SET
007 047C 79     MOV     A;C      / GET STATUS AGAIN
008 047D E640    ANI     PPIFCN   / CHECK FOR FUNCTION FLAG
009 047F C29604  JNZ    PPI040   / BRANCH IF SET
010 0482 79     MOV     A;C      / A ← STATUS
011 0483 E601    ANI     PPICNT   / CHECK FOR COUNT FLAG
012 0485 C2C204  JNZ    PPI045   / BRANCH ON COUNT FLAG
013 0488 3E02    MVI     A;ASCSTX / A ← STX
014 048A B8     CMP     B        / CHARACTER AN STX?
015 048B C20E05  JNZ    PPI100   / NO, IGNORE IT
016 048E 3E40    MVI     A,PPIFCN / A ← NEW RECEIVER STATUS
017 0490 32AEFD STA     PPISTA   / LOAD IT
018 0493 C30E05  JMP     PPI100   / CONTINUE
019 /
020 /
021 /****FUNCTION CHARACTER
022 /
023 0496 3E01    PPI040, MVI     A;.FF+ASCSTX / CREATE MESSAGE CHKSUM
024 0498 32B1FD  STA     PPICLK   / INITIALIZE CHECKSUM
025 049B 3E01    MVI     A;PPICNT / A ← NEW RECEIVER STAT S
026 049D 32AEFD STA     PPISTA   / LOAD IT
027 04A0 78     MOV     A;B      / GET CHAR
028 04A1 FED0    CFI     ASCNAK   / WAS IT NAK?
029 04A3 C2DB04  JNZ    PPI060   / NO, GO ON...

```

```

030
031 / RECEIVED A "NAK"; SHUT DOWN TRANSMIT
032
033 04A6 F1 POP PSW / CLEAR XMIT STATUS
034 04A7 AF CLA / X
035 04A8 F5 PUSH PSW / X
036
037 04A9 C5 PUSH B / SAVE CHAR
038
039 04AA 0180FF LXI B;PPOBUF / GET XMIT BUFBAS
040 04AD 112000 LXI D;PPOBFL / GET XMIT BUFLN
041 04B0 21A2FD LXI H;PPOBLK / GET XMIT BUFBK
042 04B3 CD2401 CALL BFINIT / RESET BUFFER
043
044 04B6 AF CLA / O XMIT FLAG
045 04B7 32AFFD STA FPOSTA / X
046
047 04BA 3E25 MVI A;PPCMD / GET COMMAND BYTE
048 04BC D33A OUT SP1CTL / DISABLE XMIT INTERRUPT
049
050 04BE C1 POP B / RESTORE CHAR
051 04BF C3DB04 JMP PPI060 / AND CONTINUE
052
053 /
054 /***COUNT CHARACTER
055 /
056 04C2 78 PPI045, MOV A;B / A ← CHARACTER
057 04C3 D604 SUI :04 / ALLOW FOR STX AND FCN
058 04C5 32B0FD STA MSGLEN / LOAD COUNTER
059 04C8 3E90 MVI A;PPIMSG / A ← NEW STATUS
060 04CA 32AEFD STA PPISTA / LOAD STATUS
061 04CD C3DB04 JMP PPI060 / AND CONTINUE
062
063 /***DATA CHARACTER
064 /
065 04D0 3AB0FD PPI050, LDA MSGLEN / MSGLEN.EQ.0 =>
066 04D3 B7 TST / THIS CHAR IS CHECKSUM
067 04D4 CAEA04 JZ PPI070 / BRACH FOR CHECKSUM
068 04D7 3D DCR A / DECREMENT CHARACTER COUNT
069 04D8 32B0FD STA MSGLEN / STORE IT
070
071 /***BUFFER DATA CHARACTER
072 /
073 04DB 78 PPI060, MOV A;B / A ← DATA CHARACTER
074 04DC 019CFD LXI B;PPIBLK / [B,C] ← BUFFER BLOCK
075 04DF CD2E01 CALL BFCH / BUFFER CHARACTER
076 04E2 21B1FD LXI H;PPICLK / [H,L] ← CHECKSUM ADDR BS
077 04E5 86 ADD M / A ← NEW RUNNING CHECKSUM
078 04E6 77 MOV M;A / STORE IT
079 04E7 C30E05 JMP PPI100 / CONTINUE
080
081 /***CHECKSUM CHARACTER
082 /
083 04EA 3AB1FD PPI070, LDA PPICLK / A ← COMPUTED CHECKSUM
084 04ED B8 CMP B / A.EQ.B => CHECKSUM OKAY
085 04EE C2F904 JNZ PPI080 / A.NE.B => CHECKSUM BAD
086 04F1 3E20 MVI A;PPIDON / INDICATE MESSAGE DONE
087 04F3 32AEFD STA PPISTA / SET RECEIVER STATUS
088 04F6 C30E05 JMP PPI100 / CONTINUE
089
090 /
091 /***CHECKSUM ERROR
092 /
093 04F9 3E02 PPI080, MVI A;PPICER / A ← STATUS
094
095 /***COMMON ERROR CODE
096 /
097 04FB F604 PPI090, ORI PPIRET / SET RETRAN FLAG
098 04FD 32AEFD STA PPISTA / SET NEW STATUS
099 0500 DB3B IN SP1IN / CLEAR RECEIVER
100 0502 01A0FF LXI B;PPIBUF / GET PARAMETERS
101 0505 112000 LXI D;PPIBFL / TO INITIALIZE
102 0508 219CFD LXI H;PPIBLK / RECEIVER BUFFER
103 050B CD2401 CALL BFINIT / INITIALIZE BUFFER
104
105 EJECT / CONTINUE PROCESSING

```



```

001 /
002 /***CHECK FOR TRANSMITTER INTERRUPT
003 /
004 050E F1 PPI100, POP PSW / A ← INTERFACE STATUS
005 050F E601 ANI SPSTRY / CHECK FOR TRANSMIT READY
006 0511 CA2A05 JZ PPIX / BRANCH IF NOT READY
007 0514 3AA7FD LDA PPOBLK+BFUSE / A ← BUFFER COUNT
008 0517 B7 TST / CHECK FOR BUFFER EMPTY
009 0518 CA2605 JZ PPI110 / BRANCH IF BUFFER EMPTY
010 051B 01A2FD LXI B,PPOBLK / [B,C] ← BUFBLK ADDRESS
011 051E CD5601 CALL UBFCB / GET CHARACTER FROM BUFFER
012 0521 D33E OUT SP1OUT / WRITE OUT CHARACTER
013 0523 C32A05 JMP PPIX / GO TO EXIT
014 /
015 0526 3E25 PPI110, MVI A,PPCMD / A ← COMMAND BYTE
016 0528 D33A OUT SP1CTL / DISABLE INTERRUPT
017 /
018 052A C9 PPIX, RET / EXIT
019 EJECT

SUBJOB CURSOR ILLUMINATION

001 /
002 /
003 /***SUBROUTINE CURSOR
004 /
005 /***MOVE CURSOR FROM OLD LOCATION TO NEW LOCATION
006 /
007 /***CALLING SEQUENCE:
008 /
009 / CALL CURSOR
010 /
011 /***PARAMETERS..
012 /
013 / B : XXXXYYYY
014 / C : WWWWZZZZ
015 /
016 / WHERE : XXXX = OLD CURSOR LINE
017 / YYY Y = OLD CURSOR COLUMN
018 / WWW W = NEW CURSOR LINE
019 / ZZZ Z = NEW CURSOR COLUMN
020 /
021 /***REGISTER USAGE:
022 /
023 / A : SCRATCH
024 / [B,C] : PARAMETERS (PRESERVED)
025 / [D,E] : SCRATCH
026 / [H,L] : SCRATCH
027 /
028 052B AF CURSOR, CLA / A ← 0
029 052C B8 CMP B / B.EQ.0 => SET CURSOR
030 052D CA3805 JZ CUR010 / BRANCH TO SET CURSOR
031 /
032 0530 CD4705 CALL CUR100 / GET LOCATION
033 0533 3680 MVI M,DMAFAN / CLEAR REVERSE VIDEO
034 0535 19 DAD D / GET SECOND LINE
035 0536 3680 MVI M,DMAFAN / CLEAR REVERSE VIDEO
036 /
037 0538 78 CUR010, MOV A,B / SWAP
038 0539 41 MOV B,C / B AND C
039 053A 4F MOV C,A / C FOR NEW LOCATION
040 053B CD4705 CALL CUR100 / GET LOCATION
041 053E 3690 MVI M,FACREV / TURN ON REVERSE VIDEO
042 0540 19 DAD D / GET SECOND LINE
043 0541 3690 MVI M,FACREV / TURN ON REVERSE VIDEO
044 0543 78 MOV A,B / PUT B
045 0544 41 MOV B,C / AND C
046 0545 4F MOV C,A / BACK
047 0546 C9 RET / EXIT
048 EJECT

001 /
002 /***COMPUTE START OF NODE ADDRESS
003 /
004 0547 78 CUR100, MOV A,B / A ← [LINE,COL]
005 0548 E6F0 ANI ROWMSK / ISOLATE LINE NUMBER
006 054A FE80 CPI ASMR0W / CHECK FOR STATUS LINE

```

```

007 054C CA6805      JZ      CUR120      / BRANCH TO HANDLER
008 054F 210BF8      LXI     H; DSPLOG+DSPPOW / [H,L] <- START OF LOGIC
009 0552 11A000      LXI     D, ROWB+ROWB  / [H,L] <- LOGIC ROW LENGTH
010 0555 CF          NSWP          / SHIFT
011                  / A TO FORM
012                  / COUNTER
013                  / FOR LOOP
014 0556 CD7805      CALL    CUR200      / GET LINE ADDRESS
015 0559 115000      LXI     D; ROWB      / [D,E] <- ROW LENGTH
016 055C D5          PUSH    D           / SAVE ROW LENGTH
017 055D 110700      LXI     D; DSPNUD    / [D,E] <- NODE LENGTH
018 0560 78          MOV     A; B        / A <- CURSOR LOCATION
019                  /
020 0561 E60F        CUR110, ANI     COLMSK / ISOLATE COLUMN
021 0563 CD7805      CALL    CUR200      / GET NODE ADDRESS
022 0566 D1          POP     D           / RESTORE ROW LENGTH
023 0567 C9          RET             / EXIT
024                  /
025 0568 21DAFC      CUR120, LXI     H; DSPREF / [H,L] <- REF AREA
026 056B 114E00      LXI     D; ROWD+1    / [D,E] <- ROW LENGTH
027 056E D5          PUSH    D           / SAVE ROW LENGTH
028 056F 110600      LXI     D; DSPNOD-1  / SHORTER NODE SIZE
029 0572 78          MOV     A; B        / A <- CURSOR POSITION
030 0573 D605        SUI     5           / OFFSET IT
031 0575 C36105      JMP     CUR110      / DO AN EXIT
032                  /
033 0578 3D          CUR200, DCR     A     / LOOP TO BUMP ADDRESS
034 0579 C8          RZ             / EXIT WHEN COUNT ZERO
035 057A 19          DAD     D           / BUMP ADDRESS
036 057B C37805      JMP     CUR200      / CONTINUE
037                  /

```

SUBJOB ERROR HANDLER

```

001                  /
002                  /
003                  / ***ERROR HANDLER
004                  /
005                  / ***PARAMETERS:
006                  /
007                  / [D,E] - MESSAGE ADDRESS
008                  /
009                  / ***REGISTER USAGE:
010                  /
011                  / A - SCRATCH
012                  / [B,C] - SCRATCH
013                  / [D,E] - MESSAGE ADDRESS (DESTROYED)
014                  / [H,L] - SCRATCH
015                  /
016 057E D5          ERROR, PUSH    D     / SAVE ADDRESS
017 057F 21BBFC      LXI     H; DSPERR    / [H,L] <- DESTINATION
018 0582 160C        MVI     D; ERRFLD-1 / D <- FIELD LENGTH
019 0584 CD1903      CALL    ROWN10     / CLEAR ERROR FIELD
020 0587 D1          POP     D           / RESTORE ADDRESS
021 0588 21BCFC      LXI     H; DSPERR+1 / [H,L] <- FIELD ADDRESS
022 058B CD0301      CALL    MOVSTR     / LOAD MESSAGE TO FIELD
023 058E 3E1E        MVI     A; ERRTMR   / A <- ERROR TIMER PRESET
024 0590 3293FD      STA    TMRERR     / ENABLE ERROR TIMER
025 0593 3A7CFE      LDA    KSTATE     / A <- STATE VECTOR
026 0596 F620        ORI    KRESET     / SET RESET FLAG
027 0598 327CFE      STA    KSTATE     / LOAD STATE VECTOR
028 059B CD531F      CALL    KU03      / CLEAR SHIFT FIELD
029 059E C9          RET             / EXIT
030                  /

```

SUBJOB SOURCE FILE DEMARKATION

```

001                  /
002                  /
003                  / ***P180XX.P1 : END-OF-FILE
004                  /
005                  / ***THIS MARKS THE END OF SOURCE FILE P180XX.P1.
006                  /
007                  /

```

```

001
002
003
004
005
006

```

```

008
009
010
\*** P18023.P2 : START-OF-FILE
012
013
014
\*****

```

```

016          EJECT

001          SUBJOB  KEYBOARD FUNCTIONS
002          /
003          /***KEYBOARD HANDLER
004          /
005          /***ROUTINES:
006          /
007          /          KBDINI - INITIALIZE OPERATOR INTERFACE
008          /          KBDINT - INTERRUPT HANDLER
009          /          KBDFUN - FUNCTION HANDLER
010          /          KF01  - DISABLE
011          /          KF02  - CURSOR CONTROL
012          /          KF03  - CONTACTS
013          /          KF04  - VERTICALS
014          /          KF05  - NUMERICS
015          /          KF06  - SHIFT
016          /          KF07  - FORCE
017          /          KF08  - GET
018          /          KF09  - GET NETWORK
019          /          KF10  - SEARCH
020          /          KF11  - CLEAR
021          /          KF12  - DELETE
022          /          KF13  - START NEXT
023          /          KF14  - ENTER
024          /          KF15  - SPARE
025          /          KF16  - UNDEFINED
026          /          KF17  - CLEAR RESET
027          /          KF18  - DISCRETE UPDATE
028          /          KF19  - SUPERVISORY
029          /          KF20  - POWER DISPLAY
030          /          KF21  - LED DISPLAY
031          /
032          EJECT

```

```

001          /
002          /***SUBROUTINE KBDINI
003          /
004          /***INITIALIZE OPERATOR INTERFACE
005          /
006          /***CALLING SEQUENCE:
007          /
008          /          CALL    KBDINI
009          /
010          /***PARAMETERS.
011          /
012          /          NONE
013          /
014          /***REGISTER USAGE:
015          /
016          /          A      : SCRATCH
017          /          [B,C] : SCRATCH
018          /          [D,E] : SCRATCH
019          /          [H,L] : SCRATCH
020          /
021          EJECT

```

```

001 059F 0170FF      KBDINI, LXI      B,KBDBUF      / [B,C] ← KEYBOARD BFBASE
002 05A2 111000      LXI      D,KBDBFL      / [D,E] ← KEYBOARD BFLEN
003 05A5 21A8FD      LXI      H,KBDBLK      / [H,L] ← KEYBOARD BFBL
004 05A8 CD2401      CALL     BFINIT      / INITIALIZE BUFFER
005 05AB 3EFF         MVI     A,FF         / SET INITIAL KEYSTROKE
006 05AD 3283FE      STA     LASTKY      / FOR ROLLOVER
007 05B0 3282FE      STA     NEWKEY      / FOR DEBOUNCING
008
009 05B3 CD8A1F      CALL     KU06        / RESET LOGIC
010

```



```

011 05B6 111427      LXI      D;MSGHI      / [D,E] ← MESSAGE ADDRESS
012 05B9 CD681F      CALL     KU04         / TO ADVISORY FIELD
013
014 05BC 3E14        MVI      A;KCLEAR+KCLADV / SET FLAGS FOR
015 05BE 3270FE      STA      KSTATE      / INITIAL STATE
016
017 05C1 3E30        MVI      A;ASCO       / A ← ASCII 0
018 05C3 0604        MVI      B;4          / B ← COUNTER
019 05C5 2118FD      LXI      H;DSPSTP     / [H,L] ← STEP DATA ADDR
020
021 05C8 77          KBDI10, MOV     M;A      / MOVE IN A ZERO
022 05C9 23          INX      H            / BUMP ADDRESS
023 05CA 05          DCR      B            / DECREMENT
024 05CB C2C805      JNZ      KBDI10      / BRANCH IF NOT DONE
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058

```

/
 /***COMPUTE MEMORY SIZE-AND USAGE
 /
 KBDI15, LXI H;CMDBUF+3 / [H,L] ← COMMAND BUFFER
 LXI D;ADRCON / [D,E] ← ADDRESS
 MOVDE / STORE DATA
 /
 LXI D;CMDRED+CMD02!:100+LENRED / SET PARMS
 CALL PIO / READ CONTROLLER DATA
 JNZ KBDI15 / BRANCH ON ERROR
 CALL CLRERR / CLEAR ANY ERROR MESSAGES
 LXI B;RSPBUF+3 / [B,C] ← SOURCE ADDRESS
 LXI H;SCONF1 / [H,L] ← DESTINATION ADDR
 LDAX B / A ← CONFIGURATION BYTE 1
 MOV M;A / STORE IT
 MOV D;A / D ← DATA
 INX B / BUMP ADDRESS
 INX H / BUMP ADDRESS
 LDAX B / A ← CONFIGURATION BYTE 2
 MOV M;A / STORE IT
 LXI B;:0100 / [B,C] ← 256 FOR COUNTING
 LXI H;0 / CLEAR [H,L]
 MOV A;D / A ← LOGIC RAM CONFIG
 RRC / SHIFT A
 RRC / TO FORM
 RRC / COUNTER
 ANI :1F / FOR MEMORY SIZE
 /
 KBDI20, DAD B / COUNT 256 BYTES
 DCR A / DONE?
 JNZ KBDI20 / BRANCH IF NOT DONE
 XCHG / SWAP REGISTERS
 LXI H;MEMSIZ / [H,L] ← POINTER
 MOVDE / STORE DATA
 /
 LXI H;CMDBUF+3 / [H,L] ← COMMAND BUFFER
 LXI D;ADRUSE / [D,E] ← START-OF-USER LO
 MOVDE / STORE DATA IN BUFFER
 LXI D;NOEOL!:400 / [D,E] ← DATA
 MOVDE / STORE DATA IN BUFFER
 LXI D;0 / [D,E] ← MASK
 MOVDE / STORE INTO BUFFER
 /
 KBDI25, LXI D;CMDSCH!:100+LENSCH / SET PARAMETERS
 CALL PIO / DO SEARCH
 JNZ KBDI25 / HARD FAILURE
 EJECT
 /
 / SEE IF WE GOT A GOOD ANSWER; SET UP
 / USAGE IF SO
 /
 LXI H;RSPBUF+1 / [H,L] ← POINTER
 GETHL / GET ADDR OF SEARCH RETURN
 /
 LXI D;-1 / SET FOR FAILURE TEST
 DCMPL / ADDR = FAILURE?
 JZ KBDI50 / YES, FAILED TO FIND TDL

```

059 0627 EB          XCHG          / SWAP
060 0628 13          INX           D          / ADD 2 FOR EOL NODE
061 0629 13          INX           D          / ITSELF!
062 062A 2188FE      LXI           H, MEMUSE  / [H, L] ← DESTINATION
063 062D EF          MOVDE          / STORE USAGE COUNT
064 062E EB          XCHG          / SWAP
065 062F 111DFD      LXI           D, DSPUSE  / [D, E] ← BCD DESTINATION
066 0632 CDC201      CALL          BNBCD4    / CONVERT TO BCD
067

```

```

001 /
002 /***INITIALIZE LOGIC DATA
003 /
004 0635 010000      LXI           B, 0      / [B, C] ← 0
005 0638 218AFE      LXI           H, STPNUM / [H, L] ← DESTINATION
006 063B D7          MOVBC         / CLEAR STEP NUMBER
007 063C 218CFE      LXI           H, ADRSON / [H, L] ← DESTINATION
008 063F D7          MOVBC         / CLEAR STARTING ADDR
009 0640 218EFE      LXI           H, ADREON / [H, L] ← DESTINATION
010 0643 D7          MOVBC         / CLEAR ENDING ADDR
011 /
012 0644 3E01        MVI           A, DISTMR / A ← DISCRETE TIMER
013 0646 3294FD      STA          TMRDIS    / SET TIMER
014 0649 3E06        MVI           A, ASMOOL / A ← STARTING COLUMN
015 064B 3281FE      STA          DISPTR    / INITIALIZE POINTER
016 /
017 064E CD4524      CALL          KU21     / START LED + PWR TIMERS
018 0651 C35A06      JMP          KBDIX    / YES, GO TO EXIT
019 /
020 0654 115B06      KBDISO, LXI   D, KBDIMS / [D, E] ← MESSAGE ADDR
021 0657 CD7E05      CALL          ERROR   / DISPLAY MESSAGE
022 /
023 065A C9          KBDIX, RET          / EXIT
024 /
025 /***MESSAGE
026 /
027 065B 08          KBDIMS, DB      KBDIMX
028 065C 494E4954    DA          'INIT MEM'
029 0660 204D454D    KBDIMX= .-KBDIMS-1 / MESSAGE LENGTH
030 EJECT

001 /
002 /***SUBROUTINE KBDINT
003 /
004 /***KEYBOARD INTERRUPT (POLLING) ROUTINE
005 /
006 /***CALLING SEQUENCE:
007 /
008 /          CALL    KBDINT
009 /
010 /***PARAMETERS:
011 /
012 /          NONE
013 /
014 /***REGISTER USAGE:
015 /
016 /          A      - SCRATCH
017 /          [B, C] - SCRATCH
018 /          [D, E] - SCRATCH
019 /          [H, L] - SCRATCH
020 /
021 EJECT

001 0664 3AB4FD      KBDINI, LDA      POSAVE / A ← CURRENT OUTPUT STATE
002 0667 E6C0        ANI          POPWR+POBEEP / SAVE LED AND BEEP STATE
003 0669 1608        MVI          D, 8      / D ← COUNTER
004 066B 5F          MOV          E, A      / E ← OUTPUTS
005 066C 210000      LXI          H, 0      / [H, L] ← OFFSET
006 /
007 066F 7B          KBD010, MOV      A, E    / A ← PORT CONTROL
008 0670 DB3E        OUT          PAROUT   / SELECT ROW
009 0672 DB3E        IN           PARIN    / READ DATA
010 0674 E7          TST         / A.EQ.0 => NO KEYS
011 0675 CABE06      JZ          KBD040    / A.NE.0 => KEY(S)

```

```

012 0678 0607          MVI      B:7          / B ← COUNTER
013 067A 0E01          MVI      C:01          / C ← MASK
014
015 067C F5           KBD020, PUSH     PSW          / SAVE DATA
016 067D A9           XRA      C              / LOOK FOR MATCH
017 067E C2B206       JNZ      KBD030         / WANT ONE KEY ONLY
018 0681 3A82FE       LDA      NEWKEY         / A ← LAST KEYSTROKE
019 0684 BD           CMP      L              / MATCH?
020 0685 CA8FD6       JZ       KBD021         / YES, MUST BE OKAY
021
022 0688 7D           MOV      A:L            / NO, A ← NEW KEYSTROKE
023 0689 3282FE       STA      NEWKEY         / INDICATE NEW KEYSTROKE
024 068C C3AE06       JMP      KBD025         / AND CONTINUE
025
026 068F 3A83FE       KBD021, LDA      LASTKY  / A ← LAST VALID KEY
027 0692 BD           CMP      L              / MATCH?
028 0693 CAAE06       JZ       KBD025         / YES, ALREADY BUFFERED
029
030 0696 7D           MOV      A:L            / A ← NEW KEYSTROKE
031 0697 3282FE       STA      NEWKEY         / INDICATE LATEST KEY
032 069A 3283FE       STA      LASTKY         / AND LAST VALID KEY
033 069D 01A8FD       LXI      B:KDBLK        / [B,C] ← BLOCK ADDRESS
034 06A0 CD2E01       CALL     BFCH           / BUFFER KEYSTROKE
035 06A3 3A7CFE       LDA      KSTATE         / A ← STATE VECTOR
036 06A6 E620         ANI      KRESET         / CHECK FOR ERROR
037 06A8 C2AE06       JNZ      KBD025         / NO BEEP ON ERROR STATE
038 06AB CD1401       CALL     BEEP10         / TURN ON BEEP
039
040 06AE F1           KBD025, POP     PSW          / CLEAR STACK
041 06AF C3CC06       JMP      KBDX           / EXIT
042
043 06B2 23           KBD030, INX      H          / BUMP INDEX
044 06B3 79           MOV      A:C            / SHIFT C
045 06B4 81           ADD      C              / LEFT ONE PLACE
046 06B5 4F           MOV      C:A            / BY AN ADD
047 06B6 F1           POP     PSW             / RESTORE ROW DATA
048 06B7 05           DCR      B              / DECREMENT COUNTER
049 06B8 C27C06       JNZ      KBD020         / CONTINUE LOOP
050 06BB C3CC06       JMP      KBDX           / IGNORE MULTIPLES
051
052 06BE 1C           KBD040, INR      E          /
053 06BF 010700       LXI      B:7            /
054 06C2 09           DAD      B              /
055 06C3 15           DCR      D              /
056 06C4 C26F06       JNZ      KBD010         /
057 06C7 3EFF         MVI      A:FF           / INDICATE NO KEYS
058 06C9 3283FE       STA      LASTKY         / STORE FLAG
059
060 06CC C9           KBDX,  RET             /
061  EJECT

001                      SUBJOB  KEYBOARD FUNCTION DISPATCHER
002
003  /***SUBROUTINE KBD CMD
004  /
005  /***KEYBOARD FUNCTION HANDLER
006  /
007  /***CALLING SEQUENCE:
008  /
009  /          CALL    KBD CMD
010  /
011  /***PARAMETERS:
012  /
013  /          NONLE
014  /
015  /***REGISTER USAGE:
016  /
017  /          A      : SCRATCH
018  /          [B,C] : SCRATCH
019  /          [D,E] : SCRATCH
020  /          [H,L] : SCRATCH
021  /
022  EJECT

001 06CD 01A8FD       KBD CMD, LXI      B:KDBLK  / [B,C] ← KBD BUFFER BLOCK
002 06D0 CD5601       CALL     UBFC           / GET KEYSTROKE
003 06D3 C2E506       JNZ      KBD CMX       / EXIT IF BUFFER EMPTY

```


004 06D6 21F206	LXI	H,KBDTAB	/ [H,L] ← DISPATCH TABLE
005 06D9 F5	PUSH	PSW	/ SAVE CHARACTER
006 06DA 0600	MVI	B,0	/ BC ← 0
007 06DC 87	ADD	A	/ WORD-ORIENTATED TABLE
008 06DD 4F	MOV	C,A	/ [B,C] ← OFFSET
009 06DE 09	DAD	B	/ [H,L] ← ADDRESS OF FC
010 06DF 7E	MOV	A,M	/ A ← ADDRLO
011 06E0 23	INX	H	/ BUMP ADDR
012 06E1 66	MOV	H,M	/ H ← ADDRHI
013 06E2 6F	MOV	L,A	/ L ← ADDRLO
014 06E3 F1	POP	PSW	/ RESTORE CHARACTER
015 06E4 E9	PCHL		/ DISPATCH
016			
017 06E5 0170FF	KBDCMA, LXI	B,KBDBUF	/ REINITIALIZE
018 06E8 111000	LXI	D,KBDBFL	/ KEYBOARD BUFFER
019 06EB 21A8FD	LXI	H,KBDELK	/ AFTER NO DATA
020 06EE CD2401	CALL	BFINIT	/ EXIT
021 06F1 C9	RET		
022	EJECT		

001	/		
002	/***KEYBOARD TABLE		
003	/		
004 06F2 2E15	KBDTAB, DW	KF14	/ ENTER
005 06F4 5908	DW	KF02	/ CURSOR UP
006 06F6 971B	DW	KF17	/ ERROR RESET
007 06F8 9709	DW	KF03	/ LATCH
008 06FA 9709	DW	KF03	/ COIL
009 06FC B30A	DW	KF04	/ VERTICAL SHORT
010 06FE B30A	DW	KF04	/ VERTICAL OPEN
011 0700 5908	DW	KF02	/ CURSOR RIGHT
012 0702 5908	DW	KF02	/ CURSOR DOWN
013 0704 5908	DW	KF02	/ CURSOR LEFT
014 0706 350B	DW	KF06	/ SHIFT
015 0708 D60A	DW	KF05	/ 1 / TIMER 0.01
016 070A D60A	DW	KF05	/ 7 / ADD
017 070C D60A	DW	KF05	/ 4 / SUBTRACT
018 070E 0110	DW	KF10	/ SEARCH
019 0710 731B	DW	KF15	/ SPARE 7
020 0712 731B	DW	KF15	/ SPARE 6
021 0714 D60A	DW	KF05	/ 0 / COUNTER
022 0716 D60A	DW	KF05	/ 2 / TIMER T.01
023 0718 D60A	DW	KF05	/ 8 / CONVERT
024 071A D60A	DW	KF05	/ 5
025 071C 720C	DW	KF09	/ GET PREVIOUS
026 071E 720C	DW	KF09	/ GET NEXT
027 0720 1B0C	DW	KF08	/ GET
028 0722 7111	DW	KF11	/ CLEAR
029 0724 D60A	DW	KF05	/ 3 - TIMER 1.0
030 0726 D60A	DW	KF05	/ 9 - MULTIPLY
031 0728 D60A	DW	KF05	/ 6 - DIVIDE
032 072A 841B	DW	KF16	/ NOT USED
033 072C 841B	DW	KF16	/ NOT USED
034 072E 841B	DW	KF16	/ NOT USED
035 0730 731B	DW	KF15	/ SPARE 5
036 0732 731B	DW	KF15	/ SPARE 4
037 0734 9709	DW	KF03	/ HORIZONTAL SHORT
038 0736 9709	DW	KF03	/ HORIZONTAL OPEN
039 0738 841B	DW	KF16	/ NOT USED
040 073A 841B	DW	KF16	/ NOT USED
041 073C 841B	DW	KF16	/ NOT USED
042 073E 9709	DW	KF03	/ NEG TRANSITIONAL
043 0740 9709	DW	KF03	/ POS TRANSITIONAL
044 0742 9709	DW	KF03	/ NORMALLY OPEN RELAY
045 0744 9709	DW	KF03	/ NORMALLY CLOSED RELAY
046 0746 841B	DW	KF16	/ NOT USED
047 0748 841B	DW	KF16	/ NOT USED
048 074A 841B	DW	KF16	/ NOT USED
049 074C 560B	DW	KF07	/ FORCE
050 074E 731B	DW	KF15	/ SPARE 3
051 0750 861C	DW	KF19	/ SUPERVISORY
052 0752 9D11	DW	KF12	/ DELETE
053 0754 841B	DW	KF16	/ NOT USED
054 0756 841B	DW	KF16	/ NOT USED
055 0758 841B	DW	KF16	/ NOT USED
056 075A 6207	DW	KF01	/ DISABLE
057 075C 731B	DW	KF15	/ SPARE 2

058 075E D314
059 0760 731B
060

DW KF13 / START NEXT
DW KF15 / SPARE 1
EJECT

001
002

SUBJOB KEY DEFINITION

003	0000	KEYENT=	:00	/ ENTER
004	0001	KEYUP=	:01	/ CURSOR UP
005	0002	KEYERR=	:02	/ ERROR RESET
006	0003	KEYLAT=	:03	/ LATCH
007	0004	KEYCOL=	:04	/ COIL
008	0005	KEYVSH=	:05	/ VERTICAL SHORT
009	0006	KEYVOP=	:06	/ VERTICAL OPEN
010	0007	KEYRGT=	:07	/ CURSOR RIGHT
011	0008	KEYDWN=	:08	/ CURSOR DOWN
012	0009	KEYLFT=	:09	/ CURSOR LEFT
013	000A	KEYHT=	:0A	/ SHIFT
014	000B	KEY1=	:0B	/ 1 / TIMER 0.01
015	000C	KEY7=	:0C	/ 7 / ADD
016	000D	KEY4=	:0D	/ 4 / SUBTRACT
017	000E	KEYSCH=	:0E	/ SEARCH
018	000F	KEYSP7=	:0F	/ SPARE 7
019	0010	KEYSP6=	:10	/ SPARE 6
020	0011	KEY0=	:11	/ 0 / COUNTER
021	0012	KEY2=	:12	/ 2 / TIMER T.01
022	0013	KEY8=	:13	/ 8 / CONVERT
023	0014	KEY5=	:14	/ 5
024	0015	KEYPRE=	:15	/ GET PREVIOUS
025	0016	KEYNXT=	:16	/ GET NEXT
026	0017	KEYGET=	:17	/ GET
027	0018	KEYCLR=	:18	/ CLEAR
028	0019	KEY3=	:19	/ 3 - TIMER 1.0
029	001A	KEY9=	:1A	/ 9 - MULTIPLY
030	001B	KEY6=	:1B	/ 6 - DIVIDE
031	001C	KEYNU1=	:1C	/ NOT USED
032	001D	KEYNU2=	:1D	/ NOT USED
033	001E	KEYNU3=	:1E	/ NOT USED
034	001F	KEYSP5=	:1F	/ SPARE 5
035	0020	KEYSP4=	:20	/ SPARE 4
036	0021	KEYHZS=	:21	/ HORIZONTAL SHORT
037	0022	KEYHZO=	:22	/ HORIZONTAL OPEN
038	0023	KEYNU4=	:23	/ NOT USED
039	0024	KEYNU5=	:24	/ NOT USED
040	0025	KEYNU6=	:25	/ NOT USED
041	0026	KEYNE6=	:26	/ NEG TRANSITIONAL
042	0027	KEYPOS=	:27	/ POS TRANSITIONAL
043	0028	KEYNOR=	:28	/ NORMALLY OPEN RELAY
044	0029	KEYNCR=	:29	/ NORMALLY CLOSED RELAY
045	002A	KEYNU7=	:2A	/ NOT USED
046	002B	KEYNU8=	:2B	/ NOT USED
047	002C	KEYNU9=	:2C	/ NOT USED
048	002D	KEYFOR=	:2D	/ FORCE
049	002E	KEYSP3=	:2E	/ SPARE 3
050	002F	KEYSUP=	:2F	/ SUPERVISORY
051	0020	KEYDEL=	:20	/ DELETE
052	0021	KEYNUA=	:21	/ NOT USED
053	0022	KEYNUB=	:22	/ NOT USED
054	0023	KEYNUC=	:23	/ NOT USED
055	0024	KEYDIS=	:24	/ DISABLE
056	0025	KEYSP2=	:25	/ SPARE 2
057	0026	KEYSTR=	:26	/ START NEXT
058	0027	KEYSP1=	:27	/ SPARE 1
059		EJECT		

001
002

SUBJOB KEY FUNCTION : KF01 : DISABLE

003
004
005
006
007
008
009

/

/**KEY FUNCTION : KF01 : DISABLE

/

/**DISABLE COMPLEMENTS THE DISABLE STATE.

/

/**COILS ARE DISABLED ONLY IN NETWORKS.

/**INPUTS ARE DISABLED ONLY IN REFERENCE AREA

/

010 0762 CD281F
011 0765 CD491F
012 0768 CA7107

KF01, CALL KU01 / CHECK FOR RESET
CALL KU02 / CHECK FOR SHIFT
JZ KF0110 / BRANCH ON NO SHIFT

013 076B CD791F	CALL	KF005	/ DISPLAY ERROR
014 076E C34808	JMP	KF01X	/ EXIT
015			
016 0771 CD0B23	KF0110, CALL	K012	/ GET CURSOR POINTERS
017 0774 78	MOV	A, B	/ A ← CURSOR
018 0775 E4F0	ANI	ROWMGR	/ ISOLATE ROW
019 0777 FE80	DFI	ASMR0W	/ CHECK FOR LOGIC AREA
020 0779 C2DA07	JNZ	KF0150	/ BRANCH ON LOGIC AREA
021			
022			
023			
024 077C E5	PUSH	H	/ SAVE POINTER
025 077D 23	INX	H	/ BUMP TO REFERENCE TYPE
026 077E 7E	MOV	A, M	/ A ← REFERENCE TYPE
027 077F FE31	CPI	ASC1	/ MUST BE '1XXX'
028 0781 C48807	JZ	KF0120	/ BRANCH IF OKAY
029 0784 E1	POP	H	/ CLEAN STACK
030 0785 C3D107	JMP	KF01ER	/ GO TO ERROR
031			
032 0788 D1	KF0120, POP	D	/ [D, E] ← POINTER TO FIELD
033 0789 D5	PUSH	D	/ STACK IT AGAIN
034 078A 13	INX	D	/ BUMP TO REFERENCE TYPE
035 078B 13	INX	D	/ BUMP TO FIRST DIGIT
036 078C 210000	LXI	H, 0	/ INITIALIZE BINARY RESULT
037 078F CD8E01	CALL	BCDBN3	/ CONVERT REF TO BINARY
038 0792 2B	DCX	H	/ MAKE RELATIVE BASE 0
039 0793 5D	MOV	E, L	/ E ← BINARY
040 0794 1620	MVI	D, IOFLD	/ D ← FIELD TYPE
041 0796 2193FE	LXI	H, CMDBUF+3	/ [H, L] ← CMD BUFFER
042 0799 EF	MOVDE		/ STORE ADDRESS
043			
044 079A E1	POP	H	/ [H, L] ← FIELD ADDR
045 079B F5	PUSH	H	/ STACK IT AGAIN
046 079C 114F00	LXI	D, ROWD+2	/ [D, E] ← OFFSET
047 079F 19	DAD	D	/ [H, L] ← CURRENT STATE
048 07A0 110008	LXI	D, INPDIS: 100	/ [D, E] ← DISABLE
049 07A3 7E	MOV	A, M	/ A ← STATE
050 07A4 FE44	CPI	ASCD	/ CHECK FOR DISABLED
051 07A6 C2AC07	JNZ	KF0130	/ BRANCH IF NOT
052 07A9 110000	LXI	D, 0	/ [D, E] ← NO FLAG
053			
054 07AC 2195FE	KF0130, LXI	H, CMDBUF+5	/ [H, L] ← DESTINATION
055 07AF EF	MOVDE		/ STORE DATA
056			
057 07B0 11FFF7	LXI	D, DISMSK	/ [D, E] ← MASK
058 07B3 EF	MOVDE		/ STORE MASK
059			
060 07B4 110A21	LXI	D, CMDWRT+CMDO2: 100+LENWRT	/ SET PATMS
061 07B7 CD8125	CALL	P10	/ DO WRITE
062 07BA E1	POP	H	/ GET POINTER OFF STACK
063 07BB C24808	JNZ	KF01X	/ EXIT ON ERROR
064			
065 07BE 114F00	LXI	D, ROWD+2	/ [D, E] ← OFFSET
066 07C1 19	DAD	D	/ [H, L] ← ADDR OF D FLAG
067 07C2 3E44	MVI	A, ASCD	/ A ← 'D'
068 07C4 BE	CMF	M	/ CHECK IF WAS DISABLED
069 07C5 C2CD07	JNZ	KF0140	/ BRANCH IF ENABLED
070			
071 07C8 3620	MVI	M, ASCBLK	/ CLEAR DISABLE
072 07CA C34808	JMP	KF01X	/ EXIT
073			
074 07CB 77	KF0140, MOV	M, A	/ SET DISABLE
075 07CE C34808	JMP	KF01X	/ EXIT
076			
077 07D1 11120F	KF01ER, LXI	D, KF07N1	/ [D, E] ← MESSAGE ADDR
078 07D4 CD7E0C	CALL	ERR0R	/ SET ERROR STATE
079 07D7 C34808	JMP	KF01X	/ AND EXIT
080	EJECT		
001			
002			
003			
004 07DA CDAA23	KF0150, CALL	K017	/ [H, L] ← PTR TO NODE TYPE
005 07DB 7E	MOV	A, M	/ A ← NODE TYPE
006 07DE C16507	LXI	B, NODCIL: 100+NODCOL	/ [B, C] ← PATTERN
007 07E1 B3	CMF	B	/ CHECK FOR MATCH
008 07E2 CAFA0C	JZ	KF0175	/ BRANCH ON IT

009	07E5	010A08	LXI	B, NOLATC!: 100+NODLAT	/ [B, C] ← PATTERN
010	07E8	B8	CMP	B	/ CHECK FOR MATCH
011	07E9	CAFA07	JZ	KF0175	/ BRANCH ON IT
012	07EC	010709	LXI	B, NODCOL!: 100+NOCOIL	/ [B, C] ← PATTERN
013	07EF	B8	CMP	B	/ CHECK FOR MATCH
014	07F0	CAFA07	JZ	KF0175	/ BRANCH ON IT
015	07F3	01080A	LXI	B, NODLAT!: 100+NOLATC	/ [B, C] ← PATTERN
016	07F6	B8	CMP	B	/ CHECK FOR MATCH
017	07F7	C24908	JNZ	KF01RR	/ BRANCH ON ERROR
018					
019	07FA	E5	KF0175, PUSH	H	/ SAVE PATTERN
020	07FB	C5	PUSH	B	/ SAVE POINTER
021					
022	07FC	CD5124	CALL	KU22	/ [H, L] ← COLTAB POINTER
023	07FF	E7	GETHL		/ [H, L] ← STARTING ADDR
024	0800	CD0423	CALL	KU11	/ A ← ROW
025					
026	0803	3D	KF0185, DCR	A	/ DECREMENT COUNT
027	0804	CA0C08	JZ	KF0190	/ BRANCH ON DONE
028	0807	23	INX	H	/ BUMP POINTER
029	0808	23	INX	H	/ TWICE FOR NODE
030	0809	C30308	JMP	KF0185	/ CONTINUE
031					
032	080C	EB	KF0190, XCHG		/ SNAP
033	080D	2193FE	LXI	H, CMDBUF+3	/ [H, L] ← POINTER
034	0810	EF	MOVDE		/ STORE ADDRESS
035	0811	C1	POP	B	/ GET NODE TYPES
036	0812	79	MOV	A, C	/ A ← NEW TYPE
037	0813	07	RLC		/ SHIFT TO
038	0814	07	RLC		/ FOR NODE
039	0815	57	MOV	D, A	/ D ← BYTE 0
040	0816	1E00	MVI	E, 0	/ E ← BYTE 1
041	0818	EF	MOVDE		/ STORE INTO BUFFER
042					
043	0819	C5	PUSH	B	/ SAVE NODE TYPES
044	081A	11FF83	LAI	D, -NODMSK!: 100-1	/ [D, E] ← MASK
045	081D	EF	MOVDE		/ STORE INTO BUFFER
046					
047	081E	110A21	LAI	D, CMDWRT+CMD02!: 100+LENNRT	/ SET PARMS
048	0821	CD8125	CALL	PIO	/ DO WRITE
049	0824	C1	POP	B	/ CLEAN
050	0825	E1	POP	H	/ STACK
051	0826	C24908	JNZ	KF01X	/ EXIT, ON ERROR
052					
053	0829	71	MOV	M, C	/ SET NODE TYPE IN MATRIX
054	082A	79	MOV	A, C	/ A ← NEW NODE TYPE
055	082B	21FD09	LXI	H, NODTAB+NODCON	/ [H, L] ← TABLE ADDR
056	082E	110900	LXI	D, NODRCL	/ [D, E] ← OFFSET
057					
058	0831	BE	KF0195, CMP	M	/ CHECK FOR MATCH
059	0832	CA3908	JZ	KF01A0	/ BRANCH ON MATCH
060	0835	19	DAD	D	/ BUMP POINTER
061	0836	C33108	JMP	KF0195	/ CONTINUE
062					
063	0839	11FAFF	KF01A0, LXI	D, NODDIS-NODECON	/ [D, E] ← OFFSET
064	083C	19	DAD	D	/ [H, L] ← DISPLAY
065	083D	E5	PUSH	H	/ SAVE POINTER
066	083E	CD0B23	CALL	KU12	/ GET CURSOR POINTERS
067	0841	D1	POP	D	/ [D, E] ← SOURCE
068	0842	23	INX	H	/ BUMP TO DISPLAY
069	0843	0605	MVI	B, DSPNOD-2	/ B ← COUNT
070	0845	CD0601	CALL	MOVSTO	/ MOVE STRING
071					
072	0848	C9	KF01X, RET		/ EXIT
073					
074					
075					
076					
077	0849	115008	KF01RR, LXI	D, KF01NC/ PTR TO MSG	
078	084C	CD7E05	CALL	ERRR	/ DISPLAY IT
079	084F	C9	RET		/ DONE
080					
081	0850	08	KF01NC, DB	KF01NA	
082	0851	4E4F5420	DB	"NOT COIL"	
	0855	434F494C			
083		0008	KF01NA, -KF01NC-1		
084			EJECT		

```

001          SUBJOB KEY FUNCTION : KF02: CURSOR CONTROL
002
003          /***KEY FUNCTION : KF02: CURSOR CONTROL
004
005
006 0859 CD381F      KF02: CALL      KU01          / CHECK FOR RESET
007
008 085C F5          PUSH      PSW          / SAVE KEYSTROKE
009 085D FE01        CPI        KEYUP        / LOOK FOR CURSOR UP
010 085F CA9105      JZ         KF0220       / BRANCH ON UP
011 0862 FE08        CPI        KEYDOWN       / LOOK FOR CURSOR DOWN
012 0864 CA9F08      JZ         KF0230       / BRANCH ON DOWN
013 0867 FE09        CPI        KEYLEFT      / LOOK FOR CURSOR LEFT
014 0869 CAD408      JZ         KF0250       / BRANCH ON LEFT
015
016          /***CURSOR RIGHT
017
018 086C CD8509      CALL      K02SUB        / GET DATA
019
020          IF ACTUAL CURSOR NOT = DISPLAY CURSOR;
021          FORCE NEW CURSOR TO 1,1
022
023 086F F8          CMP      B          / SAME?
024 0870 CA7508      JZ         KF0205       / YES, DO REGULAR
025
026          NO, FORCE TO THIS ROW, COL 1
027
028 0873 E6F0        ANI      ROWMSK        / ISOLATE THIS ROW
029
030          HERE FOR REGULAR PROLESS
031
032          KF0203
033 0875 D601        RUI      C01          / MOVE IT RIGHT
034 0877 4F          MOV      C/A          / C ← NEW CURSOR
035 0878 E60F        ANI      COLMSK        / CHECK FOR WRAP-AROUND
036 087A FE0C        CPI      MAXCOL+1     / WILL GO TO LEFT POSITI N
037 087C FAE708      JM      KF0280        / OF SAME ROW, BRANCH OKAY
038 087F 79          MOV      A/C          / A ← NEW CURSOR
039 0880 E6F0        ANI      ROWMSK        / MASK OUT COLUMN
040 0882 FE80        CPI      ASMROW        / CHECK FOR ASM/REF ROW
041 0884 C88C08      JZ      KF0210        / BRANCH IF THERE
042 0887 FA01        ORI      C01          / LOGIC ROW - COL 1
043 0889 C3E608      JMP      KF0270        / GO TO COMMON CODE
044
045 088C F606        KF0210, ORI      C06          / ASM/REF - COL 6
046 088E C3E608      JMP      KF0270        / GO TO COMMON CODE
047          EJECT
048
049          /***CURSOR UP
050
051 0891 CD8509      KF0220, CALL     K02SUB        / GET DATA
052 0894 DA10        SUI      C10          / SHIFT UP
053 0896 4F          MOV      C/A          / C ← NEW CURSOR
054 0897 E6F0        ANI      ROWMSK        / CHECK FOR TOP OF SCREEN
055 0899 C2E708      JNZ     KF0260        / NO, SET CURSOR
056 089C C38009      JMP      KF02A        / YES, NO MOVEMENT
057
058          /***CURSOR DOWN
059
060 089F CD8509      KF0230, CALL     K02SUB        / GET DATA
061 08A2 C610        ADI      C10          / SHIFT DOWN
062 08A4 4F          MOV      C/A          / C ← NEW CURSOR
063 08A5 E6F0        ANI      ROWMSK        / ISOLATE ROW
064 08A7 FE80        CPI      ASMROW        / CHECK FOR ASM/REF ROW
065 08A9 C2BC08      JNZ     KF0240        / BRANCH IF NOT
066 08AC 79          MOV      A/C          / A ← NEW CURSOR
067 08AD E60F        ANI      COLMSK        / ISOLATE COLUMN
068 08AF FE06        CPI      C06          / CHECK VALID MOVE
069 08B1 F2E708      JP      KF0280        / OKAY, PROCEED...
070
071          NOT ABOVE AREA, FORCE TO LEFT SIDE OF AREA
072
073 08B4 79          MOV      A/C          / GET NEW CURSOR
074 08B5 E6F0        ANI      ROWMSK        / ISOLATE ROW
075 08B7 F606        ORI      C06          / FORCE TO LEFT SIDE
076 08B9 C3E608      JMP      KF0270        / OF ASSEMBLY AND GO

```



```

077
078 08BC FE90      KFO240, CPI      ASMR0W+.10      / CHECK FOR OVERFLOW
079 08BE CA8007    JZ      KFO2X      / BRANCH FOR NO MOVEMENT
080 08C1 C3E708    JMF     KFO280     / EXECUTE
081
082
083 /
084 /
085 08C4 CD8509    KFO250, CALL     K028DB      / GET DATA
086 08E7 D601     SUB     .01         / MOVE IT LEFT
087 08C9 4F       MOV     C/A        / C ← NEW CURSOR
088 08CA E60F     ANI    COLMSK     / ISOLATE COLUMN
089 08CC C2D708    JNZ    KFO280     / BRANCH IF NOT WRAP-AROUND
090 08CF 79       MOV     A/C        / A ← CURSOR
091 08D0 E6F0     ANI    ROWMSK     / SAVE ROW
092 08D2 F60E     ORI    :0B        / MOVE TO RIGHT BOUNDARY
093 08D4 C3E608    JMP     KFO270     / GO TO COMMON CODE
094
095 08D7 FE05      KFO260, CPI      .05         / CHECK FOR ILLEGAL MOVE
096 08D9 C2E708    JNZ    KFO280     / BRANCH IF OKAY
097 08DC 79       MOV     A/C        / A ← CURSOR
098 08DD E6F0     ANI    ROWMSK     / CHECK ROW
099 08DF FE80     CPI    ASMR0W     / LOOK FOR ASM/REF
100 08E1 C2E708    JNZ    KFO280     / BRANCH IF NOT
101 08E4 F60E     ORI    :0B        / GO TO RIGHT MARGIN IF SO
102
103 08E6 4F       KFO270, MOV     C/A        / C ← NEW CURSOR
104
105 08E7 CD2B05    KFO250, CALL     CURSOR     / MOVE CURSOR
106 08EA 79       MOV     A/C        / A ← NEW CURSOR
107 08EB 327EFE    STA    CURACT     / STORE NEW ACTUAL POS
108 08EE 327DFE    STA    CURDISP    / STORE NEW DISPLAY POS
109
110 08F1 E6F0     ANI    ROWMSK     / ISOLATE ROW
111 08F3 FE80     CPI    ASMR0W     / CHECK FOR ASSEMBLY ROW
112 08F5 C2FF08    JNZ    KFO280     / BRANCH IF NOT
113
114 08F8 AF       CLA                     / A ← 0
115 08F9 3280FE    STA    CURLUN     / SET NODE TYPE
116
117 08FC C38009    JMF     KFO2X      / EXIT
118

```

✓ CHECK FOR MOVING ALONG COIL EXTENSION

```

002
003 KFO280,
004
005 / CHECK TO SEE IF WE HAVE MOVED TO THE
006 / COIL COLUMN
007
008 08FF 3A7EFE    LDA    CURACT     / GET THE CURRENT POS
009 0902 E60F     ANI    COLMSK     / ISOLATE THE COL
010 0904 FE0B     CPI    MAXCOL     / ARE WE IN COIL COL?
011 0906 CA4E09    JZ     KFO2CL     / YES, GO TO SPECIAL PROC
012
013 / NOT IN COIL COL; SEE IF ON A DASHED
014 / COIL EXTENSION
015
016 0909 3A7EFE    LDA    CURACT     / GET POSITION
017 090C 47       MOV     B/A        / SET B FOR CALL
018 090D CD8D09    CALL   DKDASH     / ARE WE ON A DASHED LINE?
019 0910 C27909    JNZ    KFO296     / NO, ALL SET TO EXIT
020
021 / ON A DASH! SPIN UNTIL NOT
022
023 KFO2LF,
024 /
025 0913 F1       POP     PSW        / GET KEYSTROKE AGAIN
026 0914 F5       PUSH    PSW        / X
027
028 / IS IT RIGHT OR LEFT?
029
030 0915 FE09     CPI    KEYLEFT     / WHICH?
031 0917 C28009    JNZ    KFO290     / IT IS RIGHT!
032
033 / HERE FOR LEFT - DECR AND CHECK FOR LEFT RAIL
034

```



```

035 091A 0D      /  /  / -1 TO COL
036 091B 79      /  /  / GET NEW COL.
037 091C F60F    /  /  / ISOLATE IT; AT LEFT RAIL?
038 091E C23A09  /  /  / NO, GO MOVE IT
039
040              /  /  / FORCE TO RIGHT RAIL, WE HAVE WRAPAROUND!
041
042 0921 79      /  /  / GET CURSOR POS
043 0922 C40B    /  /  / WRAP TO RIGHT COL
044 0924 4F      /  /  / RESET C TO NEW CURSOR
045 0925 C33A09  /  /  / MOVE AND LOOP
046
047              /  /  / HERE FOR RIGHT-HAND MOVE...
048
049              /  /  / KF0290,
050 0928 0C      /  /  / INR      C      / STEP RIGHT ONE COL
051 0929 79      /  /  / MOV      A,C    / GET IT
052 092A F60F    /  /  / ANI     CULMSK  / ISOLATE COLUMN
053 092C FE0C    /  /  / CPI     MAXCOL+1 / IS IT AT RIGHT RAIL?
054 092E DA3609  /  /  / JZ      KF0292  / NO, OKAY TO MOVE
055
056              /  /  / FORCE TO LEFT RAIL, WE HAVE WRAP-AROUND!
057
058 0931 79      /  /  / MOV      A,C    / GET CURSOR POS
059 0932 F6F0    /  /  / ANI     ROWMSK  / SAVE CURRENT ROW
060 0934 3C      /  /  / INR     A      / SET TO COL 1
061 0935 4F      /  /  / MOV      C,A    / RESET C TO NEW POS
062              /  /  / FALL TO COMMON CODE
063
064              /  /  / HERE TO MOVE CURSOR AND LOOP
065
066              /  /  / KF0292,
067 0936 3A7DFE   /  /  / LDA     CURDSP  / GET PRESENT CURSOR POS
068 0939 47      /  /  / MOV     B,A    / TO B FOR MOVE
069 093A CD2B05   /  /  / CALL   CURSOR  / MOVE IT FROM PRES TO NEW
070 093D 79      /  /  / MOV     A,C    / GET NEW
071 093E 327DFE   /  /  / STA     CURDSP  / SET DISPLAY
072 0941 327EFE   /  /  / STA     CURACT  / SET ACTUAL (MATRIX)
073
074              /  /  / SEE IF STILL ON A DASH
075
076 0944 47      /  /  / MOV     B,A    / SET FOR CALL
077 0945 CD8D09   /  /  / CALL   UNDA5H  / STILL DASH?
078 0948 C2FF08   /  /  / JNZ    KF0285  / NO, NOW GO CHECK COL.
079 094B C31309   /  /  / JMP    KF02LP  / YES, LOOP AND MOVE
080
081              /  /  / EJECT
082
083              /  /  / HERE WHEN CURSOR IS ON COIL COLUMN
084 094E CDAA23   /  /  / CALL   KU17    / GET PTR TO "MATROW"
085 0951 7E      /  /  / MOV     A,H    / GET NODE TYPE
086 0952 3280FE   /  /  / STA     CURCON  / SET CURSOR NODE TYPE
087
088              /  /  / SEE IF IT IS A REAL TYPE OR NOT
089
090 0955 B7      /  /  / TST                    / 0? OR REAL?
091 0956 C28008   /  /  / JNZ    KF02X   / REAL, SO ALL SET
092
093              /  /  / IT IS 0, SO DO WE HAVE A BLANK NODE
094              /  /  / ON SCREEN OR AN EXTENDED COIL?
095
096 0959 3A7EFE   /  /  / LDA     CURACT  / GET CURSOR LOC
097 095C 47      /  /  / MOV     B,A    / TO B FOR CALL
098 095D CD4705   /  /  / CALL   CUR100  / GET PTR TO DISPLAY AREA
099 0960 23      /  /  / INX     H      / STEP PAST ATTRIBUTE
100 0961 7E      /  /  / MOV     A,H    / GET CHAR
101 0962 FE0C    /  /  / CPI     ASOBLK  / IS IT A SPACE? (BLANK NODE?)
102 0964 C48009   /  /  / JZ     KF02X   / YES, ALL SET TO EXIT
103
104              /  /  / WE HAVE JUST MOVED TO A DASHED COIL;
105              /  /  / BACK "CURACT" UP TO THE VALID ENTRY
106              /  /  / IN "MATROW" WHERE THE TYPE IS
107
108              /  /  / KF0290,
109 0967 3A7EFE   /  /  / LDA     CURACT  / STEP BACKWARDS UNTIL
110              /  /  / NODE TYPE FOUND
111 0969 3E      /  /  / DCR     A      / COL-1

```

```

112 096B 327EFE      STA      CURACT   / X
113 096E CDAA23     CALL     KU17     / GET PRT TO TYPE
114                  / @ CURACT
115 0971 7E         MOV      A;M      / GET TYPE
116 0972 B7         TST     / O OR REAL?
117 0973 CA6709     JZ      KF0295   / O, KEEP STEPPING BACK
118 0976 C37C09     JMP     KF0297   / REAL, GO STORE AND XIT
119                  EJECT

```

```

\   HERE WHEN FINAL CURSOR POS IS KNOWN,
\   SET UP CONTACT TYPE @ CURSOR

```

```

122
123                  KF0296,
124 0979 CDAA23     CALL     KU17     / GET MATRIX POINTER
125                  KF0297,
126 097C 7E         MOV      A;M      / A ← CONTACT TYPE
127 097D 3280FE     STA      CURCON   / STORE NODE TYPE
128                  /
129 0980 F1         KF02X, POP      PSW     / CLEAN STACK
130 0981 CD531F     CALL     KU03     / CLEAR SHIFT
131 0984 C9         RET      / EXIT
132                  EJECT
133                  /
134                  /***SUBROUTINE TO SET POINTERS
135                  /
136 0985 3A7DFE     KO2SUB, LDA      CURDSP   / A ← DISPLAY CURSOR
137 0988 47         MOV      B;A      / AND SET FOR MOVE
138 0989 3A7EFE     LDA      CURACT   / GET CURSOR LOC REAL
139 098C C9         RET      / EXIT
140                  /
141                  /*** SUBR TO CHECK DISPLAY FOR A DASH
142                  /
143                  / B = CURRENT CURSOR POSITION
144                  / H/L MUST BE FREE!
145                  /
146                  CKDASH,
147 098D CD4705     CALL     CUR100   / GET SCREEN PTR @ CURSOR
148 0990 23         INX      H        / STEP PAST ATTRIBUTE
149 0991 7E         MOV      A;M      / GET CHAR AT NODE
150 0992 E6FE     ANI     -1-CATHI / KILL L. S. BIT
151 0994 FE72     CPI     ASCDSH   / IS IT A DASH?
152                  / Z SET IF DASH
153                  / Z RESET IF NOT
154 0996 C9         RET      / DONE
155                  EJECT
001                  SUBJOB KEY FUNCTION : KF03 : CONTACTS
002                  /
003                  /***KEY FUNCTION : KF03 : CONTACTS
004                  /
005 0997 CD281F     KF03,  CALL     KU01     / CHECK FOR RESET
006 099A CD491F     CALL     KU02     / CHECK FOR SHIFT
007 099D CAA609     JZ      KF0310   / BRANCH ON NO SHIFT
008 09A0 CD791F     CALL     KU05     / DISPLAY MESSAGE
009 09A3 C3E609     JMP     KF03X    / GO TO EXIT
010                  /
011                  /***ENTRY POINT FOR NON-RELAY NODES FROM KF05
012                  /
013 09A6 21F609     KF0310, LXI     H;NODTAB / [H,L] ← START OF TABLE
014 09A9 110900     LXI     D;NODRCL / [D,E] ← TABLE ENTRY LEN
015 09AC 0615     MVI     B;NODTBL / B ← NUMBER OF ENTRIES
016                  /
017 09AE BE         KF0320, CMP      M        / CHECK FOR KEY MATCH
018 09AF CAC009     JZ      KF0330   / BRANCH WHEN FOUND
019 09B2 19         DAD     D        / BUMP POINTER
020 09B3 05         DCR     B        / DECREMENT COUNTER
021 09B4 C2AE09     JNZ     KF0320   / CONTINUE LOOP
022                  /
023 09B7 11EA09     LXI     D;KF03M1 / [D,E] ← ERROR MESSAGE
024 09BA CD7E05     CALL     ERROR    / DISPLAY MESSAGE
025 09BD C3E609     JMP     KF03X    / GO TO COMMON EXIT
026                  /
027 09C0 23         KF0330, INX     H        / BUMP ADDRESS
028 09C1 EB         XCHG     / SWAP [H,L] AND [D,E]
029 09C2 FE22     CPI     KEYHZU   / CHECK FOR HORIZONTAL OPEN
030 09C4 CADC09     JZ      KF0340   / BRANCH ON IT
031 09C7 FE21     CPI     KEYHZS   / CHECK FOR SHORT
032 09C9 C2D909     JNZ     KF0360   / BRANCH IF NOT

```

```

033
034 09DC 3E20      KF0340, MVI      A, ASUBLK      / A ← BLANK
035 09DE 2100FD      LXI      H, DSPNUM+2      / [H,L] ← START OF FIELD
036 09D1 0605      MVI      B, 5              / B ← FIELD LENGTH
037
038 09D3 77        KF0350, MOV      M, A      / STORE BLANK
039 09D4 23        INX      H              / BUMP POINTER
040 09D5 05        DCR      B              / DECREMENT COUNTER
041 09D6 C2D309      JNZ      KF0350      / CONTINUE LOOP
042
043 09D9 21B3FD      KF0360, LXI      H, DSPCON      / [H,L] ← DESTINATION
044 09DC 0605      MVI      B, DSPNUM-2      / B ← COUNTER
045 09DE CD0601      CALL     MOVSI0      / MOVE TO CONTACT FIELD
046 09E1 13        INX      D              / BUMP TO NODE TYPE
047 09E2 1A        LDAX    A              / A ← CONTACT TYPE
048 09E3 327FFE      STA     ASMCON      / SAVE IT
049
050 09E6 CD531F      KF03X,  CALL     KU03      / TURN OFF SHIFT
051 09E9 C9        RET                      / EXIT
052
053 /***MESSAGES
054
055 09EA 0B        KF03M1, DB      KF03MX
056 09EB 554E4B20      DA     'UNK CONTACT'
057 09EF 434F4E54
058 09F3 414354
057      000B      KF03MX= -KF03M1-1
058      EJECT

001
002 /***TABLE FOR CONTACT TYPES
003
004      0000      NODKEY= 0              / KEYSTROKE
005      0001      NODDIS= NODKEY+1      / DISPLAY STRING (5 CHR)
006      0006      NODTYP= NODDIS+5      / NODE TYPE; 1,2 OR 3 NODES
007      0007      NODCON= NODTYP+1      / CONTACT TYPE
008      0008      NODVAL= NODCON+1      / VALID REFERENCE TYPES
009
010      0009      NODRCL= NODVAL+1      / RECORD LENGTH
011
012 /***VALID REFERENCE TYPE FLAGS
013
014      0001      NODOUT= :01          / 0XXX - COILS
015      0002      NODINP= :02          / 1XXX - INPUTS
016      0004      NODSEQ= :04          / 2YXX - SEQUENCERS
017      0008      NODIRG= :08          / 3XXX - INPUT REGISTERS
018      0010      NODHRG= :10          / 4XXX - HOLDING REGISTERS
019      0020      NODCST= :20          / 0YYY - CONSTANT
020      0040      NODBLK= :40          / BBBB - BLANKS
021      :80          / NOT USED
022
023 09F6 28        NODTAB, DB      KEYNOR      / NORMALLY OPEN RELAY
024 09F7 74746076      DB      :74::74::60::76::62
025 09FB 62
025 09FC 01        DB      1              / SINGLE-NODE CONTACT
026 09FD 03        DB      NOOREL
027 09FE 07        DB      NODOUT+NODINP+NODSEQ
028
029 09FF 29        DB      KEYNCR      / NORMALLY CLOSED RELAY
030 0A00 7474607C      DB      :74::74::60::7C::62
031 0A04 62
031 0A05 01        DB      1              / SINGLE-NODE CONTACT
032 0A06 04        DB      NOCREL
033 0A07 07        DB      NODOUT+NODINP+NODSEQ
034
035 0A08 27        DB      KEYPOS      / POSITIVE TRANSITIONAL
036 0A09 74746078      DB      :74::74::60::78::62
037 0A0D 62
037 0A0E 01        DB      1              / SINGLE-NODE CONTACT
038 0A0F 05        DB      NOPOST
039 0A10 07        DB      NODOUT+NODINP+NODSEQ
040
041 0A11 26        DB      KEYNEG      / NEGATIVE TRANSITIONAL
042 0A12 7474607A      DB      :74::74::60::7A::62
043 0A16 62
043 0A17 01        DB      1              / SINGLE-NODE CONTACT
044 0A18 06        DB      NONEGT

```


045	0A19	07	DB	NODOUT+NODINF+NODSEQ	
046					/
047	0A1A	04	DB	KEYCUL-	/ COIL
048	0A1B	7474686C	DB	:74::74::68::6C::6A	
	0A1F	6A			
049	0A20	01	DB	1	/ SINGLE-NODE CONTACT
050	0A21	07	DB	NOCOIL	
051	0A22	01	DB	NODOUT	
052					/
053	0A23	03	DB	KEYLAT	/ LATCH
054	0A24	7474686E	DB	:74::74::68::6E::6A	
	0A28	6A			
055	0A29	01	DB	1	/ SINGLE-NODE CONTACT
056	0A2A	08	DB	NDLATC	
057	0A2B	01	DB	NODOUT	
058					/
059	0A2C	22	DB	KEYHZD	/ HORIZONTAL OPEN
060	0A2D	0D0D0D0C	DB	:0D::0D::0D::0C::0D	
	0A31	0D			
061	0A32	01	DB	1	/ SINGLE-NODE CONTACT
062	0A33	0B	DB	NOHZD	
063	0A34	40	DB	NODEBLK	
064					/
065	0A35	21	DB	KEYH2S	/ HORIZONTAL SHORT
066	0A36	74747474	DB	:74::74::74::74::74	
	0A3A	74			
067	0A3B	01	DB	1	/ SINGLE-NODE CONTACT
068	0A3C	0C	DB	NOH2S	
069	0A3D	40	DB	NODEBLK	
070					/
071	0A3E	FF	DB	:FF	/ DISABLED COIL
072	0A3F	7466686C	DB	:74::66::68::6C::6A	
	0A43	6A			
073	0A44	01	DB	1	/ SINGLE-NODE CONTACT
074	0A45	09	DB	NODCOL	
075	0A46	01	DB	NODOUT	
076					/
077	0A47	FF	DB	:FF	/ DISABLED LATCH
078	0A48	7466686E	DB	:74::66::68::6E::6A	
	0A4C	6A			
079	0A4D	01	DB	1	/ SINGLE-NODE CONTACT
080	0A4E	0A	DB	NODLAT	
081	0A4F	01	DB	NODOUT	
082					/
083	0A50	11	DB	KEYO	/ COUNTER
084	0A51	20435452	DA	< CTR	
	0A55	20			
085	0A56	02	DB	2	
086	0A57	0F	DB	NOCTR	
087	0A58	10	DB	NODHRG	
088					/
089	0A59	19	DB	KEY3	/ TIMER 1.0 SECS
090	0A5A	2054312E	DA	< T1.0	
	0A5E	30			
091	0A5F	02	DB	2	
092	0A60	10	DB	NOT100	
093	0A61	10	DB	NODHRG	
094					/
095	0A62	12	DB	KEY2	/ TIMER 0.1 SECS
096	0A63	2054302E	DA	< T0.1	
	0A67	31			
097	0A68	02	DB	2	
098	0A69	11	DB	NOT010	
099	0A6A	10	DB	NODHRG	
100					/
101					/
102	0A6B	0B	DB	KEY1	/ TIMER .01 SECS
103	0A6C	20542E30	DA	< T.01	
	0A70	31			
104	0A71	02	DB	2	
105	0A72	12	DB	NOT001	
106	0A73	10	DB	NODHRG	
107					/
108					/
109	0A74	0C	DB	KEY7	/ ADD
110	0A75	20414444	DA	< ADD	
	0A79	20			

111	0A7A	03	DB	3	
112	0A7B	16	DB	NOCALL	
113	0A7C	10	DB	NODHRG	
114					
115					
116	0A7D	0D	DB	KEY4	/ SUBTRACT
117	0A7E	20535542	DA	" SUB	
	0A82	20			
118	0A83	03	DB	3	
119	0A84	16	DB	NOCALL	
120	0A85	10	DB	NODHRG	
121					
122					
123	0A86	1A	DB	KEY9	/ MULTIPLY
124	0A87	204D554C	DA	" MUL	
	0A8E	20			
125	0A8C	03	DB	3	
126	0A8D	16	DB	NOCALL	
127	0A8E	10	DB	NODHRG	
128					
129					
130	0A8F	1B	DB	KEY6	/ DIVIDE
131	0A90	20444956	DA	" DIV	
	0A94	20			
132	0A95	03	DB	3	
133	0A96	16	DB	NOCALL	
134	0A97	10	DB	NODHRG	
135					
136	0A98	13	DB	KEY8	/ CONVERT
137	0A99	20434F4E	DA	" CON	
	0A9D	20			
138	0A9E	02	DB	2	
139	0A9F	13	DB	NOCON	
140	0AA0	13	DB	NODINF+NODOUT+NODHRG	
141					
142	0AA1	00000000	DB	0; 0; 0; 0; 0; 0	/ PRESET CONSTANT
	0AA5	0000			
143	0AA7	FF	DB	: FF	
144	0AA8	0D	DB	NOCPRE	
145	0AA9	20	DB	NODCST	
146					
147	0AAA	00000000	DB	0; 0; 0; 0; 0; 0	/ PRESET REGISTER
	0AAE	0000			
148	0AB0	FF	DB	: FF	
149	0AB1	0E	DB	NORPRE	
150	0AB2	18	DB	NODIRG+NODHRG	
151					
152	-				
153	0015		NODTBL=	-NODTAB&NODRCL	/ NUMBER OF ENTRIES
154			EJECT		
001			SUBJOB	KEY FUNCTION : KF04 : VERTICALS	
002					
003			/**KEY	FUNCTION : KF04 : VERTICALS	
004					
005	0AB3	0D281F	KF04,	CALL	K001 / CHECK FOR RESET
006	0AB6	0D491F		CALL	K002 / CHECK FOR SHIFT
007	0AB9	02D20A		JNZ	KF0420 / BRANCH ON ERROR
008	0ABC	01E4D0		LXI	B, CA1101!: 100+CA0011 / ASSUME SHORT
009	0ABF	FE06		CFI	KEYVDF / CHECK FOR OPEN
010	0AC1	02C70A		JNZ	KF0410 / BRANCH ON SHORT
011	0AC4	0120E0		LXI	B, CA1100!: 100+ASCBLK / LOAD FOR OPEN
012					
013	0AC7	78	KF0410,	MOV	A, B / A ← UPPER CHARACTER
014	0AC8	32B8FC		STA	DISPVER / STORE IT
015	0ACB	79		MOV	A, C / A ← LOWER CHARACTER
016	0ACC	3205FD		STA	DISPVER+ROWD / STORE IT
017	0ACF	03D50A		JMP	KF04A / GO TO EXIT
018					
019	0AD2	0D791F	KF0420,	CALL	K003 / DISPLAY ERROR MESSAGE
020					
021	0AD5	09	KF04A,	RET	/ EXIT
022			EJECT		

```

001          SUBJOB KEY FUNCTION : KF05 : NUMERICS
002          /
003          /***KEY FUNCTION : KF05
004          /
005          /***NUMERIC KEYS
006          /
007          /***PARAMETERS:
008          /
009          /      A      - KEYSTROKE
010          /
011 0AD6 CD281F      KF05,  CALL    KU01      / TEST FOR RESET
012 0AD9 CD491F      CALL    KU02      / TEST FOR SHIFT
013 0ADC C2A609      JNZ     KF0310   / SHIFT SET => CONTACT
014 0ADF 47          MOV     B;A      / SAVE KEYSTROKE
015 0AE0 3A7CFE      LDA     KSTATE   / GET KEYBOARD STATE
016 0AE3 E610        ANI     KCLEAR   / CHECK FOR CLEAR FLAG
017 0AE5 CA000B      JZ      KF0520   / YES, DON'T CLEAR
018 0AE8 2100FD      LXI    H;DSPNUM+2 / POINT TO MS POS
019 0AEB 3620        MVI    M;ASCBLK  / BLANK IT
020 0AED 23          INX    H      / STEP TO NEXT CHAR
021 0AEE 3E30        MVI    A;ASC0   / SET UP TO ZERO ASSEMBL
022 0AF0 1604        MVI    D;4      / SET UP TO LOOP 4 0 IN
023          /
024 0AF2 77          KF0510, MOV    M;A      / LOOP TO CLEAR FIELD
025 0AF3 23          INX    H      / BUMP POINTER
026 0AF4 15          DCR    D      / DECREMENT COUNT
027 0AF5 C2F20A      JNZ    KF0510   / LOOP UNTIL DONE
028 0AF8 3A7CFE      LDA     KSTATE   / GET STATE VECTOR
029 0AFB E6EF        ANI     -1-KCLEAR / CLEAR FLAG
030 0AFD 327CFE      STA     KSTATE   / SET NEW STATE
031          /
032 0B00 0E03        KF0520, MVI    C;3      / LOOP TO SHIFT FIELD LEFT
033 0B02 2104FD      LXI    H;DSPNUM+6 / [H,L] <- RIGHT-MOST DI IT
034 0B05 56          MOV    D;M      / GET DIGIT
035          /
036 0B06 2B          KF0530, DCX    H      / MOVE POINTER
037 0B07 5E          MOV    E;M      / E <- CURRENT CHAR
038 0B08 72          MOV    M;D      / STORE DIGIT FROM RIGHT
039 0B09 53          MOV    D;E      / D <- OLD DIGIT
040 0B0A 0D          DCR    C      / DECREMENT POINTER
041 0B0B C2060B      JNZ    KF0530   / LOOP UNTIL DONE
042          /
043          /***BUILD TABLE ENTRY
044          /
045 0B0E 78          MOV    A;B      / A <- KEYSTROKE
046 0B0F 21210B      LXI    H;KF05TB / [H,L] <- TABLE ADDRESS
047          /
048 0B12 BE          KF0540, CMP    M      / CHECK FOR MATCH
049 0B13 CA1B0E      JZ     KF0550   / BRANCH WHEN FOUND
050 0B16 23          INX    H      / BUMP POINTER
051 0B17 23          INX    H      / TO NEXT ENTRY
052 0B18 C3120B      JMP    KF0540   / CONTINUE
053          /
054 0B1B 23          KF0550, INX    H      / BUMP TO DISPLAY CHARACTER
055 0B1C 7E          MOV    A;M      / A <- CHARACTER
056 0B1D 3204FD      STA     DSPNUM+6 / STORE INTO DISPLAY
057          /
058 0B20 C9          RET                    / EXIT
059          EJECT

```

```

001          /
002          /***NUMERIC KEYSTROKE TABLE
003          /
004 0B21 1130      KF05TB, DB     KEY0;ASC0
005 0B23 0B31      DB     KEY1;ASC1
006 0B25 1232      DB     KEY2;ASC2
007 0B27 1933      DB     KEY3;ASC3
008 0B29 0D34      DB     KEY4;ASC4
009 0B2B 1435      DB     KEY5;ASC5
010 0B2D 1B36      DB     KEY6;ASC6
011 0B2F 0C37      DB     KEY7;ASC7
012 0B31 1338      DB     KEY8;ASC8
013 0B33 1A39      DB     KEY9;ASC9
014          EJECT

```



```

001          SUBJOB KEY FUNCTION : KF06 : SHIFT KEY
002          /
003          /***KEY FUNCTION : KF06 : SHIFT KEY
004          /
005          /
006 OB35 CD281F      KF06,   CALL    KU01           / CHECK FOR RESET
007 OB38 217CFE      LXI     H,KSTATE       / [H,L] ← ADDRESS
008 OB3B 7E          MOV     A,M           / A ← STATE VECTOR
009 OB3C E680        ANI     KSHIFT        / ISOLATE SHIFT FLAG
010 OB3E C24A0B     JNZ     KF0610       / BRANCH IF SET
011 OB41 7E          MOV     A,M           / A ← STATE VECTOR
012 OB42 F680        ORI     KSHIFT        / SET SHIFT FLAG
013 OB44 115390     LXI     D,FACREV!:100+ASC$ / [D,E] ← FIELD VALUE
014 OB47 C3500B     JMP     KF06X         / GO TO COMMON EXIT
015          /
016 OB4A 7E          KF0610, MOV    A,M           / A ← STATE VECTOR
017 OB4B E67F        ANI     -1-KSHIFT      / CLEAR SHIFT FLAG
018 OB4D 112080     LXI     D,FACNOR!:100+ASCBLK / [D,E] ← FIELD VALUE

019          /
020 OB50 77          KF06X,  MOV    M,A           / STORE STATE VECTOR
021 OB51 2108FD     LXI     H,DSPSHT      / [H,L] ← FIELD ADDRESS
022 OB54 EF         MOVDE           / STORE FIELD
023 OB55 C9         RET              / EXIT
024          EJECT

```

```

001          SUBJOB KEY FUNCTION : KF07 : FORCE
002          /
003          /***KEY FUNCTION : KF07 : FORCE
004          /
005          /***FORCE COMPLEMENTS THE STATE OF A DISABLED CONTACT.
006          /
007          / RULES:
008          /      1-YOU CAN ONLY FORCE WHEN A CONTACT
009          /      IS DISABLED
010          /      2-YOU CAN ONLY FORCE 1XXX IN THE DISCRETE AREA;
011          /      NOT NETWORK!
012          /      3-YOU CAN ONLY FORCE 0XXX IN THE NETWORK;
013          /      NOT DISCRETE AREA!
014          /
015          /
016 OB56 CD281F      KF07,   CALL    KU01           / CHECK FOR RESET
017 OB59 CD491F      CALL    KU02           / CHECK FOR SHIFT
018 OB5C CA650B      JZ      KF0705       / BRANCH ON NO SHIFT
019 OB5F CD791F      CALL    KU05           / SET ERROR STATE
020 OB62 C3040C      JMP     KF07X         / EXIT
021          /
022 OB65 CD0B23     KF0705, CALL   KU12           / GET CURSOR POINTERS
023          /
024 OB68 78          MOV     A,B           / A ← CURSOR
025 OB69 E6F0        ANI     ROWMSK        / ISOLATE ROW
026 OB6B FE80        CPI     ASMR0W       / LOOK FOR ASSEMBLY AREA
027 OB6D C2B20B     JNZ     KF0725       / BRANCH IF NOT
028          /

```

PROCESS 1XXX

```

030          /
031 OB70 23          INX     H           / STEP TO MS DIGIT
032 OB71 7E          MOV     A,M           / GET IT
033 OB72 FE31        CPI     ASC1          / IS IT A 1?
034 OB74 CA7D0B     JZ      KF0707       / YES, GO ON
035 OB77 11120C     LXI     D,KF07N1     / NO, ERROR:
036 OB7A C3010C     JMP     KF07RR       / "NOT 1XXX"
037          /
038          KF0707,
039 OB7D EB          XCHG           / SWAP
040 OB7E 19          DAD     D           / [H,L] ← VALUE FIELD
041 OB7F 7E          MOV     A,M           / A ← DISABLE FLAG
042 OB80 FE44        CPI     ASCD          / MUST BE 'D'
043 OB82 C2FE0B     JNZ     KF07ER       / BRANCH IF NOT
044          /
045 OB85 23          INX     H           / BUMP TO CONTACT STATE
046 OB86 7E          MOV     A,M           / A ← FIRST CHARACTER
047 OB87 FE20        CPI     ASCBLK       / CHECK FOR BLANK
048 OB89 010004     LXI     B,INPSTA!:100 / [B,C] ← 'ON'
049 OB8C C2920B     JNZ     KF0710       / BRANCH IF 'OFF'
050 OB8F 010000     LXI     B,0         / B,C ← 'OFF'

```

```

051 /
052 0B92 2195FE KF0710, LXI H, CMDBUF+5 / [H, L] ← CMDBUF ADDR
053 0B95 D7 / MOVBC / LOAD WRITE DATA
054 /
055 0B96 01FFFF LXI B, -INPSTA!:100-1/ [B, C] ← MASK
056 0B99 D7 / MOVBC / LOAD MASK
057 /
058 0B9A 13 INX D / BUMP TO
059 0B9B 13 INX D / REFERENCE FIELD
060 0B9C 210000 LXI H, 0 / [H, L] ← 0
061 0B9F CD8E01 CALL BCDBN3 / CONVERT TO BINARY
062 0BA2 2B DCX H / MAKE RELATIVE BASE 0
063 0BA3 65 MOV H, L / H ← DATA0
064 0BA4 2E20 MVI L, IOFLD / L ← DATA1
065 0BA6 2293FE SHLD CMDBUF+3 / STORE ADDR
066 /
067 0BA9 110A21 LXI D, CMDWRT+CMD02!:100+LENWRT / PARMS
068 0BAC CD8125 CALL PIO / DO WRITE
069 0BAF C3040C JMP KF07X / AND EXIT
070 EJECT

```

LOGIC AREA (I. E. OXXX)

```

002 /
003 0BB2 23 KF0725, INX H / BUMP POINTER
004 0BB3 23 INX H / TO DISABLE FIELD
005 0BB4 7E MOV A, M / A ← DISABLE FIELD
006 0BB5 E6FE ANI -1-CATHI / MASK OUT LSB
007 0BB7 FE66 CPI ASCDIS / DISABLE CHARACTER?
008 0BB9 C2FE0B JNZ KF07ER / ERROR IF NOT
009 /
010 0BBC 19 DAD D / [H, L] ← REFERENCE
011 0BBD EB XCHG / SWAP
012 0BBE 13 INX D / BUMP POINTER
013 0BBF 210000 LXI H, 0 / INITIALIZE BINARY
014 0BC2 CD8E01 CALL BCDBN3 / CONVERT
015 0BC5 2B DCX H / MAKE RELATIVE BASE 0
016 0BC6 E5 PUSH H / SAVE REFERENCE
017 0BC7 65 MOV H, L / SWAP H, L
018 0BC8 2E20 MVI L, IOFLD / SET FIELD FLAG
019 0BCA 2293FE SHLD CMDBUF+3 / STORE INTO BUFFER
020 /
021 0BCD 110611 LXI D, CMDRED+CMD02!:100+LENRED / PARMS
022 0BD0 CD8125 CALL PIO / DO READ
023 0BD3 E1 POP H / CLEAN STACK
024 0BD4 C2040C JNZ KF07X / EXIT ON ERROR
025 /
026 0BD7 AF CLA / A ← 0
027 0BD8 BC CMP H / CHECK COIL TYPE
028 0BD9 C2E40B JNZ KF0730 / BRANCH ON INTERNAL
029 /
030 0BDC 0602 MVI B, OUTSTA / B ← FLAG
031 0BDE 11FFFF LXI D, -OUTSTA!:100-1/ [D, E] ← MASK
032 0BE1 C3E90B JMP KF0735 / GO TO COMMON CODE
033 /
034 0BE4 0601 KF0730, MVI B, INTSTA / B ← FLAG
035 0BE6 11FFFE LXI D, -INTSTA!:100-1/ [D, E] ← MASK
036 /
037 0BE9 3AABFE KF0735, LDA RSPBUF+3 / A ← COIL STATE
038 0BEC A8 XRA B / COMPLEMENT STATE
039 0BED 0E00 MVI C, 0 / C ← 0
040 0BEF 47 MOV B, A / B ← WRITE DATA
041 0BF0 2195FE LXI H, CMDBUF+5 / [H, L] ← POINTER
042 0BF3 D7 MOVBC / STORE DATA
043 0BF4 EF MOVDE / STORE MASK
044 /
045 0BF5 110A21 LXI D, CMDWRT+CMD02!:100+LENWRT / PARMS
046 0BF8 CD8125 CALL PIO / DO WRITE
047 0BFB C3040C JMP KF07X / GO TO EXIT
048 EJECT

001 /
002 /***ERROR HANDLER
003 /
004 0BFE 11050C KF07ER, LXI D, KF07M5 / [D, E] ← MESSAGE ADDR
005 KF07RR,

```

```

006 0001 0D7E05          CALL      ERROR          / SET ERROR
007
008 0004 09            KF07X,  RET              / EXIT
009
010                    /***MESSAGES
011
012 0005 0C            KF07MS, DB          KF07MX
013 0006 4E4F5420      DA          "NOT DISABLED"
      000A 44495341
      000E 424C4544
014                    KF07MX= -KF07MS-1
015
016 0012 08            KF07N1, DB          KF07NA
017 0013 4E4F5420      DA          "NOT 1XXX"
      0017 31585858
018                    KF07NX= -KF07N1-1
019                    EJECT

001                    SUBJOB KEY FUNCTION : KF08 : GET
002
003                    /***KEY FUNCTION : KF08 : GET
004
005                    /***THIS FUNCTION FETCHES DISCRETE VALUES FOR
006                    /***DISPLAY IN THE REFERENCE AREA
007
008                    /***ALLOWABLE DISCRETES:
009
010                    /          0XXX - COILS
011                    /          1XXX - INPUTS
012                    /          3XXX - INPUT REGISTERS
013                    /          4XXX - HOLDING REGISTERS
014
015                    /***CURSOR MUST BE IN ASSEMBLY AREA
016
017 001B 0D281F      KF08,   CALL      KU01          / CHECK FOR RESET
018 001E 0D491F          CALL      KU02          / CHECK FOR SHIFT
019 0021 0A2A0C          JZ       KF0805         / BRANCH ON NOSHIFT
020 0024 0D791F          CALL      KU05          / DISPLAY ERROR
021 0027 03650C          JMP      KF08X          / EXIT
022
023 002A 3A7EFE      KF0805, LDA      CURACT          / A ← CURSOR
024 002D E6F0          ANI     ROWMSK          / ISOLATE ROW
025 002F FE80          CPI     ASMROW          / CHECK FOR ASSEMBLY
026 0031 0A3D0C          JZ       KF0810         / BRANCH IF OKAY
027
028 0034 11660C          LXI     D, KF08M1       / [D,E] ← MESSAGE ADDR
029
030 0037 0D7E05      KF08ER, CALL     ERROR          / SET ERROR STATE
031 003A 03650C          JMP      KF08X          / EXIT
032
033 003D 3A01FD      KF0810, LDA     DSPNUM+3     / A ← REFERENCE TYPE
034 0040 11411B      LXI     D, KF14M1       / [D,E] ← MESSAGE ADDR
035 0043 FE32          CPI     ASCZ            / CHECK FOR SEQUENCER
036 0045 0A370C          JZ       KF08ER         / ERROR ON SEQUENCER
037
038 0048 3E1E          MVI     A, NODINP+NODOUT+NODIRG+NODHRG / A ← MASK
039 004A 0DC71F          CALL     KU07           / VALIDATE REFERENCE
040 004D 02650C          JNZ     KF08X          / BRANCH IF NOT VALID
041
042 0050 0D0B23          CALL     KU12           / GET CURSOR POINTERS
043 0053 23            INX     H               / BUMP BEYOND FIELD ATTRIB
044 0054 0604          MVI     B, REFLen      / B ← STRING LENGTH
045 0056 1101FD      LXI     D, DSPNUM+3     / D ← SOURCE
046 0059 0D0601      CALL     MOVSI0         / MOVE IN DATA
047
048 005C 11490C          LXI     D, ROWD-REFLEN  / [D,E] ← OFFSET
049 005F 19            DAD     D               / MOVE TO VAL FIELD
050 0060 1604          MVI     D, REFLen      / D ← FIELD LENGTH
051 0062 0D1B03      CALL     ROWNZ0         / CLEAR VALUE FIELD
052
053 0065 09            KF08X,  RET              / EXIT
054
055                    /***MESSAGE
056
057 0066 0B            KF08M1, DB          KF08MA

```



```

058 0C67 4E4FC420      DA      'NOT ALLOWED'
      0C6B 414C4C4F
      0C6F 574544
059      000B      KF08MX= -KF08M1-1
060      EJECT

001      SUBJOB KEY FUNCTION : KF09 : GET NETWORK
002      /
003      /***KEY FUNCTION : KF09 : GET NETWORK
004      /
005 0C72 CD281F      KF09,   CALL      KU01      / CHECK FOR RESET
006 0C75 CD491F      CALL      KU02      / CHECK FOR SHIFT
007 0C78 CA7F0C      JZ       KF0905     / BRANCH ON NO SHIFT
008 0C7B CD791F      CALL      KU05      / SET ERROR STATE
009 0C7E C9          RET       / EXIT
010      /
011 0C7F FE15      KF0905, CPI       KEYPRE     / IS THIS GET PREVIOUS?
012 0C81 C28D0C      JNZ      KF0915     / NO, GET NEXT
013      /
014      /***FETCH PREVIOUS - FIND PRECEEDING START NODE
015      /
016 0C84 01FFFF      LXI      B,-1       / [B,C] <- DECREMENT
017 0C87 218CFE      LXI      H,ADRSON   / [H,L] <- STARTING ADDR
018 0C8A C3930C      JMP      KF0920     / CONTINUE
019      /
020 0C8D 010100     KF0915, LXI      B,1       / [B,C] <- INCREMENT
021 0C90 218EFE      LXI      H,ADREON   / [H,L] <- STARTING ADDR
022      /
023      /***LOOK FOR PREV/NEXT START NODE
024      /
025 0C93 E7         KF0920, GETHL      / [H,L] <- ADDRESS
026      /
027      / GET THE NETWORK!
028      /
029 0C94 CD980C      CALL     GETNET    / DONE
030 0C97 C9          RET       / RETURN
031      EJECT

```

THIS POINT IS AN
ENTRY TO GET A NETWORK ONTO
THE SCREEN.

```

004      /
005      / B/C=STEP VALUE FOR CTRLR ADDR (+/-/0)
006      / H/L=AN ADDRESS TO START SEARCH
007      / FOR THE START NODE
008      /
009      GETNET,
010      /
011      / IF H/L = 0 AND B/C IS NEG; WE ARE AT START
012      / OF LOGIC. DISPLAY ERROR AND QUIT
013      /
014 0C98 7C          MOV      A,H       / GET MS BYTE
015 0C99 B5          DRA      L         / GET AND TEST BOTH
016 0C9A C2A80C      JNZ      KF0927    / NOT ZERO, GO ON
017      /
018 0C9D 78          MOV      A,B       / GET STEPPER SIGN
019 0C9E B7          TST     / + OR -?
020 0C9F F2A80C      JP       KF0927    / POS, GO ON
021      /
022      / AT START OF LOGIC; GIVE ERROR
023      /
024 0CA2 110B27      LXI      D;MSGSQL/ GET PTR
025 0CA5 C3CA0C      JMP      KF09ER    / DISPLAY AND EXIT
026      /
027      / HERE TO STEP AND SEARCH NEXT NODE
028      /
029      KF0927,
030 0CA8 09          DAD     B         / [H,L] <- NEXT
031 0CA9 09          DAD     B         / ADDRESS FOR READ
032 0CAA EB          XCHG    / SWAP
033 0CAB C5          PUSH   B         / STACK INC/DEC
034 0CAC D5          PUSH   D         / STACK ADDRESS
035 0CAD 2193FE      LXI      H,CMDBUF+3 / [H,L] <- DESTINATION
036 0CE0 EF          MOVDE   / STORE DATA
037      /
038 0CE1 110611      LXI      D;CMDRED+CMD02!:100+LENRED / SET PARMS

```

```

039 0CB4 CD8125 CALL P10 / DO READ
040 0CB7 E1 POP H / GET ADDRESS
041 0CB8 C1 POP B / GET INC/DEC
042 0CB9 C0 RNZ / EXIT ON ERROR
043 /
044 0CBA 3AABFE LDA RSFBUF+3 / A ← NODE - BYTE 0
045 0CBD FE00 CPI NDSN!.04 / CHECK FOR START NODE
046 0CBF CACE00 JZ / BRANCH ON IT
047
048
049 0CC2 FE04 CPI NOEOL!.04 / CHECK FOR END-OF-LOGIC
050 0CC4 C2980C JNZ GETNET / BRANCH IF NOT
051 /
052 0CC7 11D727 LXI D,MSGEOL / [D,E] ← MESSAGE ADDR
053 /
054 0CCA CD7E05 KF09ER, CALL ERROR / SET ERROR
055 0CCD C9 RET / EXIT
056 EJECT

001 /
002 /***BUILD NETWORK
003 /
004 0CCE EB KF09FD, XCHG / SWAP
005 0CCF 3A7CFF LDA RSTATE / A ← STATE VECTOR
006 0CD2 F608 ORI KNET / INDICATE NETWORK ACTIV
007 0CDA 327CFF STA RSTATE / SET STATE
008 0CD7 218CFF LXI H,ADRSON / [H,L] ← DESTINATION
009 0CDA EF MOVDE / STORE DATA
010 0CDB 218EFE LXI H,ADREON / [H,L] ← DESTINATION
011 0CDE EF MOVDE / STORE DATA
012 0CDF D5 PUSH D / SAVE ADDRESS
013 0CE0 CD2121 CALL KU08 / INC/DEC STEP NUMBER
014 /
015 0CE3 CD8A1F CALL KU0e / CLEAR LOGIC DATA
016 /
017 0CE6 CD8C23 CALL KU16 / DISPLAY POWER RAIL
018 /
019 0CE9 D1 KF09FD, POP D / POP ADDRESS
020 0CEA 13 INX D / BUMP
021 0CEB 13 INX D / ADDRESS
022 0CEC 2193FE LXI H,CMDBUF+3 / [H,L] ← DESTINATION
023 0CEF EF MOVDE / STORE DATA
024 0CF0 D5 PUSH D / SAVE ADDRESS
025 /
026 0CF1 110A11 LXI D,CMDRED+CMD02!.100+LENRED / SET PARMS
027 0CF4 CD8125 CALL P10 / DO READ
028 0CF7 C25D0F JNZ / EXIT ON ERROR
029 /
030 0CFA 3AABFE LDA RSFBUF+3 / A ← BYTE 0 OF NODE
031 0CFD E67C ANI NODMsk / ISOLATE NODE TYPE
032 0CFF 0F RRC / NORMALIZE
033 0D00 0F RRC / FOR OFFSET
034 0D01 87 ADD A / DOUBLE IT
035 0D02 0600 MVI B,0 / B ← 0
036 0D04 4F MOV C,A / C ← OFFSET
037 0D05 210A0D LXI H,KF09TB / [H,L] ← TABLE BASE
038 0D08 09 DAB B / [H,L] ← ADDR OF BRANCH
039 0D09 DF DSPTAG / EXECUTE ROUTINE
040 EJECT

001 /
002 /***DISPATCH TABLE FOR NODE TYPES
003 /
004 0D0A 5D0F KF09TB, DW KF0999 / START-OF-NETWORK
005 0D0C 5D0F DW KF0999 / END-OF-LOGIC
006 0D0E 480D DW K09100 / END-OF-COLUMN
007 0D10 AA0D DW K09200 / NORMALLY OPEN RELAY
008 0D12 AA0D DW K09200 / NORMALLY CLOSED RELAY
009 0D14 AA0D DW K09200 / POSITIVE TRANSITIONAL
010 0D16 AA0D DW K09200 / NEGATIVE TRANSITIONAL
011 0D18 AA0D DW K09200 / COIL
012 0D1A AA0D DW K09200 / LATCH
013 0D1C AA0D DW K09200 / DISABLED COIL
014 0D1E AA0D DW K09200 / DISABLED LATCH
015 0D20 0F0E DW K09300 / HORIZONTAL OPEN
016 0D22 0F0E DW K09300 / HORIZONTAL SHORT
017 0D24 1C0E DW K09400 / PRESET CONSTANT

```

```

018 0D26 2E0E      DW      K09500      / PRESET REGISTER
019 0D28 590E      DW      K09600      / COUNTER
020 0D2A 590E      DW      K09600      / TIMER - 1.00 SECS.
021 0D2C 590E      DW      K09600      / TIMER - 0.10 SECS.
022 0D2E 590E      DW      K09600      / TIMER - 0.01 SECS.
023 0D30 D00E      DW      K09700      / CALC : C-NODE : CONSTANT
024 0D32 D00E      DW      K09700      / CALC : C-NODE : REGIST R
025 0D34 F30E      DW      K09800      / CALCULATE
026 0D36 230F      DW      K09900      / CONVERT
027 0D38 4D0F      DW      K09A00      / NULL
028 0D3A 4D0F      DW      K09A00      / NOT USED
029 0D3C 4D0F      DW      K09A00      / NOT USED
030 0D3E 4D0F      DW      K09A00      / NOT USED
031 0D40 4D0F      DW      K09A00      / NOT USED
037 0D42 4D0F      DW      K09A00      / NOT USED
033 0D44 4D0F      DW      K09A00      / NOT USED
034 0D46 4D0F      DW      K09A00      / NOT USED
035                EJECT

001                /
002                /***END OF COLUMN NODE
003                /
004 0D48 CD5124    K09100, CALL      R022      / [H,L] ← COLTAB POINTER
005 0D4B 010400    EXT      B, E0CH1      / [B,C] ← OFFSET
006 0D4E 09                DAD      D                / [H,L] ← EOC NODE
007 0D4F 3A08                MVI      R0, R0E0C1, 04      / STORE BYTE 0
008 0D51 23                INX      H                / BUMP POINTER
009 0D52 3AACFE    LDA      RSPBUF+4      / A ← BYTE 1
010 0D55 77                MOV      M, A            / STORE CONNECTIVITY
011 0D56 3A7EFE    LDA      CURACT        / A ← CURSOR
012 0D59 E60F                ANI      COLNSK        / ISOLATE COLUMN
013 0D5B F610                ORI      .10           / FAKE TO ROW 1
014 0D5D 47                MOV      B, A            / B ← FAKED CURSOR
015 0D5E CD4705    CALL     CUR100        / [H,L] ← FIRST NODE AD R
016 0D61 110600    LXI      D, DSPNOD-1   / [D,E] ← OFFSET
017 0D64 19                DAD      D                / [H,L] ← VERTICAL COLUMN
018 0D65 115000    LXI      D, ROWB        / [D,E] ← OFFSET
019                /
020 0D68 3AACFE    LDA      RSPBUF+4      / A ← CONNECTIVITY
021 0D6B 0607                MVI      B, MAXROW      / B ← COUNTER
022                /
023 0D6D 4F                MOV      C, A            / C ← CONNECTIVITY
024 0D6E E680                ANI      .80            / CHECK FOR POWER FROM U
025 0D70 C2820D    JNZ     K09140        / BRANCH ON IT
026 0D73 79                MOV      A, C            / A ← CONNECTIVITY
027 0D74 E640                ANI      .40            / CHECK FOR POWER DOWN
028 0D76 C27D0D    JNZ     K09130        / BRANCH ON IT
029 0D79 19                DAD      D                / NO CHANGE
030 0D7A C39B0D    JMP     K09170        / CONTINUE
031                /
032 0D7D 36D0    K09130, MVI      R0, CA1101      / DOWN VERTICAL - DISPLAY
033 0D7F C39B0D    JMP     K09160        / CONTINUE
034                /
035 0D82 79                MOV      A, C            / A ← CONNECTIVITY
036 0D83 E640                ANI      .40            / CHECK FOR DOWN VERTICAL
037 0D85 C2960D    JNZ     K09150        / BRANCH ON IT
038                /
039                / IF WE HAVE A DASH HERE, DO NOT CONNECT
040                / THE VERTICAL DOWN. DASH MEANS NO
041                / CONNECTION
042                /
043 0D88 7E                MOV      A, M            / GET CHAR THERE NOW
044 0D89 E6FE                ANI      -1-CATHI      / KILL THE HILITE BIT
045 0D8B FE72                CFI      ASLDsh        / IS IT A DASH?
046 0D8D CA920D    JZ      K09145        / YES, SKIP OVER CHAR
047 0D90 36DC                MVI      R0, CA1110      / NO, LOAD DISPLAY
048                /
049 0D92 19                DAD      D                / BUMP POINTER
050 0D93 C39B0D    JMP     K09170        / CONTINUE
051                /
052 0D96 36E8    K09150, MVI      R0, CA1111      / LOAD DISPLAY
053                /
054 0D98 19                DAD      D                / BUMP ADDRESS
055 0D99 36E4                MVI      R0, CA0011      / LOAD DISPLAY
056                /
057 0D9B 19                DAD      D                / BUMP ADDRESS

```


058	0D9C	79	MOV	A,C	/ A ← CONNECTIVITY
059	0D9D	07	RL		/ ROTATE IT
060	0D9E	05	LDI	D	/ DECREMENT COUNTER
061	0D9F	02A00D	CALL	R09120	/ CONTINUE UNTIL DONE
062					
063	0DA2	D1	POP	D	/ [D,E] ← ADDR
064	0DA3	D5	PIBH	D	/ STACK IT AGAIN
065	0DA4	0D930F	CALL	R0924	/ DO UPDATES
066	0DA7	03E90C	POP	R09250	/ CONTINUE
067			EBE01		
N SINGLE-NODE					
002					
003	0DA6	3AABFE	LDA	R09200	/ A ← BYTE 0
004	0DA9	E67C	AND	000001	/ ISOLATE NODE TYPE
005	0DAF	0F	RRR		/ RIGHT-JUSTIFY
006	0DB0	0F	RRR		/ NODE TYPE
007	0DB1	0D1A7A	CALL	R015	/ DISPLAY CONTACT
008					
009	0DB4	0D0B29	CALL	R012	/ SET POINTERS
010	0DB7	115700	CALL	R090B+2	/ [D,E] ← OFFSET
011	0DBA	19	ORL		/ [H,L] ← REF AREA
012	0DBB	EB	XCHG		/ SWAP
013					
014	0DBE	3AABFE	LDA	R09200+3	/ A ← NODE TYPE
015	0DBF	47	MOV	B,H	/ B ← NODE TYPE
016	0DC0	E603	AND	0E0FLG	/ ISOLATE REFERENCE TYPE
017	0DC2	0E30	MOV	C,0	/ C ← 0
018	0DC4	FE01	CFI	000FLG	/ CHECK FOR OUTPUT COIL
019	0DC6	0AD50D	CALL	R09210	/ BRANCH ON IT
020	0DC9	FE02	CFI	INTFLG	/ CHECK FOR INTERNAL COIL
021	0DCB	0AD50D	CALL	R09210	/ BRANCH ON IT
022	0DCE	FE03	CFI	SEQFLG	/ CHECK FOR SEQUENCER
023	0DD0	0AE50D	CALL	R09230	/ BRANCH TO SEQ
024	0DD3	0E31	MOV	C,ASC1	/ INPUT REFERENCE
025					
026	0DD5	79	MOV	A,C	/ A ← REFERENCE TYPE
027	0DD6	17	STAX	D	/ STORE IT
028	0DD7	13	INX	D	/ BUMP POINTER
029					
030	0DD8	3AABFE	LDA	R09200+4	/ A ← REFERENCE NUMBER
031	0DDB	6F	MOV	L,A	/ L ← REFERENCE NUMBER
032	0DDC	2600	CFI	MOV	
033	0DDE	23	INR	H	/ NAME RELATIVE BASE 1
034	0DDF	78	MOV	A,B	/ A ← NODE TYPE
035	0DE0	E603	AND	0E0FLG	/ ISOLATE REFERENCE TYPE
036	0DE2	FE02	CFI	INTFLG	/ INTERNAL COIL?
037	0DE4	02E80D	CALL	R09220	/ NO, CONTINUE
038	0DE7	24	INR	H	/ BUMP HI-ORDER VALUE
039					
040	0DE8	0DD501	CALL	R09220	/ CONVERT + DISPLAY
041	0DEB	03090F	POP	R09140	/ CONTINUE
042					
043	0DEF	3E32	MOV	H,AB02	/ SEQUENCE REFERENCE
044	0DF0	17	STAX	D	/ DISPLAY "2"
045	0DF1	13	INX	D	/ BUMP POINTER
046	0DF2	3AABFE	LDA	R09200+4	/ A ← SEQ DATA
047	0DF5	47	MOV	B,A	/ B ← BACKUP
048	0DF6	E603	AND	R09080	/ ISOLATE REGISTER RET
049	0DF8	07	RL		/ ROTATE
050	0DF9	07	RL		/ TO FURN
051	0DFA	07	RL		/ CONSTANT
052	0DFE	0631	CALL	R091	/ A ← REG REFERENCE
053	0DF0	17	STAX	D	/ STORE IT
054	0DFE	13	INX	D	/ BUMP POINTER
055					
056	0DFE	78	MOV	H,C	/ A ← REGISTER REF
057	0E00	E61E	AND	000008	/ ISOLATE STEP NUMBER
058	0E02	23	INR	H	/ Bump H
059	0E03	AF	MOV	L,C	/ L ← STEP NUMBER
060	0E04	2A00	CALL	R09	/ H ← 0
061	0E06	0DE801	CALL	R09010	/ CONVERT AND DISPLAY
062					
063	0E09	0D570F	CALL	R09000	/ DO COMMON CODE
064	0E0C	03E90C	POP	R09250	/ CONTINUE
065					

```

001
002
003
004 0E0F 3AABFE K09300, LDA RSPBUF+3 / A ← BYTE 0
005 0E12 E67C ANI NODMSK / ISOLATE NODE TYPE
006 0E14 0F RRC / SHIFT
007 0E15 0F RRC / RIGHT
008 0E16 CD1323 CALL K013
009 0E19 C3090E JMF K09240
010
011
012 /***PRESET CONSTANT/B-NODE CONSTANT
013 /
014 0E1C CD4423 K09400, CALL K014 / DISPLAY TOP LINE
015 /
016 0E1F EB K09405, XCHG / SWAP
017 0E20 21ABFF LXI H, RSPBUF+3 / [H,L] ← POINTER
018 0E23 E7 GETHL / [H,L] ← NODE
019 0E24 7C MOV A, H / A ← BYTE 0
020 0E25 E603 ANI -1-NODMSK-EOCFLG / ISOLATE HI-ORDER DATA
021 0E27 67 MOV H, H / H ← HI-ORDER DATA
022 0E28 CD0201 CALL BNBCD4 / DISPLAY
023 0E2B C3090E JMF K09240 / COMMON CODE
024
025
026 /***PRESET REGISTER / B-NODE REGISTER
027 /
028 0E2F CD6423 K09500, CALL K014 / DISPLAY TOP LINE
029 /
030 0E31 EB K09505, XCHG / SWAP
031 0E32 21ABFE LXI H, RSPBUF+3 / [H,L] ← POINTER
032 0E35 E7 GETHL / [H,L] ← NODE
033 0E36 7C MOV A, H / A ← BYTE 0
034 0E37 2600 MVI H, 0 / H ← 0
035 0E39 23 INX H / BUMP TO MAKE BASE 1
036 0E3A EB XCHG / SWAP
037 0E3B E603 ANI EOCFLG / ISOLATE REGISTER TYPE
038 0E3D 3634 MVI M, ASL4 / ASSUME HOLDING REGISTER
039 0E3F FE00 CPI HLDFLG / CHECK FOR IT
040 0E41 CA510E JZ K09515 / BRANCH OKAY
041 0E44 FE02 CFI DUMFLG / CHECK FOR DUMMY REG
042 0E46 C24F0E JNZ K09510 / BRANCH IF NOT
043 0E49 110000 LXI D, 0 / CLEAR [D,E]
044 0E4C C3510E JMF K09515 / CONTINUE
045 /
046 0E4F 3633 K09510, MVI H, ASL3 / INPUT REGISTER
047 /
048 0E51 EB K09515, XCHG / SWAP
049 0E52 13 INX D / BUMP POINTER
050 0E53 CD0501 CALL BNBCD3 / CONVERT AND DISPLAY
051 0E56 C3090E JMF K09240 / CONTINUE
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```


025	0E7D	114B00	K09625	LXI	D, ROWD+2	/ [D, E] ← OFFSET
026	0E7F	19		DAD	D	/ [H, L] ← NEXT LINE
027	0E80	3605		MVI	M, ASCLB	/ DISPLAY BOARDER
028	0E82	23		INX	H	/ BUMP POINTER
029	0E83	EB		XCHG		/ SWAP
030						
031	0E84	21ABFE		LXI	H, RSPBUF+3	/ [H, L] ← POINTER
032	0E87	E7		GETHL		/ [H, L] ← NODE
033	0E88	7C		MOV	H, H	/ A ← BYTE 0
034	0E89	2600		MVI	H, 0	/ H ← 0
035	0E8B	23		JNA	H	/ BUMP REFERENCE
036	0E8C	EB		XCHG		/ SWAP
037	0E8D	47		MOV	B, A	/ SAVE BYTE 0
038	0E8E	E67C		ANI	NODMSK	/ ISOLATE NODE TYPE
039	0E90	FF4C		CFI	NOCCON:04	/ CHECK FOR CONVERT
040	0E92	0AC10E		JZ	K09680	/ BRANCH ON CONVERT
041						
042	0E95	78		MOV	A, B	/ REFETCH NODE TYPE
043	0E96	E603		ANI	SEUFLG	/ ISOLATE REFERENCE TYPE
044	0E98	3614		MVI	M, ASC4UN	/ ASSUME HOLDING
045	0E9A	FE00		CFI	HLDFLG	/ CHECK FOR HOLDING
046	0E9C	0AADD0E		JZ	K09640	/ BRANCH ON IT
047	0E9F	FE07		CFI	DUMFLG	/ CHECK FOR DUMMY
048	0EA1	02AA0E		JNZ	K09630	/ BRANCH ON INPUT REGIST R
049	0EA4	11C000		LXI	D, 0	/ [D, E] ← 0
050	0EA7	03AC0E		JMP	K09640	/ CONTINUE
051						
052	0EAA	3613	K09630	MVI	M, ASC5UN	/ INPUT REGISTER
053						
054	0EAC	EB	K09640	XCHG		/ SWAP
055	0EAD	13		INX	D	/ BUMP POINTER
056	0EAF	05		PUSH	D	/ SAVE POINTER
057	0EAF	0DD501		CALL	BNB0D3	/ DISPLAY REFERENCE
058						
059	0EB2	E1		POP	H	/ GET POINTER
060	0EB3	0A03		MVI	B, 3	/ B ← COUNTER
061						
062	0EB5	7E	K09600	MOV	A, M	/ A ← REFERENCE DIGIT
063	0EB6	0A20		SUI	ASC0-ASCOUN	/ UNDER LINE DIGIT
064	0EB8	77		MOV	M, A	/ STORE IN DISPLAY
065	0EB9	23		INX	H	/ INCREMENT POINTER
066	0EBA	05		DCR	B	/ DECREMENT COUNTER
067	0EBE	02B50E		JNZ	K09600	/ LOOP
068	0EBE	03090E		JMP	K09240	/ COMMON CODE
069						
070	0EC1	3610	K09680	MVI	M, ASCOUN	/ ASSSUME COIL OUTPUT
071	0EC3	78		MOV	A, B	/ A ← BYTE 0
072	0EC4	E603		ANI	DRGFLG	/ ISOLATE DESTINATION
073	0EC6	FE02		CFI	DINFLG	/ CHECK FOR COIL OUTPUT
074	0EC8	0AADD0E		JZ	K09680	/ BRANCH ON IT
075						
076	0ECE	3614		MVI	M, ASC4UN	/ REGISTER OUTPUT
077	0ECD	03AC0E		JMP	K09640	/ AND CONTINUE
078				EJECT		
001						
002						/**C-NODE CONSTANT / REGISTER
003						
004	0ED0	0D0B23	K09700	CALL	K012	/ B ← CURSOR
005	0ED3	23		INX	H	/ BUMP
006	0ED4	3605		MVI	M, ASCLB	/ DISPLAY LEFT BOARDER
007	0ED6	110400		LXI	D, 4	/ [D, E] ← OFFSET
008	0ED9	19		DAD	D	/ [H, L] ← RIGHT BOARDER
009	0EDA	3609		MVI	M, ASCRB	/ DISPLAY RIGHT BOARDER
010	0EDC	23		INX	H	/ BUMP
011	0EDD	36E0		MVI	M, CA1100	/ DISPLAY CONNECTOR
012	0EDF	114B00		LXI	D, ROWB-DSPNOD+2	/ [D, E] ← OFFSET
013	0EE2	19		DAD	D	/ [H, L] ← NEXT ROW
014	0EE3	3605		MVI	M, ASCLB	/ DISPLAY LEFT BOARDER
015	0EE5	23		INX	H	/ BUMP
016						
017	0EE6	3AABFE		LDA	RSPBUF+3	/ A ← BYTE 0
018	0EE9	E67C		ANI	NODMSK	/ ISOLATE NODE TYPE
019	0EEB	FE50		CFI	NOCCON:04	/ CHECK FOR CONSTANT
020	0EED	0A1F0E		JZ	K09405	/ BRANCH ON IT

325

326

```

021 0EF0 C3310E          JMP      K09505          / BRANCH FOR REGISTER
022                      /
023                      /***CALCULATE NODES
024                      /
025 0EF3 CD0B23      K09800, CALL      KU12          / B ← CURSOR
026 0EF6 1163FF          LXI      D, -ROWB-ROWB+3 / [D,E] ← OFFSET
027 0EF9 19            DAD      D              / [H,L] ← ADDR FOR SYMB
028 0EFA 3AABFE          LDA      RSPBUF+3       / A ← NODE
029 0EFD E603          ANI      DIVFLG         / ISOLATE TYPE
030 0EFF 362B          MVI      M, ASCPLS     / ASSUME ADD
031 0F01 FE00          CPI      ADDFLG        / CHECK
032 0F03 CA160F          JZ       K09810        / BRANCH OKAY
033 0F06 362D          MVI      M, ASCMIN     / ASSUME SUB
034 0F08 FE01          CPI      SUBFLG        / CHECK
035 0F0A CA160F          JZ       K09810        / BRANCH OKAY
036 0F0D 362A          MVI      M, ASCAST     / ASSUME MULTIPLY
037 0F0F FE02          CPI      MPXFLG        / CHECK
038 0F11 CA160F          JZ       K09810        / BRANCH OKAY
039 0F14 362F          MVI      M, ASCSLH     / MUST BE DIVIDE
040                      /
041 0F16 119E00      K09810, LXI      D, ROWB+ROWB-2 / [D,E] ← OFFSET
042 0F19 19            DAD      D              / [H,L] ← DISPLAY
043 0F1A 11560F          LXI      D, KF09MS     / DO FIRST LINE
044 0F1D CD0301          CALL     MOVSTR        / VIA MOVSTR
045 0F20 C37C0E          JMP      K09625        / CONTINUE
046                      /

```

```

001                      /
002                      /***CONVERT NODES
003                      /
004 0F23 3AABFE      K09900, LDA      RSPBUF+3       / A ← NODE TYPE
005 0F26 E603          ANI      DRGFLG        / ISOLATE TYPE DATA
006 0F28 FE02          CPI      DINFLG        / CHECK FOR DESTINATION
007 0F2A F2590E          JF       K09600        / BRANCH ON DESTINATION
008                      /
009 0F2D CD6423          CALL     KU14          / DISPLAY TOP LINE
010 0F30 EB            XCHG                     / SWAP
011 0F31 21ABFE          LXI      H, RSPBUF+3   / SET POINTER
012 0F34 E7            GETHL                    / [H,L] ← NODE DATA
013 0F35 7C            MOV      A, H          / A ← BYTE 0
014 0F36 2600          MVI      H, 0          / CLEAR HI-ORDER BYTE
015 0F38 23            INX      H              / BUMP FROM ZERO BASE
016 0F39 EB            XCHG                     / SWAP
017                      /
018 0F3A 3631          MVI      M, ASC1       / ASSUME INPUT SOURCE
019 0F3C E603          ANI      DRGFLG        / ISOLATE SOURCE TYPE
020 0F3E FE00          CPI      SINFLG        / WAS IT FROM INPUT?
021 0F40 CA450F          JZ       K09905        / BRANCH ON INPUT
022 0F43 3634          MVI      M, ASC4       / SOURCE IS REGISTER
023                      /
024 0F45 23          K09905, INX      H          / BUMP POINTER
025 0F46 EB            XCHG                     / SWAP
026 0F47 CD0501          CALL     BNBCD3        / DISPLAY REFERENCE
027 0F4A C3090E          JMP      K09240        / AND CONTINUE
028                      /

```

```

001                      /
002                      /***USED OR NORMALLY INVALID NODE TYPES
003                      /
004 0F4D 11EA07      K09A00, LXI      D, KF03M1     / [D,E] ← MESSAGE ADDR
005 0F50 CD7E05          CALL     ERROR         / SET ERROR STATE
006 0F53 C35D0F          JMP      KF0999        / AND EXIT
007                      /

```

```

001                      /
002                      /***MESSAGE
003                      /
004 0F56 06          KF09MS, DB      KF09MX
005 0F57 05301B20      DB      ASCLB; ASCBLK; ASCADN; ASCBLK; ASCRB; CA1100
006 0F5B 09E0          KF09MX= .-KF09MS-1
007                      /

```

```

\ HERE WHEN DONE WITH NETWORK DISPLAY
002
003      KF099A)
004      / NOW, PUT CURSOR AT HOME (1,1) UNLESS
005      / WE HAVE A COIL AT 1,1. IN THAT CASE,
006      / PUT IT AT 1, MAX COL
007
008 0F5D 3A7DFE      LDA      CURDSP / GET PRESENT LOC
009 0F60 47         MOV      B,A      / TO B FOR OLD
010
011 0F61 3E11      MOV      A,11     / GET "HOME"
012 0F63 327EFE      STA      CURACT  / SET REAL
013 0F6A 327DFE      STA      CURDSP  / SET DISPLAY-TO-BE
014
015 0F69 0DAA33     CALL     K017     / GET PTR TO "MATROW"
016                  / FOR NODE TYPE
017 0F6C 7E         MOV      A,M      / GET TYPE
018 0F6D 3280FE      STA      CURCON  / SET IT UP
019
020 0F70 0D7635     CALL     [COIL   / IS IT A COIL TYPE?
021 0F73 0A7B0F     CL      KF099A  / NO, ALL SET
022
023      / HERE WHEN NODE @ 1,1 IS A COIL TYPE
024
025 0F74 3E1B      MOV      A,10+MAXCOL / SET ROW 1, COIL COL
026 0F78 327DFE      STA      CURDSP    / DISPLAY HERE, INSTEAD
027
028      / HERE TO PUT CURSOR CORRECTLY
029
030      LF099A)
031 0F7B 3A7DFE      LDA      CURDSP  / GET NEW DISPLAY POS
032 0F7E 4F         MOV      C,A      / SET FOR CALL
033 0F7F 0D2B03     CALL     CURSOR
034
035 0F82 0D4524     CALL     K021     / START TIMERS
036 0F85 01         POP      D        / CLEAN STACK
037
038 0F86 09         RET                    / EXIT
039      EJECT

```

```

\ **** SUBROUTINE K09Z

```

```

\ **** SUBROUTINE K09ZZ

```

```

003      /
004      / ***THIS SUBROUTINE PERFORMS COMMON CODE DURING
005      / ***THE PROCESSING OF A NODE.
006      /
007      / ***PARAMETERS.
008      /
009      / CURACT . ACTUAL CURSOR LOCATION
010      / CURDSP . DISPLAY CURSOR LOCATION
011      /
012      / ***ACTIVITIES:
013      /
014      / 1. LOAD NODE TYPE INTO MATROW (K09Z ONLY)
015      / 2. UPDATES ADREUN
016      / 3. UPDATES COLTAB
017      / 4. MOVES CURSOR TO NEXT DISPLAY POSITION
018      /
019      / K09Z)
020
021      / THIS ENTRANCE BEGINS BY UPDATING "MATROW"
022      / WITH THE CURRENT NODE TYPE
023
024 0F87 0DAA33     CALL     K017     / [H,L] <= MATRIX POINTE
025 0F8A 3AABFE     LDA      B+FBUF+3 / A <- BYTE 0
026 0F8D FA7C      AND     ROUNSK   / ISOLATE NODE TYPE
027 0F8F 0F         RRC                    / SHIFT
028 0F90 0F         RRC                    / RIGHT
029 0F91 77         MOV      B,A      / STORE IN MATRIX
030
031      / THIS IS THE ALTERNATE ENTRY WHICH DOES NOT
032      / UPDATE MATROW BY DOES EVERYTHING ELSE.
033      /
034      / K09ZZ)
035
036      / NOW UPDATE THE "CURRENT" (-2) CONTROLLER

```

```

037 / END OF NETWORK ADDRESS. (IT WILL BE
038 / THE END OF THIS NETWORK WHEN THE
039 / WHOLE NETWORK IS DISPLAYED.)
040
041 0F92 218EFE LXI H,ADREON / [H,L] ← SOURCE
042 0F95 E7 GETHL / [H,L] ← LAST ADDRESS
043 0F96 23 INX H / BUMP
044 0F97 23 INX H / ADDRESS
045 0F98 EB XCHG / SWAP
046 0F99 218EFE LXI H,ADREON / [H,L] ← DESTINATION
047 0F9C EF MOVDE / STORE DATA
048
049 / NOW, KEEP TRACK OF THE LOWEST AND HIGHEST
050 / CONTROLLER ADDR FOR THIS COLUMN.
051
052 0F9D 3A7EFE LDA CURACT / A ← CURSOR
053 0FA0 E60F ANI COLMSK / ISOLATE COLUMN
054 0FA2 010600 LXI B,COLBKL / [B,C] ← OFFSET
055 0FA5 21E7FD LXI H,COLTAB-COLBKL / [H,L] ← ADDRESS
056
057 0FAB 09 K09Z10, DAD B / BUMP ADDRESS
058 0FA9 3D DCR A / DECREMENT COUNTER
059 0FAA C2A80F JNZ K09Z10 / LOOP UNTIL DONE
060
061 0FAD BE CMP M / TEST FOR START OF COLUMN
062 0FAE C2BB0F JNZ K09Z20 / BRANCH IF NOT
063 0FB1 23 INX H / BUMP POINTER
064 0FB2 BE CMP M / TEST AGAIN
065 0FB3 2B DCA H / DECREMENT POINTER
066 0FB4 C2BB0F JNZ K09Z20 / BRANCH IF NOT ZERO
067 0FB7 EF MOVDE / STORE ADDR START
068 0FB8 C3BD0F JMP K09Z25 / CONTINUE
069
070 0FBB 23 K09Z20, INX H / BUMP TO
071 0FBC 23 INX H / ADDR END
072
073 0FBD EF K09Z25, MOVDE / STORE LAST ADDR
074
075 0FBE 3AABFE LDA RSPBUF+3 / A ← BYTE 0 OF NODE
076 0FC1 E680 ANI EOCFLG / CHECK FOR EOC FLAG
077 0FC3 CAE50F JZ K09Z40 / BRANCH IF NOT
078
079 0FC6 3A7EFE K09Z30, LDA CURACT / A ← CURSOR
080 0FC9 47 MOV B,A / B ← CURSOR
081 0FCA E60F ANI COLMSK / ISOLATE COLUMN
082 0FCC FE0B CPI MAXCOL / CHECK FOR LAST COLUMN
083 0FCE CA0010 JZ K09ZX / EXIT IF DONE
084 0FD1 C611 ADI :11 / GO TO START NEXT COLUMN
085
086 0FD3 4F K09Z35, MOV C,A / C ← NEW CURSOR
087 0FD4 327EFE STA CURACT / STORE NEW REAL CURSOR
088 0FD7 3A7DFE LDA CURDSP / GET "OLD" CURSOR POS
089 0FDA 47 MOV B,A / SET UP FOR MOVE
090 0FDB CD2B05 CALL CURSOR / DISPLAY NEW CURSOR
091 0FDE 79 MOV A,C / A ← NEW DISPLAY CURSOR
092 0FDF 327DFE STA CURDSP / STORE IT
093 0FE2 C30010 JMP K09ZX / EXIT
094
095 0FE5 3AABFE K09Z40, LDA RSPBUF+3 / A ← NODE TYPE
096 0FEB E67C ANI ROWMSK / ISOLATE IT
097 0FEA FE08 CPI NOEUC!:04 / CHECK FOR EOC NODE
098 0FEC CAC60F JZ K09Z30 / BRANCH ON IT
099
100 0FEF 3A7EFE LDA CURACT / A ← CURSOR
101 0FF2 47 MOV B,A / B ← CURSOR
102 0FF3 E6F0 ANI ROWMSK / ISOLATE ROW
103 0FF5 FE70 CPI MAXROW!:10 / CHECK FOR MAX
104 0FF7 CA0010 JZ K09ZX / EXIT IF IT IS
105 0FFA 78 MOV A,B / A ← OLD CURSOR
106 0FFB C610 ADI :10 / DROP TO NEXT ROW
107 0FFD C3D30F JMP K09Z35 / CONTINUE
108
109 1000 C9 K09ZX, RET / EXIT
110 EJECT

```



```

001          SUBJOB KEY FUNCTION : KF10 : SEARCH
002
003          /***KEY FUNCTION : KF10 : SEARCH
004
005 1001 CD281F      KF10: CALL      KU01          / CHECK FOR RESET
006 1004 CD491F      CALL      KU02          / CHECK FOR SHIFT
007 1007 C21010     JNZ       KF1005       / BRANCH ON SHIFT
008
009 100A 110200     LXI       D,ADRUSE       / SEARCH FROM START
010 100D C31710     JMP       KF1010       / OF LOGIC
011
012 1010 218EFE     KF1005: LXI       H,ADREON      / SEARCH FROM CURRENT
013 1013 E7         GETHL          / NETWORK
014 1014 23        INX       H              / BUMP TO
015 1015 23        INX       H              / NEXT NETWORK
016 1016 EB        XCHG          / SWAP
017
018 1017 2193FE     KF1010: LXI       H,CMDBUF+3    / SET POINTER
019 101A EF        MOVDE          / LOAD ADDRESS
020
021 101B 110000     LXI       D,0           / [D,E] ← INITIAL DATA
022 101E EF        MOVDE          / LOAD IN BUFFER
023
024 101F 11FFFF     LXI       D,FFFF       / [D,E] ← INITIAL MASK
025 1022 EF        MOVDE          / LOAD IN BUFFER
026
027          /
028          INITIALIZE RUNNING STEP COUNT
029 1023 210000     LXI       H,0          / SET TO ZERO
030 1026 220BFF     SHLD     SRCHST       / X
031          EJECT

```

\ HERE TO CHECK CONTACT FIELD
 \ FOR SEARCH TYPE

```

034
035 1029 3AB3FC     LDA       DSPCON       / A ← CONTACT FIELD
036 102C FE1F      CPI       ASCCBK      / CHECK FOR UNDEFINED
037 102E CA4110     JZ        KF1015      / BRANCH ON UNDEFINED
038
039 1031 3A97FE     LDA       CMDBUF+7     / A ← MASKHI
040 1034 E683      ANI       -1-NODMSK   / CLEAR NODE TYPE FIELD
041 1036 3297FE     STA       CMDBUF+7     / SET NEW MASKHI
042
043 1039 3A7FFE     LDA       ASMCN       / A ← CONTACT TYPE
044 103C 07        RLC          / SHIFT
045 103D 07        RLC          / LEFT
046 103E 3295FE     STA       CMDBUF+5     / STORE NEW DATAHI
047          EJECT

```

\ HERE TO CHECK NUMERIC FIELD
 \ FOR SEARCH TYPE

```

050
051 1041 3A01FD     KF1015: LDA       DSPNUM+3    / A ← NUMERIC FIELD
052 1044 FE1D      CPI       ASCNBK      / CHECK FOR UNDEFINED
053 1046 CA6710     JZ        KF1020      / BRANCH ON UNDEFINED
054
055 1049 3EFF      MVI       A,FF        / ALLOW ALL TYPES
056 104B CDC71F     CALL     KU07         / VERIFY REFERENCE
057 104E C25711     JNZ     KF10X        / BRANCH ON ERROR
058
059 1051 EB        XCHG          /
060 1052 2195FE     LXI       H,CMDBUF+5   / SET POINTER
061 1055 7E        MOV       A,M         / A ← DATAHI
062 1056 B2        ORA       D           / SET REFERENCE HIGH BIT
063 1057 77        MOV       M,A         / STORE DATAHI
064
065 1058 23        INA       H           / BUMP POINTER
066 1059 73        MOV       M,E         / STORE DATALO
067
068 105A 23        INA       H           / BUMP POINTER (=CMDBUF+7)
069 105B 7E        MOV       A,M         / A ← MASKHI
070 105D FEFF      CPI       :FF         / ARE WE LOOKING FOR
071          / A CONTACT?
072 105E CA6410     JZ        KF1017      / YES, IGNORE REF
073          / HIGH BITS FOR NOW
074          / NO.....

```

```

075 1061 EAF0          ANI      -1-SEMFLG      / CLEAR REFERENCE HIGH BITS
076 1063 77           MOV      M,A              / STORE MASKHI
077                   KF1017,
078 1064 23           INA      H              / BUMP POINTER
079 1065 3600         NVI      M;0            / CLEAR MASKLO
080                   EUBCT

```

\ HERE TO ACTUALLY
\ DO THE SEARCH!

```

083
084 1067 110A30       KF1020, LXI      D;CNDSCN!:100+LENSCH / SET PARMS
085 106A 008125       CALL     P10              / DO SEARCH
086 106D 025711       JNZ      KF10X           / BRANCH ON ERROR
087
088                   /      KEEP A RUNNING COUNT OF THE # OF S. O. N.
089                   /      NODES PASSED DURING SEARCH
090
091 1070 21ADFE       LXI      H;RSPBUF+5 / PTR TO RETURNED COUNT
092 1073 E7           GETHL           / # TO...
093 1074 EB           XCHG           / D/E
094
095 1075 210BFF       LXI      H;SRCHST/ PTR TO RUNNING COUNT CELLS
096 1078 E7           GETHL           / GET CURRENT #
097 1079 19           DAD      D          / ADD NEW VALUE
098 107A EB           XCHG           / TO D/E FOR STORE
099 107B 210BFF       LXI      H;SRCHST/ PTR AGAIN
100 107E EF           MOVDE          / SAVE RUNNING TOTAL
101
102                   /      DID WE FIND ANYTHING? OR NO MATCH
103
104 107F 21A9FE       LXI      H;RSPBUF+1 / GET THE ADDR RETURNED
105 1082 E7           GETHL           / INTO H/L
106
107 1083 11FFFF       LXI      D,-1          / SEE IF VALID OR NOT
108 1086 F7           DUMP           / MATCH OR NO MATCH?
109 1087 CA5B11       JZ       KF12NM / NO MATCH, TAKE ERROR EXIT
110
111                   /      DID WE GET THE FAMOUS "END-OF-LOGIC"
112                   /      NODE RETURNED. IF SO, "NO MATCH"
113
114 108A 3AABFE       LDA      RSPBUF+3/ GET NODE TYPE
115 108D E67C         ANI      NODMSK / ISOLATE IT
116 108F FE04         CPI      NOEOL!4 / IS IT END OF LOGIC?
117 1091 CA5B11       JZ       KF12NM / YES, TAKE NO MATCH EXIT
118
119                   /      OKAY, WE FOUND A MATCH TO THE REQUEST.
120                   /      NOW, SEE IF WE WERE LOOKING FOR A # ONLY. IF SO,
121                   /      THIS IS A SPECIAL PROBLEM
122
123 1094 3AB3FC       LDA      DSPCON / GET CONTACT TYPE
124 1097 FE1F         CPI      ASCCBK / BLANK?
125 1099 021911       JNZ      KF1080 / NO, SKIP SPECIAL CHECK
126
127                   /      CONTACT IS BLANK; SEE IF NUMBER IS NOT
128                   /      BLANK
129
130 109C 3A01FD       LDA      DSPNUM+3/ GET MS DIGIT OF #
131 109F FE1D         CPI      ASCNBK / IS THERE A #?
132 10A1 CA1911       JZ       KF1080 / NO, NOT SPECIAL
133                   EJECT

```

\ HERE WHEN ONLY # SEARCH!

```

135
136                   /      THIS IS A SPECIAL CASE. WE HAVE A MATCH ON
137                   /      THE L. S. BYTE OF REFERENCE ONLY. NOW WE
138                   /      MUST SEE IF THE NODE TYPE WITH MATCHING
139                   /      # IS TRULY A MATCH.
140
141                   /      THERE ARE BASICALLY 2 PERTINENT SERIES OF #S:
142                   /      1-IF USER PUT IN 0XXX TO 2XXX, MATCH ON
143                   /      TYPE MUST BE IN RANGE OF OPEN RELAY TO
144                   /      DIS LATCH COIL. ALSO, SPECIAL
145                   /      TEST ON CONVERT NODE FOR 0XXX,1XXXX
146                   /      2-IF USER PUT IN 3XXX TO 4XXX, MATCH
147                   /      TYPE MUST BE ON RANGE FROM PRESET REG/B TO C-NODE
148                   /      REG, EXCEPT C-NODE CONSTANT. ALSO, CALC NODE

```

```

149 / IS A SPECIAL CASE; MUST BE 4XXX ONLY.
150
151 / DECIDE WHICH SERIES: 0-2XXX OR 3-4XXX
152
153 10A4 FE33 CPI A5C3 / < OR >=?
154 10A6 D2E310 JNC KF10RG / >= WE HAVE A REGISTER
155 / < FALL TO 0-2XXX
156 EJECT

```

```

\ HERE WHEN USER WANTS
\ OXXX-2XXX

```

```

003
004 / GET NODE TYPE AND SEE IF VALID
005
006 10A9 3AABFE LDA RSPBUF+3/ GET NODE TYPE BYTE
007 10AC E67C ANI NODMSK / ISOLATE TYPE
008 10AE 0F RRC / TO LOW END
009 10AF 0F RRC / X
010
011 / IF CONVERT, SPECIAL
012
013 10B0 FE13 CPI NOCON / CONVERT?
014 10B2 CAC810 JZ KF1025 / YES, GO SPECIAL
015
016 / SEE IF IN REGULAR RANGE
017
018 10B5 FE03 CPI NOOREL / LOWER THAN LOW?
019 10B7 DA0F11 JC KF1070 / YES, MATCH FAIL
020 10BA FE0B CPI NODLAT+1/ HIGHER THAN HIGH?
021 10BC D20F11 JNC KF1070 / YES, FAIL
022
023 / PASSES RANGE TEST; SEE IF REF # HIGH BITS
024 / MATCH UP.
025
026 10BF CD6411 CALL COMPHI / SEE IF MATCH...
027 10D2 C20F11 JNZ KF1070 / NO, FAIL. LOOK AGAIN
028 10D5 C31911 JMP KF1080 / SUCCESS! TAKE IT.
029 EJECT

```

```

\ HERE FOR CONVERT NODE. SEE IF USER #
\ 1XXX OR 1-256

```

```

003
004 KF1025,
005 10C8 CD6411 CALL COMPHI / IF THEY MATCH AND ARE 00,
006 / OKAY. HAVE 1XXX MATCH
007 10CB C2D510 JNZ KF1030 / NO MATCH, CHECK FOR 1-256
008 10CE B7 TST / MATCH! ARE THEY 00?
009 10CF CA1911 JZ KF1080 / YES, TAKE IT
010 10D2 C30F11 JMP KF1070 / NO, FAILURE
011
012 / NOT 1XXX. SEE IF USER ASKED FOR
013 / 1-256 (A=01) AND GOT 1-256 (B=11)
014
015 KF1030,
016 10D5 FE01 CPI 01 / DID USER ASK FOR 1-256?
017 10D7 C20F11 JNZ KF1070 / NO, FAILURE
018 / YES, CHECK RESULT
019 10DA 78 MOV A:B / GET IT
020 10DB FE03 CPI 3 / IS IT 1-256 IN NODE?
021 10DD C20F11 JNZ KF1070 / NO, FAILURE
022 10E0 C31911 JMP KF1080 / YES, TAKE IT
023 EJECT

```

```

\ HERE WHEN USER WANTS
\ 3XXX-4XXX

```

```

003
004 KF10RG,
005
006 / IF NODE TYPE IS C-NODE CONSTANT; FAILURE.
007 / IF NODE TYPE IS CALC, SPECIAL CHECK
008 / ELSE CHECK RANGE FROM B-NODE REG TO C-NODE REG
009
010 10E3 3AABFE LDA RSPBUF+3/ GET NODE TYPE
011 10E6 E67C ANI NODMSK / ISOLATE IT
012 10E8 0F RRC / TO LOW END
013 10E9 0F RRC / X

```



```

014
015 / CALC?
016
017 10EA FE16 CPI NOCALC / IS IT?
018 10EC CA0711 JZ KF1040 / YES, GO SPECIAL
019
020 / C-NODE CONSTANT?
021
022 10EF FE14 CPI NOCCON / IS IT?
023 10F1 CA0F11 JZ KF1070 / YES, FAIL
024
025 / CHECK THE RANGE
026
027 10F4 FE0E CPI NORPRE / IS IT LOWER THAN LOW?
028 10F6 DA0F11 JC KF1070 / YES, FAIL
029 10F9 FE16 CPI NOCREG+1 / HIGHER THAN HIGH?
030 10FB D20F11 JNC KF1070 / YES, FAIL
031
032 / IN RANGE! MATCH ON REF # HI-BITS
033
034 10FE CD6411 CALL COMPHI / MATCH?
035 1101 C20F11 JNZ KF1070 / NO, FAIL
036 1104 C31911 JMP KF1080 / YES, TAKE IT
037 EJECT

```

\ HERE FOR CALC NODE.
 \ MUST BE 4XXX REQUEST.

```

003
004 KF1040,
005 1107 3A01FD LDA DSPNUM+3/ GET MS DIGIT OF #
006 110A FE34 CPI ASC4 / IS IT 4?
007 110C CA1911 JZ KF1080 / YES, TAKE IT
008 / NO, FALL TO FAILURE COMMON
009 EJECT

```

\ HERE WHEN WE DECIDE THIS SEARCH
 \ IS A FAILURE

```

012
013 KF1070,
014
015 / STEP TO NEXT CONTROLLER ADDR
016 / AND LOOP BACK TO SEARCH I/O
017 / STAT:
018 / H/L = CURRENT CTRLR ADDR
019
020 110F EB XCHG / TO D/E
021 1110 13 INX D / STEP TO NEXT
022 1111 13 INX D / CONTROLLER ADDR
023 1112 2193FE LXI H; CMDBUF+3 / POINT TO ADDR AREA
024 1115 EF MOVDE / LOAD ADDR FOR I/O
025 1116 C36710 JMP KF1020 / GO SEARCH AGAIN
026 EJECT

```

\ HERE WHEN WE DECIDE THIS MATCH
 \ IS GOOD!

```

003
004 KF1080,
005
006 / STAT:
007 / H/L = CONTROLLER ADDR W/GOOD MATCH
008
009 1119 EB XCHG / TO D/E FOR NOW
010
011 / GET AND SAVE OLD STEP # TIL LATER
012
013 111A 218AFE LXI H; STPNUM / [H,L] <- PTR TO SEQ NUMBE
014 111D E7 GETHL / [H,L] <- OLD SEQUENCE #
015 111E E5 ER
016 111F EB PUSH H / STACK IT!
017 XCHG / SWAP ADDR TO H/L
018
019 / NOW FIGURE OUT THE STEP PARAMETER. IF NO
020 / CONDITION ON SEARCH, STEP = 0.
021 / IF ANY CONDITION, STEP = -1.

```

```

022 1120 01FFFF          LXI    B,-1      / ASSUME CONDITIONS
023 1123 3AB3FC          LDA    DSPCON    / CHECK CONTACT FIELD
024 1126 FE1F            CPI    ASCLBF    / UNDEFINED?
025 1128 C23411          JNZ    KF1090    / NO, GO
026
027 112B 3A01FD          LDA    DSPNUM+3/ CHECK NUMERIC FIELD
028 112E FE1D            CPI    ASCNBK    / UNDEFINED?
029 1130 C23411          JNZ    KF1090    / NO, GO
030
031 1133 03              INX    B          / BOTH UNDEFINED; SET 0000
032                      EJECT
033                      / GET THE NETWORK ONTO THE SCREEN!!!!
034
035                      KF1090,
036 1134 CD980C          CALL   GETNET    / FETCH THIS NETWORK
037
038 1137 D1              POP    D          / GET OLD SEQ #
039 1138 210BFF          LXI    H,SRCHST  / POINT TO # OF SONS
040 113B E7              GETHL             / GET THEM!
041 113C 3A7CFE          LDA    KSTATE    / A ← STATE VECTOR
042 113F E620            ANI    KRESET    / CHECK FOR ERRORS
043 1141 C25711          JNZ    KF10X     / EXIT ON ERROR
044
045 1144 CD491F          CALL   KU02      / CHECK FOR SHIFT
046 1147 CA4B11          JZ     KF1095    / BRANCH ON NO SHIFT
047
048 114A 19              DAD    D          / [H,L] ← NEW STEP NUMBER
049
050 114B EB              KF1095, XCHG     / SWAP
051 114C 218AFE          LXI    H,STENUM  / SET DESTINATION
052 114F EF              MOVDE             / LOAD NEW SEQUENCE NUMBER
053
054 1150 EB              XCHG             / SWAP
055 1151 1118FD          LXI    D,DSPSTP  / [D,E] ← DESTINATION
056 1154 CDC201          CALL   BNBLD4    / DISPLAY BCD
057
058 1157 CD531F          KF10X, CALL   KU03    / CLEAR SHIFT
059 115A C9              RET              / EXIT
060
061                      / HERE WHEN SEARCH COMMAND ANSWER
062                      / SAYS "NO MATCH"
063
064                      KF12NM,
065 115B 11EF27          LXI    D,MSGSCH/ GET PTR TO MSG
066 115E CD7E05          CALL   ERROR     / DISPLAY IT
067 1161 C35711          JMP    KF10X     / EXIT
068
069                      / COMPHI IS A SUBR TO GET AND COMPARE
070                      / THE REF # HIGH BITS
071
072                      / A & B MUST BE FREE
073
074                      / EXIT:
075                      / A = HIGH BITS FROM "CMDBUF+5"
076                      / B = HIGH BITS FROM "RSPBUF+3"
077                      / Z BIT SET IF =; RESET IF NOT
078
079                      COMPHI,
080 1164 3AABFE          LDA    RSPBUF+3/ GET ANSER HIGH BITS
081 1167 E603            ANI    3          / ISOLATE THEM
082 1169 47              MOV    B:A        / SAVE IN B FOR EXIT
083 116A 3A95FE          LDA    CMDBUF+5/ GET HIGH BITS FROM REQUEST
084 116D E603            ANI    3          / ISOLATE THEM
085 116F B8              CMP    B          / SET,RESET Z-BIT
086 1170 C9              RET
087                      EJECT
001                      SUBJOB KEY FUNCTION : KF11 : CLEAR
002
003                      /***KEY FUNCTION : KF11 : CLEAR
004
005 1171 CD281F          KF11, CALL   KU01    / CHECK FOR RESET
006 1174 CD491F          CALL   KU02      / CHECK FOR SHIFT
007 1177 CA7B11          JZ     KF1110    / BRANCH ON NO SHIFT
008
009                      / SHIFT/CLEAR MAKES THE P130 RESET!
010

```

```

011 117A C7          RST      0          / DO IT!
012
013                /      REGULAR CLEAR; JUST THE ASSEMBLY AREA
014
015 117B 21B1FC     KF1110, LXI      H; DSPASM   / [H,L] ← CONTACT FIELD
016 117E 118803     LXI      D; DMAST3  / [D,E] ← SOURCE DATA
017 1181 CD0301     CALL     MOVSTR     / LOAD DATA
018 1184 21FEFC     LXI      H; DSPNUM   / [H,L] ← NUMERIC FIELD
019 1187 119303     LXI      D; DMAST4  / [D,E] ← SOURCE DATA
020 118A 3A7CFE     LDA      KSTATE     / A ← STATE VECTOR
021 118D F610       ORI      KCLEAR     / SET CLEAR NUMERIC FLAG
022 118F 327CFE     STA      KSTATE     / STORE VECTOR
023 1192 CD0301     CALL     MOVSTR     / LOAD DATA
024 1195 CD531F     CALL     KU03       / CLEAR SHIFT
025 1198 AF         CLA                     / A ← 0
026 1199 327FFE     STA      ASMC0N     / CLEAR CONTACT TYPE
027 119C C9        RET                      / EXIT
028                EJECT

001                SUBJOB  KEY FUNCTION : KF12 : DELETE
002                /
003                /***KEY FUNCTION : KF12 : DELETE
004                /
005                /***PARAMETERS:
006                /
007                /      SHIFT. EQ. 0 => DELETE NODE
008                /      SHIFT. EQ. 1 => DELETE NETWORK
009                /
010 119D CD281F     KF12,  CALL     KU01   / CHECK FOR RESET
011 11A0 CD491F     CALL     KU02   / CHECK FOR SHIFT
012 11A3 C21013     JNZ      KF1265 / BRANCH TO DELETE NETWORK
013                /
014                /***DELETE NODE
015                /
016 11A6 3A7EFE     LDA      CURACT     / A ← CURSOR
017 11A9 47         MOV      B; A       / B ← CURSOR
018 11AA E6F0       ANI      ROWMSK     / ISOLATE ROW
019 11AC FE80       CPI      ASMR0W     / CHECK FOR ASSEMBLY
020 11AE C2C811     JNZ      KF1210   / BRANCH IF NOT
021                /
022                /***CLEAR REFERENCE SLOT
023                /
024 11B1 CD4705     CALL     CUR100    / [H,L] ← CURSOR LOCATION
025 11B4 23        INX      H          / STEP BEYOND FIELD ATTRIBUTE
026 11B5 3E04       MVI      A; REFLN   / A ← COUNT
027 11B7 0620       MVI      B; ASCBLK  / B ← BLANK
028                /
029 11B9 70         KF1205, MOV      M; B     / CLEAR BYTE
030 11BA 114E00     LXI      D; ROWD+1 / [D,E] ← OFFSET
031 11BD 19        DAD      D          / BUMP TO NEXT LINE
032 11BE 70         MOV      M; B     / CLEAR BYTE
033 11BF 11B3FF     LXI      D; -ROWD  / [D,E] ← OFFSET
034 11C2 19        DAD      D          / BACK TO PREVIOUS LINE
035 11C3 3D        DCR      A         / DECREMENT COUNTER
036 11C4 C2B911     JNZ      KF1205   / LOOP UNTIL DONE
037 11C7 C9        RET                      / GO TO EXIT
038                EJECT

```

***** DELETE NETWORK NODE(S)**

```

002                /
003 11C8 3A7CFE     KF1210, LDA      KSTATE / A ← STATE VECTOR
004 11CB E608       ANI      KNET       / NETWORK ACTIVE?
005 11CD 11E427     LXI      D; MSGNET  / [D,E] ← MESSAGE ADDR
006 11D0 CADA13     JZ       KF12ER     / BRANCH ON ERROR
007                /
008                /      VERIFY THAT THE CURSOR IS ON A NODE
009                /      THAT WE CAN DELETE. THESE ARE:
010                /      NON-BLANK; TOP OF NODE MULTI-NODE ITEMS
011                /
012 11D3 3A80FE     LDA      CURCON    / GET CURSOR NODE TYPE
013 11D6 B7         TST                     / BLANK?
014 11D7 CAE911     JZ       KF12NV    / YES, "INVALID" ERROR EXIT
015                /
016                /      ON A REAL NODE; NOW IS IT THE TOP
017                /      NODE IF M-NODE ITEM?
018

```



```

019 11DA FE0D      CPI      NOCPRE / IS IT A SINGLE NODE ITEM?
020 11DC FAF011    JM       KF1212 / YES, ALL SET
021
022 11DF FE0F      CPI      NOCTR  / IS IT A PRESET/B-NODE?
023 11E1 FAF511    JM       KF1214 / YES, POSSIBLY GOOD
024
025 11E4 FE13      CPI      NOCON  / IS IT A CONVERT NODE?(SPECIAL)
026 11E6 CA1712    JZ      KF1216 / YES, SPECIAL PROCESS
027
028 /             / CURSOR IS NOT ON A VALID NODE!
029
030 KF12NV,
031 11E9 11411E    LXI     D,KF14M1/ PTR TO "INVALID"
032 11EC CD7E05    CALL    ERROR  / DISPLAY IT
033 11EF C9        RET     / DONE
034
035 /             / HERE FOR SINGLE NODE ITEMS;
036 /             / SET COUNT=1 AND GO
037
038 KF1212,
039 11F0 0E01      MVI     C,1    / SET NODE COUNT
040 11F2 C32F12    JMP     KF1220 / GO VERIFY POSITION
041 EJECT
042 /             / HERE WHEN PRESET/B-NODE TYPE.
043 /             / MAKE SURE WE ARE @ TOP OF CTR, TMR, CALC
044
045 KF1214,
046 11F5 0E02      MVI     C,2    / SET COUNT AT LEAST TO 2
047
048 /             / NOW LOOK AT THE NODE BELOW CURRENT; MAKE
049 /             / SURE IT IS EITHER PRESET/C (I. E. 2ND NODE OF
050 /             / CALC) OR CTR/TMR
051
052 11F7 3A7EFE    LDA     CURACT / GET CURRENT POSITION
053 11FA C610      ADI     :10    / STEP TO NEXT ROW
054 11FC CD4614    CALL    GETYPE / GET THE TYPE THERE
055
056 /             / CTR/TMR?
057
058 11FF FE0F      CPI     NOCTR  / TEST LOW END OF RANGE
059 1201 FAE911    JM     KF12NV  / LOWER, ERROR (NEVER HAPPEN)
060 1204 FE13      CPI     NOCON  / TEST HIGH END
061 1206 FA2F12    JM     KF1220 / IN RANGE, GOV VERIFY POS
062
063 /             / NOT CTR/TMR, IT BETTER BE THE C-NODE/PRESET
064
065 1209 0C        INR     C     / SET COUNT = 3
066
067 120A FE14      CPI     NOCON  / TEST LOW END
068 120C FAE911    JM     KF12NV  / ERROR
069 120F FE16      CPI     NOCALC / TEST HIGH END
070 1211 FA2F12    JM     KF1220 / OK; GO VER POS
071 1214 C3E911    JMP     KF12NV  / INVALID!
072 EJECT
073 /             / HERE FOR SPECIAL CONVERT NODE TEST.
074 /             / THE NODE BELOW MUST BE CONVERT, TOO!
075
076 KF1216,
077
078 1217 0E02      MVI     C,2    / SET COUNT = 2
079 /             / NOW LOOK AT NODE BELOW. IT MUST BE CONVERT
080
081 1219 3A7EFE    LDA     CURACT / GET CURRENT POS
082 121C C610      ADI     :10    / STEP TO NEXT ROW
083 121E 47        MOV     B,A    / SAVE IT
084 121F E6F0      ANI     ROWMSK / ISOLATE ROW
085 1221 FE80      CPI     MAXROW+1!@16 / DID WE GO > MAX?
086 1223 D2E911    JNC     KF12NV  / YES, ERROR
087
088 1226 78        MOV     A,B    / NOT OFF BOTTOM; GET TYPE
089 1227 CD4614    CALL    GETYPE / NOW THE TYPE BELOW IS IN A
090 122A FE13      CPI     NOCON  / IS IT ALSO CONVERT?
091 122C C2E911    JNZ     KF12NV  / NO, ERROR
092 /             / YES, FALL TO COMMON CODE
093 EJECT

```

HERE WHEN THE CURRENT NODE IS
ABLE TO BE DELETED!

```

003
004 /
005 / NOW, THE ITEM MUST BE THE LAST IN ITS
006 / COLUMN. IF IT IS AND IT IS IN THE TOP ROW,
007 / IT MUST BE THE LAST IN ITS ROW!
008 KF1220,
009 / STAT:
010 / C = ROW
011
012 122F CD0423 CALL KU11 / GET CURRENT ROW IN L. S NIBBLE
013 1232 81 ADD C / CALCULATE ROW+1 OF DELETABLE
014 1233 FE08 CPI MAXROW+1 / ARE WE DEALING WITH LAST ROW?
015 1235 D25712 JNC KF1225 / YES, THIS MUST BE THE LAST
016 / ITEM IN COLUMN
017
018 / NOT AT BOTTOM OF NETWORK,
019 / SEE WHAT'S BELOW
020
021 1238 CF NSWP / GET ROW BEYOND TO
022 / MOST SIG NIBBLE
023 1239 47 MOV B:A / SAVE IT
024 123A 3A7EFE LDA CURACT / GET PRESENT POSITION
025 123D E60F ANI COLMSK / SAVE COLUMN
026 123F B0 ORA B / CONCAT W/ TEST ROW
027 1240 CD4614 CALL GETYPE / GET THE NODE TYPE BELOW
028 1243 B7 TST / IS IT BLANK?
029 1244 CA5712 JZ KF1225 / YES, GO CHECK TOP ROW
030
031 /
032 / HERE FOR ERROR WHEN SOMETHING IS EITHER
033 / BELOW OR TO THE RIGHT IN TOP ROW.
034 / ERROR EXIT WITH "NOT LAST"
035 KF12NL,
036 1247 114E12 LXI D:KM12NL / PTR TO MSG
037 124A CD7E05 CALL ERROR / DISPLAY AND GO
038 124D C9 RET / X
039
040 124E 08 KM12NL, DB KM12X1
041 124F 4E4F5420 DA 'NOT LAST'
042 1253 4C415354
043 0008 KM12X1 = -KM12NL-1
EJECT

001 / WE HAVE THE LAST ITEM IN THE COL.
002 / IS IT THE TOP ROW?
003
004 KF1225,
005 1257 CD0423 CALL KU11 / GET ROW IN LS NIBBLE
006 125A FE01 CPI 1 / AT TOP ROW?
007 125C C27312 JNZ KF1230 / NO, OK TO DELETE
008
009 / IN TOP ROW, ARE WE IN LAST COL?
010
011 125F 3A7EFE LDA CURACT / GET POSITION
012 1262 47 MOV B:A / SAVE IT, TOO
013 1263 E60F ANI COLMSK / ISOLATE COL
014 1265 FE08 CPI MAXCOL / AT RIGHT RAIL?
015 1267 CA7312 JZ KF1230 / YES, OK TO DELETE
016
017 / NOT LAST COL, MAKE SURE IT IS LAST
018 / NODE IN ROW
019
020 126A 78 MOV A:B / REFETCH CURRENT CURSOR
021 126B 3C INR A / STEP ONE COL
022 126C CD4614 CALL GETYPE / GET THE NODE TYPE THERE
023 126F B7 TST / IS IT BLANK?
024 1270 C24712 JNZ KF12NL / NO, ERROR "NOT LAST"
025 / YES, FALL TO DELETE
026 EJECT

```

HERE TO ACTUALLY DO DELETE!

```

002
003 / FIRST, CALC CONTROLLER ADDR OF NODE(S)
004 / TO DELETE, AND SET UP I/O BUFFER
005

```

```

006 KF1230,
007 1273 CD5124 CALL KUZZ / GET PTR TO COLTAB
008 1276 E5 PUSH H / SAVE IT
009
010 1277 E7 GETHL / GET THE START ADDR OF THIS COL
011
012 1278 CD0423 CALL KU11 / GET THE CURRENT ROW
013 127B 3D DCR A / MAKE IT 0 REL
014 127C 87 ADD A / AND DOUBLE IT
015 127D 5F MOV E,A / SET IT AS LS BYTE
016 127E 1600 MVI D,0 / CLR MS BYTE
017
018 1280 19 DAD D / NOW H/L = ADDR OF NODE TO DELETE
019 1281 EB XCHG
020
021 1282 2193FE LXI H,CMDBUF+3 / GET PTR TO STORE ADDR
022 1285 EF MOVDE / STORE CNTRLR ADDR IN I/O BUFF
023
024 / NOW SEE IF THERE ARE ANY VERTICALS HERE AT ALL
025
026 1286 E1 POP H / GET COLTAB PTR BACK
027 1287 110400 LXI D,EOCHI / GET OFFSET TO E-O-C NODE
028 128A 19 DAD D / CALC PTR TO EOC
029 128B AF CLA / FOR TEST
030 128C BE CMP M / ANY VERTICALS?
031 128D C29612 JNZ KF1240 / YES, GO HANDLE BELOW
032
033 / YAY!, NO VERTICALS. JUST DELETE
034
035 1290 3E00 MVI A,CMDDEC / GET DEL @ EOC FUNCTION
036 1292 CD4F14 CALL DELTIO / DO DELETE AND UPDATE
037 1295 C9 RET / DONE
038 EJECT

```

HERE WHEN THERE ARE VERTICALS IN THE COLUMN

```

003
004 / NEED TO TEST AND HANDLE 3 CONDITIONS:
005
006 / 1-THERE ARE VERTICALS, BUT THIS DELETE
007 / DOES NOT TOUCH THEM
008 / 2-THERE ARE VERTICALS, AND THIS DELETE
009 / REMOVES ALL OF THEM
010 / 2-THERE ARE VERTICALS, AND THIS DELETE
011 / REMOVES SOME OF THEM
012
013 KF1240,
014
015 / CREATE A MASK BASED UPON COUNT
016 / AND CURRENT ROW WHICH WILL AND OUT
017 / THE VERTICALS BEING DELETED
018
019 1296 CD0423 CALL KU11 / GET ROW #
020 1299 57 MOV D,A / TO D FOR COUNT
021
022 129A 3E80 MVI A,80 / GET BIT FOR 1ST ROW W/DELETE
023
024 KF1242,
025 129C 0F RRC / SHIFT 1ST ROW BIT LEFT
026 129D 15 DCR D / COUNT DOWN # OF ROWS
027 129E C29C12 JNZ KF1242 / LOOP UNTIL 1ST ROW BIT
028 / IS IN POSITION
029
030 / NOW A-REG = POSITION FOR 1ST ROW TO BE DELETED.
031 / NEXT, PUT IN MORE BITS FOR > 1 NODE
032
033 12A1 5F MOV E,A / SAVE RESULT
034 12A2 57 MOV D,A / SET UP A TRAVELER BIT
035 12A3 C5 PUSH B / SAVE COUNT
036
037 12A4 0D DCR C / COUNT DOWN # OF NODES
038 12A5 CAB012 JZ KF1245 / DONE!, E HAS MASK
039
040 / > 1 NODE, SO CREATE ANOTHER BIT IN MASK
041

```



```

042 12A8 7A      MOV      A;D      / GET TRAVELING BIT
043 12A9 0F      RRC                      / SHIFT TO NEXT LOWER ROW POS
044 12AA 57      MOV      D;A      / SAVE TRAVELER
045
046 12AB B3      ORA      E          / CONCATENATE WITH MASK RESULT
047 12AC 5F      MOV      E;A      / SAVE RESULT
048 12AD C3A412  JMP      KF1244    / LOOP TIL MASK MADE
049      EJECT

001
002      /      HERE WHEN MASK IS BUILT. ONE'S COMPLEMENT
003      /      IT TO CREATE THE "AND" MASK
004
005      KF1245,
006 12B0 C1      POP      B          / GET COUNT BACK
007 12B1 7B      MOV      A;E      / GET MASK
008 12B2 2F      LMA                      / NOW A = FINAL MASK
009 12B3 47      MOV      B;A      / SAVE IT IN B
010
011      /
012      /      NOW LOOK AT END-OF-COLUMN NODE STORED
013      /      LOCALLY AND SEE WHAT, IF ANY, VERTICAL
014      /      BITS ARE CHANGING.
015 12B4 CD5124  CALL     KU22      / GET PTR TO COLTAB
016
017 12B7 110500  LXI      D;EOCLO  / OFFSET TO EOC VERT BITS
018 12BA 19      DAD      D          / SET H/L AS PTR TO THEM
019 12BB 7E      MOV      A;M      / GET THEM
020 12BC A0      ANA      B          / MASK OUT THE DELETED ONES
021
      \      NOW, THE BIG DECISION!
023
024 12BD CACA12  JZ       KF1250    / GO IF WE HAVE DELETED ALL!
025
      \      DID WE DELETE ANY?
027
028 12C0 BE      CMP      M          / SAME AS BEFORE?
029 12C1 C2D612 JNZ      KF1260    / NO, GOT RID OF SOME!
030
      \      HERE WHEN NO CHG TO VERTICALS!
032
033 12C4 3E60      MVI     A;CMDDEL/ DELETE NOT AT E-O-C
034
035 12C6 CD4F14  CALL     DELTIO   / DO DELETE AND UPDATE
036 12C9 C9      RET                      / DONE
037      EJECT

      \
      \      HERE WHEN THIS DELETE
      \      ALSO REMOVES ALL VERTICALS
003
004      KF1250,
005 12CA 0C      INR      C          / ADD 1 TO COUNT TO
006                      / REMOVE EOC NODE
007 12CB 2B      DCX      H          / STEP COLTAB PTR BACK TO
008                      / MS BYTE OF EOC NODE
009 12CC 110000  LXI      D;0        / CLEAR EOC NODE IN COLTAB
010 12CF EF      MOVDE                      / X
011
012 12D0 3E00      MVI     A;CMDDEC/ GET FUNCTION TO DELETE @ EOC
013 12D2 CD4F14  CALL     DELTIO   / DO IT
014 12D5 C9      RET                      / DONE
015      EJECT

      \
      \      HERE WHEN DELETING SOME VERTICALS
      \      BUT NOT ALL
018
019      KF1260,
020
021      /      THIS CASE IS THE TRICKIEST. WE NEED TO
022      /      WRITE THE END OF COL NODE W/MASK
023      /      TO REMOVE SOME VERTICAL BITS; THEN DELETE
024      /      NODE(S).
025      /      STAT.
026      /
027      /      H/L = PTR TO EOC BITS IN COLTAB (EOCLO)
028      /      A = THOSE BITS 'ANDED' BY MASK (BITS LEFT OVER)
029      /      B = FINAL MASK
029      /      C = COUNT

```

```

030
031 / WE MUST DO A LOT OF FANCY FOOTWORK TO SAVE
032 / VALUES AND WRITE VERTICALS. WATCH CLOSELY!
033
034 12D6 EB XCHG / SAVE EOC PTR IN D/E
035 12D7 2A93FE LHL D CMDBUF+3/ GET THE ADDR OF NODE(S)
036 / TO DELETE AND
037 12DA E5 PUSH H / SAVE ON STACK FOR LATER
038
039 12DB EB XCHG / RESTORE EOC PTR
040 12DC E5 PUSH H / SVE EOC PTR
041 12DD C9 PUSH B / SAVE MASK, COUNT
042 12DE F5 PUSH PSW / SAVE FINAL BITS
043
044 / BUILD I/O COMMAND FOR RE-WRITING EOC NODE
045
046 12DF 11FDFF LXI D,COLEHI-EOCLO / GET PTR BACK TO
047 / END OF COL ADDR
048 12E2 19 DAD D / X
049
050 12E3 E7 GETHL / GET ADDR FOR WRITE
051 12E4 EB XCHG / INTO D/E
052
053 12E5 2193FE LXI H,CMDBUF+3 / POINT TO I/O BUFFER
054
055 12E8 EF MOVDE / STORE ADDR FOR WRITE
056 EJECT
057 / BUILD DATA & MASK FOR WRITE
058
059 12E9 F1 POP PSW / GET FINAL BITS
060 12EA F5 PUSH PSW / X
061 12EB 5F MOV E,A / SET AS "DATALO"
062 12EC 1600 MVI D,0 / CLEAR "DATAHI"
063 12EE EF MOVDE / STORE DATA TO WRITE
064
065 12EF 1100FF LXI D,FF00 / GET MASK FOR WRITE
066 12F2 EF MOVDE / STORE IN CMDBUF
067 EJECT
068 / DO THE I/O!
069
070 12F3 110A21 LXI D,CMDWRT+CMDOZ!!:100+LENWRT / PARAMS
071 12F6 CD8125 CALL PID / DO WRITE!
072
073 12F9 C20B13 JNZ KF1264 / EXIT IF ERROR, NO UPDATE
074
075 / I/O OKAY, NOW....
076 / RELOAD ALL VALUES AND PTRS
077
078 12FC F1 POP PSW / GET FINAL BITS
079 12FD C1 POP B / GET MASK, COUNT
080 12FE D1 POP D / GET EOC PTR TO D/E
081 12FF E1 POP H / RELOAD DELETE ADDR
082
083 / HERE TO DELETE AND UPDATE
084
085 1300 2293FE SHLD CMDBUF+3/ RESTORE DELETE ADDR TO
086 / I/O BUFFER
087
088 / UPDATE BITS IN COLTAB
089
090 1303 EB XCHG / SET H/L = PTR TO EOC BITS
091 1304 77 MOV M,A / SET BITS!
092
093 / DO DELETE AND UPDATE THE WORLD!
094
095 1305 3E60 MVI A,CMDEL/ GET FUNCTION
096 1307 CD4F14 CALL DELTID / DO IT
097 130A C9 RET / DONE
098
099
100 / HERE WHEN ERROR, CLEAR STACK AND EXIT
101
102 KF1264,
103 130B E1 POP H / DISCARD 4 ITEMS
104 130C E1 POP H / X

```

```

105 130D E1      PUP      H      / X
106 130E E1      PUP      H      / X
107 130F C9      RET      / EXIT
108              EJECT

```

*** DELETE NETWORK

```

002 /
003 1310 11E427  KF1265. LXI      D:MSGNET      / [D,E] ← MESSAGE ADDR
004 1313 3A7CFE      LDA      KSTATE      / A ← STATE VECTOR
005 1316 E608        ANI      KNET         / NETWORK ACTIVE?
006 1318 CADA13      JZ       KF12ER      / NO. ERROR
007 /
008 /              DELETE NETWORK A COLUMN AT A TIME;
009 /              BEGINNING WITH RIGHT AND WORKING LEFT
010 /
011 131B 3E1B        MVI      A: 10+MAXCOL / SET ACTUAL CURSOR
012 131D 327EFE      STA      CURACT      / TO RIGHT SIDE
013 /
014 /              BEGIN LOOP AND DELETE BY COLUMN
015 /
016 KF1270.
017 /
018 /              GET PTR TO COLUMN BEING DELETED
019 /
020 1320 CD5124      CALL     KU22        / NOW H/L HAS PTR
021 1323 44          MOV      B:H         / SAVE IN B/C
022 1324 4D          MOV      L:L         / FOR LATER
023 /
024 1325 E7          GETHL           / GET START ADDR
025 1326 EB          XCHG          / TO D/E
026 1327 7A          MOV      A:D         / IS IT 0? IF SO, DO
027 1328 B3          ORA      E         / NOTHING ON THIS COL.
028 1329 CA5A13      JZ       KF1275     / GO DO NEXT COLUMN!
029 /
030 /              SET UP ADDR FOR DELETE
031 /
032 132C 2193FE      LXI      H:CMDBUF+3 / PTR TO I/O BUFF
033 132F EF          MOVDE          / SET IT UP
034 /
035 /              FIGURE OUT HOW MANY NODES TO DELETE (1-8)
036 /              CALC: (END-START)/2+1
037 /
038 1330 7A          MOV      A:D         / TWO'S COMPLEMENT START
039 1331 2F          CMA          / AND PUT IT IN D/E
040 1332 57          MOV      D:A         / X
041 1333 7B          MOV      A:E         / X
042 1334 2F          CMA          / X
043 1335 5F          MOV      E:A         / X
044 1336 13          INX      D         / DONE
045 EJECT
046 /              GET END ADDR AND SUBTRACT IT
047 /
048 1337 210200      LXI      H:COLEHI/ GET OFFSET TO END ADR
049 133A 09          DAD      B         / NOW H/L=PTR TO END
050 133B E7          GETHL           / GET END
051 133C 19          DAD      D         / END-START
052 /
053 /              NOW DIVIDE RESULTS BY 2 AND ADD 1
054 /              FOR FINAL NODE COUNT
055 /
056 133D 7D          MOV      A:L         / GET L. S. RESULT
057 /              / (MS CAN BE IGNORED)
058 133E 0F          RRC          / DIVIDE BY 2
059 133F E60F        ANI      :F         / PURIFY IT
060 1341 3C          INR      A         / NOW A-REG = NODE COUNT!
061 /
062 /              SET UP B/C FOR USAGE DECR IF GOOD I/O.
063 /              B/C MUST = -(BYTE COUNT)
064 /
065 1342 F5          PUSH     PSW        / SAVE COUNT
066 /
067 1343 07          RLC          / *2 FOR BYTE COUNT
068 1344 2F          CMA          / TWO'S COMP IT
069 1345 3C          INR      A         / X
070 1346 4F          MOV      C:A         / SET L. S. BYTE
071 1347 06FF        MVI      B:FF       / SET M. S. BYTE

```



```

072 1349 F1      POP      PSW      / RESTORE NODE COUNT
073 134A C5      PUSH     B        / SAVE -BYTE COUNT
074
075              / NOW BUILD I/O COMMAND
076
077 134B F660    ORI      CMDDDEL / CREATE FUNCTION
078 134D 57      MOV     D/A      / TO D
079 134E 1E06    MVI     E,LENDEL / SET LENGTH FOR CALL
080 1350 CD8125  CALL    P10      / DO DELETE!
081 1353 C1      POP     B        / REFETCH -BYTE COUNT
082
083 1354 C27813  JNZ     KF1295   / ERROR ON DELETE; HANDLE BELOW
084
085              / UPDATE USAGE COUNT
086
087 1357 CDCB23  CALL    KU18     / DONE
088  EJECT

001              / OKAY, COLUMN IS GONE. STEP TO NEXT
002              / AND CHECK IF DONE
003
004              / KF1275,
005 135A 3A7EFE  LDA     CURACT   / GET CURRENT COLUMN
006 135D 3D      DCR     A        / -1 FROM COL
007 135E FE10    CPI     10      / ARE WE DONE?
008 1360 CA6913  JZ      KF1280   / YES, GET RID OF S. O. N.
009
010              / NOT DONE; SET UP AND LOOP
011
012 1363 327EFE  STA     CURACT   / SET TO NEXT COL AT LEFT
013 1366 C32013  JMP     KF1270   / LOOP TIL DONE
014
015              / HERE WHEN ALL BUT S. O. N. NODE DELETED
016
017              / KF1280,
018 1369 218CFE  LXI    H,ADRSON / POINT TO S. O. N
019 136C E7      GETHL           / GET IT
020 136D EB      XCHG          / TO D/E
021 136E 2193FE  LXI    H,CMDBUF+3 / PTR TO BUFF
022 1371 EF      MOVDE          / SET IT UP
023
024              / DELETE THE SON
025
026 1372 110661  LXI    D,CMDDDEL+CMDO2!:100+LENDEL / PARAMS
027 1375 CD8125  CALL    P10      / DO IT
028  EJECT

001              / ALL DONE; NOW GET NEXT NETWORK ON SCREEN
002
003              / ALSO HERE FOR ERROR.
004              / WE GET THE SAME NETWORK AGAIN AND REDISPLAY
005              / IT TO SHOW USER WHAT WAS NOT DELETED
006
007              / KF1295,
008 1378 CD8A1F  CALL    KU06     / CLEAR SCREEN
009 137B CD531F  CALL    KU03     / CLEAR SHIFT
010
011              / SEE WHAT WE HAVE AT THE OLD "START-OF-NET" ADDR:
012              / -IF WE HAVE AN "END-OF-LOGIC" NODE, WE DELETED
013              / LAST NETWORK IN DATABASE, SO WE WILL GET
014              / THE PREVIOUS NETWORK.
015              / -IF NOT, WE WILL GET THE NEXT NETWORK.
016              / -IF WE DELETED THE LAST AND ONLY NETWORK,
017              / "GET PREV" WILL GIVE "START OF LOGIC" MSG.
018
019 137E 218CFE  LXI    H,ADRSON / POINT TO OUR OLD START
020 1381 E7      GETHL           / GET ADDR
021 1382 F5      PUSH    H        / SAVE IT
022 1383 EB      XCHG          / TO D/E
023 1384 2193FE  LXI    H,CMDBUF+3 / POINT TO I/O
024 1387 EF      MOVDE          / SET UP ADDR TO READ
025 1388 110611  LXI    D,CHDRED+CMDO2!:100+LENRED
026 138B CD8125  CALL    P10      / READ NODE!
027
028 138E E1      POP     H        / IN CASE OF ERROR
029 138F C0      RND           / ERROR, QUIT NOW

```

```

030 1390 E5      PUSH    H          / GOOD, RESAVE ADRSON
031
032             /      SEE WHAT NODE IS AT S. O. N. ADDR
033
034 1391 3A0EFE   LDA     RSPBUF+3/  GET NODE TYPE
035 1394 FE04     CPI     NDEUL!4 /  IS IT END-OF-LOGIC?
036 1396 C2BF13   JNZ    KF1297 /    NO, GO LOOK FOR NEXT
037
038             /      WE DELETED THE NETWORK AT END OF DATA.
039             /      SET UP TO LOOK FOR PREVIOUS.
040             /      NOTE: IF WE DELETED THE LAST AND ONLY,
041             /      CLEAR NETWORK ACTIVE AND STEP #
042
043 1399 F1       POP     H          / GET SON
044 139A 01FFFF   LXI    B,-1      / GET STEP FOR GETNET
045
046 139D 110200   LXI    D,ADRUSE/  SEE IF WE ARE AT BEGINNING
047 13A0 F7       DCRH   /          OF USER LOGIC.
048 13A1 C20B13   JNZ    KF1298 /    NO, GO GET PREV
049
050             /      YES, CLEAR NET ACTIVE AND STEP #
051
052 13A4 C5       PUSH   B          / SAVE REGS
053 13A5 E5       PUSH   H          / X
054
055             /      CLEAR STEP # AND DISPLAY
056
057 13A6 1118FD   LXI    D,DSPSTP/  PTR TO DISPLAY AREA
058 13A9 210000   LXI    H,0       / CLEAR VALUE
059 13AC 228AFE   SHLD  STPNUM /  CLEAR COUNTER
060 13AF CDC201   CALL  BNBCD4 /  CLEAR DISPLAY
061
062 13B2 3A70FE   LDA    KSTATE /  CLEAR "NETWORK ACTIVE"
063 13B5 E6F7     ANI    -KNET-1 / X
064 13B7 3270FE   STA    KSTATE / X
065
066 13BA E1       POP     H          / RESTORE SON
067 13BB C1       POP     B          / RESTORE STEP
068 13BC C30B13   JMF    KF1298 /  GO GET PREV, IT WILL
069             /      "START OF LOGIC" ON DISPLAY
070
071             /      EJECT
072             /      HERE TO GET NEXT NETWORK
073             /      (NOT LAST DELETED)
074
075             /      KF1297,
076 13BF 01FFFF   LXI    B,-1      / DECR STEP # CAUSE 'GETNET'
077 13C2 CD2121   CALL  KU08 /    INCRS IT!
078
079 13C5 E1       POP     H          / GET SON ADDR BACK
080 13C6 2B       DCX    H          / STEP BACK 2 BECAUSE 'GETNET'
081 13C7 2B       DCX    H          / STEPS FWD 2
082 13C8 010100   LXI    B,1      / STEPPER
083
084             /      NOW CLEAR OLD START/END ADDR
085             /      KF1298,
086 13CB E5       PUSH   H          / SAVE ADDR FOR 'GETNET'
087
088 13CC 210000   LXI    H,0       / SET CLEAR
089 13CF 2280FE   SHLD  ADRSON /  CLEAR START
090 13D2 228EFE   SHLD  ADREON /  CLEAR END
091
092 13D5 E1       POP     H          / RESTORE ADDR
093
094             /      GO GET NETWORK FROM 484! (FINALLY!)
095
096 13D6 CD980C   CALL  GETNET /  DONE
097 13D9 C9       RET     / EXIT
098
099
100
101
102
103
104             /      HERE FOR BAD DELETE,
105             /      TAKE ERROR EXIT
106             /      KF12ER,
107 13DA CD7E05   CALL  ERROR /  DISPLAY MSG
108 13DD C9       RET
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

```

```

001      /***SUBROUTINE TO CLEAR NODES AND UPDATE COLUMN TABLE
002      /
003 13DE 05      K12SUB, PUSH      B      / SAVE COUNT
004 13DF 79      MOV      A,C      / A ← COUNT
005 13E0 81      ADD      C      / DOUBLE IT FOR
006      / SCREEN ROW CNT
007 13E1 4F      MOV      C,A      / AND STORE IT BACK
008 13E2 3A7EFE  LDA      CURACT      / CURRENT CURSOR POS
009 13E5 47      MOV      B,A      / SET B FOR CALL
010 13E6 CD4705  CALL     CUR100      / GET DISPLAY POINTERS
011
012      / FIRST, DELETE ALL BUT 1 CHAR OF
013      / THE 1ST ROW AND SEE IF THERE IS
014      / A VERTICAL UP CHAR AS LAST. IF SO,
015      / SET IT TO BE UP ONLY.
016
017 13E9 1605      MVI     D,DSPNOD-2 / CLEAR ALL BUT LAST
018 13EB CD1B03  CALL     ROWN20      / DONE
019
020      / NOW H/L IS POINTING TO LAST CHAR IN ROW
021
022 13EE 7E      MOV     A,M      / GET CHAR THERE
023 13EF FE08      CPI     CA1111 / VERT UP/DN?
024 13F1 CAFE13  JZ      K12S02 / YES, GO FIX FOR VERT UP
025 13F4 FEDC      CPI     CA1110 / VERT UP?
026 13F6 CAFE13  JZ      K12S02 / YES, GO FIX VERT UP
027
028      / NO VERT UP, SO BLANK IT
029
030 13F9 3620      MVI     M,ASCBLK / DONE
031 13FB C30014  JMP     K12S04 / GO JOIN REG
032
033      / HERE TO SET UP VERT ONLY
034
035      K12S02,
036 13FE 36DC      MVI     M,CA1110 / SET
037
038      / DONE WITH ROW; STEP H/L ONCE TO
039      / KEEP IN SYNC WITH "ROWN20"
040
041      K12S04,
042 1400 23      INX     H      / SET
043 1401 0D      DCR     C      / ACCOUNT FOR 1ST ROW DONE
044
045 1403 114900  LXI     D,ROWB-DSPNOD / STEP TO NEXT ROW
046 1405 19      DAD     D      / DONE
047      EJECT
048
049      / NOW LOOP AND BLANK REST OF NODES
050
051 1406 1606      K12S10, MVI     D,DSPNOD-1 / D ← COUNT
052
053      / IF DASHED LINE, SKIP IT AND QUIT
054
055 1408 23      INX     H      / TO 1ST CHAR
056 1409 7E      MOV     A,M      / GET CHAR
057 140A EAFE      ANI     -CATHI-1 / REMOVE HILITE BIT
058 140C FE72      CPI     ASCD3H / DASH?
059 140E CA2B14  JZ      K12S15 / YES, ALL DONE
060 1411 2B      DCX     H      / NO, RESET FOR CLEAR
061
062 1412 CD1B03  CALL     ROWN20      / CLEAR NODE
063 1415 114900  LXI     D,ROWB-DSPNOD / ED,EJ ← OFFSET
064 1418 19      DAD     D      / BUMP TO NEXT ROW
065 1419 0D      DCR     C      / DECREMENT COUNTER
066 141A D20614  JNZ     K12S10      / LOOP UNTIL DONE
067
068      / NOW FIX LAST VERT:
069
070 141D 110600  LXI     D,DSPNOD-1 / GET STEP TO LAST VERT
071 1420 19      DAD     D      / NOW H/L POINTS TO LAST
072
073      / IF DASH, DON'T CLEAR
074
075 1421 7E      MOV     A,M      / GET CHAR
076 1422 EAFE      ANI     -CATHI-1 / REMOVE HILITE

```



```

077 1424 FE72
078 1426 CA2B14
079 1429 3620
080
081
082 142B C1
083 142C C5
084
085 142D 3A7EFE
086 1430 47
087
088
089
090
091 1431 CDB123
092 1434 3600
093
094 1436 78
095 1437 C610
096 1439 47
097 143A 0D
098 143B C23114
099
100
101
102 143E CD5124
103 1441 C1
104 1442 CD0025
105 1445 C9
106

001
002
003
004
005
006
007
008
009
010
011
012
013
014
015 1446 D5
016 1447 E5
017
018 1448 CDB123
019
020 144B 7E
021
022 144C E1
023 144D D1
024 144E C9
025

001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020

001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020

```

```

CPI      ASCDSH / DASH?
JZ       K12S15 / YES, DONE
MVI      M:ASCBLK/ NO, CLEAR IT
EJECT

K12S15,
POP      B / GET COUNT
PUSH     B / STACK IT AGAIN

LDA      CURACT / GET CURRENT POS AGAIN
MOV      B:A / SET B FOR LOOP

/ LOOP CLEAR "MATROW" NODE TYPES

K12S20,
CALL     KU17A / GET PTR TO MATROW
MVI      M:0 / CLEAR TYPE BYTE

MOV      A:B / GET POSITION
ADI      :10 / STEP TO NEXT ROW
MOV      B:A / SAVE IT
DCR      C / COUNT DOWN # OF NODES
JNZ      K12S20 / AND LOOP TIL CLEARED

/ NOW UPDATE "COLTAB"

CALL     KU22 / GET COLTAB PTR
POP      B / GET COUNTER
CALL     COLDEC / FIX IT
RET
EJECT

SUBJOB GETYPE = GET NODE TYPE FROM 'MATROW'

/ GETYPE IS A SUBR TO GET THE NODE TYPE BASED ON A
/ ROW, COL INDEX
/
/ *ENTRY
/ A = ROW, COL
/
CALL     GETYPE
/
/ *EXIT
/ A = NODE TYPE
/

GETYPE,
PUSH     D / SAVE
PUSH     H / X

CALL     KU17A / GET PTR TO MATROW

MOV      A:H / GET NODE TYPE @ PTR

POP      H / RESTORE AND EXIT
POP      D / X
RET      / X
EJECT

SUBJOB DELTIO = DELETE I/O AND UPDATE SUBR

/ DELTIO IS A SUBR TO DO THE I/O WHICH DELETES
/ NODES. THEN, IF NO ERROR, IT DOES THE UPDATE ON:
/ SCREEN
/ 'MATROW' (NODE MATRIX TABLE)
/ 'COLTAB' (COLUMN ADDR TABLE)
/
/ *ENTRY
/ "CMDBUF+3" ALREADY HAS THE ADDRESS
/ C = NODE COUNT (1,2,3)
/ A = BASIC COMMAND, OR END-OF-COL COMMAND
/ (60 OR C0)
/
CALL     DELTIO
/
/ *EXIT
/ A = ?
/ C = NODE COUNT
/

```

```

021 / EXIT IS SAME WHETHER I/O IS GOOD OR BAD.
022 / UPDATES ARE SKIPPED IF I/O BAD.
023 /
024 DELT10,
025 144F E5 PUSH H / SAVE ALL
026 1450 D5 PUSH D / X
027 1451 C5 PUSH B / X
028
029 / CREATE I/O COMMAND
030
031 1452 B1 ORA C / A NOW HAS FINAL FUNCTION
032 1453 57 MOV D, A / SET D= DELETE FUNCTION
033
034 / SPECIAL TEST, IF CURSOR IS ON TOP ROW,
035 / THEN THIS IS THE LAST NODE(S) IN COLUMN.
036 / SET THE I/O FUNCTION TO REGULAR DELETE
037 / (NOT DELETE AT E-O-C) SO WE DON'T SET
038 / EOC BIT ON PREVIOUS COLUMN OR
039 / S. O. N. NODE
040
041 1454 CD0423 CALL K011 / GET CURRENT ROW
042 1457 FE01 CPI :01 / TOP ROW?
043 1459 C26014 JNZ DELT10 / NO, OK
044
045 145C 3E60 MVI A, CMDDEL / REGULAR DELETE
046 145E B1 ORA C / SET FINAL FUNCTION
047 145F 57 MOV D, A / TO D
048
049 / HERE FOR I/O
050
051 DELT10,
052 1460 1E06 MVI E, LENDEC / SET E= LENGTH OF COMMAND
053 1462 CD8125 CALL PIQ / DO IT
054
055 / IF I/O ERROR, SKIP UPDATE
056
057 1465 C27414 JNZ DELTEX / BAD I/O, SKIP
058
059 / UPDATE SCREEN, MATROW, COLTAB, USAGE
060
061 1468 C1 POP B / RELOAD COUNT
062 1469 C5 PUSH B / X
063
064 146A CDDE13 CALL K12SUB / UPDATES DONE
065
066 146D AF CLA / CLEAR CURRENT CURSOR NODE
067 146E 3280FE STA CURCON / TYPE
068
069 / ERASE DASHES IF THE DELETE WAS OF A COIL
070
071 1471 CD7814 CALL DELDASH / DO IT
072
073 / EXIT
074
075 DELTEX,
076 1474 C1 POP B / RESTORE ALL
077 1475 D1 POP D / X
078 1476 E1 POP H / X
079 1477 C9 RET
080 EJECT

001 SUBJOB DELDASH = DELETE DASHES IF COIL EXTENSION
002
003 / DELDASH IS A SUBR WHICH DECIDES IF WE ARE REMOVING
004 / A COIL EXTENSION. IF SO, IT BLANKS THE DASHES ON
005 / THE SCREEN AND BACKS THE CURSOR UP TO THE TRUE
006 / CURACT LOCATION.
007 /
008 / *ENTRY
009 / CURACT AND CURDSP ARE USED
010 /
011 / CALL DELDASH
012 /
013 / *EXIT
014 / IF CURACT=CURDSP, NO ACTION
015 / IF NOT, DASHES DELETED AND CURSOR AND CURDSP

```

```

016 / ARE SET TO CURACT
017 /
018 DELD3H,
019 1478 C5 PUSH B / SAVE ALL
020 1479 D5 PUSH D / X
021 147A E5 PUSH H / X
022 147B F5 PUSH PSW / X
023
024 / SEE IF WE HAVE ANY WORK TO DO; DOES 'CURACT' =
025 / 'CURDSP'?
026
027 147C 3A7DFE LDA CURDSP / GET DISPLAY POSITION
028 147F 47 MOV B,A / . TO B FOR COMPARE AND POSSIBLE M
VE
029
030 1480 3A7EFE LDA CURACT / GET TRUE POSITION
031 1483 B8 CMP B / SAME?
032 1484 CACE14 JZ DELDEX / YES, EXIT
033
034 / WE HAVE DASHES! BACK UP CURSOR AND GET
035 / RID OF THEM!
036
037 1487 4F MOV C,A / SET FOR CURSOR MOVE
038 1488 327DFE STA CURDSP / ALSO UPDATE NEW DISPLAY POS
039
040 148B CD2B05 CALL CURSOR / MOVE IT
041
042 / NOW LOOP ACROSS THE ROW AND DELETE DASHES
043 / AND COIL
044
045 DELD10
046 148E 41 MOV B,C / GET CURRENT LOCATION
047 148F CD4705 CALL CUR100 / GET PTR TO SCREEN @ TRUE NODE
048 1492 23 INX H / STEP PAST ATTRIBUTE
049
050 1493 1606 MVI D,DSPNOD-1 / SET COUNTER
051 1495 1E20 MVI E,ASCBLK / SET CLEAR CHAR
052
053 / LOOP AND CLEAR THE NODE OF DASHES
054
055 DELD20,
056 1497 7E MOV A,M / GET CHAR THERE NOW
057 1498 E6FE ANI -1-CATHI / ISOLATE CHAR W/O HIGHLIGHT
058 149A FE72 CPI ASCDSH / IS IT A DASH?
059 149C C2A014 JNZ DELD25 / NO, LEAVE WHAT'S THERE, THERE
060 149F 73 MOV M,E / YES, BLANK IT
061 DELD25,
062
063 / NOW, IF CHAR ABOVE IS A VERT SHORT,
064 / REPLACE BLANK WITH "HOR AND VERT UP"
065 / CHAR TO MAKE VERTICAL COMPLETE
066
067 14A0 D5 PUSH D / SAVE
068 14A1 E5 PUSH H / SAVE PRESENT LOC
069
070 14A2 11B0FF LXI D,-ROWB / STEP TO PREV ROW
071 14A5 19 DAD D / NOW H/L POINTS THERE
072 14A6 7E MOV A,M / GET CHAR
073 14A7 E6FE ANI -1-CATHI / GET RID OF HILITE BIT
074 14A9 FEE4 CPI CA0011 / IS THERE A VERT SHORT?
075 14AB E1 POP H / (RESTORE IN EITHER CASE)
076 14AC D1 POP D / (DITTO)
077 14AD C2B214 JNZ DELD30 / NO, GO ON
078 14B0 36DC MVI M,CA1110 / YES, SET HOR AND VERT UP
079 DELD30,
080 14B2 23 INX H / STEP TO NEXT CHAR
081 14B3 15 DCR D / COUNT DOWN
082 14B4 C29714 JNZ DELD20 / NOT DONE WITH NODE, LOOP
083
084 / DONE WITH NODE, STEP AND CHECK FOR DONE
085
086 14B7 0C INR C / STEP TO NEXT COL. POS
087 14B8 79 MOV A,C / GET ROW, COL
088 14B9 E60F ANI COLMSK / ISOLATE COL
089 14BB FE0B CPI MAXCOL / AT RIGHT?
090 14BD C28E14 JNZ DELD10 / NO, LOOP TIL DONE
091 EJECT

```



```

092 / NOW GET RID OF COIL ITSELF
093
094 14C0 1606 MVI D, DSPNOD-1 / COUNT TO CLEAR
095 14C2 CD1B03 CALL RDOWN20 / CLEAR 1ST ROW
096 14C5 114900 LXI D, ROWB-DSPNOD / OFFSET TO NEXT
097 14C8 19 DAD D / SET H/L TO NEXT ROW
098 14C9 1606 MVI D, DSPNOD-1 / SET COUNT AGAIN
099 14CB CD1B03 CALL RDOWN20 / CLEAR 2ND ROW
100
101 / EXIT
102
103 DELDEX,
104 14CE F1 POP PSW / RESTORE ALL
105 14CF E1 POP H / X
106 14D0 D1 POP D / X
107 14D1 D1 POP B / X
108 14D2 C9 RET / DONE
109 EJECT

001 SUBJOB KEY FUNCTION : KF13 : START NEXT
002 /
003 /***KEY FUNCTION : KF13 : START NEXT
004 /
005 14D3 CD281F KF13, CALL KU01 / CHECK FOR RESET
006 14D6 CD491F CALL KU02 / CHECK FOR SHIFT
007 14D9 CAE214 JZ KF1305 / BRANCH ON NO SHIFT
008 14DC CD791F CALL KU05 / DISPLAY ERROR
009 14DF C32715 JMP KF13X / GO TO EXIT
010 /
011 14E2 218EFE KF1305, LXI H, ADREON / [H, L] <- ADDR
012 14E5 46 MOV B, M / B <- ADDRHI
013 14E6 23 INX H / BUMP POINTER
014 14E7 4F MOV C, M / C <- ADDRLO
015 14E8 03 INX B / BUMP ADDRESS
016 14E9 03 INX B / TO NEXT NODE
017 14EA 2193FE LXI H, CMDBUF+3 / [H, L] <- POINTER
018 14ED D7 MOVBC / STORE ADDRESS
019 14EE 010000 LXI B, NOSON!: 400 / [B, C] <- START NODE
020 14F1 D7 MOVBC / STORE NODE
021 /
022 14F2 110851 LXI D, CMDINS+CMD02!: 100+LENINS / SET FARMS
023 14F5 CD8125 CALL P10 / DO INSERT
024 /
025 14F8 C22715 JNZ KF13X / BRANCH ON ERROR
026 14FB CD8A1F CALL KU06 / INITIALIZE LOGIC DATA
027 /
028 14FE 010100 LXI B, 1 / [B, C] <- INCREMENT
029 1501 CD2121 CALL KU08 / INCREMENT STEP NUMBER
030 /
031 1504 112815 LXI D, KF13MS / [D, E] <- MESSAGE ADDR
032 1507 CD681F CALL KU04 / DISPLAY MESSAGE
033 /
034 150A CD8C23 CALL KU16 / DISPLAY POWER RAIL
035 EJECT
036 150D 3E18 MVI A, KCLEAR+KNET / A <- NEW STATE VECTOR
037 150F 327DFE STA KSTATE / LOAD NEW STATE VECTOR
038 1512 218EFE LXI H, ADREON / SET UP POINTER
039 1515 E7 GETHL / [H, L] <- END OF LAST NET
040 1516 23 INX H / BUMP POINTER
041 1517 23 INX H / TO NEXT NODE ADDR
042 1518 EB XCHG / SWAP
043 1519 218CFE LXI H, ADRSON / [H, L] <- POINTER
044 151C EF MOVDE / STORE NEW START ADDR
045 151D 218EFE LXI H, ADREON / [H, L] <- POINTER
046 1520 EF MOVDE / STORE NEW END ADDR
047 1521 010200 LXI B, 2 / INCREMENT
048 1524 CDCB23 CALL KU13 / MEMORY USAGE
049 /
050 1527 C9 KF13X, RET / EXIT
051 /
052 /***MESSAGE
053 /
054 1528 05 KF13MS, DB KF13MX
055 1529 53544152 DB START
056 152D 54 KF13NX- / MESSAGE LENGTH
057 EJECT

```

```

001          SUBJOB KEY FUNCTION : KF14 : ENTER
002          /
003          /***KEY FUNCTION : KF14: ENTER
004          /
005 152E CD281F      KF14,  CALL    KU01      / CHECK FOR RESET
006 1531 CD491F      CALL    KU02      / CHECK FOR SHIFT
007 1534 CA3D15      JZ     K14005     / BRANCH ON NO SHIFT
008 1537 CD791F      CALL    KU05      / DISPLAY ERROR
009 153A C3011B      JMP     KF14X     / GO TO EXIT
010          /
011 153D 3A7CFE      K14005, LDA    KSTATE  / A ← STATE VECTOR
012 1540 E608        ANI    KNET      / CHECK FOR NETWORK FLAG
013 1542 C24B15      JNZ    K14010     / BRANCH ON IT
014 1545 11E427      LXI    D,MSGNET  / [D,E] ← MESSAGE ADDR
015 1548 C3FE1A      JMP     KF14ER   / GO TO ERROR CODE
016          /
017 154B 3A7EFE      K14010, LDA    CURACT  / A ← CURSOR
018 154E 47          MOV    B:A      / B ← CURSOR
019 154F E6F0        ANI    ROWMSK   / ISOLATE ROW
020 1551 FE80        CPI    ASMROW   / CHECK FOR ASSEMBLY AREA
021 1553 CABA1A      JZ     K14900   / BRANCH IF ASSEMBLY ARE
022          EJECT
023          /
024          /
025          /
026          /
027          /
028 1556 3A7FFE      LDA    ASMCON   / GET NEW TYPE
029 1559 CD7A25      CALL   ISCOIL  / IS IT A COIL?
030 155C DA8D15      JC     K14014  / NO, GO DO OTHER VALID
031          /
032          /
033          /
034 155F 3A05FD      LDA    DSPVER+ROWD / GET ASSEMBLY VERT
035 1562 FEE4        CPI    CA0011   / SHORT?
036 1564 CAA815      JZ     KF14NV   / YES, ERROR
037          /
038          /
039          /
040          /
041          /
042 1567 3A7EFE      LDA    CURACT   / GET TRUE POS
043 156A CD021B      CALL   KF14Z   / BLANK HERE?
044 156D C2AD17      JNZ    K14135  / NO, GO TO REPLACE
045          /
046          /
047          /
048          /
049 1570 3A7EFE      LDA    CURACT   / START HERE
050 1573 47          MOV    B:A     / SET B AS HOLDER
051          /
052          /
053          /
054          /
055          /
056 1574 78          K14012, MOV    A:B     / GET CURRENT LOC
057 1575 CD021B      CALL   KF14Z   / BLANK HERE?
058 1578 C28715      JNZ    KF14NR  / NO, ERROR
059          EJECT
060          /
061          /
062 157B 78          MOV    A:B     / GET POSITION
063 157C E60F        ANI    COLMSK  / ISOLATE COLUMN
064 157E FE0B        CPI    MAXCOL  / AT RIGHT-SIDE?
065 1580 CAAE15      JZ     K14015  / YES, ALL DONE AND GOOD
066 1583 04          INR    B      / NO, GO BACK AND CHECK
067 1584 C37415      JMP    K14012  / NEXT COLUMN IN ROW
068          /
069          /
070          /
071          /
072          /
073 1587 11671B      KF14NR, LXI    D,KF14M5 / GET PTR TO ERROR
074 158A C3FE1A      JMP    KF14ER  / DISPLAY AND EXIT
075          EJECT
076          /
          IF IN BOTTOM ROW OR RIGHT-HAND COLUMN;

```



```

077 / NO VERTICAL SHORT ALLOWED
078
079 K14014,
080 158D 3A7DFE LDA CURDSP / GET CURRENT CURSOR LOC
081 1590 47 MOV B:A / SAVE IT, TOO
082 1591 E6F0 ANI ROWMSK / ISOLATE ROW
083 1593 FE70 CPI MAXROW!@16 / ARE WE IN BOTTOM ROW?
084 1595 CAA015 JZ KF14VT / YES, GO CHECK VERTS
085
086 / NOT IN BOTTOM ROW; SEE IF RIGHT-HAND COL
087
088 1598 78 MOV A:B / GET CURENT LOC AGAIN
089 1599 E60F ANI COLMSK / ISOLATE COLUMN
090 159B FE0B CPI MAXCOL / ARE WE IN RIGHT COL?
091 159D C2AE15 JNZ K14015 / NO, GO AHEAD
092
093 / HERE TO MAKE SURE NO VERTICAL SHORT
094
095 KF14V1,
096 15A0 3A05FD LDA DSPVER+ROWD / GET ASSEMBLY VERT
097 15A3 FEE4 CPI CA0011 / IS IT SHORT?
098 15A5 C2AE15 JNZ K14015 / NO, GO AHEAD
099
100 / ERROR! NO VERT
101
102 KF14NV,
103 15A8 115F1B LXI D:KF14M4/ GET PTR
104 15AB C3FE1A JMP KF14ER / GO ERROR EXIT
105 EJECT

```

\ HERE TO ENTER INTO A NETWORK!

```

002
003 15AE 3A7EFE K14015, LDA CURACT / A ← CURSOR
004 15B1 CD021B CALL KF14Z / CHECK FOR BLANK
005 15B4 C2AD17 JNZ K14135 / BRANCH FOR REPLACEMENT
006

```

\ HERE TO ENTER NEW NODE!

```

008
009 15B7 11491B LXI D:KF14M2 / [D,E] ← ERROR MESSAGE
010 15BA 3A01FD LDA DSPNUM+3 / CHECK FOR ALL FIELDS
011 15BD FE1D CPI ASCNBK / DEFINED
012 15BF CAFE1A JZ KF14ER / BRANCH ON NUMERIC ERROR
013 15C2 3AB3FC LDA DSPCON / A ← CONTACT
014 15C5 FE1F CPI ASCCBK / CHECK FOR BLANK
015 15C7 CAFE1A JZ KF14ER / BRANCH ON IT
016 15CA 3AB3FC LDA DSPVER / A ← VERTICAL
017 15CD FE1E CPI ASCVBK / CHECK FOR BLANK
018 15CF CAFE1A JZ KF14ER / BRANCH ON IT
019
020 15D2 3A7EFE LDA CURACT / A ← CURSOR
021 15D5 FE11 CPI .11 / CHECK FOR HOME POSITIO
022 15D7 CAF215 JZ K14025 / BRANCH AT HOME
023 15DA 47 MOV B:A / BACK UP CURSOR
024 15DB E6F0 ANI ROWMSK / ISOLATE ROW
025 15DD FE10 CPI .10 / CHECK FOR TOP ROW
026 15DF 78 MOV A:B / RESTORE CURSOR
027 15E0 C2E715 JNZ K14016 / BRANCH IF NOT TOP
028 15E3 3D DCR A / CHECK TO LEFT ON TOP ROW
029 15E4 C3E915 JMP K14020 / CONTINUE
030
031 15E7 D610 K14016, SUI .10 / CHECK ABOVE NODE
032
033 15E9 CD021B K14020, CALL KF14Z / CHECK FOR BLANK
034 15EC 11411B LXI D:KF14M1 / [D,E] ← MESSAGE ADDR
035 15EF CAFE1A JZ KF14ER / BRANCH ON BLANK NODE
036
037 15F2 21FD09 K14025, LXI H:NUDTAB+NUDDCON / [H,L] ← TABLE ADDR
038 15F5 110700 LXI D:NUDRCL / [D,E] ← ENTRY LENGTH
039 15F8 3A7FFE LDA ASMCON / A ← CONTACT TYPE
040
041 15FB BF K14030, CMP M / CHECK FOR MATCH
042 15FC CA0316 JZ K14035 / BRANCH ON MATCH
043 15FF 19 DAD D / BUMP POINTER
044 1600 C3FB15 JMP K14030 / LOOP UNTIL FIND
045 EJECT

```



```

001 1603 2B          K14035, DCX      H          / MOVE TO NODE SIZE
002 1604 3E01        MVI          A;1          / A ← 1
003 1606 BE          CMP          M          / CHECK FOR MULTI-NODE
004 1607 C29416      JNZ          K14075        / BRANCH MULTI-NODE CONTACT
005
\
ENTER A 1 NODE ITEM
007
008 160A 23          INX          H          / STEP TO REFERENCE NODE
009 160B 23          INX          H          / BUMP POINTER
010 160C 7E          MOV          A;M          / A ← REFERENCE FIELD MASK
011
/
012 160D CDC71F      K14065, CALL     KU07          / VERIFY REFERENCE NUMBER
013 1610 C2011B      JNZ          KF14X         / BRANCH ON ERROR
014
/
015                  / IF NOT ENHANCED, DONT TAKE
016                  / TRANSITIONAL CONTACTS
017
018 1613 3A85FE      LDA          SCNF2        / GET CONFIG BYTE
019 1616 E602        ANI          SYSENH       / ISOLATE ENHANCED BIT
020 1618 C22816      JNZ          K14070       / GO IF ENHANCED
021
/
022                  / NOT ENHANCED; CHECK TRANSITIONALS
023
024 161B 3A7FFE      LDA          ASMCUN       / GET ASSEMBLY CONTACT
025 161E FE05        CPI          NOPOST       / POSI. TRANS?
026 1620 CAFB1A      JZ          KF14IV        / YES, INVALID
027 1623 FE06        CPI          NONEGT       / NEG. TRANS?
028 1625 CAFB1A      JZ          KF14IV        / YES, INVALID
029
/
030                  / HERE TO PUT CONTACT IN
031
032
K14070,
033 1628 3A7FFE      LDA          ASMCUN       / A ← CONTACT
034 162B 07          RLC                    / ROTATE TO
035 162C 07          RLC                    / FORM NODE
036 162D B4          DRA          H          / A ← BYTE 0
037 162E 65          MOV          H;L         / SWAP H AND L
038 162F 6F          MOV          L;A         / FOR SHLD
039 1630 2295FE      SHLD        CMDBUF+5     / STORE INTO BUFFER
040 1633 0E01        MVI          C;1         / SINGLE NODE INSERT
041 1635 CD3321      CALL        KU09         / DO INSERT
042 1638 C2011B      JNZ          KF14X         / BRANCH ON ERROR
043
/
044 163B 3A7FFE      LDA          ASMCUN       / A ← CONTACT TYPE
045 163E CD7A25      CALL        ISCOIL       / IS IT COIL TYPE?
046 1641 DA6D16      JC          K14040       / GO IF NOT
047
/
048 1644 3A7EFE      LDA          CURACT       / EXTEND ROW FOR COILS
049 1647 47          MOV          B;A         / B ← CURSOR
050 1648 4F          MOV          C;A         / C ← CURSOR
051
/
052 1649 79          K14037, MOV        A;C         / A ← CURSOR
053 164A E60F        ANI          COLMSK       / ISOLATE COLUMN
054 164C FE0C        CPI          MAXCOL+1    / AT RIGHT RAIL?
055 164E CA6716      JZ          K14039       / YES, DONE
056 1651 CD2B05      CALL        CURSOR       / MOVE CURSOR
057 1654 CD4705      CALL        CUR100       / GET POINTERS
058 1657 23          INX          H          / SKIP FIELD ATTRIBUTE
059 1658 1606        MVI          D;DSPNOD-1  / D ← COUNTER
060 165A 1E72        MVI          E;ASCDSH    / E ← DASH
061
/
062
K14038,
063 165C 73          MOV          M;E         / STORE DASH
064 165D 23          INX          H          / BUMP POINTER
065 165E 15          DCR          D          / DECREMENT COUNTER
066 165F C25C16      JNZ          K14038       / LOOP UNTIL DONE
067
/
068 1662 41          MOV          B;C         / B ← NEW CURSOR
069 1663 0C          INR          C          / C ← NEXT CURSOR
070 1664 C34916      JMP          K14037       / CONTINUE
071
/
072 1667 79          K14039, MOV        A;C         / A ← CURSOR
073 1668 3D          DCR          A          / ADJUST CURSOR
074 1669 327DFE      STA          CURDSP       / LOAD DATA
075
/
076 166C CD0B23      K14040, CALL     KU12         / SET CURSOR POINTERS

```

```

077 166F 23          INX      H          / BUMP ADDRESS
078 1670 11B3FD     LXI      D, DSPCON      / [D,E] ← SOURCE
079 1673 0606       MVI      B, DSPNOD-1    / B ← LENGTH
080 1675 CD0601     CALL     MOVSI0         / MOVE DATA
081 1678 114A00     LXI      D, ROWB-DSPNOD+1 / [D,E] ← BUMP TO NEXT
082 167B 19         DAD      D          / ROW FOR DISPLAY
083 167C 1100FD     LXI      D, DSPNUM+2    / [D,E] ← SOURCE
084 167F 0606       MVI      B, DSPNOD-1    / B ← LENGTH
085 1681 CD0601     CALL     MOVSI0         / MOVE DATA
086 1684 114F00     LXI      D, ROWB-1      / STEP TO NEXT ROW
087 1687 19         DAD      D          / BELOW VERT
088 1688 CD6224     CALL     FIXVER        / GO FIX LAST VERT CHAR
089 168B CDE323     CALL     KU19          / EXTEND POWER FROM RAIL
090 168E CD0A24     CALL     KU20          / CONNECT VERTICALS
091 1691 C3011B     JMP      KF14X         / GO TO EXIT
092                EJECT

```

ENTER A 2 OR 3 NODE ITEM

```

002
003 1694 CD0423     K14075, CALL     KU11          / A ← ROW
004 1697 86        ADD      M          / CHECK FOR OVERFLOW
005 1698 FE09       CPI      MAXROW+2    / BEYOND ROW 7
006 169A 11521B     LXI      D, KF14M3    / [D,E] ← MESSAGE ADDR
007 169D F2FE1A     JF       KF14ER      / BRANCH ON ERROR
008
009 16A0 3E02       MVI      A, 2        / CHECK FOR CALCULATE-TYPE
010 16A2 BE        CMP      M          / NODES (3 NODES)
011 16A3 C2E119     JNZ      K14200      / BRANCH ON A TRIPLE
012

```

HERE TO ENTER A 2 NODE ITEM

```

014
015 16A6 3A7FFE     LDA      ASMDON      / A ← CONTACT TYPE
016 16A9 FE13       CPI      NODCON      / CHECK FOR CONVERT NODE
017 16AB CAD916     JZ       K14085      / HANDLE CONVERT SEPARATELY
018

```

2 NODE: CTR, TMRS

```

020
021 16AE 3E38       MVI      A, NODCST+NODIRG+NODHRG / A ← MASK
022 16B0 CDC71F     CALL     KU07         / VALIDATE REFERENCE
023 16B3 C2011B     JNZ      KF14X        / EXIT ON ERROR
024
025 16B6 3A01FD     LDA      DSPNUM+3    / A ← REFERENCE TYPE
026 16B9 FE30       CPI      ASC0        / CHECK FOR CONSTANT
027 16BB 3E34       MVI      A, NODPRE!!04 / ASSUME CONSTANT
028 16BD CAC216     JZ       K14080      / BRANCH ON CONSTANT
029 16C0 3E38       MVI      A, NORPRE!!04 / REGISTER PRESET
030
031 16C2 B4         K14080, ORA      H          / A ← DATA HI
032 16C3 65        MOV      H, L        / H ← DATA LO
033 16C4 6F        MOV      L, A        / L ← DATA HI
034 16C5 2295FE     SHLD    CMDBUF+5     / LOAD BUFFER
035
036 16C8 3A7FFE     LDA      ASMDON      / A ← CONTACT TYPE
037 16CB 07        RLC          / ROTATE
038 16CC 07        RLC          / LEFT
039 16CD F602       ORI      DUMFLG      / SET DUMMY REG FLAG
040 16CF 3297FE     STA      CMDBUF+7    / LOAD NODE2 DATA HI
041 16D2 AF        CLA          / A ← 0
042 16D3 3298FE     STA      CMDBUF+10   / LOAD NODE2 DATA LO
043 16D6 C33817     JMP      K14110      / CONTINUE
044

```

2 NODE: CONVERT NODE

```

002
003 / MAKE SURE IT IS ENHANCED SET TO
004 / ACCEPT "CONVERT"
005
006 K14085,
007 16D9 3A85FE     LDA      BCONF2     / GET CONFIG BYTE
008 16DC E602       ANI      SYSENI     / ISOLATE ENHANCED?
009 16DE CAFB1A     JZ       KF14IV     / NO, ERROR
010 / OK, GO AHEAD
011
012 16E1 3A01FD     LDA      DSPNUM+3    / GET REFERENCE TYPE
013 16E4 FE31       CPI      ASC1        / CHECK FOR INPUT SOURCE

```


014	16E6	C22317	JNZ	K14100	/ BRANCH ON IT
015					
016	16E9	1102FD	LXI	D; DSPNUM+4	/ [D,E] ← BCD SOURCE
017	16EC	210000	LXI	H; 0	/ INITIALIZE BINARY
018	16EF	CD8E01	CALL	BCDBN3	/ CONVERT TO BINARY
019	16F2	2B	DCX	H	/ MAKE ZERO RELATIVE
020	16F3	E5	PUSH	H	/ STACK IT
021					
022	16F4	3A85FE	LDA	SCONF2	/ A ← COIL RAM CONFIG
023	16F7	E6F0	ANI	SYS256+SYS192+SYS128+SYS064	/ ISOLATE
024	16F9	CF	NSWP		/ COIL CONFIGURATION
025					/ AND ROTATE TO SET
026					/ UP TEST FOR ALLOWABLE
027					/ RANGE FOR I/O
028	16FA	21CBFF	LXI	H; -253	/ [H,L] ← INTIAL MAX
029	16FD	11C0FF	LXI	D; -264	/ [D,E] ← OFFSET
030					
031	1700	0F	K14090. RRC		/ ROTATE MASK
032	1701	DA0817	JC	K14095	/ BRANCH WHEN DONE
033	1704	19	DAD	D	/ ELSE, UP MAX
034	1705	C30017	JMP	K14090	/ AND CONTINUE
035					
036	1708	EB	K14095. XCHG		/ SWAP
037	1709	E1	POP	H	/ GET BINARY
038	170A	E5	PUSH	H	/ RESTACK IT
039	170B	19	DAD	D	/ CHECK FOR OVERFLOW
040	170C	E1	POP	H	/ GET BINARY
041	170D	11411B	LXI	D; KF14M1	/ SET ERROR ADDRESS
042	1710	DAFE1A	JC	KF14ER	/ BRANCH ON OVERFLOW
043					
044	1713	65	MOV	H, L	/ H ← NODE1 DATA LO
045	1714	2E4C	MVI	L; NOCON!: 04+SINFLG	/ L ← NODE1 DATA HI
046	1716	2295FE	SHLD	CMDBUF+5	/ LOAD BUFFER
047	1719	26FF	MVI	H; FF	/ H ← NODE1 DATA LO
048	171B	2E4F	MVI	L; NOCON!: 04+DRGFLG	/ L ← NODE2 DATA HI
049	171D	2297FE	SHLD	CMDBUF+7	/ LOAD BUFFER
050	1720	C33817	JMP	K14110	/ AND CONTINUE
051					
052	1723	3E10	K14100. MVI	A; NODHRG	/ CONVERT REGISTER SOURCE
053	1725	CDC71F	CALL	KU07	/ VALIDATE REFERENCE
054	1728	C2011B	JNZ	KF14X	/ EXIT ON ERROR
055					
056	172B	65	MOV	H; L	/ H ← NODE 1 DATA LO
057	172C	2E4D	MVI	L; NOCON!: 04+SRGFLG	/ L ← NODE 1 DATA HI
058	172E	2295FE	SHLD	CMDBUF+5	/ LOAD BUFFER
059	1731	26FF	MVI	H; FF	/ H ← NODE 2 DATA LO
060	1733	2E4E	MVI	L; NOCON!: 04+DINFLG	/ L ← NODE 2 DATA HI
061	1735	2297FE	SHLD	CMDBUF+7	/ LOAD BUFFER
062			EJECT		

COMMON CODE FOR 2 NODE ITEMS. . .

002					
003	1738	0E02	K14110. MVI	C; 2	/ C ← NODE COUNT
004	173A	CD3321	CALL	KU09	/ INSERT DOUBLE-NODE
005	173D	C2011B	JNZ	KF14X	/ EXIT ON ERROR
006					
007	1740	CD0B23	CALL	KU12	/ SET CURSOR POINTERS
008	1743	23	INX	H	/ STEP OVER ATTRIBUTE
009	1744	11081B	LXI	D; MULLN1	/ [D,E] ← SOURCE
010	1747	0605	MVI	B; DSPNOD-2	/ B ← LENGTH
011	1749	CD0601	CALL	MOVSI0	/ MOVE DATA
012	174C	3AB8FC	LDA	DSPVER	/ A ← VERTICAL
013	174F	77	MOV	M; A	/ STORE IT
014	1750	114B00	LXI	D; ROWB-DSPNOD+2	/ MOVE POINTER
015	1753	19	DAD	D	/ TO NEXT LINE
016	1754	3605	MVI	M; ASCLB	/ SET LEFT BOARDER
017	1756	23	INX	H	/ BUMP TO NEXT POSITION
018	1757	1101FD	LXI	D; DSPNUM+3	/ [D,E] ← SOURCE
019	175A	0604	MVI	B; 4	/ B ← LENGTH
020	175C	CD0601	CALL	MOVSI0	/ MOVE DATA
021	175F	3A05FD	LDA	DSPVER+ROWD	/ A ← VERTICAL
022	1762	77	MOV	M; A	/ STORE IT
023	1763	CDE323	CALL	KU19	/ POWER FROM RAIL
024	1766	CD0A24	CALL	KU20	/ CONNECT VERTICALS
025					

026	1769	3A7FFE	LDA	ASMC0N	/ A ← CONTACT TYPE
027	176C	110700	LXI	D, MULACL	/ [D, E] ← RECORD LENGTH
028	176F	21131B	LXI	H, MULTAB+1	/ [H, L] ← TABLE ADDR
029			/		
030	1772	BE	K14115, CMP	M	/ CHECK KEY MATCH
031	1773	CA7A17	JZ	K14120	/ BRANCH ON IT
032	1776	19	DAD	D	/ BUMP POINTER
033	1777	C37217	JMP	K14115	/ CONTINUE
034			/		
035	177A	E5	K14120, PUSH	H	/ SAVE ADDRESS
036	177B	3A7DFE	LDA	CURDSP	/ A ← CURSOR
037	177E	C610	ADI	:10	/ FAKE IT TO NEXT
038	1780	327DFE	STA	CURDSP	/ COLUMN FOR INSERT
039	1783	CD0B23	CALL	KU12	/ SET UP CURSOR POINTERS
040	1786	23	INX	H	/ STEP OVER ATTRIBUTE
041	1787	D1	POP	D	/ [D, E] ← DISPLAY SOURCE
042	1788	13	INX	D	/ FOR DISPLAY
043	1789	0605	MVI	B, DSPNOD-2	/ B ← LENGTH
044	178B	CD0601	CALL	MOVS10	/ MOVE DATA
045	178E	36E0	MVI	M, CA1100	/ DO VERTICAL
046	1790	114E00	LXI	D, ROWB-DSPNOD+2	/ [D, E] ← OFFSET
047	1793	19	DAD	D	/ MOVE POINTER TO NEXT ROW
048	1794	110D1B	LXI	D, MULLN2	/ [D, E] ← SOURCE
049	1797	0605	MVI	B, DSPNOD-2	/ B ← LENGTH
050	1799	CD0601	CALL	MOVS10	/ MOVE DATA
051	179C	CDE323	CALL	KU19	/ EXTEND POWER
052	179F	CD0A24	CALL	KU20	/ CONNECT VERTICALS
053			/		
054	17A2	3A7DFE	LDA	CURDSP	/ A ← CURSOR
055	17A5	D610	SUI	:10	/ MOVE IT BACK
056	17A7	327DFE	STA	CURDSP	/ STORE IT
057	17AA	C3011B	JMP	KF14X	/ AND EXIT
058			EJECT		

*** NODE REPLACEMENT

002			/		
003			K14135,		
004	17AD	3A7FFE	LDA	ASMC0N	/ GET REQUESTED TYPE
005	17B0	CD8023	CALL	KU15A	/ SEE IF M-NODE TYPE
006	17B3	C2FB1A	JNZ	KF141V	/ YES, "INVALID"
007					/ NO, OKAY TO GO ON
008	17B6	CD5124	CALL	KU22	/ GET PTR TO "COLTAB"
009	17B9	E7	GETHL		/ [H, L] ← LAST ADDRESS
010	17BA	CD0423	CALL	KU11	/ A ← ROW
011	17BD	3D	DCR	A	/ MAKE RELATIVE TO ZERO
012	17BE	87	ADD	A	/ TWO BYTES PER NODE
013	17BF	1600	MVI	D:0	/ D ← 0
014	17C1	5F	MOV	E:A	/ E ← OFFSET
015	17C2	19	DAD	D	/ [H, L] ← ADDRESS OF MO
016	17C3	EB	XCHG		/ SWAP
017	17C4	2193FE	LXI	H: CMDBUF+3	/ [H, L] ← DESTINATION
018	17C7	EF	MOVDE		/ STORE ADDRESS
019			/		
020	17C8	110000	LXI	D:0	/ [D, E] ← NULL DATA
021	17CB	EF	MOVDE		/ STORE DATA
022			/		
023	17CC	11FFFF	LXI	D:FFFF	/ [D, E] ← MASK
024	17CF	EF	MOVDE		/ STORE MASK
025			/		
026			/		
027			/		
028					
029	17D0	3AB7FC	LDA	DSPCON+4	/ GET LAST CHAR OF HOR CONT
CT					
030	17D3	FE1F	CPI	ASCCBK	/ IS IT "NULL" (CLEARED)?
031	17D5	CA0A18	JZ	K14150	/ YES, GO PROCESS #
032			EJECT		

\ HERE TO PROCESS NEW CONTACT TYPE

034			/		
035			/		
036					
037	17D8	CD7D23	CALL	KU15	/ MULTI-NODE REPLACE?
038	17DB	C2FB1A	JNZ	KF141V	/ YES, NOT ALLOWED
039			/		

```

040 K14145,
041
042 / IF AT RIGHT RAIL, REPLACEMENT MUST
043 / BE COIL TYPE
044 / IF NOT AT RIGHT RAIL, REPLACEMENT
045 / TYPE MUST NOT BE COIL
046
047 17DE 3A7DFE LDA CURDSP / GET CURRENT DISPLAY LOC
048 17E1 E60F ANI CDLMSK / ISOLATE COLUMN
049 17E3 FE0B CPI MAXCOL / AT RIGHT RAIL?
050 17E5 CAF417 JZ K14146 / YES, MAKE SURE ITS A COIL
051
052 / NOT AT RIGHT RAIL; MUST NOT BE A COIL
053
054 17E8 3A7FFE LDA ASMC0N / GET NEW TYPE
055 17EB CD7A25 CALL ISCOIL / IS IT A COIL?
056 17EE DAFD17 JC K14147 / NO, OKAY
057 17F1 C3FB1A JMP KF14IV / YES, INVALID
058
059 / AT RIGHT RAIL, MUST HAVE A COIL!
060
061 K14146,
062 17F4 3A7FFE LDA ASMC0N / GET NEW TYPE
063 17F7 CD7A25 CALL ISCOIL / IS IT A COIL?
064 17FA DAFB1A JC KF14IV / NO, INVALID
065
066 / OKAY TO USE CONTACT
067
068 K14147,
069 17FD 3A7FFE LDA ASMC0N / GET CONTACT TYPE
070 1800 07 RLC / SHIFT LEFT
071 1801 07 RLC / FOR NODE FORMAT
072 1802 3295FE STA CMDBUF+5 / STORE IN BUFFER
073
074 1805 3E83 MVI A, -1-NODMSK / A ← NEW MASK
075 1807 3297FE STA CMDBUF+7 / STORE IN BUFFER
076 EJECT

```

*** NUMERIC REPLACEMENT

```

002 /
003 180A 3A01FD K14150, LDA DSPNUM+3 / A ← NUMERIC FIELD
004 180D FE1D CPI ASCNBK / UNDEFINED?
005 180F CA8718 JZ K14158 / YES, SKIP
006
007 / USE THE #, SO SET THE "TO BE CLEARED" FLAG
008
009 1812 3A7CFE LDA KSTATE / GET FLAGS
010 1815 F610 ORI KCLEAR / SET "TO BE CLEARED"
011 1817 327CFE STA KSTATE / X
012
013 / BRANCH ON CONTACT TYPE
014
015 181A 3A80FE LDA CURCON / A ← CURRENT CONTACT
016 181D FE0D CPI NOCPRE / CHECK FOR CONSTANT PRESET
017 181F CA3E18 JZ K14152 / BRANCH ON CONSTANT PRESET
018 1822 FE14 CPI NOCCON / CHECK FOR C-NODE CONST NT
019 1824 CA3E18 JZ K14152 / BRANCH ON IT
020 1827 FE0E CPI NORPRE / CHECK FOR REGISTER PRESET
021 1829 CA3118 JZ K14151 / BRANCH ON IT
022 182C FE15 CPI NOCREG / CHECK FOR C-NODE REG
023 182E C25618 JNZ K14154 / BRANCH IF NOT
024
025 1831 47 K14151, MOV B, A / B ← NODE TYPE
026 1832 3A01FD LDA DSPNUM+3 / A ← REFERENCE TYPE
027 1835 FE30 CPI ASC0 / CHECK FOR CONSTANT
028 1837 C25618 JNZ K14154 / BRANCH IF NO CHANGE
029 183A 05 DCR B / CHANGE NODE TYPE
030 183B C34818 JMP K14153 / AND CONTINUE
031
032 183E 47 K14152, MOV B, A / B ← NODE TYPE
033 183F 3A01FD LDA DSPNUM+3 / A ← REFERENCE TYPE
034 1842 FE30 CPI ASC0 / CHECK FOR CONSTANT
035 1844 CA5618 JZ K14154 / BRANCH ON CONSTANT
036 1847 04 INR B / CHANGE NODE TYPE
037
038 1848 78 K14153, MOV A, B / A ← NEW NODE TYPE

```


039	1849	3280FE		STA	CURCON	/ UPDATE CURSOR NODE TYF
040	184C	07		RLC		/ ROTATE TO
041	184D	07		RLC		/ FORM NODE
042	184E	3295FE		STA	CMDBUF+9	/ STORE IN BUFFER
043	1851	3E80		MVI	A, EDCFLG	/ A ← MASKHI
044	1853	3297FE		STA	CMDBUF+7	/ LOAD MASKHI
045				EJECT		
001			/			VERIFY REFERENCE # TO CONTACT TYPE +
002			/			SET UP I/O COMMAND.
003						
004	1856	3A7FFE	K14154,	LDA	ASMC0N	/ A ← ASSEMBLY NODE TYF
005	1859	B7		TST		/ CHECK IF DEFINED
006	185A	C26018		JNZ	K14155	/ BRANCH IF DEFINED
007			/			
008	185D	3A80FE		LDA	CURCON	/ A ← CURSOR NODE TYPE
009			/			
010	1860	21FD09	K14155,	LXI	H, NODTAB+NODCON	/ [H, L] ← POINTER
011	1863	110900		LXI	D, NODRCL	/ [D, E] ← OFFSET
012			/			
013	1866	BE	K14156,	CMF	M	/ LOOK FOR MATCH
014	1867	CA6E18		JZ	K14157	/ BRANCH ON MATCH
015	186A	19		DAD	D	/ MOVE TO NEXT ENTRY
016	186B	C36618		JMP	K14156	/ CONTINUE
017			/			
018	186E	23	K14157,	INX	H	/ BUMP TO REFERENCE MASK
019	186F	7E		MOV	A, M	/ A ← REFERENCE MASK
020	1870	CDC71F		CALL	KU07	/ VERIFY REFERENCE
021	1873	C2011B		JNZ	KF14X	/ EXIT ON ERROR
022			/			
023	1876	EB		XCHG		/ SWAP
024	1877	2195FE		LXI	H, CMDBUF+5	/ SET POINTER
025	187A	7A		MOV	A, D	/ A ← HI-ORDER REF
026	187B	B6		ORA	M	/ SET CORRECT BITS
027	187C	77		MOV	M, A	/ SET FINAL BYTE 0
028			/			
029	187D	23		INX	H	/ BUMP POINTER
030	187E	73		MOV	M, E	/ SET BYTE 1
031	187F	23		INX	H	/ BUMP TO MASKHI
032	1880	7E		MOV	A, M	/ A ← MASKHI
033	1881	E6FC		ANI	-1-SEQFLG	/ CLEAR HI-ORDER REF FIELD
034	1883	77		MOV	M, A	/ STORE IN BUFFER
035	1884	AF		CLA		/ A ← 0
036	1885	23		INX	H	/ BUMP TO MASKLO
037	1886	77		MOV	M, A	/ STORE MASKLO
038			/			
039	1887	110A21	K14158,	LXI	D, CMDWRT+CMDOZ!: 100+LENWRT	/ SET PARMS
040	188A	CD8125		CALL	PID	/ DO WRITE
041	188D	C2011B		JNZ	KF14X	/ EXIT ON ERROR
042			/			
043			/			UPDATE "MATROW" WITH NEW NODE TYPE.
044			/			IF "ASMC0N" BLANK, USE CURSOR NODE TYPE
045						
046	1890	3A7FFE		LDA	ASMC0N	/ GET ASSEMBLY NODE TYPE
047	1893	B7		TST		/ BLANK?
048	1894	C29A18		JNZ	K14159	/ NO, USE IT
049	1897	3A80FE		LDA	CURCON	/ YES, USE CURRENT NODE
050			K14159,			
051	189A	F5		PUSH	PSW	/ SAVE TYPE
052	189B	CDAA23		CALL	KU17	/ GET PTR TO MATROW
053	189E	F1		POP	PSW	/ GET NODE TYPE
054	189F	77		MOV	M, A	/ UPDATE MATROW
055			/			
056			/			UPDATE DISPLAY WITH NEW CONTACT, IF THERE IS ONE
057						
058	18A0	3AB3FC		LDA	DSPC0N	/ A ← CONTACT FIELD
059	18A3	FE1F		CPI	ASCCBK	/ CHECK FOR BLANK
060	18A5	CABD18		JZ	K14160	/ DO CHANGE NEEDED
061			/			
062	18A8	CD7D23		CALL	KU15	/ CHECK FOR MULTI-NODE
063	18AB	C2BD18		JNZ	K14160	/ DON'T CHANGE
064			/			
065	18AE	CD0B23		CALL	KU12	/ B ← CURSOR
066	18B1	23		INX	H	/ BUMP OVER ATTRIBUTE
067	18B2	11B3FC		LXI	D, DSPC0N	/ SET SOURCE


```

068 18B5 0605          MVI      B, DSPNOD+2      / B ← COUNT
069 18B7 CD0601        CALL     MOVSI0           / PAINT NEW DISPLAY
070 18BA CDE323        CALL     KU19            / EXTEND POWER
071                    EJECT
\   SEE IF # TO BE DISPLAYED
073
074 18BD 3A01FD        K14160, LDA     DSPNUM+3      / A ← NUMERIC FIELD
075 18C0 FE1D          CPI      ASCNBK         / UNDEFINED?
076 18C2 CAF118        JZ       K14170         / YES, SKIP
077
\   MOVE # FROM ASSEMBLY TO NETWORK
079
080 18C5 CD0B23        CALL     KU12            / B ← CURSOR
081 18C8 19            DAD     D              / BUMP TO REF FIELD
082 18C9 23            INX     H              / SKIP ATTRIBUTE
083 18CA 23            INX     H              / TO MS DIGIT IN NETWORK
084 18CB 0604          MVI     B, 4           / SET LENGTH
085
086                    /   NOW SEE IF THE ASSEMBLY AREA IS SPACES OR #
087
088 18CD 1101FD        LXI     D, DSPNUM+3      / SET SRC PTR
089 18D0 1A            LDAX   D              / GET MS DIGIT
090 18D1 FE20          CPI     ASCBLK         / IS IT BLANK NOW?
091 18D3 CAE118        JZ      K14164         / YES, JUST MOVE #
092
093                    /   HAVE #, SO SEE IF UNDERLINE OR NOT
094
095 18D6 7E            MOV     A, M           / GET # IN SCREEN NET
096 18D7 FE20          CPI     ASCBLK         / IS IT BLANK NOW?
097 18D9 CAE118        JZ      K14164         / YES, JUST MOVE #
098
099                    /   # IS ON SCREEN; SEE IF IT IS ALREADY
100                   /   UNDERLINED
101
102 18DC FE30          CPI     ASC0           / CHECK FOR UNDERLINE
103 18DE DAE718        JC     K14165         / BRANCH TO UNDERLINE
104                   K14164,
105 18E1 CD0601        CALL     MOVSI0         / DISPLAY REFERENCE
106 18E4 C3F118        JMP     K14170         / CONTINUE
107
108 18E7 1A            K14165, LDAX   D           / A ← NUMBER
109 18E8 D620          SUI     ASC0-ASCOUN    / UNDERLINE IT
110 18EA 77            MOV     M, A          / DISPLAY IT
111 18EB 13            INX     D             / BUMP
112 18EC 23            INX     H             / POINTERS
113 18ED 05            DCR     B             / DONE?
114 18EE C2E718        JNZ    K14165         / NO, CONTINUE LOOP
115                    EJECT
\*** VERTICAL REPLACEMENTS
002
003 18F1 3AB9FC        K14170, LDA     DSPVER      / A ← VERTICAL FIELD
004 18F4 FE1E          CPI     ASCVBK         / UNDEFINED?
005 18F6 CA011B        JZ      KF14X         / YES, EXIT
006
007 18F9 CD0423        CALL     KU11           / A ← ROW
008 18FC 47            MOV     B, A          / B ← COUNTER
009 18FD 3E80          MVI     A, 80         / A ← INITIAL MASK
010
011 18FF 0F            K14172, RRC           / ROTATE MASK
012 1900 05            DCR     B             / DECREMENT COUNTER
013 1901 C2FF18        JNZ    K14172         / LOOP UNTIL DONE
014 1904 4F            MOV     C, A          / SAVE MASK FOR A SEC
015
016                    /   NOW GET PTR TO COLTAB FOR CURRENT POS
017
018 1905 CD5124        CALL     KU22           / NOW H/L = PTR
019 1908 E5            PUSH   H             / SAVE POINTER
020 1909 110400        LXI     D, EDCI       / [D, E] ← OFFSET
021 190C 19            DAD     D             / [H, L] ← EDC NODE
022 190D 59            MOV     E, C         / E ← MASK
023 190E AF            CLA           / A ← 0
024 190F BE            CMP     M             / ANY VERTICALS THIS ROW
025 1910 C24C19        JNZ    K14175         / YES, CONTINUE
026
027 1913 3A05FD        LDA     DSPVER+ROWD    / A ← VERTICAL TYPE

```

028	1916	FE20	CPI	ASCBLK	/ IS IT BLANK?
029	1918	C21F19	JNZ	K14174	/ NO, MUST DO AN INSERT
030	1918	E1	POP	H	/ YES, CLEAN STACK
031	191C	C3011B	JMF	KF14X	/ AND EXIT WITH NO ACTION
032					
033	191F	1608	MVI	B, NDECC1, 04	/ D ← NODE TYPE
034	1921	2B	DCX	H	/ MOVE POINTER BACK TO
035	1922	2B	DCX	H	/ LAST ADDRESS IN COLUMN
036	1923	E7	GETHL		/ [H, L] ← LAST ADDRESS
037	1924	23	INX	H	/ BUMP TO MOVE OVER
038	1925	23	INX	H	/ LAST NODE IN COL
039	1926	44	MOV	B, H	/ B ← ADDRHI
040	1927	4D	MOV	C, E	/ C ← ADDRLO
041	1928	2193FE	LXI	H, UMDBUF+3	/ [H, L] ← DESTINATION
042	192B	D7	MOVDE		/ LOAD ADDRESS
043	192C	EF	MOVDE		/ LOAD DATA
044	192D	D5	PUSH	B	/ SAVE DATA
045					
046	192E	1108B1	LXI	B, CNDINC+CMDO2!, 100+LENINC	/ SET PARMS
047	1931	CD8125	CALL	PIQ	/ INSERT NEW EOC NODE
048	1934	D1	POP	H	/ GET EOC NODE
049	1935	E1	POP	H	/ GET COLUMN POINTER
050	1936	C2011B	JNZ	KF14X	/ EXIT ON ERROR
051					
052	1939	E5	PUSH	H	/ STACK POINTER
053	193A	010400	LXI	B, EDCI	/ [B, C] ← OFFSET
054	193D	09	DAD	B	/ [H, L] ← EOC DATA
055	193E	EF	MOVDE		/ LOAD NEW EOC DATA
056	193F	E1	POP	H	/ RESTORE POINTER
057	1940	010100	LXI	B, CMDO2	/ [B, C] ← NODE COUNT
058	1943	CDB824	CALL	COLINC	/ UPDATE COLUMN TABLE
059	1946	CD0F22	CALL	KU09UF	/ UPDATE MATROW AND USEAGE
060	1949	C30419	JMP	K14190	/ AND DISPLAY VERTICAL
061			EJECT		

*** UPDATE EXISTING VERTICAL

002					
003				K14175.	
004	194C	23	INX	H	/ POINT AT VERTICALS
005	194D	7B	MOV	A, E	/ A ← MASK
006	194E	4B	MOV	C, E	/ C ← MASK
007	194F	5E	MOV	E, M	/ E ← VERTICALS
008	1950	1600	MVI	D, 0	/ D ← 0
009	1952	B3	DRA	E	/ SET FLAG
010	1953	5F	MOV	E, A	/ STORE IT
011	1954	3A05FD	LDA	DSFVER+ROWD	/ A ← VERTICAL FIELD
012	1957	FE20	CPI	ASCBLK	/ ANY VERTICAL?
013	1959	C29E19	JNZ	K14180	/ YES
014					
015	195C	79	MOV	A, C	/ A ← MASK
016	195D	AB	XRA	E	/ CLEAR FLAG
017	195E	5F	MOV	E, A	/ SET VERTICAL FLAGS
018	195F	C29E19	JNZ	K14180	/ BRANCH IF STILL VERTICALS
019					
020	1962	01FDFE	LXI	B, -EUCLU+COLEHI	/ [B, C] ← OFFSET
021	1965	09	DAD	B	/ [H, L] ← ADDRESS
022	1966	E7	GETHL		/ [H, L] ← ADDRESS OF EOC N
DE					
023	1967	EB	XCHG		/ SWAP
024	1968	2193FE	LXI	H, UMDBUF+3	/ [H, L] ← DESTINATION
025	196B	EF	MOVDE		/ LOAD ADDRESS
026					
027	196C	1106C1	LXI	B, CMDDC+CMDO2!, 100+LENDEC	/ SET PARMS
028	196F	CD8125	CALL	PIQ	/ DELETE EOC NODE
029	1972	E1	POP	H	/ GET POINTER
030	1973	C2011B	JNZ	KF14X	/ EXIT ON ERROR
031					
032	1976	E5	PUSH	H	/ SAVE PTR TO COL
033	1977	010400	LXI	B, EDCI	/ GET OFFSET TO EOC DATA
034	197A	09	DAD	B	/ COMPUTE ADDR OF EOC
035	197B	010000	LXI	B, 0	/ CLEAR THE EOC NODE
036	197E	D7	MOVDE		/ DONE
037	197F	E1	POP	H	/ RESTORE PTR TO COL
038	1980	010100	LXI	B, CMDO2	/ [B, C] ← NODE COUNT
039	1983	CD0035	CALL	COLDEC	/ UPDATE COLUMN TABLE
040	1986	CD0B33	CALL	KU12	/ GET DISPLAY POINTERS


```

041 1989 110600      LXI      D, DSPNDD-1      / [D, E] ← OFFSET
042 198C 19          DAD      D                  / BUMP TO VERTICAL SPOT
043 198D 34E0        MVI      M, CA1100        / CLEAR VERTICAL
044 198F 115000      LXI      D, ROWB          / [D, E] ← OFFSET
045 1992 19          DAD      D                  / BUMP TO NEXT LINE
046 1993 3496        MVI      M, ASOBLK        / BLANK ENTRY
047 1995 19          DAD      D                  / BUMP TO NEXT ROW
048 1996 3E20        MVI      A, ASOBLK        / GET A SPACE
049 1998 CD6224      CALL     FIXVER           / FIX LAST VERT CHAR
050 199B C3011B      JMP      KF14X            / AND EXIT
051

```

HERE TO PUT A NEW VERT IN
ON A COLUMN WITH OTHER VERTS

```

054
055
056 199E 2195FE      K14180, LXI      H, CMDBUF+5 / SET DESTINATION
057 19A1 EF          MOVDE                      / STORE DATA
058
059 19A2 1100FF      LXI      D, FF00          / SET MASK
060 19A5 EF          MOVDE                      / LOAD MASK
061
062 19A6 E1          POP      H                / GET POINTER
063 19A7 E5          PUSH     H                / SAVE IT
064 19A8 110200      LXI      D, COLEHI        / [D, E] ← OFFSET
065 19AB 19          DAD      D                / [H, L] ← LAST ADDR
066 19AC E7          GETHL                      / [H, L] ← LAST ADDR OF AT

067 19AD EB          XCHG                      / SWAP
068 19AE 2193FE      LXI      H, CMDBUF+3      / [H, L] ← DESTINATION
069 19B1 EF          MOVDE                      / LOAD ADDRESS
070
071 19B2 110A21      LXI      D, CMDWRT+CMDOZ!:100+LENWRT / SET PATMS
072 19B5 CD8125      CALL     F10              / DO WRITE
073 19B8 E1          POP      H                / CLEAN STACK
074 19B9 C2011B      JNZ     KF14X            / EXIT ON ERROR
075
076 19BC 110500      LXI      D, EOCL0        / [D, E] ← OFFSET
077 19BF 19          DAD      D                / [H, L] ← CONNECTIVITY YT

078 19C0 3A96FE      LDA      CMDBUF+6        / A ← NEW CONNECTIVITY BYT
079 19C3 77          MOV      M, A            / LOAD NEW CONNECTIVITY BYT

080
081 K14190,
082 19C4 CD0B23      CALL     KU12             / B ← CURSOR
083 19C7 110600      LXI      D, DSPNDD-1      / GET OFFSET
084 19CA 19          DAD      D                / [H, L] ← VERTICAL SLOT
085 19CB 3AB8FC      LDA      DSPVER           / A ← VERTICAL
086 19CE 77          MOV      M, A            / DISPLAY IT
087 19CF 115000      LXI      D, ROWB          / [D, E] ← OFFSET
088 19D2 19          DAD      D                / MOVE POINTER
089 19D3 3A05FD      LDA      DSPVER+ROWD      / A ← VERTICAL
090 19D6 77          MOV      M, A            / DISPLAY IT
091 19D7 19          DAD      D                / STEP TO ROW BELOW VERT
092 19D8 CD6224      CALL     FIXVER           / GO FIX LAST VERT CHR
093 19DB CD0A24      CALL     KU20             / CONNECT VERTICALS
094 19DE C3011B      JMP      KF14X            / AND EXIT
095

```

*** CALCULATE NODES (3 NODES)

```

002
003 / MAKE SURE ENHANCED SET TO ACCEPT CALC
004
005 K14200,
006 19E1 3A55FE      LDA      SCNF2           / GET CONFIG BYTE
007 19E4 E602        ANI      SYSEINH         / IS IT ENHANCED?
008 19E6 CAFB1A      JZ       KF14IV         / NO, ERROR
009 / OK, GO AHEAD
010
011 19E9 3E38        MVI      A, NODCST+NODIRG+NODHRG / A ← MASK
012 19EB CDC71F      CALL     KU07            / VALIDATE REFERENCE
013 19EE C2011B      JNZ     KF14X            / EXIT ON ERROR
014

```


015	19F1	3A01FD	LDA	DSPNUM+3	/ A ← REFERENCE TYPE
016	19F4	FE30	CFI	ASCO	/ CHECK FOR CONSTANT
017	19F6	3E34	MVI	A,NDOPRE!!04	/ ASSUME CONSTANT
018	19F8	CAF015	OZ	K14201	/ BRANCH ON CONSTANT
019	19FB	3E38	MVI	A,NDOPRE!!04	/ REGISTER PRESET
020			/		
021	19FD	B4	ORA	H	/ CREATE DATAHI
022	19FE	60	MOV	H,L	/ H ← DATA0
023	19FF	6F	MOV	L,A	/ L ← DATAHI
024	1A00	2295FE	SHLD	CMDBUF+5	/ STORE INTO BUFFER
025			/		
026	1A03	2600	MVI	H,0	/ H ← DATA0
027	1A05	2E06	MVI	L,NDOPRE!!04+DUMPLG	/ L ← DATAHI
028	1A07	2297FE	SHLD	CMDBUF+7	/ LOAD BUFFER
029			/		
030	1A0A	21351B	LXI	H,K14TAB	/ [H,L] ← START OF TABL
031	1A0D	110300	LXI	D,3	/ D ← TABLE LENGTH
032	1A10	3AB4FC	LDA	DSPNUM+1	/ A ← FIRST CHARACTER
033			/		
034	1A13	BE	JMP	H	/ LOOK FOR MATCH
035	1A14	0A1B1A	OZ	K14210	/ BRANCH ON MATCH
036	1A17	19	DAD	D	/ BUMP TO NEXT ENTRY
037	1A18	0B131A	JMP	K14205	/ AND CONTINUE
038			/		
039	1A1B	23	INX	H	/ BUMP TO DISPLAY CHAR
040	1A1C	E5	PUSH	H	/ STACK POINTER
041	1A1D	23	INX	H	/ BUMP TO SUB-FIELD TYPE
042	1A1F	7F	MOV	A,M	/ A ← SUB-FIELD
043	1A1F	F658	ORI	NDOPRE!!04	/ SET NODE TYPE
044	1A21	6F	MOV	L,A	/ SET DATA HI
045	1A22	26FF	MVI	H,FF	/ SET DATA LO
046	1A24	2299FE	SHLD	CMDBUF+11	/ LOAD BUFFER
047			/		
048	1A27	0E03	MVI	D,3	/ D ← NODE COUNT
049	1A29	0D3321	CALL	KD09	/ INSERT NODE
050	1A2C	E1	POP	H	/ CLEAN STACK
051	1A2D	02011B	JNZ	KF14X	/ EXIT ON ERROR
052			/		
053	1A30	E5	PUSH	H	/ RESTACK POINTER
054	1A31	0D0B23	CALL	KU12	/ SET CURSOR POINTERS
055	1A34	23	INX	H	/ SET OVER ATTRIBUTE
056	1A35	11081B	LXI	D,MULLN1	/ [D,E] ← SOURCE
057	1A38	0605	MVI	B,DSPNOD-2	/ B ← LENGTH
058	1A3A	0D0601	CALL	MOV510	/ DISPLAY TOP ROW
059			/		
060	1A3D	3AB8FC	LDA	DSPVER	/ A ← VERTICAL
061	1A40	77	MOV	M,A	/ DISPLAY VERTICAL
062	1A41	114C00	LXI	D,ROWB-DSPNOD+3	/ [D,E] ← OFFSET
063	1A44	19	DAD	D	/ BUMP TO NEXT ROW
064	1A45	3605	MVI	M,ASCLB	/ SET LEFT BORDER
065	1A47	23	INX	H	/ INCREMENT POINTER
066	1A48	1101FD	LXI	D,DSPNUM+3	/ [D,E] ← SOURCE
067	1A48	0605	MVI	B,5	/ B ← COUNTER
068	1A4D	0D0601	CALL	MOV510	/ DISPLAY REFERENCE
069			/		
070	1A50	0DE323	CALL	KU19	/ EXTEND POWER
071	1A53	0D0A24	CALL	KU20	/ CONNECT VERTICALS
072			/		
073	1A56	3A7DFE	LDA	CURDSP	/ A ← CURSOR
074	1A59	0610	ADI	:10	/ FAKE IT TO NEXT ROW
075	1A5B	327DFE	STA	CURDSP	/ AND STORE IT
076			EJED1		
			/		
001	1A5E	0D0B23	CALL	KU12	/ A ← CURSOR
002	1A61	23	INX	H	/ STEP OVER ATTRIBUTE
003	1A62	3605	MVI	M,ASCLB	/ DISPLAY LEFT BORDER
004	1A64	23	INX	H	/ STEP
005	1A65	23	INX	H	/ TO ACTIVITY POSITION
006	1A66	D1	POP	D	/ GET POINTER TO CHAR
007	1A67	1A	LDHA	D	/ A ← CHARACTER
008	1A68	77	MOV	M,A	/ DISPLAY IT
009	1A69	23	INX	H	/ BUMP
010	1A6A	23	INX	H	/ POINTER
011	1A6B	3605	MVI	M,ASCLB	/ DISPLAY BORDER
012	1A6D	23	INX	H	/ NOW DO VERTICAL
013	1A6F	3AE0	MVI	M,CA1100	/ DISPLAY VERTICAL
014			/		

015	1A70	114B00	LXI	D, ROWB-DSPNOD+2	/ [D, E] ← OFFSET
016	1A73	19	DAD	D	/ BUMP TO NEXT ROW
017	1A74	3605	MVI	M, ASCLB	/ DISPLAY BOARDER
018	1A76	23	INX	H	/ BUMP POINTER
019	1A77	EB	XCHG		/ SWAP
020	1A78	21A00F	LXI	H, @4000	/ [H, L] ← DUMMY REGISTER
021	1A7B	CDC201	CALL	BNBCD4	/ DISPLAY IT
022	1A7E	CDE323	CALL	KU19	/ EXTEND POWER
023	1A81	CDA24	CALL	KU20	/ CONNECT VERTICALS
024					
025	1A84	3A7DFE	LDA	CURDSP	/ A ← CURSOR
026	1A87	C610	ADI	:10	/ FAKE IT TO NEXT ROW
027	1A89	327DFE	STA	CURDSP	/ FOR ENTRY
028					
029	1A8C	CD0B23	CALL	KU12	/ SET POINTERS
030	1A8F	23	INX	H	/ SKIP ATTRIBUTE
031	1A90	3605	MVI	M, ASCLB	/ DISPLAY BOARDER
032	1A92	23	INX	H	/ MOVE
033	1A93	23	INX	H	/ POINTER
034	1A94	361B	MVI	M, ASCADN	/ INSERT DOWN ARROW
035	1A96	23	INX	H	/ MOVE
036	1A97	23	INX	H	/ POINTER
037	1A98	3609	MVI	M, ASCRB	/ DISPLAY RIGHT BOARDER
038	1A9A	23	INX	H	/ BUMP POINTER
039	1A9B	36E0	MVI	M, CA1100	/ DISPLAY VERTICAL
040	1A9D	114B00	LXI	D, ROWB-DSPNOD+2	/ [D, E] ← OFFSET
041	1AA0	19	DAD	D	/ MOVE POINTER
042	1AA1	110D1B	LXI	D, MULLN2	/ [D, E] ← SOURCE
043	1AA4	0605	MVI	B, 5	/ B ← LENGTH
044	1AA6	CD0601	CALL	MOV810	/ DISPLAY DATA
045	1AA9	CDE323	CALL	KU19	/ EXTEND POWER
046	1AAC	CDA24	CALL	KU20	/ CONNECT VERTICALS
047					
048	1AAF	3A7DFE	LDA	CURDSP	/ A ← CURSOR
049	1AB2	D620	SUI	:20	/ UNFAKE IT
050	1AB4	327DFE	STA	CURDSP	/ TO REAL PLACE
051	1AB7	C3011B	JMP	KF14X	/ AND EXIT
052			EJECT		
001					
002					/***HOLDING REGISTER UPDATE
003					
004	1ABA	CD4705	CALL	CUR100	/ SET CURSOR POINTERS
005	1ABD	23	INX	H	/ SKIP ATTRIBUTE
006	1ABE	7E	MOV	A, M	/ A ← REFERENCE TYPE
007	1ABF	FE34	CPI	ASC4	/ MUST BE 4XXX
008	1AC1	C2FB1A	JNZ	KF14IV	/ BRANCH ON ERROR
009					
010	1AC4	23	INX	H	/ STEP OVER REFERENCE TYPE
011	1AC5	EB	XCHG		/ SWAP
012	1AC6	210000	LXI	H, 0	/ INITIALIZE BINARY RESULT
013	1AC9	CD8E01	CALL	BCDBN3	/ CONVERT TO BINARY
014	1ACC	EB	XCHG		/ SWAP
015	1ACD	1640	MVI	D, REGFLD	/ SET ADDRHI
016	1ACF	1C	INR	E	/ SET ADDRLO
017	1AD0	2193FE	LXI	H, CMDBUF+3	/ [H, L] ← DESTINATION
018	1AD3	EF	MOVDE		/ STORE ADDR
019					
020	1AD4	1101FD	LXI	D, DSPNUM+3	/ [D, E] ← NUMERIC FIELD
021	1AD7	1A	LDAX	D	/ A ← HIGH-ORDER DIGIT
022	1AD8	FE30	CPI	ASC0	/ MUST BE ZERO
023	1ADA	CAE31A	JZ	K14905	/ BRANCH ON NO ERROR
024					
025	1ADD	11491B	LXI	D, KF14M2	/ [D, E] ← MESSAGE ADDR
026	1AE0	C3FE1A	JMP	KF14ER	/ COMMON CODE
027					
028	1AE3	210000	LXI	H, 0	/ INITIALIZE BINARY
029	1AE6	CD8101	CALL	BCDBN4	/ CONVERT TO BINARY
030					
031	1AE9	EB	XCHG		/ SWAP
032	1AEA	2195FE	LXI	H, CMDBUF+5	/ SET DESTINATION
033	1AED	EF	MOVDE		/ STORE NEW DATA
034					
035	1AEE	110000	LXI	D, 0	/ [D, E] ← MASK
036	1AF1	EF	MOVDE		/ SET MASK
037					

```

038 1AF2 110A21          LXI      D, CMDWRT+CMD02!: 100+LENWRT / SET PARMS
039 1AF5 0B8123          CALL    F10           / DO WRITE
040 1AF8 03011B          JMP     KF14X         / AND EXIT
041                      EJECT

001                      KF14IV,
002 1AFB 11411B          LXI      D, KF14M1     / GET PTR TO 'INVALID'
003 1AFE 0D7E05          KF14ER, CALL    ERROR / SET ERROR STATE
004                      /
005 1B01 09              KF14X,  RET          / RET
006                      EJECT

001                      SUBJOB  KF14Z = CHECK FOR A BLANK NODE
002                      /
003                      /
004                      /      A = CURRENT CURSOR (ROW, COL)
005                      /
006                      /***CHECK FOR BLANK NODE
007                      /
008                      /***Z-BIT. EQ. 0 -> NON-BLANK.
009                      /***Z-BIT. EQ. 1 -> BLANK.
010                      /
011                      KF14Z,
012 1B02 0DB123          CALL    KU17A         / GET "MATROW" PTR
013 1B05 AF              CLA          / TEST FOR BLANK NODE TO 0
014 1B06 BE              CMP     M           / TEST DONE; FLAGS SET
015 1B07 09              RET          / WITH Z: BIT SET/RESET
016                      EJECT

001                      SUBJOB  KEY_FUNCTION : KF14 : ENTER - DATA TABLES
002                      /
003                      /***TOP ROW - MULTINODE CONTACTS
004                      /
005 1B08 02030303        MULLN1, DB          ASCTL; ASCUB; ASCUB; ASCUB; ASCUB; ASCUB
006                      1B0C 04
007                      /
008                      /***BOTTOM ROW - MULTINODE CONTACTS
009                      /
009 1B0D 05141010        MULLN2, DB          ASCLB; ASC4UN; ASCOUN; ASCOUN; ASCOUN
010                      1B11 10
011                      /
012                      /***DISPLAY TABLE
013                      /
013                      0000          MULKEY= 0           / KEY
014                      0001          MULNOD= MULKEY+1       / NODE TYPE
015                      0002          MULDIS= MULNOD+1       / DISPLAY
016                      /
017                      0007          MULRCL= MULDIS+5       / RECORD LENGTH
018                      /
019 1B12 110F            MULTAB, DB          KEY0; NOCTR         / COUNTER
020 1B14 05435452        DB          ASCLB; ASCO; ASCO; ASCO; ASCO; ASCO; ASCO
021                      1B18 09
022                      /
022 1B19 1910            DB          KEY3; NOT100          / TIMER 1.0
023 1B1B 07312E30        DB          ASCTMR; ASC1; ASCDOT; ASCO; ASCRB
024                      1B1F 09
024                      /
025 1B20 1211            DB          KEY2; NOT010          / TIMER 0.1
026 1B22 07302E31        DB          ASCTMR; ASCO; ASCDOT; ASC1; ASCRB
027                      1B26 09
027                      /
028 1B27 0B12            DB          KEY1; NOT001          / TIMER 0.01
029 1B29 072E3031        DB          ASCTMR; ASCDOT; ASCO; ASC1; ASCRB
030                      1B2D 09
030                      /
031 1B2E 1313            DB          KEYS; NOCON          / CONVERT
032 1B30 05434F4E        DB          ASCLB; ASCO; ASCO; ASCN; ASCRB
033                      1B34 09
033                      EJECT
034 1B35 41              K14TAB, DA          'A'           / ADD
035 1B36 2B00            DB          ASCPLS; ADDFLG
036                      /
037 1B38 53              DA          'S'           / SUBTRACT
038 1B39 2D01            DB          ASCMIN; SUBFLG
039                      /

```



```

040 1B3E 4D          DA          'M'          / MULTIPLY
041 1B3C 0E02        DB          ASCMPX; MPXFLG
042                  /
043 1B3E 44          DA          'D'          / DIVIDE
044 1B3F 0A03        DB          ASCDIV; DIVFLG
045                  EJECT

001                  /
002                  /***MESSAGES
003                  /
004 1B41 07          KF14M1, DB          K14M1X
005 1B42 494E5641    DA          'INVALID'
006 1B46 4C4944
007 0007            KF14M1X= .-KF14M1-1
008                  /
008 1B49 08          KF14M2, DB          K14M2X
009 1B4A 42414420    DA          'BAD NODE'
010 1B4E 4E4F4445
011 0008            KF14M2X= .-KF14M2-1
012                  /
012 1B52 0C          KF14M3, DB          K14M3X
013 1B53 434F4C20    DA          'COL TOO LONG'
014 1B57 544F4F20
015 1B5B 4C4F4E47
016 000C            KF14M3X= .-KF14M3-1
017                  /
016 1B5F 07          KF14M4, DB          K14M4X
017 1B60 4E4F2056    DA          'NO VERT'
018 1B64 455254
019 0007            KF14M4X= .-KF14M4-1
020                  /
020 1B67 0B          KF14M5, DB          K14M5X
021 1B68 4E4F4445    DA          'NODE IN WAY'
022 1B6C 20494E20
023 1B70 574159
024 000B            KF14M5X= .-KF14M5-1
025                  EJECT

001                  /          SUBJOB KEY FUNCTION : KF15 : SPARE KEY
002                  /
003                  /***KEY FUNCTION : KF15 : SPARE KEYS
004                  /
005                  /***PRELIMINARY VERSION
006                  /
007 1B73 117A1B      KF15,   LXI      D;KF15MS      / [D,E] <- MESSAGE ADDR
008 1B76 CD7E05      CALL     ERROR      / SET ERROR STATE
009 1B79 C9          RET       / EXIT
010                  /
011                  /***MESSAGE
012                  /
013 1B7A 09          KF15MS, DB          KF15MX
014 1B7B 53504152    DA          'SPARE KEY'
015 1B7F 45204B45
016 1B83 59
017 0009            KF15MX= .-KF15MS-1          / MESSAGE LENGTH
018                  EJECT

001                  /          SUBJOB KEY FUNCTION : KF16 : ILLEGAL KEYS
002                  /
003                  /***KEY FUNCTION : KF16 : ILLEGAL KEYS
004                  /
005 1B84 118B1B      KF16,   LXI      D;KF16MS      / [D,E] <- MESSAGE ADDR
006 1B87 CD7E05      CALL     ERROR      / SET ERROR STATE
007 1B8A C9          RET       / EXIT
008                  /
009                  /***MESSAGE
010                  /
011 1B8E 0B          KF16M3, DB          KF16MX
012 1B8C 494C4C45    DA          'ILLEGAL KEY'
013 1B90 47414C20
014 1B94 4B4559
015 000B            KF16MX= .-KF16MS-1          / MESSAGE LENGTH
016                  EJECT

```

```

001          SUBJOB KEY FUNCTION : KF17 : ERROR RESET
002          /
003          /***KEY FUNCTION : KF17 : ERROR RESET
004          /
005          KF17,
006 1B97 CD9F1E          CALL      CLRERR          / CLEAR ERROR LINE AND STATE
007
008          /          NOW, IF ERROR CAME FROM HARD P180 I/O, RESET WORD!
009
010 1B9A 78             MOV      A;B          / GET OLD STATE BYTE
011 1B9B E640          ANI      KERROR          / HARD I/O ERROR?
012 1B9D C8             RZ              / NO, GO BACK TO REGULAR
013 1B9E C7             RST      0          / YES!, RESET P180
014
015
016          \ SUBROUTINE "CLRERR"
017
018          / THIS ROUTINE CLEARS THE ERROR LINE AND THE
019          / RESET AND SHIFT BITS IN THE STATE BYTE
020
021          CLRERR,
022 1B9F 217CFE          LXI      H,KSTATE          / [H,L] ← STATE VECTOR
023 1BA2 7E             MOV      A;M          / LOAD STATE VECTOR
024 1BA3 47             MOV      B;A          / SAVE IT, TOO
025 1BA4 E65F          ANI      -1-KRESET-KSHIFT / CLEAR RESET FLAG
026 1BA6 77             MOV      M;A          / STORE STATE VECTOR
027 1BA7 AF             CLA              / CLEAR A
028 1BA8 3293FD          STA      TMREERR          / RESET ERROR TIMER
029 1BAB 21BBFC          LXI      H,DISPERR          / [H,L] ← ERROR FIELD ADDRESS
030 1BAE 160C          MVI      D,ERRFLD-1          / D ← FIELD LENGTH
031 1BB0 CD1903          CALL      ROWNIO          / CLEAR ERROR FIELD
032
033 1BB3 C9             RET              / EXIT
034
001          SUBJOB KEY FUNCTION : KF18 : DISCRETE UPDATE
002          /
003          /***KEY FUNCTION : KF18 : DISCRETE UPDATE
004          /
005          /***FUNCTION ACTIVATED BY CLKINT VIA SPOOLER
006          /
007 1BB4 3E06          KF18,   MVI      A;ASMMNUM          / A ← COUNTER
008          /
009 1BB6 F5             KF1805, PUSH     PSW          / STACK COUNTER
010 1BB7 3A81FE          LDA      DISPTR          / A ← POINTER
011 1BBA F680          ORI      ASMROW          / SET MASK
012 1BBC 47             MOV      B;A          / B ← DISPLAY COORDINATES
013 1BED CD4705          CALL      CUR100          / COMPUTE POINTERS
014 1BC0 23             INX      H          / BUMP TO REFERENCE TYPE
015 1BC1 7E             MOV      A;M          / A ← REFERENCE TYPE
016 1BC2 FE20          CPI      ASCBLK          / CHECK FOR BLANK
017 1BC4 C2DD1B          JNZ      KF1815          / BRANCH IF NOT
018          /
019 1BC7 3A81FE          LDA      DISPTR          / A ← POINTER
020 1BCA 3C             INR      A          / BUMP IT
021 1BCB FE0C          CPI      MAXCOL+1          / CHECK FOR WRAP-AROUND
022 1BCD C2D21B          JNZ      KF1810          / BRANCH IF NOT
023 1BD0 3E06          MVI      A;ASMCOL          / RESET POINTER
024          /
025 1BD2 3281FE          KF1810, STA      DISPTR          / STORE POINTER
026 1BD5 F1             POP      PSW          / GET COUNTER
027 1BD6 3D             DCR      A          / DECREMENT COUNTER
028 1BD7 C2B61B          JNZ      KF1805          / LOOP IF NOT DONE
029 1BDA C3781C          JMP      KF18X          / EXIT
030          /
031 1BDD F1             KF1815, POP      PSW          / CLEAN STACK
032 1BDE E5             PUSH     H          / SAVE POINTER
033 1BDF 23             INX      H          / BUMP TO ADDRESS
034 1BE0 EB             XCHG          / SWAP
035 1BE1 210000          LXI      H,0          / INITIALIZE BINARY RESULT
036 1BE4 CD8E01          CALL      B0DBN3          / CONVERT
037          /
038 1BE7 2D             DCR      H          / MAKE RELATIVE BASE 0
039 1BE8 EB             XCHG          / SWAP

```

401

402

040	1BE9	E1	POP	H	/ GET POINTER
041	1BEA	E5	PUSH	H	/ STACK IT AGAIN
042	1BEB	7E	MOV	A;M	/ A ← REFERENCE TYPE
043	1BEC	FE34	CPI	ASC4	/ CHECK FOR HOLDING REG
044	1BEE	CA431C	JZ	KF1830	/ BRANCH ON IT
045	1BF1	FE33	CPI	ASC3	/ CHECK FOR INPUT REG
046	1BF3	CA4A1C	JZ	KF1835	/ BRANCH ON IT
047					
048	1BF6	D5	PUSH	D	/ SAVE TYPE
049	1BF7	1620	MVI	D; IOFLD	/ SET I/O FIELD FLAG
050	1BF9	2193FE	LXI	H; CMDBUF+3	/ [H,L] ← POINTER
051	1BFC	EF	MOVDE		/ STORE ADDRESS
052			EJECT		
001	1BFD	110611	LXI	D; CMDRED+CMD02!: 100+LENRED	/ SET PARMS
002	1C00	CD8125	CALL	PIO	/ DO READ
003	1C03	D1	POP	D	/ CLEAN (REFERENCE TYPE)
004	1C04	E1	POP	H	/ STACK (POINTER)
005	1C05	C2781C	JNZ	KF18X	/ EXIT ON ERROR
006					
007	1C08	0601	MVI	B; INTSTA	/ B ← MASK
008	1C0A	AF	CLA		/ A ← 0
009	1C0B	BA	CMF	D	/ CHECK FOR INTERNAL COIL
010	1C0C	C22C1C	JNZ	KF1820	/ BRANCH ON IT
011	1C0F	0602	MVI	B; OUTSTA	/ B ← MASK
012	1C11	7E	MOV	A;M	/ A ← REFERENCE TYPE
013	1C12	FE30	CPI	ASC0	/ CHECK FOR COIL
014	1C14	CA2C1C	JZ	KF1820	/ BRANCH ON IT
015	1C17	0604	MVI	B; INPSTA	/ B ← MASK
016	1C19	E5	PUSH	H	/ SAVE POINTER
017	1C1A	114E00	LXI	D; ROWD+1	/ [D,E] ← OFFSET
018	1C1D	19	DAD	D	/ BUMP TO DISABLE AREA
019	1C1E	3AABFE	LDA	RSPBUF+3	/ A ← STATE
020	1C21	E608	ANI	INPDIS	/ ISOLATE DISABLE FLAG
021	1C23	D1	POP	D	/ [D,E] ← POINTER
022	1C24	3620	MVI	M; ASCBLK	/ ASSUME ENABLED
023	1C26	CA2B1C	JZ	KF1816	/ BRANCH IF ENABLED
024	1C29	3644	MVI	M; ASCD	/ ELSE, INDICATE DISABLED
025					
026	1C2B	EB	KF1816, XCHG		/ SWAP POINTERS BACK
027					
028	1C2C	3AABFE	KF1820, LDA	RSPBUF+3	/ A ← STATE BYTE
029	1C2F	A0	ANA	B	/ ISOLATE DISCRETE STATE
030	1C30	114F00	LXI	D; ROWD+2	/ D ← OFFSET
031	1C33	19	DAD	D	/ MOVE POINTER
032	1C34	117E1C	LXI	D; KF18M1	/ [D,E] ← 'OFF' MESSAGE
033	1C37	C23D1C	JNZ	KF1825	/ BRANCH ON OFF
034	1C3A	11821C	LXI	D; KF18M2	/ [D,E] ← 'ON' MESSAGE
035					
036	1C3D	CD0301	KF1825, CALL	MOVSTR	/ DISPLAY STATE
037	1C40	C36C1C	JMP	KF1899	/ GO TO COMMON EXIT
038			EJECT		
001					
002					
003					
004	1C43	1C	KF1830, INR	E	/ HOLDING REGISTER
005	1C44	1C	INR	E	/ STEP TO BASE ADDRESS
006	1C45	1640	MVI	D; REGFLD	/ SET TO FIELD TYPE
007	1C47	C3511C	JMP	KF1840	/ GO TO COMMON CODE
008					
009	1C4A	7D	KF1835, MOV	A;L	/ A ← INPUT REG - 1
010	1C4B	85	ADD	L	/ DOUBLE IT
011	1C4C	C6C0	ADI	INPBAS	/ ADD IN BASE ADDR
012	1C4E	6F	MOV	L;A	/ L ← SPD ADDRESS
013	1C4F	2660	MVI	H; SPDFLD	/ SET FIELD TYPE
014					
015	1C51	2193FE	KF1840, LXI	H; CMDBUF+3	/ [H,L] ← POINTER
016	1C54	EF	MOVDE		/ LOAD ADDRESS
017					
018	1C55	110611	LXI	D; CMDRED+CMD02!: 100+LENRED	/ SET PARMS
019	1C58	CD8125	CALL	PIO	/ DO READ
020	1C5B	E1	POP	H	/ CLEAN STACK
021	1C5C	C2781C	JNZ	KF18X	/ EXIT ON ERROR
022					

023 1C5F 114D00	LXI	D; ROWD	/ [D, E] ← OFFSET
024 1C62 19	DAD	D	/ [H, L] ← DESTINATION
025 1C63 EB	XCHG		/ SWAP
026 1C64 13	INX	D	/ INCREMENT BINARY VALUE
027 1C65 21ABFE	LXI	H; RSPBUF+3	/ SET POINTER TO VALUE
028 1C68 E7	GETHL		/ [H, L] ← BINARY REGISTER
ALUE			
029 1C69 CDC201	CALL	BNBCD4	/ CONVERT AND DISPLAY
030	/		
031 1C6C 2181FE	KF1899, LXI	H; DISPTR	/ [H, L] ← POINTER
032 1C6F 34	INR	M	/ BUMP POINTER
033 1C70 7E	MOV	A; M	/ A ← POINTER
034 1C71 FE0C	CPI	MAXCOL+1	/ CHECK FOR WRAP-AROUND
035 1C73 C2781C	JNZ	KF18X	/ BRANCH IF NOT
036	/		
037 1C76 3606	MVI	M; ASMCOL	/ RESET POINTER
038	/		
039 1C78 3E01	KF18X, MVI	A; DISTMR	/ A ← TIMER RESET
040 1C7A 3294FD	STA	TMRDIS	/ REINIT TIMER
041 1C7D C9	RET		/ EXIT
042	EJECT		
043	/		
044	/***MESSAGES		
045	/		
046 1C7E 03	KF18M1, DB	K18M1X	
047 1C7F 204F4E	DA	'ON'	
048	0003	K18M1X=	.-KF18M1-1
049	/		
050 1C82 03	KF18M2, DB	K18M2X	
051 1C83 4F4646	DA	'OFF'	
052	0003	K18M2X=	.-KF18M2-1
053	EJECT		
001		SUBJOB KEY FUNCTION :	KF19 : SUPERVISORY STATE
002	/		
003	/***KEY FUNCTION :	KF19 : SUPERVISORY STATE	
004	/		
005	/***ENTRY POINTS:		
006	/		
007	/	KF19 - SET SUPERVISORY STATE	
008	/	KF1920 - EXECUTE SUPERVISORY COMMAND	
009	/		
010	/***NOTE:		
011	/		
012	/	SUPERVISORY STATE CREATES ITS OWN DISPLAY.	
013	/	WHEN IN SUPERVISORY STATE (KSUPER. EQ. 1)	
014	/	ALL KEYS, EXCEPT RESET ARE PROCESSED HERE.	
015	/		
016 1C86 CD281F	KF19, CALL	KU01	/ CHECK FOR RESET
017 1C89 CD491F	CALL	KU02	/ CHECK FOR SHIFT
018 1C8C CA951C	JZ	KF1905	/ BRANCH ON NO SHIFT
019 1C8F CD791F	CALL	KU05	/ SET ERROR STATE
020 1C92 C3131D	JMP	KF19X	/ EXIT
021	/		
022 1C95 CD0002	KF1905, CALL	SPLINI	/ CLEAR SPOOLER
023 1C98 AF	CLA		/ A ← 0
024 1C99 3294FD	STA	TMRDIS	/ STOP DISCRETE UPDATES
025 1C9C 3292FD	STA	TMRPWR	/ STOP POWER DISPLAY
026 1C9F 3291FD	STA	TMRLED	/ STOP LED DISPLAY
027	/		
028 1CA2 CD8A1F	CALL	KU06	/ CLEAR SCREEN
029 1CA5 3E2F	MVI	A; KEYSUP	/ A ← SUPERVISORY KEY
030 1CA7 3283FE	STA	LASTKY	/ INDICATE LAST KEYSTROKE
031 1CAA 3282FE	STA	NEWKEY	/ AND CURRENT KEYSTROKE
032	/		
033 1CAD 3E02	MVI	A; KSUPER	/ A ← SUPERVISORY STATE
034 1CAF 327CFE	STA	KSTATE	/ SET STATE
035	/		
036 1CB2 CD0B23	CALL	KU12	/ B ← CURSOR
037 1CB5 3680	MVI	M; DMAFAN	/ CLEAR DISPLAY CURSOR TOP
038 1CB7 19	DAD	D	/ BUMP TO NEXT ROW
039 1CB8 3680	MVI	M; DMAFAN	/ CLEAR DISPLAY CURSOR B T
040	/		
041 1CBA 21B1FC	LXI	H; DSPASM	/ [H, L] ← POINTER


```

007 1D15 110480      K19100, LXI      D;CMDSTP!:100+LENSTP  / SET PARMs
008 1D18 CD8125      CALL      PID          / STOP CONTROLLER
009 1D1B C0          RNZ          / ERROR, EXIT NOW
010
011 /              NOW READ SYSTEM STATE TO BE SURE!
012
013 1D1C CD5B2C      CALL      DELHLF / WAIT .5 SEC
014 1D1F CD641D      CALL      RDSYS / GET IT
015 1D22 C0          RNZ          / I/O EROR, QUIT
016
017 1D23 E610        ANI      SYSSTP / ARE WE STOPPED?
018 1D25 CA5D1D      JZ       K19NG  / NO! FAILURE
019
020 /              DISPLAY MSG "STOP OK" ON ADVISORY
021
022 1D28 11D61D      LXI      D;K191MS      / GET PTR
023 1D2B CD681F      CALL      KU04         / DO IT
024 1D2E C3131D      JMP      KF19X        / EXIT
025
026 /***START
027 /
028 1D31 110490      K19200, LXI      D;CMDGO!:100+LENGO    / SET PARMs
029 1D34 CD8125      CALL      PID          / GO COMMAND
030 1D37 C0          RNZ          / ERROR, EXIT NOW
031
032 /              NOW READ SYSTEM STATE TO BE SURE!
033
034 1D38 CD5B2C      CALL      DELHLF / WAIT .5 SEC
035 1D3B CD641D      CALL      RDSYS / GET IT!
036 1D3E C0          RNZ          / I/O ERROR, QUIT
037
038 1D3F E680        ANI      SYSRUN / ARE WE RUNNING?
039 /              (GO OUT AND CATCH IT!)
040 1D41 CA5D1D      JZ       K19NG  / NO, FAILURE
041
042 /              DISPLAY "START OK"
043
044 1D44 11DE1D      LXI      D;K192MS      / GET PTR TO MSG
045 1D47 CD681F      CALL      KU04         / DISPLAY IT
046 1D4A C3131D      JMP      KF19X        / EXIT
047 EJECT
048 /
049 /***INITIALIZE
050 /
051 1D4D 1104A0      K19300, LXI      D;CMDINI!:100+LENINI  / SET PARMs
052 1D50 CD8125      CALL      PID          / INITIALIZE
053 1D53 C0          RNZ          / ERROR, EXIT NOW
054
055 /              DISPLAY MSG "INIT OK"
056
057 1D54 11E71D      LXI      D;K193MS      / PTR TO MSG
058 1D57 CD681F      CALL      KU04         / DISPLAY IT
059 1D5A C3131D      JMP      KF19X        / EXIT
060
061 /
062 /              HERE WHEN START OR STOP FAILED!
063
064 K19NG,
065 1D5D 11EF1D      LXI      D;K194MS / GET PTR
066 1D60 CD7E05      CALL      ERROR / DISPLAY IT
067 1D63 C9          RET          / EXIT
068 EJECT

001 SUBJOB RDSYS = READ SYSTEM STATE BYTE
002
003 / RDSYS IS A SUBR TO READ THE SYS STATE BYTE
004 /
005 / *ENTRY
006 /     A MUST BE FREE
007 /
008 /     CALL RDSYS
009 /
010 / *EXIT
011 /     A = BYTE IF GOOD READ;

```



```

012 / Z-BIT RESET IF BAD I/O; SET IF OK
013 /
014 RDSYS,
015 1D64 C5 PUSH B / SAVE MAJOR
016 1D65 D5 PUSH D / X
017 1D66 E5 PUSH H / X
018
019 1D67 2193FE LXI H;CMBUF+3 / PTR TO I/O BUFF
020 1D6A 11BD60 LXI D;ADRSYS / CONTROLLER ADDR OF SYS STATE
021 1D6D EF MOVDE / STORE IN BUFF
022
023 1D6E 110611 LXI D;CMDRED+CMD02!:100+LENRED
024 1D71 CD8125 CALL PIO / READ IT!
025 1D74 C27B1D JNZ RDSYSX / BAD, EXIT
026
027 1D77 AF CLA / GOOD! SET Z-BIT
028 1D78 3AABFE LDA RSPBUF+3/ GET SYS STATE BYTE
029
030 RDSYSX,
031 1D7B E1 POP H / RESTORE AND EXIT
032 1D7C D1 POP D / X
033 1D7D C1 POP B / X
034 1D7E C9 RET / X
035
001 1D7F DSPSUP=
002 1D7F 0A80 DSPST0, DB DSPS0X;DMAFAN
003 1D81 3020D20 DA '0 - EX'
004 1D85 4558
005 1D87 80 DB DMAFAN
006 1D88 4954 DA 'IT'
007 000A DSPS0X=.-DSPST0-1
008 1D8A 0A80 DSPST1, DB DSPS1X;DMAFAN
009 1D8C 3120D20 DA '1 - ST'
010 1D90 5354
011 1D92 80 DB DMAFAN
012 1D93 4F50 DA 'OP'
013 000A DSPS1X=.-DSPST1-1
014 1D95 0B80 DSPST2, DB DSPS2X;DMAFAN
015 1D97 3220D20 DA '2 - ST'
016 1D9B 5354
017 1D9D 80 DB DMAFAN
018 1D9E 415254 DA 'ART'
019 000B DSPS2X=.-DSPST2-1
020 1DA1 1180 DSPST3, DB DSPS3X;DMAFAN
021 1DA3 3320D20 DA '3 - IN'
022 1DA7 494E
023 1DA9 80 DB DMAFAN
024 1DAA 49544941 DA 'ITIALI'
025 1DAE 4C49
026 1DB0 80 DB DMAFAN
027 1DB1 5A45 DA 'ZE'
028 0011 DSPS3X=.-DSPST3-1
029 1DB3 0A80 DSPST4, DB DSPS4X;DMAFAN
030 1DB5 3420D20 DA '4 - LO'
031 1DB9 4C4F
032 1DBB 80 DB DMAFAN
033 1DBC 4144 DA 'AD'
034 000A DSPS4X=.-DSPST4-1
035 1DBE 0A80 DSPST5, DB DSPS5X;DMAFAN
036 1DC0 3520D20 DA '5 - DU'
037 1DC4 4455
038 1DC6 80 DB DMAFAN
039 1DC7 4D50 DA 'MP'
040 000A DSPS5X=.-DSPST5-1
041 1DC9 0C80 DSPST6, DB DSPS6X;DMAFAN
042 1DCB 3620D20 DA '6 - VE'
043 1DCF 5645

```

```

042 1DD1 80          DB      DMAFAN
043 1DD2 52494659    DA      'RIFY'
044          000C     DSPTS6X= .-DSPST6-1
045          /
046          0007     SUPOPT= 7          / NUMBER OF COMMANDS
047
048
049          /          SUPERVISORY MSGS
050
051          K191MS,
052 1DD6 07          DB      K191ME / LENGTH
053 1DD7 53544F50    DA      'STOP OK'
      1DD8 204F4B
054          0007     K191ME=.-K191MS-1
055
056          K192MS,
057 1DDE 08          DB      K192ME
058 1DDF 53544152    DA      'START OK'
      1DE3 54204F4B
059          0008     K192ME=.-K192MS-1
060
061          K193MS,
062 1DE7 07          DB      K193ME
063 1DE8 494E4954    DA      'INIT OK'
      1DEC 204F4B
064          0007     K193ME=.-K193MS-1
065
066          K194MS,
067 1DEF 0A          DB      K194ME
068 1DF0 46434E20    DA      'FCN FAILED'
      1DF4 4641494C
      1DF8 4544
069          000A     K194ME=.-K194MS-1
070          EJECT

001          SUBJOB KEY FUNCTION : KF20 : POWER DISPLAY
002          /
003          /          THIS ROUTINE IS ACTIVATED BY THE
004          /          SPOOLER FROM THE CLOCK INTERRUPT ROUTINE
005          /
006          /***KEY FUNCTION : KF20 : POWER DISPLAY
007          /
008 1DFA 3A7CFE      KF20, LDA      KSTATE          / A ← STATE VECTOR
009 1DFD E608        ANI      KNET           / NETWORK ACTIVE?
010 1DFF CA9A1E      JZ       KF20X          / NO, EXIT
011          /
012 1E02 218AFE      LXI      H,STPNUM       / [H,L] ← SOURCE
013 1E05 E7          GETHL          / [H,L] ← SEQUENCE NUMBER
014 1E06 EB          XCHG           / SWAP
015 1E07 2193FE      LXI      H,CMDBUF+3      / [H,L] ← DESTINATION
016 1E0A EF          MOVDE          / LOAD BUFFER
017          /
018 1E0B 110640      LXI      D,CMDPWR!:100+LENPWR / SET PARMS
019 1E0E CD8125      CALL     PIO             / GET POWER DATA
020 1E11 C29A1E      JNZ      KF20X          / EXIT ON ERROR
021          /
022 1E14 210FF8      LXI      H,L1C01U+4      / [H,L] ← DISPLAY POINTER
023 1E17 11ABFE      LXI      D,RSPBUF+3     / [D,E] ← POWER DATA POINT
R
024 1E1A 3E01        MVI      A,1             / A ← COLUMN INDICATOR
025          /
026 1E1C E5          KF2005, PUSH     H             / STACK DISPLAY POINTER
027 1E1D D5          PUSH     D             / STACK POWER POINTER
028 1E1E F5          PUSH     PSW           / STACK COLUMN COUNTER
029          /
030 1E1F 1A          LDAX     D             / A ← POWER BYTE FOR CO
031 1E20 57          MOV      D,A           / D ← POWER BYTE
032 1E21 0602        MVI      B,2           / B ← COUNTER
033 1E23 CDA01E      CALL     K20SUB        / HIGHLIGHT BEFORE VERTS
034          /
035 1E26 F1          POP      PSW           / GET COLUMN COUNT
036 1E27 F5          PUSH     PSW           / RESTACK IT
037 1E28 FE0B        CPI      MAXCOL        / CHECK IF DONE
038 1E2A C9C01E      JZ       KF2040        / BRANCH WHEN DONE
039 1E2D 21F7FD      LXI      H,COLTAB-COLBKL / [H,L] ← TABLE POINTER

```

413

414

```

040 1E30 110600          LXI      D; COLBKL      / [D,E] ← ENTRY LENGTH
041                      /
042 1E33 19             KF2010, DAD      D      / BUMP POINTER
043 1E34 3D             DCR      A      / DECREMENT COUNT
044 1E35 C2331E         JNZ      KF2010      / LOOP UNTIL DONE
045                      /
046 1E38 110500         LXI      D; EOCLD      / [D,E] ← OFFSET
047 1E3B 19             DAD      D      / SET POINTER TO EOC
048 1E3C AF             CLA                      / A ← 0
049 1E3D BE             CMP      M      / ANY VERTICALS?
050 1E3E CA7D1E         JZ       KF2035      / NO, CONTINUE
051                      EJECT

001                      /
002                      /***SOLVE VERTICAL CONNECTIVITY
003                      /
004 1E41 4E             MOV      C; M      / C ← CONNECTIVITY BYTE
005 1E42 210200         LXI      H; 2      / [H,L] ← OFFSET
006 1E45 39             DAD      SP      / [H,L] ← ADDR OF POINTER
007 1E46 7E             MOV      A; M      / A ← DATA0
008 1E47 23             INX      H      / BUMP POINTER
009 1E48 66             MOV      H; M      / H ← DATAHI
010 1E49 6F             MOV      L; A      / L ← DATA0
011 1E4A 56             MOV      D; M      / D ← POWER BYTE
012 1E4B AF             CLA                      / A ← 0
013 1E4C BA             CMP      D      / CHECK FOR ANY POWER
014 1E4D CA7D1E         JZ       KF2035      / NO POWER => NO WORK
015 1E50 0601           MVI      B; 01      / B ← MASK BIT
016                      /
017 1E52 79             KF2020, MOV     A; C      / A ← CONNECTIVITY DATA
018 1E53 A0             ANA      B      / LOOK FOR VERTICAL
019 1E54 CA6C1E         JZ       KF2030      / BRANCH IF NO VERT
020 1E57 1E00           MVI      E; 0      / CLEAR NEW MASK
021                      /
022 1E59 B3             KF2025, ORA      E      / SET BITS
023 1E5A 5F             MOV      E; A      / SAVE MASK
024 1E5B 78             MOV      A; B      / A ← ROTATING MASK
025 1E5C 07             RLC                      / SHIFT IT LEFT
026 1E5D 47             MOV      B; A      / AND SAVE IT
027 1E5E A1             ANA      C      / LOOK FOR CONTINUED VER
028 1E5F C2591E         JNZ      KF2025      / LOOP UNTIL END
029                      /
030 1E62 78             MOV      A; B      / A ← ROTATING MASK
031 1E63 B3             ORA      E      / A ← POWER MASK
032 1E64 5F             MOV      E; A      / SAVE IT
033 1E65 A2             ANA      D      / LOOK FOR ANY POWER
034 1E66 CA6C1E         JZ       KF2030      / NO POWER, CONTINUE
035                      /
036 1E69 7A             MOV      A; D      / A ← POWER BYTE
037 1E6A B3             ORA      E      / PASS POWER ON VERTS
038 1E6B 57             MOV      D; A      / UPDATE POWER BYTE
039                      /
040 1E6C 78             KF2030, MOV     A; B      / A ← ROTATING MASK
041 1E6D FE80           CPI      :80      / DONE?
042 1E6F 07             RLC                      / ROTATE MASK
043 1E70 47             MOV      B; A      / UPDATE IT
044 1E71 C2521E         JNZ      KF2020      / LOOP UNTIL DONE
045                      /
046 1E74 210200         LXI      H; 2      / [H,L] ← OFFSET
047 1E77 39             DAD      SP      / [H,L] ← ADDR OF PTR
048 1E78 7E             MOV      A; M      / A ← DATA0
049 1E79 23             INX      H      / BUMP POINTER
050 1E7A 66             MOV      H; M      / H ← DATAHI
051 1E7B 6F             MOV      L; A      / L ← DATA0
052 1E7C 72             MOV      M; D      / LOAD IN BUFFER
053                      /
054                      EJECT

001                      /
002                      /***DISPLAY POWER AFTER VERTICALS
003                      /
004 1E7D F1             KF2035, POP     PSW      / GET DATA FROM STACK
005 1E7E D1             POP      D      / POWER POINTER
006 1E7F E1             POP      H      / DISPLAY POINTER
007 1E80 E5             PUSH     H      / RESTACK DATA

```


008 1E81 D5	PUSH	D	/ TO SAVE
009 1E82 F5	PUSH	PSW	/ FOR FUTURE USE
010	/		
011 1E83 1A	LDAX	D	/ A ← POWER BYTE
012 1E84 57	MOV	D; A	/ D ← POWER BYTE
013 1E85 23	INX	H	/ MOVE POINTER
014 1E86 23	INX	H	/ TO VERTICAL AREA
015 1E87 0605	MVI	B; DSPNOD-2	/ B ← COUNTER
016 1E89 CDA01E	CALL	K20SUB	/ DO POWER
017	/		
018 1E8C F1	KF2040, POP	PSW	/ GET COLUMN COUNTER
019 1E8D D1	POP	D	/ GET POWER POINTER
020 1E8E E1	POP	H	/ GET DISPLAY POINTER
021	/		
022 1E8F 13	INX	D	/ BUMP TO NEXT POWER BYTE
023 1E90 3C	INR	A	/ BUMP TO NEXT COLUMN
024 1E91 010700	LXI	B; DSPNOD	/ [B, C] ← OFFSET
025 1E94 09	DAD	B	/ BUMP TO NEXT COLUMN
026 1E95 FE0C	CPI	MAXCOL+1	/ CHECK IF DONE
027 1E97 FA1C1E	JM	KF2005	/ LOOP UNTIL DONE
028	/		
029 1E9A 3E02	KF20X, MVI	A; PWR TMR	/ A ← TIMER VALUE
030 1E9C 3292FD	STA	TMRPWR	/ LOAD TIMER
031 1E9F C9	RET		/ EXIT
032	EJECT		
001	/		
002	/***SUBROUTINE TO HIGHLIGHT POWER		
003	/		
004 1EA0 0E0E	K20SUB, MVI	C; MAXROW+MAXROW	/ C ← COUNTER
005	/		
006 1EA2 7A	K20S05, MOV	A; D	/ A ← POWER BYTE
007 1EA3 07	RLC		/ SHIFT LEFT TO LSB
008 1EA4 57	MOV	D; A	/ AND SAVE IT
009	/		
010 1EA5 C5	K20S10, PUSH	B	/ SAVE POINTERS
011	/		
012 1EA6 CDC71E	K20S15, CALL	K20SSR	/ DO NEXT CHARACTER
013 1EA9 23	INX	H	/ BUMP POINTER
014 1EAA 05	DCR	B	/ DECREMENT COUNTER
015 1EAB C2A61E	JNZ	K20S15	/ LOOP UNTIL DONE
016	/		
017 1EAE 015000	LXI	B; ROWB	/ [B, C] ← OFFSET
018 1EB1 09	DAD	B	/ BUMP TO NEXT LINE
019 1EB2 C1	POP	B	/ GET COUNTERS
020 1EB3 58	MOV	E; B	/ E ← FIELD LENGTH
021	/		
022 1EB4 2B	K20S20, DCX	H	/ MOVE POINTER
023 1EB5 1D	DCR	E	/ DECREMENT COUNT
024 1EB6 C2B41E	JNZ	K20S20	/ LOOP UNTIL DONE
025	/		
026 1EB9 0D	DCR	C	/ DECREMENT COUNTER
027 1EBA CAC61E	JZ	K20SX	/ EXIT ON ERROR
028 1EBD 79	MOV	A; C	/ A ← COUNT REMAINING
029 1EBE E601	ANI	:01	/ ISOLATE LSB
030 1EC0 C2A51E	JNZ	K20S10	/ BRANCH IF ODD
031 1EC3 C3A21E	JMP	K20S05	/ ELSE, ROTATE POWER
032	/		
033 1EC6 C9	K20SX, RET		/ EXIT
034	EJECT		
001	/		
002	/***HIGHLIGHT CHARACTER ROUTINE		
003	/		
004 1EC7 7E	K20SSR, MOV	A; M	/ A ← CHARACTER
005 1EC8 FE60	CPI	ASCLRE	/ DO RANGE CHECK
006 1ECA DAE01E	JC	K20SSX	/ BRANCH OUT-OF-RANGE
007 1ECD FE7F	CPI	FACNOR-1	/ DO RANGE CHECK
008 1ECF DAD71E	JC	K20SS1	/ BRANCH IN RANGE
009 1ED2 FE00	CPI	CA0101	/ DO RANGE CHECK
010 1ED4 DAE01E	JC	K20SSX	/ BRANCH OUT-OF-RANGE
011	/		
012 1ED7 7A	K20SS1, MOV	A; D	/ A ← POWER BYTE
013 1ED8 E601	ANI	CATHI	/ ISOLATE LSB
014 1EDA 5F	MOV	E; A	/ E ← POWER STATE

```

015 1EDB 7E          MOV      A;M          / A ← CHARACTER
016 1EDC E6FE        ANI      -1-CATHI     / TURN OFF HIGHLIGHT
017 1EDE B3          ORA      E            / TURN OFF/ON HIGHLIGHT
018 1EDF 77          MOV      M;A          / LOAD IN DISPLAY
019 /
020 1EE0 C9          K20SSX, RET          / EXIT
021 EJECT

001 SUBJOB KEY FUNCTION : KF21 : LED DISPLAY
002 /
003 /***KEY FUNCTION : KF21 : LED DISPLAY
004 /
005 1EE1 CD0423       KF21,  CALL      KU11          / A ← ROW
006 1EE4 FE08        CPI      ASMR0W%:10      / ASSEMBLY AREA?
007 1EE6 CA221F       JZ       KF21X          / YES, EXIT
008 /
009 1EE9 47          MOV,     B;A          / B ← COUNTER
010 1EEA 110800       LXI     D;ROWBKL       / [D,E] ← OFFSET
011 1EED 21ADFD       LXI     H;ROWTAB-ROWBKL / [H,L] ← START OF TABLE
012 /
013 1EF0 19          KF2120, DAD      D          / BUMP POINTER
014 1EF1 3D          DCR      A            / DECREMENT COUNTER
015 1EF2 C2F01E       JNZ     KF2120         / LOOP UNTIL DONE
016 /
017 1EF5 11F8FF       KF2130, LXI     D;-ROWBKL   / [D,E] ← -OFFSET
018 1EF8 0E00        MVI     C;0           / C ← 0
019 /
020 1EFA 7E          KF2140, MOV      A;M          / A ← ROW FLAG
021 1EFB E680        ANI     ROWFSN        / CHECK FOR START FLAG
022 1EFD C20B1F       JNZ     KF2150        / BRANCH ON IT
023 1F00 19          DAD      D            / MOVE POINTER
024 1F01 0D          DCR      C            / DECREMENT COUNT
025 1F02 05          DCR      B            / DECREMENT COUNT
026 1F03 C2FA1E       JNZ     KF2140        / LOOP IF NOT DONE
027 1F06 AF          XRA     A            / Z-BIT ← 1
028 1F07 3C          INR     A            / Z-BIT ← 0
029 1F08 C3221F       JMP     KF21X         / GO TO EXIT
030 EJECT
031 /
032 1F0B 3E07        KF2150, MVI     A;MAXROW    / A ← MAXROW
033 1F0D 81          ADD     C            / A ← TRUE ROW IN NETWORK
034 1F0E 87          SAL     / ROTATE A
035 1F0F 87          SAL     / TO FORM
036 1F10 87          SAL     / FIRST BYTE
037 1F11 87          SAL     / OF ARGUMENT
038 1F12 4F          MOV     C;A          / C ← ROW
039 1F13 3A7EFE       LDA     CURACT        / A ← CURSOR
040 1F16 E60F        ANI     COLMSK        / ISOLATE COLUMN
041 1F18 81          ADD     C            / A ← ROW/COL
042 1F19 3293FE       STA     CMDBUF+3      / STORE IT
043 /
044 1F1C 110570       LXI     D;CMDLED!:100+LENLED / SET PARMS
045 1F1F CD8125       CALL    PIO          / ISSUE COMMAND
046 /
047 1F22 3E1E        KF21X, MVI     A;LEDTMR    / A ← LED TIMER PRESET
048 1F24 3291FD       STA     TMRLED       / STORE TI
049 /
050 1F27 C9          RET          / EXIT
051 EJECT

001 SUBJOB KEY UTILITIES
002 /
003 /***KEY UTILITIES:
004 /
005 / KU01 - RESET/SUPERVISORY STATE TEST
006 / KU02 - SHIFT TEST
007 / KU03 - CLEAR SHIFT
008 / KU04 - ADVISORY MESSAGE
009 / KU05 - ILLEGAL SHIFT
010 / KU06 - RESET DISPLAY AND LOGIC DATA
011 / KU07 - VALIDATE REFERENCE NUMBER
012 / KU08 - INCREMENT/DECREMENT SEQUENCE NUMBER
013 / KU09 - INSERT NODE IN NETWORK
014 / KU10 -
015 / KU11 - A ← ROW

```

```

016 / KU12 - COMPUTE CURSOR POINTERS
017 / KU13 - DISPLAY NODE TYPE
018 / KU14 - DISPLAY TOP LINE - MULTINODE CONTACT
019 / KU15 - CHECK FOR MULTI-NODE CONTACT
020 / KU16 - DISPLAY POWER RAIL
021 / KU17 - COMPUTE MATRIX ADDRESS
022 / KU18 - INCREMENT/DECREMENT MEMORY USAGE
023 / KU19 - EXTEND POWER FROM RAIL
024 / KU20 - CONNECT VERTICALS
025 / KU21 - LOAD TIMERS
026 / KU22 - COMPUTE COLTAB POINTER
027 / FIXVER - FIX LAST VERTICAL CHAR
028 /
029 / EJECT

SUBJOB KEY UTILITY : KU01 : RESET/SUPER TEST
001 /
002 /
003 / ***KEY UTILITY : KU01 : RESET/SUPERVISORY TEST
004 /
005 / ***EXITS:
006 /
007 / TO CALLER IF NO RESET AND NO SUPERVISORY
008 / TO EXEC IF RESET
009 / TO SUPERVISORY IF NO RESET AND SUPER
010 /
011 1F28 F5 KU01, PUSH PSW / STACK CHARACTER
012 1F29 3A7CFE LDA KSTATE / A ← STATE VECTOR
013 1F2C E602 ANI KSUPER / TEST FOR SUPERVISORY
014 1F2E CA3E1F JZ KU0105 / BRANCH IF NOT SUPERVISORY
015 /
016 1F31 3A7CFE LDA KSTATE / TEST FOR RESET
017 1F34 E620 ANI KRESET / ISOLATE FLAG
018 1F36 C2461F JNZ KU0110 / BRANCH ON RESET
019 1F39 F1 POP PSW / RESTORE CHARACTER
020 1F3A C1 POP B / CLEAR EXIT
021 1F3B C3E31C JMP KF1920 / GO TO SUPERVISORY STATE
022 /
023 1F3E 3A7CFE KU0105, LDA KSTATE / A ← STATE VECTOR
024 1F41 E620 ANI KRESET / ISOLATE RESET FLAG
025 1F43 CA471F JZ KU0115 / BRANCH IF NOT
026 /
027 1F46 F1 KU0110, POP PSW / CLEAR CHARACTER
028 /
029 1F47 F1 KU0115, POP PSW / CLEAR STACK
030 1F48 C9 RET / EXIT
031 EJECT

SUBJOB KEY UTILITY : KU02 : TEST FOR SHIFT
001 /
002 /
003 / ***KEY UTILITY : KU02 : TEST FOR SHIFT
004 /
005 / ***ON EXIT:
006 /
007 / Z-BIT. EQ. 1 => CLEAR
008 / Z-BIT. EQ. 0 => SET
009 /
010 1F49 C5 KU02, PUSH B / SAVE [B,C]
011 1F4A 47 MOV B;A / SAVE A
012 1F4B 3A7CFE LDA KSTATE / A ← STATE VECTOR
013 1F4E E680 ANI KSHIFT / TEST FOR SHIFT FLAG
014 1F50 78 MOV A;B / RESTORE A
015 1F51 C1 POP B / RESTORE [B,C]
016 1F52 C9 RET / EXIT
017 EJECT

SUBJOB KEY UTILITY : KU03 : CLEAR SHIFT
001 /
002 /
003 / ***KEY UTILITY : KU03 : CLEAR SHIFT
004 /
005 / ***PRESERVES ALL REGISTERS
006 /
007 1F53 F5 KU03, PUSH PSW / SAVE A
008 1F54 3A7CFE LDA KSTATE / A ← STATE VECTOR
009 1F57 E67F ANI -1-KSHIFT / CLEAR SHIFT FLAG
010 1F59 327CFE STA KSTATE / STORE NEW STATE VECTOR
011 /

```


421

422

```

012 1F5C 3E20          MVI    A;ASCBLK      / A ← BLANK
013 1F5E 3209FD       STA    DSPSHT+1     / CLEAR SHIFT FIELD
014                   /
015 1F61 3E80          MVI    A;FACNOR     / A ← NORMAL FIELD ATTR B
016 1F63 3208FD       STA    DSPSHT       / CLEAR FIELD
017                   /
018 1F66 F1           POP    PSW           / RESTORE A
019 1F67 C9           RET                / EXIT
020                   EJECT

001                   SUBJOB  KEY UTILITY : KU04 : ADVISORY MESSAGE
002                   /
003                   /***KEY UTILITY : KU04 : ADVISORY MESSAGE
004                   /
005                   /***REGISTER USAGE:
006                   /
007                   /      A      - SCRATCH
008                   /      [B,C] - SCRATCH
009                   /      [D,E] - MESSAGE ADDRESS (DESTROYED)
010                   /      [H,L] - SCRATCH
011                   /
012 1F68 D5           KU04,  PUSH    D      / SAVE MESSAGE ADDRESS
013 1F69 210AFD       LXI    H;DSPADV     / [H,L] ← START OF FIELD
014 1F6C 160A         MVI    D;ADVFLD-1  / D ← FIELD LENGTH
015 1F6E CD1903       CALL   ROWN10      / CLEAR FIELD
016 1F71 D1           POP    D            / RESTORE MESSAGE ADDRESS
017 1F72 210BFD       LXI    H;DSPADV+1  / [H,L] ← DESTINATION
018 1F75 CD0301       CALL   MOVSTR      / LOAD FIELD
019 1F78 C9           RET                / EXIT
020                   EJECT

001                   SUBJOB  KEY UTILITY : KU05 : ILLEGAL SHIFT
002                   /
003                   /***KEY UTILITY : KU05 : ILLEGAL SHIFT
004                   /
005 1F79 11801F       KU05,  LXI    D;KU05MS / [D,E] ← MESSAGE ADDR
006 1F7C CD7E05       CALL   ERROR      / SET ERROR STATE
007 1F7F C9           RET                / EXIT
008                   /
009                   /***MESSAGE
010                   /
011 1F80 09           KU05MS, DB    KU05MX
012 1F81 42414420     DA    'BAD SHIFT'
013                   /
014 1F85 53484946     /
015 1F89 54           /
016                   /
017 1F8B 0009       KU05MX= .-KU05MS-1 / MESSAGE LENGTH
018                   EJECT

001                   SUBJOB  KEY UTILITY : KU06 : RESET DISPLAY/LOGIC
002                   /
003                   /***KEY UTILITY : KU06 : RESET DISPLAY AND LOGIC
004                   /
005                   /***USES ALL REGISTERS
006                   /
007 1F8A 21B5FD       KU06,  LXI    H;ROWTAB / [H,L] ← TABLE ADDRESS
008 1F8D 0638         MVI    B;ROWTAB   / B ← TABLE LENGTH
009 1F8F AF           CLA                / A ← 0
010                   /
011 1F90 77           KU0610, MOV   M;A    / CLEAR ENTRY
012 1F91 23           INX    H           / BUMP POINTER
013 1F92 05           DCR    B           / DECREMENT COUNTER
014 1F93 C2901F       JNZ    KU0610     / LOOP UNTIL DONE
015                   /
016 1F96 21EDFD       LXI    H;COLTAB   / [H,L] ← TABLE ADDRESS
017 1F99 0642         MVI    B;COLTAB   / B ← TABLE LENGTH
018                   /
019 1F9B 77           KU0620, MOV   M;A    / CLEAR ENTRY
020 1F9C 23           INX    H           / BUMP POINTER
021 1F9D 05           DCR    B           / DECREMENT COUNTER
022 1F9E C29B1F       JNZ    KU0620     / LOOP UNTIL DONE
023                   /
024 1FA1 212FFE       LXI    H;MATROW   / [H,L] ← TABLE ADDRESS
025 1FA4 064D         MVI    B;MATROL   / B ← TABLE LENGTH
026                   /
027 1FA6 77           KU0630, MOV   M;A    / CLEAR ENTRY

```

423

424

```

028 1FA7 23          INX      H          / BUMP POINTER
029 1FA8 05          DCR      B          / DECREMENT COUNTER
030 1FA9 C2A61F      JNZ      KU0630     / LOOP UNTIL DONE
031
032 1FAC 2108F8      LXI      H,DSPLUG   / [H,L] ← START OF LOGIC
033 1FAF 060E        MVI      B,ROWCNT   / B ← NUMBER OF ROWS *
034
035 1FB1 CD0503      KU0640, CALL    ROWLOG / CLEAR ROW
036 1FB4 05          DCR      B          / DECREMENT COUNTER
037 1FB5 C2B11F      JNZ      KU0640     / LOOP UNTIL DONE
038
039 1FB8 217DFE      LXI      H,CURSORP  / [H,L] ← CURSOR ADDR
040 1FBB 46          MOV      B,M        / B ← CURRENT CURSOR
041 1FBC 0E11        MVI      C,11       / C ← HOME CURSOR
042 1FBE 71          MOV      M,C        / STORE NEW CURSOR
043 1FBF 217EFE      LXI      H,CURACT   / SET DESTINATION
044 1FC2 71          MOV      M,C        / LOAD CURSOR
045 1FC3 CD3B05      CALL    CURSOR     / MOVE CURSOR HOME
046
047 1FC6 D9          RET              / EXIT
048

001                SUBJOB  KEY UTILITY :  KU07 . VALIDATE REFERENCE
002                /
003                /****KEY UTILITY :  KU07 . VALIDATE REFERENCE
004                /
005                /****PARAMETERS.
006                /
007                /          Z-BIT. EQ. 0 => REFERENCE NOT VALID
008                /          Z-BIT. EQ. 1 => REFERENCE VALID
009                /
010                /****REGISTER USAGE:
011                /
012                /          A      - ALLOWABLE TYPES MASK
013                /          [B,C] - SCRATCH
014                /          [D,E] - MESSAGE ADDRESS ON ERROR
015                /          [H,L] - REFERENCE TYPE
016                /
017                EJECT

001 1FC7 47          KU07,  MOV      B:A          / A ← REFERENCE TYPES
002
003                /          SET THE "TO BE CLEARED" FLAG FOR NUMERIC
004                /          AREA OF ASSEMBLY AREA
005
006 1FC8 3A7CPE      LDA      KSTATE     / GET FLAGS BYTE
007 1FCB F610        ORI      KCLEAR     / SET "TO BE CLRED"
008 1FCD 327CFE      STA      KSTATE     / X
009
010                /          NOW DECIDE WHAT TYPE OF * WE HAVE
011
012 1FD0 3A01FD      LDA      DSPNUM+3   / A ← FIRST REFERENCE D'3I
013
014 1FD3 FE30        CPI      ASC0       / CHECK FOR OUTPUT/CONSTANT
015 1FD5 CAF01F      JZ      KU0700     / BRANCH ON OUTPUT/CONSTANT
016 1FD8 FE31        CPI      ASC1       / CHECK FOR INPUT
017 1FDA CA0B20      JZ      KU0701     / BRANCH ON INPUT
018 1FDD FE32        CPI      ASC2       / CHECK FOR SEQUENCER
019 1FDF CA1420      JZ      KU0702     / BRANCH ON SEQUENCER
020 1FE2 FE33        CPI      ASC3       / CHECK FOR INPUT REGISTER
021 1FE4 CA1D20      JZ      KU0703     / BRANCH ON INPUT REGISTER
022 1FE7 FE34        CPI      ASC4       / CHECK FOR HOLDING REG
023 1FE9 CA2620      JZ      KU0704     / BRANCH ON HOLDING REG
024 1FEC FE20        CPI      ASCBLK    / CHECK FOR BLANK
025 1FEE CA2F20      JZ      KU0705     / BRANCH ON BLANK
026
026 1FF1 11411B      KU07ER, LXI      D,KF14M1 / [D,E] ← MESSAGE ADDR
027 1FF4 CD7E05      CALL    ERROR      / SET ERROR STATE
028 1FF7 AF          CLA              / A ← 0
029 1FF8 3C          INR      A          / Z ← 0
030 1FF9 C32021      JMP      KU07X     / EXIT
031                EJECT

```

```

001 /
002 /***OUTPUT/CONSTANT CHECK
003 /
004 1FFC 78 KU0700, MOV A;B / A ← MASK
005 1FFD E601 ANI NODOUT / OUTPUTS ALLOWED?
006 1FFF C23820 JNZ KU0710 / YES, CONTINUE
007 2002 78 MOV A;B / A ← MASK
008 2003 E620 ANI NODCST / CONSTANTS ALLOWED?
009 2005 C20021 JNZ KU0770 / YES, CONTINUE
010 2008 C3F11F JMP KU07ER / ERROR
011 /
012 /***INPUT CHECK
013 /
014 200B 78 KU0701, MOV A;B / A ← MASK
015 200C E602 ANI NODINP / INPUTS ALLOWED?
016 200E C23820 JNZ KU0710 / YES, CONTINUE
017 2011 C3F11F JMP KU07ER / NO, ERROR
018 /
019 /***SEQUENCER CHECK
020 /
021 2014 78 KU0702, MOV A;B / A ← MASK
022 2015 E604 ANI NODSEQ / SEQUENCER ALLOWED?
023 2017 C27C20 JNZ KU0730 / YES, CONTINUE
024 201A C3F11F JMP KU07ER / NO, ERROR
025 /
026 /***INPUT REGISTER CHECK
027 /
028 201D 78 KU0703, MOV A;B / A ← MASK
029 201E E608 ANI NODIRG / INPUT REGISTER ALLOWED?
030 2020 C2B220 JNZ KU0740 / YES, CONTINUE
031 2023 C3F11F JMP KU07ER / NO, ERROR
032 /
033 /***HOLDING REGISTER CHECK
034 /
035 2026 78 KU0704, MOV A;B / A ← MASK
036 2027 E610 ANI NODHRG / HOLDING REGISTER ALLOWED?
037 2029 C2D420 JNZ KU0745 / YES, CONTINUE
038 202C C3F11F JMP KU07ER / NO, ERROR
039 /
040 /***BLANK FIELD CHECK
041 /
042 202F 78 KU0705, MOV A;B / A ← MASK
043 2030 E640 ANI NODBLK / BLANKS ALLOWED?
044 2032 C20C21 JNZ KU0780 / YES, CONTINUE
045 2035 C3F11F JMP KU07ER / NO, ERROR
046 EJECT

```

***** COILS AND INPUTS (OXXX/1XXX)**

```

002 /
003 / VALIDATE THE 3 L.S. DIGITS
004 /
005 2038 210000 KU0710, LXI H;0 / [H,L] ← 0
006 203B 1102FD LXI D;DSPNUM+4 / [D,E] ← ADDRESS OF RE
007 203E CD8E01 CALL BCDBN3 / CONVERT TO BINARY
008 2041 2B DCX H / MAKE RELATIVE TO BASE 0
009 2042 7C MOV A;H / GET MS BYTE OF REF:
010 / IF 0, OKAY FOR BOTH
011 / IF 1, MUST BE OUTPUT
012 / IF > 1, ERROR!
013 2043 B7 TST / SEE IF ZERO:
014 2044 CA5320 JZ KU0713 / YES, OKAY 0,1XXX
015 2047 3D DCR A / NOT 0, SEE IF 1 OR
016 2048 C2F11F JNZ KU07ER / >1, ERROR!
017 /
018 / # IS >256; SO MUST BE OUTPUT COIL OR ELSE ERROR
019 /
020 204B 3A01FD LDA DSPNUM+3 / GET MS DIGIT
021 204E FE30 CPI ASC0 / IS IT 0 (FOR OUTPUT)
022 2050 C2F11F JNZ KU07ER / NO, ERROR
023 /
024 KU0713,
025 2053 E5 PUSH H / SAVE BINARY VALUE (ORIG)
026 /
027 / REF IS 512/256 OR LESS; SO SEE IF IT IS VALID

```



```

028 / BY THE CURRENT 484 CONFIG OF I/O
029
030 2054 3A85FE LDA SCONF2 / GET I/O CONFIG BITS
031 2057 210001 LXI H, @256 / MAX I/O
032 205A 11C0FF LXI D, -@64 / SET NEG STEP SIZE
033 KU0715,
034 205D 17 RAL / GET THE CONFIG BIT
035 205E DA6520 JC KU0717 / OK! FOUND SIZE
036 2061 19 DAD D / NOT FOUND, REDUCE M X
037 2062 C35D20 JMP KU0715 / AND LOOP TIL FOUND
038
039 / NOW REGISTER L = MAX # OF I/O POINTS:
040 / (00=256; C0=192; 80=128; 40=64)
041
042 KU0717,
043 2065 7D MOV A, L / GET MAX #
044 2066 E1 POP H / GET ORIG BIN REF #
045 2067 B7 TST / IF MAX=0, ALL OK
046 2068 CA7020 JZ KU0719 / OK, HAVE MAX
047 EJECT
048 / NOT MAX, SO CHECK REF TO SYS LIMIT
049
050 206B 3D DCR A / (MAKE CARRY CHECK WORK
051 206C BD CMP L / MATCH TO LS BYTE ONLY
052 / (MS IS IRRELEVANT)
053 206D DAF11F JC KU07ER / BAD #!; ERROR
054
055 / SUCCESS!!! REF # OK (LEAST SIG 3 DIGITS)
056
057 KU0719,
058 2070 3A01FD LDA DSPNUM+3 / A ← TYPE
059 2073 FE30 CPI ASC0 / CHECK FOR COILS
060 2075 C21F21 JNZ KU0799 / BRANCH IF NOT
061 2078 24 INR H / SET FLAG FOR OUTPUT/INT
062 2079 C31F21 JMP KU0799 / GO TO SUCCESS EXIT
063 EJECT

001 /
002 /***SEQUENCER REFERENCE (2YXX)
003 /
004 207C 3A85FE KU0730, LDA SCONF2 / A ← CONFIG BYTE
005 207F E602 ANI SYSENH / ENHANCED SET?
006 2081 CAF11F JZ KU07ER / NO, ERROR
007 2084 3A02FD LDA DSPNUM+4 / A ← REGISTER TYPE
008 2087 D630 SUI ASC0 / CONVERT TO BINARY
009 2089 FAF11F JM KU07ER / BRANCH ON ERROR
010 208C FE09 CPI :09 / CHECK HIGH RANGE
011 208E F3F11F JP KU07ER / BRANCH ON ERROR
012 2091 3D DCR A / CREATE BASE 0 REF
013 2092 0F RRC / ROTATE TO FORM
014 2093 0F RRC / MASK FOR FINAL
015 2094 0F RRC / RESULT
016 2095 F5 PUSH PSW / STACK IT
017 /
018 2096 210000 LXI H, 0 / [H, L] ← 0
019 2099 1103FD LXI D, DSPNUM+5 / [D, E] ← BCD FIELD ADDR
020 209C CD9801 CALL BCDBN2 / CONVERT TO BINARY
021 /
022 209F 2B DCX H / DECREMENT RESULT
023 20A0 3EE0 MVI A, REGMSK / A ← MASK
024 20A2 A5 ANA L / CHECK RESULT
025 20A3 CAAA20 JZ KU0735 / BRANCH OKAY
026 20A6 F1 POP PSW / REFERENCE TOO LARGE
027 20A7 C3F11F JMP KU07ER / GO TO ERROR
028 /
029 20AA F1 KU0735, POP PSW / A ← REGISTER VALUE
030 20AB B5 ORA L / SET BITS
031 20AC 6F MOV L, A / L ← BYTE 1
032 20AD 2603 MVI H, SEQFLG / H ← BYTE 0
033 20AF C31F21 JMF KU0799 / GO TO EXIT
034 EJECT

```

*** INPUT REGISTER REFERENCE (30XX); X)

```

002 /
003 20B2 3A85FE KU0740, LDA SCONF2 / A ← CONFIGURATION
004 20B5 E602 ANI SYSENH / CHECK FOR ENHANCED EXEC

```

```

005 20B7 DAF11F      JZ      KU07ER      / BRANCH IF NOT
006
007 20BA 210000      LXI     H;0          / [H,L] ← 0
008 20BD 1103FD      LXI     D;DSPNUM+4  / [D,E] ← START OF FIELD
009 20C0 CD8E01      CALL    BCDBN3     / CONVERT REF TO BINARY
010
011 20C3 2B         DCX     H           / MAKE RELATIVE BASE 0
012 20C4 AF         CLA                     / A ← 0
013 20C5 BC         CMP     H           / H MUST BE ZERO
014 20C6 C2F11F      JNZ     KU07ER     / BRANCH IF NOT
015 20C9 7D         MOV     A;L        / A ← LOW-ORDER BYTE
016 20CA D620       SUI     @32        / MUST BE < 31
017 20CC F2F11F      JP      KU07ER     / BRANCH IF NOT
018 20CF 2601       MVI     H;INPFLG   / SET INPUT REGISTER FLA
019 20D1 C31F21     JMP     KU0799     / GO TO EXIT
020
EJECT

```

*** HOLDING REGISTER REFERENCE (4XXX)

```

002
003 20D4 210000      KU0745, LXI     H;0          / [H,L] ← 0
004 20D7 1103FD      LXI     D;DSPNUM+4  / [D,E] ← BCD FIELD
005 20DA CD8E01      CALL    BCDBN3     / CONVERT TO BINARY
006
007 20DD 2B         DCX     H           / MAKE RELATIVE BASE 0
008 20DE 7C         MOV     A;H        / GET MS BYTE. IT
009
010
011
012 20DF B7         TST                     / MUST BE 0, OR ELSE
013 20E0 C2F11F      JNZ     KU07ER     / ERROR.
014 20E3 E5         PUSH    H           / ZERO? (IE 256 OR LESS?)
015
016
017
018 20E4 3A85FE      LDA     SCONF2     / NO, ERROR
019 20E7 210001      LXI     H;@256    / YES, SAVE ORIG #
020 20EA 11C0FF      LXI     D; -@64   /
021
022 20ED 17         RAL                     / OKAY, LEAST SIG 3 DIGITS ARE 1-256;
023 20EE DAF520      JC      KU0753    / NOW CHECK TO CONFIG SIZE
024 20F1 19         DAD     D           /
025 20F2 C3ED20     JMP     KU0750    /
026
027
028
029
030
031
032
033 20F5 3EFD      KU0750, MVI     A;-3     / GET I/O CONFIG BITS
034
035 20F7 85         ADD     L           / MAX POSS I/O
036
037 20F8 E1         POP     H           / NEG STEP SIZE
038 20F9 ED         CMP     L           /
039 20FA DAF11F      JC      KU0753    / GET NEXT CONFIG BIT
040 20FD C31F21     JMP     KU0799     / OK! FOUND SIZE
041
EJECT
/ NOT FOUND, REDUCE MAX
/ AND LOOP TIL FOUND
/
/ NOW L= MAX # OF I/O POINTS
/ (00=256, 00=192, 80=128, 40=64)
/ NOW SUBTRACT 2 BECAUSE THE "LAST 2"
/ HOLDING REGS ARE NOT AVAILABLE TO USER.
/
032
KU0753,
033 20F5 3EFD      MVI     A;-3     / -2 FOR THE SYSTEM
034
035 20F7 85         ADD     L           / (-1 FOR CARRY TEST)
036
037 20F8 E1         POP     H           / NOW A=MAX 4XXX FOR THIS
038 20F9 ED         CMP     L           / CONTROLLER
039 20FA DAF11F      JC      KU07ER     / GET ORIG #
040 20FD C31F21     JMP     KU0799     / MATCH ON LS BYTE
041
EJECT
/ BAD #, ERROR
/ GOOD, EXIT OK
/
001
002
003
004 2100 1101FD      KU0770, LXI     D;DSPNUM+3 / [D,E] ← BCD POINTER
005 2103 210000      LXI     H;0          / INITIALIZE RESULT
006 2106 CD8101      CALL    BCDBN4     / CONVERT TO BINARY
007 2109 C31F21     JMP     KU0799     / AND EXIT
008
009
010
011 210C 2101FD      KU0780, LXI     H;DSPNUM+3 / [H,L] ← SOURCE ADDR
012 210F 3E20       MVI     A;ASCBLK   / A ← ASCII BLANK
013 2111 0604       MVI     B;REFLEN   / B ← FIELD LENGTH
014
015 2113 BE         KU0785, CMP     M           / CHECK CHARACTER
016 2114 C2F11F      JNZ     KU07ER     / BRANCH ON ERROR
017 2117 23         INX     H           / BUMP POINTER

```

431

432

```

018 2118 05          DCR      B          / DECREMENT COUNT
019 2119 021321     JNZ      KU0785       / LOOP UNTIL DONE
020                /
021 211C 210000     LXI      H,0        / [H,L] ← 0
022                /
023 211F AF         KU0799, CLA       / A ← 0; Z-BIT ← 1
024                /
025 2120 C9        KU07X, RET        / EXIT
026                EJECT

001                SUBJOB  KEY UTILITY : KU08 : INC/DCR STEP NUMBER
002                /
003                /***KEY UTILITY : KU08 : INC/DCR STEP NUMBER
004                /
005                /***PARAMETERS:
006                /
007                /          [B,C].EQ. +1    => INCREMENT
008                /          [B,C].EQ. -1    => DECREMENT
009                /
010                /***REGISTER USAGE:
011                /
012                /          A      - SCRATCH
013                /          [B,C] - INCREMENT/DECREMENT
014                /          [D,E] - SCRATCH
015                /          [H,L] - SCRATCH
016                /
017 2121 218AFE     KU08,  LXI      H,STPNUM  / [H,L] ← ADDRESS
018 2124 E7         GETHL                    / [H,L] ← STEP NUMBER
019 2125 09        DAD      B          / MODIFY STEP NUMBER
020 2126 EB        XCHG                    / [D,E] ↔ [H,L]
021 2127 218AFE     LXI      H,STPNUM  / [H,L] ← ADDRESS
022 212A EF        MOVDE                    / UPDATE STEP NUMBER
023 212B EB        XCHG                    / [D,E] ↔ [H,L]
024 212C 1118FD     LXI      D,DSPSTP  / [D,E] ← DESTINATION
025 212F CDC201     CALL     BNBCD4      / CONVERT TO BCD AND DISPLA

026 2132 C9        RET                    / EXIT
027                EJECT

001                SUBJOB  KEY UTILITY : KU09 : INSERT NODE
002                /
003                /***KEY UTILITY : KU09 : INSERT NODE
004                /
005                /***PARAMETERS:
006                /
007                /          Z-BIT.EQ. 0 => ERROR
008                /          Z-BIT.EQ. 1 => INSERT COMPLETED
009                /
010                /          NODE IN COMMAND BUFFER
011                /          INSERT AT CURSOR (CURACT)
012                /
013                /***REGISTER USAGE:
014                /
015                /          A      - SCRATCH
016                /          B      - SCRATCH
017                /          C      - NODE COUNT
018                /          [D,E] - SCRATCH
019                /          [H,L] - SCRATCH
020                /
021                EJECT

001 2133 C5        KU09,  PUSH     B          / SAVE COUNTER
002 2134 CD5124     CALL     KU22         / [H,L] ← COLTAB POINTER
003 2137 E5        PUSH     H          / STACK POINTER
004 2138 3A7EFE     LDA      CURACT       / A ← CURSOR
005 213B E6F0      ANI      ROWMSK      / ISOLATE ROW
006 213D FE10      CPI      :10         / CHECK FOR FIRST ROW
007 213F C2AF21     JNZ      KU0930       / BRANCH IF NOT
008                /
009 2142 3A05FD     LDA      DSPVER+ROWD  / A ← VERTICAL
010 2145 FE20      CPI      ASCBLK      / CHECK FOR BLANK
011 2147 C27921     JNZ      KU0915       / BRANCH ON VERTICAL
012                /
013 214A 218EFE     LXI      H,ADREUN    / [H,L] ← SOURCE
014 214D E7         GETHL                    / [H,L] ← LAST ADDRESS
015 214E EB        XCHG                    / SWAP

```


433

434

```

016 214F 13          INX      D          / BUMP ADDRESS
017 2150 13          INX      D          / FOR NEW NODE
018 2151 2193FE     LXI      H;CMBUF+3    / [H,L] ← COMMAND BUF P R
019 2154 EF          MOVDE                     / STORE ADDRESS
020
021 2155 41          MOV      B;C          / B ← COUNT
022 2156 2193FE     LXI      H;CMBUF+3    / [H,L] ← POINTER
023 2159 110200     LXI      D;2          / [D,E] ← OFFSET
024
025 215C 19          KU0910, DAD      D          / BUMP POINTER
026 215D 05          DCR      B          / DECREMENT COUNTER
027 215E C25C21     JNZ      KU0910      / LOOP UNTIL DONE
028
029 2161 7E          MOV      A;M          / A ← DATAI
030 2162 F680       ORI      EOCFLG      / SET END-OF-COL FLAG
031 2164 77          MOV      M;A          / STORE BACK INTO BUFFER
032
033 2165 3E50       MVI      A;CMDINS     / A ← COMMAND CODE
034 2167 CDF822     CALL     KU9SUB       / DO FUNCTION
035 216A E1         POP      H          / POP POINTER
036 216B C1         POP      B          / POP COUNTER
037 216C C27621     JNZ      KU09ER      / BRANCH ON FAILURE
038 216F CDB824     CALL     COLINC      / UPDATE COLTAB
039 2172 CDCF22     CALL     KU09UP      / UPDATE MATROW AND USEA E
040 2175 C9         RET                      / DONE
041
042 2176 AF          KU09ER, CLA          / A ← 0
043 2177 3C          INR      A          / A ← 1; Z-BIT ← 0
044 2178 C9         RET                      / GO TO EXIT
045
001
002 /
003 /
004 2179 218EFE     KU0915, LXI      H;ADREON    / [H,L] ← SOURCE
005 217C E7         GETHL                     / [H,L] ← LAST ADDR
006 217D EB         XCHG                      / SWAP
007 217E 13         INX      D          / BUMP ADDRESS
008 217F 13         INX      D          / FOR NEW NODE
009 2180 2193FE     LXI      H;CMBUF+3    / [H,L] ← DESTINATION
010 2183 EF          MOVDE                     / STORE ADDR IN BUFFER
011
012 2184 41          MOV      B;C          / B ← COUNT
013
014 2185 23          KU0920, INX      H          / BUMP POINTER
015 2186 23          INX      H          / TO FIND EOC SPOT
016 2187 05          DCR      B          / DECREMENT COUNTER
017 2188 C28521     JNZ      KU0920      / LOOP UNTIL DONE
018
019 218B 114008     LXI      D;NOEOC!:400+:40 / [D,E] ← EOC NODE
020 218E EF          MOVDE                     / LOAD BUFFER
021
022 218F E1         POP      H          / GET POINTER
023 2190 C1         POP      B          / GET COUNTER
024 2191 0C         INR      C          / INCREMENT COUNT FOR EOC
025 2192 C5         PUSH     B          / STACK COUNTER
026 2193 E5         PUSH     H          / STACK POINTER
027 2194 3E50       MVI      A;CMDINS     / A ← COMMAND CODE
028 2196 CDF822     CALL     KU9SUB       / DO INSERT
029 2199 E1         POP      H          / GET POINTER
030 219A C1         POP      B          / GET COUNTER
031 219B C27621     JNZ      KU09ER      / BRANCH ON ERROR
032
033 219E E5         PUSH     H          / SAVE POINTER
034 219F 110400     LXI      D;EOCHI      / [D,E] ← OFFSET
035 21A2 19         DAD      D          / [H,L] ← EOC ADDR
036 21A3 114008     LXI      D;NOEOC!:400+:40 / [D,E] ← EOC NODE
037 21A6 EF          MOVDE                     / LOAD TABLE
038 21A7 E1         POP      H          / GET POINTER
039 21A8 CDB824     CALL     COLINC      / UPDATE COLTAB
040 21AB CDCF22     CALL     KU09UP      / UPDATE MATROW AND USEA E
041 21AE C9         RET                      /
042

```

```

001 /
002 /***INSERT IN EXISTING COLUMN
003 /
004 21AF 110400 KU0930, LXI D; EDCI / [B,C] ← OFFSET
005 21B2 19 DAD D / POINT TO EOC NODE
006 21B3 AF CLA / A ← 0
007 21B4 BE CMP M / CHECK IF ANY VERTICALS
008 21B5 C23322 JNZ KU0940 / BRANCH ON VERTICALS
009 /
010 21B8 3A05FD LDA DSPVER+ROWD / A ← VERTICAL FOR NODE
011 21BB FE20 CPI ASCBLK / ANY VERTICAL?
012 21BD C2ED21 JNZ KU0935 / YES, PROCESS
013 /
014 21C0 E1 POP H / [H,L] ← POINTER
015 21C1 E5 PUSH H / SAVE IT
016 21C2 110200 LXI D; COLLEHI / [B,C] ← OFFSET
017 21C5 19 DAD D / BUMP TO LAST ADDRESS
018 21C6 E7 GETHL / [H,L] ← LAST ADDRESS
019 21C7 23 INX H / INSERT AFTER
020 21C8 23 INX H / THIS ADDRESS
021 21C9 EB XCHG / SWAP
022 21CA 2193FE LXI H; CMDBUF+3 / POINTER TO CMDBUF
023 21CD EF MOVDE / STORE ADDRESS IN CMDBUF
024 /
025 21CE 41 MOV B;C / B ← COUNT
026 /
027 21CF 05 KU0931, DCR B / DECREMENT COUNTER
028 21D0 CAD821 JZ KU0932 / EXIT WHEN DONE
029 21D3 23 INX H / BUMP
030 21D4 23 INX H / POINTER
031 21D5 C3CF21 JMP KU0931 / AND LOOP
032 /
033 21D8 7E KU0932, MOV A;M / A ← DATA HI
034 21D9 F680 ORI EOCFLG / SET EOC FLAG
035 21DB 77 MOV M;A / AND RELOAD BUFFER
036 /
037 21DC 3EB0 MVI A; CMDINC / A ← COMMAND CODE
038 21DE CDF822 CALL KU9SUB / INSERT NODE(S)
039 21E1 E1 POP H / GET POINTER
040 21E2 C1 POP B / GET COUNTER
041 21E3 C27621 JNZ KU09ER / BRANCH ON ERROR
042 21E6 CDB824 CALL COLINC / UPDATE COLTAB
043 21E9 CDCF22 CALL KU09UP / UPDATE MATROW AND USAGE
044 21EC C9 RET / DONE
045 EJECT

001 /
002 /***INSERT IN COLUMN - NEW VERTICAL
003 /
004 21ED 2B KU0935, DCX H / DECREMENT
005 21EE 2B DCX H / POINTER
006 21EF E7 GETHL / [H,L] ← ADDRESS
007 21F0 23 INX H / INSERT AFTER
008 21F1 23 INX H / THIS ADDRESS
009 21F2 EB XCHG / SWAP
010 21F3 2193FE LXI H; CMDBUF+3 / POINTER TO CMDBUF
011 21F6 EF MOVDE / STORE ADDRESS IN CMDBUF
012 21F7 CD0423 CALL KU11 / A ← ROW
013 21FA 47 MOV B;A / B ← COUNTER
014 21FB 3E80 MVI A; 80 / A ← MASK
015 /
016 21FD 0F KU0936, RRC / SHIFT MASK
017 21FE 05 DCR B / DECREMENT COUNT
018 21FF C2FD21 JNZ KU0936 / LOOP UNTIL DONE
019 /
020 2202 1608 MVI D; NOEOC!! 04 / D ← EOC NODE
021 2204 5F MOV E;A / E ← MASK
022 /
023 2205 41 MOV B;C / B ← COUNT
024 2206 2195FE LXI H; CMDBUF+5 / [H,L] ← POINTER
025 /
026 2209 23 KU0937, INX H / BUMP
027 220A 23 INX H / POINTER
028 220B 05 DCR B / DECREMENT COUNT
029 220C C20922 JNZ KU0937 / LOOP UNTIL DONE
030 220F EF MOVDE / LOAD EOC NODE IN BUFFER
031 /

```


032	2210	E1	POP	H	/ GET POINTER
033	2211	C1	POP	B	/ GET COUNT
034	2212	0C	INR	C	/ ACCOUNT FOR EOC NODE
035	2213	43	MOV	B;E	/ BACK UP MASK
036	2214	C5	PUSH	B	/ STACK COUNTER
037	2215	E5	PUSH	H	/ STACK POINTER
038					
039	2216	3E80	MVI	A;CMDINC	/ A ← COMMAND CODE
040	2218	CDF822	CALL	KU9SUB	/ DO INSERT
041	221B	E1	POP	H	/ GET POINTER
042	221C	C1	POP	B	/ GET COUNT
043	221D	C27621	JNZ	KU09ER	/ BRANCH ON ERROR
044					
045	2220	E5	PUSH	H	/ SAVE POINTER
046	2221	110400	LXI	D;EOCHI	/ [D,E] ← OFFSET
047	2224	19	DAD	D	/ [H,L] ← EOC DATA
048	2225	3608	MVI	M;NOEOD1:04	/ LOAD EOC NODE DATA HI
049	2227	23	INX	H	/ BUMP POINTER
050	2228	70	MOV	M;B	/ LOAD CONNECTIVITY
051	2229	0600	MVI	B;0	/ ZERO COUNT
052	222B	E1	POP	H	/ GET POINTER
053	222C	CDB824	CALL	COLINC	/ UPDATE COLTAB
054	222F	CDCF22	CALL	KU09UP	/ UPADATE MATROW AND USE GE
055	2232	C9	RET		/ DONE
056			EJECT		
001					
002					/**/INSERT IN COLUMN - EXISTING VERTICAL
003					
004	2233	2B	KU0940, DCX	H	/ DECREMENT POINTER
005	2234	2B	DCX	H	/ TO EOC ADDRESS
006	2235	E7	GETHL		/ [H,L] ← EOC ADDRESS
007	2236	3A05FD	LDA	DSPVER+ROWD	/ A ← VERTICAL
008	2239	FE20	CPI	ASCBLK	/ CHECK FOR NEW VERTICAL
009	223B	C25422	JNZ	KU0950	/ BRANCH ON NEW VERTICAL
010					
011	223E	EB	XCHG		/ SWAP
012	223F	2193FE	LXI	H;CMDBUF+3	/ [H,L] ← POINTER
013	2242	EF	MOVDE		/ STORE ADDRESS
014					
015	2243	3E50	MVI	A;CMDINS	/ A ← COMMAND TYPE
016	2245	CDF822	CALL	KU9SUB	/ DO INSERT
017	2248	E1	POP	H	/ GET POINTER
018	2249	C1	POP	B	/ GET COUNT
019	224A	C27621	JNZ	KU09ER	/ BRANCH ON ERROR
020	224D	CDB824	CALL	COLINC	/ UPDATE COLTAB
021	2250	CDCF22	CALL	KU09UP	/ UPDATE MATROW AND USEA E
022	2253	C9	RET		/ DONE
023			EJECT		
001					
002					/**/INSERT IN COLUMN - ADD NEW VERTICAL
003					
004	2254	EB	KU0950, XCHG		/ SWAP
005	2255	2193FE	LXI	H;CMDBUF+3	/ [H,L] ← POINTER
006	2258	EF	MOVDE		/ STORE ADDRESS
007					
008	2259	E1	POP	H	/ GET POINTER
009	225A	E5	PUSH	H	/ SAVE IT AGAIN
010	225B	110500	LXI	D;EOCLO	/ [D,E] ← OFFSET
011	225E	19	DAD	D	/ [H,L] ← CONNECTIVITY BYT
012					
013	225F	CD0423	CALL	KU11	/ A ← ROW
014	2262	47	MOV	B;A	/ B ← ROW
015	2263	3E80	MVI	A;:80	/ B ← MASK
016					
017	2265	0F	KU0955, RRC		/ ROTATE MASK
018	2266	05	DCR	B	/ DECREMENT COUNT
019	2267	C26522	JNZ	KU0955	/ LOOP UNTIL DONE
020					
021	226A	5F	MOV	E;A	/ E ← DATA0
022	226B	1600	MVI	D;0	/ D ← DATA1
023	226D	D5	PUSH	D	/ SAVE CONNECTIVITY DATA
024					
025	226E	3E50	MVI	A;CMDINS	/ A ← COMMAND CODE

026 2270 CDF822	CALL	KU9SUB	/ DO INSERT
027 2273 D1	POP	D	/ POP CONNECTIVITY DATA
028 2274 E1	POP	H	/ POP POINTER
029 2275 C1	POP	B	/ POP COUNTER
030 2276 C0	RNZ		/ EXIT ON ERROR
031			
032 2277 CDB824	CALL	COLINC	/ UPDATE COLTAB
033 227A C5	PUSH	B	/ STACK COUNTER
034 227B E5	PUSH	H	/ STACK POINTER
035 227C D5	PUSH	D	/ STACK CONNECTIVITY BYT
036			
037 227D CDCF22	CALL	KU09UP	/ UPDATE "MATROW"
038			
039 2280 D1	POP	D	/ RELOAD CONNECT
040			
041 2281 E1	POP	H	/ RELOAD PTR
042 2282 E5	PUSH	H	/ X
043			
044 2283 D5	PUSH	D	/ SAVE CONNECTIV.
045			
046 2284 010200	LXI	B, COLEHI	/ [B, C] ← OFFSET
047 2287 09	DAD	B	/ [H, L] ← ADDR OF EOC
048 2288 E7	GETHL		/ [H, L] ← EOC ADDR
049 2289 42	MOV	B; D	/ B ← DATAHI
050 228A 4B	MOV	C; E	/ C ← DATALO
051 228B EB	XCHG		/ SWAP
052 228C 2193FE	LXI	H; CMDBUF+3	/ SET DESTINATION
053 228F EF	MOVDE		/ LOAD ADDRESS
054 2290 D7	MOVBC		/ LOAD DATA
055 2291 06FF	MVI	B; FF	/ B ← MASKHI
056 2293 79	MOV	A; C	/ A ← DATALO
057 2294 2F	CMA		/ COMPLEMENT
058 2295 4F	MOV	C; A	/ C ← MASKLO
059 2296 D7	MOVBC		/ LOAD MASK
060			
061 2297 110A21	LXI	D; CMDWRT+CMDO2!; 100+LENWRT	/ SET PARMS
062 229A CD8125	CALL	PIO	/ UPDATE EOC NODE
063 229D D1	POP	D	/ GET CONNECTIVITY BYTE
064 229E E1	POP	H	/ GET POINTER
065 229F C1	POP	B	/ GET COUNTER
066 22A0 C2AC22	JNZ	KU0960	/ BRANCH ON ERROR
067			
068 22A3 010500	LXI	B; EOCLO	/ [B, C] ← OFFSET
069 22A6 09	DAD	B	/ [H, L] ← CONNECTIVITY BYT
070 22A7 7E	MOV	A; M	/ A ← OLD CONNECTIVITY BYT
071 22A8 B3	ORA	E	/ SET NEW BIT
072 22A9 77	MOV	M; A	/ LOAD NEW CONNECTIVITY BYT
073 22AA AF	CLA		/ A ← 0
074 22AB C9	RET		/ AND EXIT
075	EJECT		
001			
002			
003			
004			
005 22AC C5	PUSH	B	/ SAVE COUNT
006 22AD E5	PUSH	H	/ SAVE COLTAB PTR
007 22AE 110200	LXI	D; COLEHI	/ [D, E] ← OFFSET
008 22B1 19	DAD	D	/ [H, L] ← ADDRESS OF DATA
009 22B2 E7	GETHL		/ [H, L] ← EOC ADDRESS
010 22B3 2B	DCX	H	/ DECREMENT
011 22B4 2B	DCX	H	/ ADDRESS
012 22B5 EB	XCHG		/ SWAP
013 22B6 2193FE	LXI	H; CMDBUF+3	/ SET DESTINATION
014 22B9 EF	MOVDE		/ LOAD BUFFER
015			
016 22BA 3E60	MVI	A; CMDDEL	/ A ← FUNCTION CODE
017 22BC 81	ADD	C	/ A ← COMMAND CODE
018 22BD 57	MOV	D; A	/ D ← COMMAND CODE
019 22BE 79	MOV	A; C	/ A ← NODE COUNT
020 22BF 3D	DCR	A	/ MAKE ZERO RELATIVE
021 22C0 87	ADD	A	/ DOUBLE IT

441

442

```

022 22C1 C606      ADI      LENDEL      / A ← COMMAND LENGTH
023 22C3 5F        MOV      E;A        / E ← COMMAND LENGTH
024 22C4 CD8125    CALL     PIO        / DELETE NODE(S)
025
026 22C7 E1        POP      H          / GET COLTAB PTR
027 22C8 C1        POP      B          / GET COUNTER
028 22C9 CD0025    CALL     COLDEC     / REPAIR COLTAB
029 22CC AF        CLA          / A ← 0
030 22CD 3C        INR      A          / Z-BIT ← 0
031 22CE C9        RET          / AND EXIT
032 EJECT

001 SUBJOB KU09UP = UPDATE "MATROW" AND USEAGE COUNT
002 /
003 /
004 / C = # OF NODES ADDED
005 /
006 KU09UP,
007 22CF 3A95FE    LDA      CMDBUF+5   / A ← DATAI
008 22D2 E67C      ANI      NODMSK    / ISOLATE NODE TYPE
009 22D4 0F        RRC          / ROTATE
010 22D5 0F        RRC          / RIGHT
011 22D6 3280FE    STA      CURCON    / AND SET TYPE
012 /
013 22D9 CDAA23    CALL     KU17      / GET MATRIX POINTERS
014 22DC 1195FE    LXI      D;CMDBUF+5 / [D,E] ← POINTER
015 /
016 22DF 1A        KU099A, LDAX     D          / A ← BYTE 0
017 22E0 E67C      ANI      NODMSK    / ISOLATE NODE TYPE
018 22E2 0F        RRC          / ROTATE
019 22E3 0F        RRC          / NODE TYPE
020 22E4 FE02      CPI      NOEOC     / CHECK FOR AN EOC
021 22E6 CAEA22    JZ       KU099B    / BRANCH ON EOC
022 22E9 77        MOV      M;A       / LOAD IN MATRIX
023 /
024 22EA 13        KU099B, INX      D          / BUMP
025 22EB 13        INX      D          / POINTER
026 22EC C5        PUSH     B         / SAVE COUNTER
027 22ED 010B00    LXI      B;MAXCOL  / [B,C] ← OFFSET
028 22F0 09        DAD     B          / BUMP TO NEXT COLUMN
029 22F1 C1        POP      B         / GET COUNTER
030 22F2 0D        DCR      C         / DECREMENT COUNTER
031 22F3 C2DF22    JNZ     KU099A     / LOOP UNTIL DONE
032 /
033 22F6 AF        CLA          / Z-BIT ← 1
034 /
035 22F7 C9        RET          / EXIT
036 EJECT

001 SUBJOB KU9SUB = SEND I/O COMMAND
002 /
003 /***SUBROUTINE TO SEND COMMAND
004 /
005 /***REGISTER USAGE:
006 /
007 / A - COMMAND CODE
008 / B - SCRATCH
009 / C - NODE COUNT
010 / [D,E] - SCRATCH
011 / [H,L] - SCRATCH
012 /
013 22F8 81        KU9SUB, ADD      C          / CREATE FUNCTION CODE
014 22F9 57        MOV      D;A       / D ← FUNCTION CODE
015 22FA 79        MOV      A;C       / A ← BYTE COUNT
016 22FB 3D        DCR      A         / ACCOUNT FOR BASE VALUE
017 22FC 87        ADD      A         / COMPUTE EXTRA LENGTH
018 22FD C608      ADI     LENINS     / COMPUTE TOTAL LENGTH
019 22FF 5F        MOV      E;A       / E ← COMMAND LENGTH
020 2300 CD8125    CALL     PIO        / ISSUE COMMAND
021 2303 C9        RET          / EXIT
022 EJECT

001 SUBJOB KEY UTILITY : KU11 : A ← CURSOR ROW
002 /
003 /***KEY UTILITY : KU11 : A ← CURSOR ROW
004 /

```

```

005 2304 3A7EFE      KU11,   LDA      CURACT      / A ← CURSOR
006 2307 E6F0        ANI      ROWMSK      / ISOLATE ROW
007 2309 CF          NSWF          / SHIFT
008                  / TO
009                  / NORMALIZE
010                  / RESULT
011 230A C9          RET          / EXIT
012                  EJECT

001                  SUBJOB KEY UTILITY : KU12 : B ← CURSOR
002                  /
003                  /***KEY UTILITY : KU12 : B ← CURSOR
004                  /
005                  / H/L=HAS SCREEN ADDR AT CURSOR
006                  / D/E=STEP VALUE TO NEXT ROW (FROM CUR100)
007                  /
008 230B 3A7DFE      KU12,   LDA      CURDSP      / A ← CURSOR
009 230E 47          MOV      B:A        / B ← CURSOR
010 230F CD4705      CALL     CUR100     / SET POINTERS
011 2312 C9          RET          / EXIT
012                  EJECT

001                  SUBJOB KEY UTILITY : KU13 : SEARCH + DISPLAY
002                  /
003                  /***KEY UTILITY : KU13 : SEARCH + DISPLAY
004                  /
005                  /***PARAMETERS:
006                  /
007                  /      A      - NODE TYPE
008                  /      [H,L] - LAST DISPLAY ADDRESS FOR NODE
009                  /
010 2313 21FD09      KU13,   LXI      H;NODTAB+NODCON / [H,L] ← STARTING ADDR
011 2316 110900      LXI      D;NODRCL      / [D,E] ← OFFSET
012                  /
013 2319 BE          KU1305, CMP      M          / CHECK FOR MATCH
014 231A CA2123      JZ       KU1310     / BRANCH ON MATCH
015 231D 19          DAD      D          / GO TO NEXT ENTRY
016 231E C31923      JMP      KU1305     / CONTINUE
017                  /
018 2321 11FAFF      KU1310, LXI      D;NODDIS-NODCON / [D,E] ← OFFSET
019 2324 19          DAD      D          / [H,L] ← START OF DATA
020 2325 E5          PUSH     H          / SAVE POINTER
021                  /
022 2326 CD7A25      CALL     ISCOIL     / COIL TYPE?
023 2329 DA5423      JC       KU1340     / NO, GO
024                  /
025 232C 3A7EFE      LDA      CURACT      / A ← CURSOR
026 232F 47          MOV      B:A        / B ← CURSOR
027 2330 4F          MOV      C:A        / C ← CURSOR
028                  /
029 2331 79          KU1320, MOV      A:C        / A ← NEXT POSITION
030 2332 E60F        ANI      COLMSK     / ISOLATE COLUMN
031 2334 FE0C        CPI      MAXCOL+1  / AT RIGHT RAIL?
032 2336 CA5023      JZ       KU1335     / YES, BRANCH
033 2339 CD2B05      CALL     CURSOR     / MOVE CURSOR
034 233C 41          MOV      B:C        / B ← NEW CURSOR
035 233D CD4705      CALL     CUR100     / GET POINTERS
036 2340 23          INX      H          / SKIP FIELD ATTRIBUTE
037 2341 1606        MVI      D;DSPNOD-1 / D ← COUNT
038 2343 3E72        MVI      A;ASCDSH   / A ← DASH EXTENSION
039                  /
040 2345 77          KU1325, MOV      M:A        / DISPLAY DASH
041 2346 23          INX      H          / BUMP POINTER
042 2347 15          DCR      D          / DECREMENT COUNTER
043 2348 C24523      JNZ     KU1325     / LOOP UNTIL DONE
044                  /
045 234B 41          KU1330, MOV      B,C        / B ← NEW CURSOR
046 234C 0C          INR      C          / C ← NEXT CURSOR
047 234D C33123      JMP      KU1320     / CONTINUE
048                  /
049 2350 78          KU1335, MOV      A:B        / A ← CURSOR
050 2351 327DFE      STA     CURDSP     / INDICATE DISPLAY CURSOR
051                  /
052 2354 CD0B23      KU1340, CALL     KU12     / B ← CURSOR
053 2357 23          INX      H          / BUMP OVER ATTRIBUTE
054 2358 D1          POP      D          / [D,E] ← SOURCE

```



```

001          SUBJOB KEY UTILITY : KU17 : SET MATRIX POINTERS
002          /
003          /***KEY UTILITY : KU17 : SET MATRIX POINTERS
004          /***KEY UTILITY : KU17A : SET MATRIX POINTERS A-REG
005          /
006          /***REGISTER USAGE:
007          /
008          /          A      - SCRATCH <OR> ROW, COL
009          /          [D, E] - SCRATCH
010          /          [H, L] - ENTRY IN MATROW TABLE FOR NODE
011          /
012          KU17,
013 23AA C5          PUSH      B              / SVE
014 23AB 3A7EFE      LDA        CURACT        / GET CURRENT CURSOR
015 23AE C3B223      JMP        KU1705        / GO JOIN COMMON
016
017          /          ALTERNATE ENTRY WITH ROW, COL IN A-REG
018
019          KU17A,
020 23B1 C5          PUSH      B              / SVE
021
022          /          COMMON CODE
023
024          KU1705,
025 23B2 47          MOV        B:A          / SAVE ROW, COL FOR BELOW
026
027 23B3 E6F0        ANI        ROWMSK        / ISOLATE ROW
028 23B5 CF          NSWF          / ROTATE
029
030
031
032
032 23B6 2124FE      LXI        H, MATROW-MAXCOL / [H, L] <- TABLE BASE
033 23B9 110B00      LXI        D, MAXCOL        / [D, E] <- OFFSET
034
035 23BC 19          KU1710, DAD      D              / BUMP TO NEXT COLUMN
036 23BD 3D          DCR        A              / DECREMENT COUNTER
037 23BE C2BC23      JNZ        KU1710        / LOOP UNTIL DONE
038
039
039 23C1 78          MOV        A, B            / A <- CURSOR
040 23C2 E60F        ANI        COLMSK        / ISOLATE COLUMN
041 23C4 3D          DCR        A              / MAKE RELATIVE BASE 0
042 23C5 1600        MVI        D, 0          / D <- 0
043 23C7 5F          MOV        E, A          / E <- ROW - 1
044 23C8 19          DAD        D              / [H, L] <- MATRIX ENTRY
045
046
046 23C9 C1          POP        B              / RESTORE
047 23CA C9          RET                    / EXIT
048          EJECT
001          SUBJOB KEY UTILITY : KU18 : INC/DEC MEM USAGE
002          /
003          /***KEY UTILITY : KU18 : INC/DEC MEMORY USAGE
004          /
005          /***REGISTER USAGE:
006          /
007          /          A      - SCRATCH
008          /          [B, C] - INC/DEC (BYTE COUNT)
009          /
010          KU18,
011 23CB C5          PUSH      B              / SAVE
012 23CC D5          PUSH      D              /
013 23CD E5          PUSH      H              / X
014
015 23CE 2188FE      LXI        H, MEMUSE      / [H, L] <- POINTER
016 23D1 E7          GETHL          / [H, L] <- MEMORY USAGE
017 23D2 09          DAD        B              / INCREMENT/DECREMENT
018 23D3 EB          XCHG          / SWAP
019 23D4 2188FE      LXI        H, MEMUSE      / [H, L] <- DESTINATION
020 23D7 EF          MOVDE          / STORE DATA
021 23D8 EB          XCHG          / SWAP
022 23D9 111DFD      LXI        D, DSPUSE      / [D, E] <- DESTINATION
023 23DC CDC201      CALL       BNBCD4        / DISPLAY MEMORY USAGE
024
025 23DF E1          POP        H              / RESTORE ALL
026 23E0 D1          POP        D              /
027 23E1 C1          POP        B              / X
028 23E2 C9          RET                    / EXIT
029          EJECT

```

```

001          SUBJOB KEY UTILITY : KU19 : POWER TO COLUMN 1
002 /
003 /****KEY UTILITY : KU19 : POWER TO COLUMN 1
004 /
005 /****REGISTER USAGE:
006 /
007 /          A      - SCRATCH
008 /          [B,C] - SCRATCH
009 /          [D,E] - SCRATCH
010 /          [H,L] - SCRATCH
011 /
012 23E3 3A7DFE KU19, LDA CURDSP / A ← CURSOR
013 23E6 E60F ANI COLMSK / ISOLATE COLUMN
014 23E8 FE01 CPI :01 / CHECK FOR FIRST COL
015 23EA C20924 JNZ KU19X / EXIT IF NOT
016 23ED CD0B23 CALL KU12 / SET DISPLAY POINTERS
017 23F0 23 INX H / SKIP FIELD ATTRIBUTE
018 23F1 1603 MVI D,3 / D ← COUNTER
019 /
020 23F3 7E KU1905, MOV A,M / A ← CHARACTER
021 23F4 FE60 CPI ASCLRE / DO RANGE CHECK
022 23F6 DA0924 JC KU19X / (EXIT IF NOT)
023 23F9 FE7F CPI FACNDR-1 / FOR HIGH-LIGHTABLE CHARS
024 23FB DA0324 JC KU1910 / (BRANCH IF IN RANGE)
025 23FE FEC0 CPI CA0101 / CHECK CHARACTER ATTRIBUTE

026 2400 DA0924 JC KU19X / EXIT IF NOT
027 /
028 2403 34 KU1910, INR M / HIGH-LIGHT POWER FLOW
029 2404 23 INX H / BUMP POINTER
030 2405 15 DCR D / DECREMENT COUNTER
031 2406 C2F323 JNZ KU1905 / LOOP UNTIL DONE
032 /
033 2409 C9 KU19X, RET / EXIT
034 EJECT

001          SUBJOB KEY UTILITY : KU20 : CONNECT VERTICALS
002 /
003 /****KEY UTILITY : KU20 : CONNECT VERTICALS
004 /
005 /****REGISTER USAGE:
006 /
007 /          A      - SCRATCH
008 /          [B,C] - PRESERVED
009 /          [D,E] - SCRATCH
010 /          [H,L] - PRESERVED
011 /
012 240A C5 KU20, PUSH B / SAVE [B,C]
013 240B E5 PUSH H / SAVE [H,L]
014 240C CD0423 CALL KU11 / A ← ROW
015 240F FE01 CPI :01 / CHECK FOR TOP ROW
016 2411 CA4224 JZ KU20X / NO WORK FOR TOP ROW
017 /
018 2414 CD0B23 CALL KU12 / SET DISPLAY POINTERS
019 2417 11B6FF LXI D, DSPNOD-1-ROWE / [D,E] ← OFFSET
020 241A 19 DAD D / [H,L] ← VERTICAL ABOVE
021 241B 7E MOV A,M / A ← VERTICAL
022 241C FEE4 CPI CA0011 / MUST BE VERTICAL
023 241E CA2624 JZ KU2005 / BRANCH OKAY
024 2421 FEE5 CPI CA0011+CATHI / CHECK HIGHLIGHTED VERT
025 2423 C24224 JNZ KU20X / EXIT ON NO VERT
026 /
027 2426 115000 KU2005, LXI D, ROWS / [D,E] ← OFFSET
028 2429 19 DAD D / [H,L] ← VERTICAL FOR NOD

029 242A 7E MOV A,M / A ← VERTICAL CHARACTE
030 242B FEE0 CPI CA1100 / CHECK FOR DOWN VERTICAL
031 242D CA3524 JZ KU2010 / BRANCH ON IT
032 2430 FEE1 CPI CA1100+CATHI / ALSO CHECK HIGHLIGHT
033 2432 C23D24 JNZ KU2015 / MUST HAVE DOWN VERT
034 /
035 2435 E601 KU2010, ANI CATHI / SAVE HIGHLIGHT BIT
036 2437 F6DC ORI CA1110 / CONNECT UPWARD
037 2439 77 MOV M,A / AND DISPLAY
038 243A C34224 JMP KU20X / EXIT
039 /

```



```

040 243D E601      KU2015, ANI      CATHI      / SAVE HIGHLIGHT BIT
041 243F F6E8      /          ORI      CA1111     / CONNECT UP AND DOWN
042 2441 77        /          MOV      M;A        / DISPLAY CONNECTION
043                /
044 2442 E1        KU20X,  PDF      H          / RESTORE [H,L]
045 2443 C1        /          PDF      B          / RESTORE [B,C]
046 2444 C9        /          RET                      / EXIT
047                / EJECT

001                SUBJOB  KEY UTILITY : KU21 : SET DISPLAY TIMERS
002                /
003                /****KEY UTILITY : KU21 : SET DISPLAY TIMERS
004                /
005                /****REGISTER USAGE:
006                /
007                /          A      - SCRATCH
008                /          [B,C] - PRESERVED
009                /          [D,E] - PRESERVED
010                /          [H,L] - PRESERVED
011                /
012                KU21,
\*****: TEMP UNTIL PWR AND LED TEST!!!!
014 2445 C9        RET
\*****: END TEMP
016 2446 3E02      MVI      A;PWR TMR      / A ← TIMER VALUE
017 2448 3292FD    STA      TMRPWR        / INITIALIZE POWER TIMER
018 244B 3E1E      MVI      A;LED TMR     / A ← TIMER VALUE
019 244D 3291FD    STA      TMRLED        / INITIALIZE TIMER
020 2450 C9        RET                      / EXIT
021                EJECT

001                SUBJOB  KEY UTILITY : KU22 : COMPUTE COLTAB POINTER
002                /
003                /****KEY UTILITY : KU22 : COMPUTE COLTAB POINTER
004                /
005                /****REGISTER USAGE:
006                /
007                /          A      - SCRATCH
008                /          [B,C] - NOT USED
009                /          [D,E] - SCRATCH
010                /          [H,L] - COLTAB POINTER
011                /
012 2451 3A7EFE    KU22,  LDA      CURACT      / A ← CURSOR
013 2454 E60F      ANI      COLMSK        / ISOLATE COLUMN
014 2456 21E7FD    LXI      H;COLTAB-COLBKL / [H,L] ← STARTING ADDR
015 2459 110600    LXI      D;COLBKL      / [D,E] ← BLOCK LENGTH
016                /
017 245C 19        KU2210, DAD      D          / BUMP POINTER
018 245D 3D        DCR      A            / DECREMENT COUNT
019 245E C25C24    JNZ      KU2210       / LOOP UNTIL DONE
020                /
021 2461 C9        RET                      / EXIT
022                EJECT

001                SUBJOB  FIXVER = FIX LAST VERTICAL CHAR
002                /          FIXVER IS A SUBR WHICH FIGURES OUT HOW TO
003                /          CONNECT (OR DISCONNECT) A VERTICAL
004                /          COMING DOWN ONTO THE ROW BELOW.
005                /          IT EXAMINES THE CHAR ON THE ROW BELOW
006                /          TO SEE WHAT IT CURRENTLY
007                /          CONNECTS SO THAT THE NEW VERTICAL
008                /          WILL NOT CHANGE OTHER CONNECTIONS.
009                /
010                /          ** ENTRY
011                /          A ← NEW VERT CHAR (THIS
012                /          DESCRIBES WHETHER WE ARE SHORTING
013                /          OR OPENING A VERTICAL
014                /          (20= SPACE; E4= SHORT9
015                /          H/L = PTR TO LAST VERTICAL CHAR
016                /
017                /          CALL    FIXVER
018                /
019                /          ** EXIT
020                /          REGS SAME
021                /
022                /
023                FIXVER,

```

```

024 2462 C5      PUSH      B          / SAVE
025 2463 47      MOV       B,A         / SAVE UPPER (NEW) VERT
026
027              /      IF LAST VER CHAR=SPACE, NO PROBLEM
028              /      BECAUSE THERE ARE NO OTHER CONNECTIONS
029
030 2464 7E      MOV       A,M         / GET PRESENT LAST VERT
031
032              /      IF CHAR THERE IS A DASH, DON'T
033              /      CONNECT AT ALL:
034
035 2465 E6FF     ANI      -CATHI-1/  STRIP OFF HILITE BIT
036 2467 FE72     CPI      ASCD6H / DASH?
037 2469 C2AE24   JNZ      FIXV05 / NO, GO ON
038 246C D1       POP      B          / YES, EXIT NOW
039 246D C9       RET              / X
040
041              FIXV05,
042 246E FE20     CPI      ASCBLK / IS IT SPACE?
043 2470 C27E24   JNZ      FIXV20 / NO, GO LOOK AT HOR.
044
045              /      HAVE SPACE NOW, WHAT SHOULD IT BE?
046
047              FIXV10,
048 2473 B8       CMP     B          / DO WE WANT A SPACE?
049 2474 D1       POP      B          / (RESTORE ORIG)
050 2475 CA7B24   JZ       FIXV15 / YES, GO STORE SPACE
051
052 2478 36DC     MVI     M,CA1110/ NO, SET VERT
053 247A C9       RET              / DONE
054
055              /      HERE TO PUT A SPACE @ LAST VERT
056
057              FIXV15,
058 247B 3620     MVI     M,ASCBLK/ SET
059 247D C9       RET              / DONE
060
061              EJECT
062              /      THE CURRENT LAST VERT IS NOT SPACE. LOOK
063              /      AT THE CHAR* TO THE LEFT, WHICH TELLS US
064              /      IF THERE IS A HORIZONTAL ENTITY PRESENT.
065
066              FIXV20,
066 247E 2B       DCX     H          / STEP TO CHAR @ LEFT
067 247F 7E      MOV     A,M         / GET IT
068 2480 23       INX     H          / RESET TO PRESENT VERT
069 2481 FE20     CPI     ASCBLK / IS IT SPACE?
070 2483 CA7324   JZ      FIXV10 / YES, GO BACK AND DECIDE
071              /      WHAT IT SHOULD BECOME
072
073
074
075
THE PRESENT CONTACTS ARE NOT
NULL. WE NEED TO ANALYSE WHAT
CHANGE IS NECESSARY
076
077              /      THERE ARE 4 TYPES OF CONNECTIONS:
078              /      1-HOR ONLY
079              /      2-HOR AND VERT ABOVE AND VERT BELOW
080              /      3-HOR AND VERT ABOVE ONLY
081              /      4-HOR AND VERT BELOW ONLY
082
083              /      WE NEED TO LEAVE THE CONNECTIVITY THAT IS
084              /      NOT RELATED TO THE NEW VERTICAL
085              /      CONDITION AND ADD IN THE NEW VERT.
086
087              /      BRANCH ON THE 4 TYPES...
088
089 2486 7E      MOV     A,M         / GET CURRENT VERT ON SCREEN
090 2487 E6FE     ANI     -CATHI-1/ STRIP OFF HILITE BIT
091 2489 FEE0     CPI     CA1100 / IS IT HOR ONLY?
092 248B CAA024   JZ      FIXV30 / YES, GO FIX
093
094 248E FEE8     CPI     CA1111 / IS IT HOR AND VERT UP/DN?
095 2490 CAAB24   JZ      FIXV40 / YES, GO FIX
096
097 2493 FEDC     CPI     CA1110 / IS IT HOR AND VERT UP ONLY?
098 2495 CAB024   JZ      FIXV50 / YES, GO FIX
099

```

```

100 / ASSUME. HOR AND VERT DOWN ONLY.
101 / HERE TO FIX IT. IF NEW VERT = SPACE,
102 / DO NOTHING. IF NOT, PUT HOR AND VERT UP/DN
103
104 2498 78      MOV      A,B      / GET NEW VERT
105 2499 C1      POP      B      / RESTORE ORIG
106 249A FE20    CPI      ASCBLK  / SPACE?
107 249C C8      RZ          / YES, DONE
108 249D 36E8    MVI     M;CA1111/ NO, SET HOR AND VERT UP/DN
109 249F C9      RET          / DONE
110 EJECT
111 / HERE TO FIX HOR ONLY. IF SPACE, N/D.
112 / IF NOT, SET HOR AND VERT UP
113
114 FIXV30,
115 24A0 78      MOV      A,B      / GET NEW VERT
116 24A1 C1      POP      B      / RESTORE ORIG
117 24A2 FE20    CPI      ASCBLK  / SPACE?
118 24A4 C8      RZ          / YES, DONE
119 24A5 36DC    MVI     M;CA1110/ NO, SET HOR AND VERT UP
120 24A7 C9      RET          / DONE
121
122 / HERE TO FIX HOR AND VERT UP/DOWN
123 / IF NEW IS SPACE, SET HOR AND VERT DN
124 / IF NOT, NO CHG
125
126 FIXV40,
127 24A8 78      MOV      A,B      / GET NEW
128 24A9 C1      POP      B      / RESTORE
129 24AA FE20    CPI      ASCBLK  / SPACE?
130 24AC C0      RNZ          / NO, NO CHG
131 24AD 36D0    MVI     M;CA1101/ YES, CHG TO HOR AND VERT DN
132 24AF C9      RET
133
134 / HERE TO FIX HOR AND VERT UP ONLY.
135 / IF NEW IS SPACE, CHANGE TO HOR ONLY.
136 / IF NOT, NO CHG
137
138 FIXV50,
139 24B0 78      MOV      A,B      / GET NEW
140 24B1 C1      POP      B      / RESTORE
141 24B2 FE20    CPI      ASCBLK  / SPACE?
142 24B4 C0      RNZ          / NO, NO CHG
143 24B5 36E0    MVI     M;CA1100/ YES, SET HOR ONLY
144 24B7 C9      RET
145 EJECT

001 SUBJOB COLINC = INCREMENT "COLTAB" ADDRESSES
002 / COLINC IS A SUBR TO INCREMENT ALL
003 / THE ADDRESSES IN COLTAB BASED UPON A #
004 / OF NODES INSERTED. IT ALSO FIXES 'ADREON'
005 /
006 / *ENTRY.
007 / C = # OF NODES ADDED
008 / H/L = PTR TO COLUMN WITH INSERT
009 /
010 / CALL COLINC
011 /
012 / *EXIT:
013 / REGS SAME
014 /
015 COLINC,
016 24B8 D5      PUSH     D      /SAVE
017 24B9 C5      PUSH     B
018 24BA E5      PUSH     H
019
020 / SET B/C FOR INCREMENT VALUE
021
022 24BB 0600    MVI     B,C      / CLEAR MS
023 24BD 60      MOV     H,B      / SET H/L FOR DOUBLING
024 24BE 69      MOV     L,C      / X
025 24BF C9      DAD     B      / NOW H/L=2*B/C
026 24C0 44      MOV     B,H      / RESET B/C WITH STEP VALUE
027 24C1 4D      MOV     C,L      / X
028
029 / SEE IF THE CURRENT COLUMN IS EMPTY;

```



```

030 / IF SO, THIS IS THE 1ST INSET IN COL.
031
032 24C2 E1 POP H / RELOAD PTR
033 24C3 E5 PUSH H / X
034 24C4 110000 LXI D,0 / TO CHECK
035 24C7 E7 GETHL / GET START ADDR
036 24C8 F7 DCMF / EMPTY?
037 24C9 CAE624 JZ COLI20 / YES, GO FIX THIS COL ONLY
038 EJECT
039 / NOT EMPTY, SO FIX LAST ADDRESS AND RIPPLE
040 / DOWN THE COLUMNS, FIXING EACH
041
042 / FETCH THE "LAST ADDR" FROM THIS COL AND STEP IT
043
044 24CC E1 POP H / RELOAD COL PTR
045 24CD E5 PUSH H / X
046 24CE 110200 LXI D,COLEHI / OFFSET TO LAST ADDR
047 24D1 19 DAD D / NOW H/L POINTS TO LAST
048 24D2 E5 PUSH H / SAVE IT
049 24D3 E7 GETHL / GET LAST ADDR
050 24D4 09 DAD B / STEP IT
051 24D5 EB XCHG / TO D/E FOR STORE
052 24D6 E1 POP H / GET PTR TO LAST ADDR
053 24D7 EF MOVDE / STORE NEW LAST ADDR
054
055 24D8 E1 POP H / RELOAD PTR
056 24D9 E5 PUSH H / X
057
058 / NOW STEP TO NEXT COLUMN AND SEE IF DONE
059
060 COLI10,
061 24DA CD4225 CALL STEPCL / STEP TO NEXT COL AND
062 / SEE IF PAST COLTAB OR NEXT I
063 24DD DAF624 JC COLI99 / DONE, FIX ADREON AND GO
064
065 / NOT DONE, FIX NEXT COLUMN
066
067 24E0 CD5A25 CALL FIXCOL / GO FIX IT
068 24E3 C3DA24 JMP COLI10 / LOOP TIL DONE
069 EJECT
070 / HERE TO FIX ONLY THIS COL WHEN IT IS EMPTY
071
072 COLI20,
073
074 / THE START ADDR = [ADREON]+2
075 / THE END ADDR = START + STEP VALUE -2
076
077 24E6 218EFE LXI H,ADREON / GET PTR TO END OF NET ADDR
078 24E9 E7 GETHL / GET IT
079 24EA EB XCHG / TO D/E FOR STORE
080 24EB 13 INX D / STEP IT TWICE
081 24EC 13 INX D / X
082 24ED E1 POP H / GET PTR TO COL
083 24EE E5 PUSH H / X
084 24EF EF MOVDE / STORE START
085 24F0 EB XCHG / SET D/E=PTR TO END;
086 / SET H/L=START ADDR
087 24F1 09 DAD B / CALC END ADDR
088 24F2 2B DCX H / ACCOUNT FOR START NODE
089 24F3 2B DCX H / X
090 24F4 EB XCHG / NOW D/E=END; H/L=PTR
091 24F5 EF MOVDE / STORE IT
092
093 / FIX THE "END OF NET" ADDR, AND USAGE
094
095 COLI99,
096 24F6 CD6B25 CALL FIXEON / DONE
097
098 24F9 CDCB23 CALL KU18 / FIX USAGE
099
100 24FC E1 POP H / RESTORE
101 24FD C1 POP B
102 24FE D1 POP D
103 24FF C9 RET
104 EJECT

```

```

001 SUBJOB COLDEC = DECREMENT "COLTAB" ADDRESSES
002 / COLDEC IS A SUBR TO DECREMENT ALL
003 / THE ADDRESSES IN COLTAB BASED UPON A #
004 / OF NODES DELETED. IT ALSO FIXES 'ADREON'
005 /
006 / *ENTRY:
007 / C = # OF NODES DELETED
008 / H/L = PTR TO COLUMN WITH DELETE
009 /
010 / CALL COLDEC
011 /
012 / *EXIT:
013 / REGS SAME
014 /
015 COLDEC,
016 2500 D5 PUSH D /SAVE
017 2501 C5 PUSH B
018 2502 E5 PUSH H
019
020 / SET B/C FOR DECREMENT VALUE
021
022 2503 0600 MVI B,0 / CLEAR MS
023 2505 60 MOV H,B / SET H/L FOR DOUBLING
024 2506 69 MOV L,C / X
025 2507 09 DAD B / NOW H/L=2*B/C
026
027 / NOW TWO'S COMP IT
028
029 2508 7D MOV A,L / GET LS BYTE
030 2509 2F CMA / ONE'S COMP
031 250A 4F MOV C,A / SET LS BYTE
032 250B 7C MOV A,H / GET MS BYTE
033 250C 2F CMA / X
034 250D 47 MOV B,A / SET MS BYTE
035 250E 03 INX B / NOW B/C = NEG STEP
036
037 / GET THE LAST ADDR IN THIS COL AND DECR
038 / IT. IF IT IS < START ADDR, THE LAST ITEM IN
039 / THIS COL IS BEING DELETED.
040
041 250F E1 POP H / RELOAD PTR
042 2510 E5 PUSH H / X
043
044 2511 110200 LXI D,COLEHI / OFFSET TO LAST
045 2514 19 DAD D / NOW H/L=PTR TO LAST
046 2515 E5 PUSH H / SAVE FOR LATER
047 2516 E7 GETHL / GET LAST ADDR
048 2517 09 DAD B / DECR IT
049 2518 EB XCHG / NOW D/E = NEW LAST ADDR
050 2519 E1 POP H / GET PTR TO LAST ADR
051 251A EF MOVDE / STORE IT
052
053 / NOW SEE IF COL IS EMPTY
054
055 251B E1 POP H / RELOAD PTR
056 251C E5 PUSH H / X
057 251D E7 GETHL / GET START
058 251E F7 DCMPL / COMPARE END:START
059 251F DA3025 JC COLD20 / END IS LESS!
060 / GO CLEAR COLUMN
061
062 / NOT LAST NODE DELETE, SO RIPPLE THRU COLTAB
063
064 2522 E1 POP H / RELOAD PTR
065 2523 E5 PUSH H / X
066 COLD10,
067 2524 CD4225 CALL STEPCL / STEP AND SEE IF DONE
068 2527 DA3825 JC COLD99 / DONE, FIX ADREON
069 252A CD5A25 CALL FIXCOL / NOT DONE, FIX THIS COL
070 252D C32425 JMP COLD10 / AND LOOP TIL DONE
071
072 / LAST NODE IN COL IS DELETED, CLEAR COLUMN IN
073 / COLTAB
074
075 COLD20,

```

```

076 2530 E1      POP      H          / GET PTR
077 2531 E5      PUSH     H          / X
078 2532 110000  LXI     D;0         / CLEAR VALUE
079 2535 EF      MOVDE   / CLEAR START ADDR
080 2536 EF      MOVDE   / CLEAR END ADDR
081 2537 EF      MOVDE   / CLEAR EOC NODE
082
083              /      FIX THE "END OF NET" ADDR
084
085              COLD99,
086 2538 CD6B25  CALL    FIXEON     / DONE
087
088 253B CDCB23  CALL    KU18      / FIX USEAGE
089
090 253E E1      POP      H          / RESTORE
091 253F C1      POP      B
092 2540 D1      POP      D
093 2541 C9      RET
094              EJECT

001              SUBJOB STEPCL = STEP A PTR TO COLTAB AND CHECK DONE
002              / STEPCL IS A SUBR TO STEP TO THE NEXT COLUMN IN COLTAB
003              / AND CHECK TO SEE IF DONE. IT IS DONE IF
004              / THE POINTER GOES PAST THE END OF TABLE OR IF THE
005              / ADDRESS IN THE NEXT COLUMN IS ZERO.
006              /
007              / *ENTRY
008              / H/L=CURRENT PTR
009              /
010              / CALL STEPCL
011              /
012              / *EXIT
013              / H/L=NEXT COLUMN
014              / C SET IF DONE OR NEXT COL IS EMPTY
015              / C RESET IF NEITHER
016              /
017              STEPCL,
018 2542 D5      PUSH     D          / SAVE
019
020 2543 110600  LXI     D;COLBKL/  SIZE (COL STEP VALUE)
021 2546 19      DAD     D          / NOW H/L POINTS TO NEW
022 2547 112EFE  LXI     D;COLTBX-1/ GET END OF TABLE
023 254A F7      DCMPL / END<START?
024 254B D1      POP      D          / (IN CASE WE EXIT)
025 254C D8      RC          / YES!, EXIT NOW
026              / C IS SET
027
028              / NOT AT END, SO, SEE IF NEXT IS 0
029
030 254D D5      PUSH     D          / SAVE AGAIN
031 254E 110000  LXI     D;0         / FOR TEST
032 2551 E5      PUSH     H          / SAVE PTR
033 2552 E7      GETHL / GET START IN NEXT COL
034 2553 F7      DCMPL / IS IT ZERO? (EMPTY)
035 2554 E1      POP      H          / (RESTORE FOR EXIT)
036 2555 D1      POP      D          / (DITTO)
037 2556 37      STC     / SET CARRY IN CASE OF ZERO
038 2557 C8      RZ          / RETURN DONE!
039
040 2558 3F      CMC     / RESET FOR NOT DONE
041 2559 C9      RET          / X
042              EJECT

001              SUBJOB FIXCOL = FIX ONE COLUMN IN "COLTAB"
002              / FIXCOL IS A SUBR TO FIX THE START AND END ADDRESSES
003              / IN ONE COLUMN OF COLTAB, USING A STEP VALUE (+/-)
004              /
005              / *ENTRY
006              / H/L=PTR TO A COLUMN
007              / B/C=STEP VALUE (+/-)
008              /
009              / CALL FIXCOL
010              /
011              / *EXIT
012              / H/L, B/C SAME
013              /
014              FIXCOL,

```



```

015 255A D5      PUSH      D      / SAVE
016 255B E5      PUSH      H
017
018 255C E7      GETHL             / GET START ADDRESS
019 255D 09      DAD        B      / STEP IT
020 255E EB      XCHG             / SET TO D/E FOR STORE
021 255F E1      POP         H      / RELOAD PTR TO START
022 2560 E5      PUSH      H      / X
023 2561 EF      MOVDE             / STORE NEW START ADDR
024
025 2562 E5      PUSH      H      / SAVE STEPPED PTR (END)
026 2563 E7      GETHL             / GET OLD END ADDR
027 2564 09      DAD        B      / STEP IT
028 2565 EB      XCHG             / TO D/E FOR STORE
029 2566 E1      POP         H      / GET PTR TO END ADDR
030 2567 EF      MOVDE             / STORE IT
031
032 2568 E1      POP         H      / RESTORE AND EXIT
033 2569 D1      POP         D      / X
034 256A C9      RET
035      EJECT

001      SUBJOB  FIXEON = FIX [THE END OF NETWORK ADR
002      / FIXEON IS A SUBR TO ADJUST THE END OF NETWORK
003      / ADDR BY A STEP VALUE (+/-)
004      /
005      / *ENTRY
006      /      B/C = STEP VALUE
007      /
008      /      CALL    FIXEON
009      /
010      / *EXIT
011      /      B/C SSAME
012      /
013      FIXEON,
014 256B D5      PUSH      D      /SAVE
015 256C E5      PUSH      H
016
017 256D 218EFE  LXI      H;ADREON/ GET PTR TO END ADDR
018 2570 E7      GETHL             / GET IT
019 2571 09      DAD        B      / FIX IT!!!!!!
020 2572 EB      XCHG             / TO D/E FOR STORE
021 2573 218EFE  LXI      H;ADREON/ GET PTR TO END AGAIN
022 2576 EF      MOVDE             / STORE NEW ADDR
023
024 2577 E1      POP         H      / RESTORE AND EXIT
025 2578 D1      POP         D
026 2579 C9      RET
027      EJECT

001      SUBJOB  ISCOIL = SEE IF CONTACT IS A COIL TYPE
002
003      / ISCOIL IS A SUBR TO SEE IF THE CONTACT TYPE
004      / IN A = COIL TYPE
005      /
006      / *ENTRY
007      /      A = CONTACT TYPE
008      /
009      /      CALL    ISCOIL
010      /
011      / *EXIT
012      /      A = SAME
013      /      CARRY SET IF NOT COIL TYPE
014      /      CARRY RESET IF COIL TYPE
015      /
016      ISCOIL,
017 257A FE07  CPI      NOCOIL / IS IT < COIL TYPE?
018 257C D8      RC              / YES, RETURN 'NOT COIL'
019
020 257D FE0B  CPI      NODLAT+1/ IS IT > COIL TYPE?
021 257F 3F      CMC              / SET CARRY PROPERLY FOR EXIT
022 2580 C9      RET              / EXIT
023      EJECT

```

```

SUBJOB PERIPHERAL I/O HANDLER
/
/****PERIPHERAL I/O HANDLER
/
/****INPUTS:
/
/      D      - FUNCTION CODE
/      E      - MESSAGE LENGTH
/      CMDBUF - FORMATTED DATA
/
/****OUTPUTS:
/
/      Z-BIT. EQ. 1 => OKAY; RESPONSE IN BUFFER
/      Z-BIT. EQ. 0 => UNABLE TO COMPLETE TRANSACTION
/
/****BUILDS MESSAGE PACKET
/
/      TRANSMIT DATA
/      CHECKS RESPONSE FOR NON-MESSAGE RELATED ERRORS
/      RETURNS
/
/****REGISTER USAGE:
/
/      A      - SCRATCH
/      [B,C] - SCRATCH
/      [D,E] - FUNCTION/LENGTH
/      [H,L] - SCRATCH
/
EJECT

001 2581 AF      PID,   CLA      / INITIALIZE
002 2582 32E2FD STA      RCOUNT / RETRY COUNT
003
004 2585 2190FE LXI      H,CMDBUF / [H,L] ← COMMAND BUFFER
005 2588 3602   MVI      M,ASCSTX / LOAD AN STX
006 258A 23     INX      H      / BUMP POINTER
007 258B EF     MOVDE   / STORE COMMAND DATA
008 258C 2190FE LXI      H,CMDBUF / [H,L] ← POINTER
009 258F 3EFF   MVI      A,FF    / A ← START OF CHECKSUM
010 2591 D5     PUSH     D      / SAVE PARAMETERS
011 2592 1D     DCR      E      / ACCOUNT FOR STX
012
013 2593 86     PID010, ADD     M      / UPDATE CHKSUM
014 2594 23     INX      H      / BUMP POINTER
015 2595 1D     DCR      E      / DECREMENT COUNTER
016 2596 C29325 JNZ     PID010   / LOOP UNTIL DONE
017 2599 77     MOV      M,A    / STORE CHECKSUM
018
019 259A 2190FE PID020, LXI      H,CMDBUF / [H,L] ← START OF BUFFER
020 259D 01A2FD LXI      B,PPOBLK / [B,C] ← BUFFER BLKADDR
021 25A0 D1     POP      D      / GET COUNTER
022 25A1 D5     PUSH     D      / STACK IT AGAIN
023
024 25A2 7E     PID030, MOV     A,M    / A ← NEXT BYTE
025 25A3 D5     PUSH     D      / SAVE COUNTER
026 25A4 E5     PUSH     H      / SAVE POINTER
027 25A5 CD7E01 CALL    BFCH    / BUFFER BYTE
028 25A8 E1     POP      H      / GET POINTER
029 25A9 D1     POP      D      / GET COUNT
030 25AA 23     INX      H      / INCREMENT POINTER
031 25AB 1D     DCR      E      / DECREMENT COUNT
032 25AC C2A225 JNZ     PID030   / LOOP UNTIL DONE
033
034 25AF CDE126 CALL    FU02    / CHECK BUSY STATUS
035 25B2 CAD425 JZ      FI0060   / BRANCH NOT BUSY
036
037 25B5 118827 PID040, LXI      D,MSGBSY / [D,E] ← SOURCE ADDR
038 25B8 216CFC LXI      H,DSPBSY / [H,L] ← DESTINATION ADDR
039 25BB CD0301 CALL    MOVSTR  / DISPLAY MESSAGE
040 25BE CD2004 CALL    FFINIT  / INITIALIZE PORT
041
042 25C1 CDE126 PID050, CALL    FU02    / CHECK FOR PORT BUSY
043 25C4 C20125 JNZ     FI0050   / WAIT UNTIL AVAILABLE
044 25C7 3A8827 LDA      MSGBSY / A ← BYTE COUNT
045 25CA 216CFC LXI      H,DSPBSY / [H,L] ← ADDR

```

```

046 250D 57          MOV      D;A          / D ← BYTE COUNT
047 250E CD1903     CALL     ROWN10       / CLEAR FIELD
048 25D1 C39A25     JMF     P10020       / START MESSAGE AGAIN
049                EJECT

001 25D4 3E3C       P10060, MVI     A;ACKTMR / A ← ACKNOWLEDGE TIMER
002 25D6 3290FD     STA     TMRACK       / LOAD TIMER
003 25D9 3E27       MVI     A,PPCMD+SPCDTR / A ← COMMAND CODE
004 25DB D33A       OUT     SP1CTL       / ENABLE INTERRUPT
005                /
006 25DD CDE124     P10070, CALL    PU02   / CHECK PORT AVAILABILIT
007 25E0 C2B525     JNZ     P10040       / BRANCH IF NOT
008 25E3 3A90FD     LDA     TMRACK       / A ← ACK TIMER
009 25E6 B7         TST     / CHECK
010 25E7 CAF925     JZ      P10085       / BRANCH IF TIME-OUT
011                /
012 25EA 21AEFD     P10080, LXI     H;PPISTA / [H,L] ← STATUS ADDR
013 25ED 7E         MOV     A;M          / A ← STATUS
014 25EE E620       ANI     PPIIDUN      / CHECK FOR DONE
015 25F0 C24126     JNZ     P10100       / BRANCH ON DONE
016 25F3 7E         MOV     A;M          / A ← STATUS
017 25F4 EA04       ANI     PPIRET       / CHECK FOR RETRAN
018 25F6 CADD25     JZ      P10070       / LOOP IF NOT
019                /
020 25F9 21B2FD     P10085, LXI     H;RCOUNT / ERROR, CHECK RETRIES
021 25FC 34         INR     M            / BUMP COUNT
022 25FD 3E05       MVI     A;MAXTRY+1   / COMPARE AGAINST MAX
023 25FF BE         CMP     M            / RETRY COUNT
024 2600 CA0E26     JZ      P10090       / BRANCH ON HARD ERROR
025                /
026 2603 3AAEFD     P10087, LDA     PPISTA   / A ← STATUS
027 2606 E6E1       ANI     -1-PPIIDVR-PPIRET-PPIPAR-PPICR / CLEAR FLAGS
028 2608 32AEFD     STA     PPISTA       / CLEAR FLAGS
029 260B C39A25     JMP     P10020       / TRY AGAIN
030                EJECT

\
HERE FOR ALL I/O ERRORS
032                /
033 260E 21AEFD     P10090, LXI     H;PPISTA / [H,L] ← STATUS ADDR
034 2611 11FB26     LXI     D;MSGOVR     / [D,E] ← MESSAGE ADDR
035 2614 7E         MOV     A;M          / A ← STATUS
036 2615 E608       ANI     PPIIDVR      / CHECK FOR OVERRUN
037 2617 C22F26     JNZ     P10ERR       / BRANCH ON OVERRUN
038 261A 11EE26     LXI     D;MSGPAR     / [D,E] ← MESSAGE ADDR
039 261D 7E         MOV     A;M          / A ← STATUS
040 261E E610       ANI     PPIPAR       / CHECK FOR PARITY/FRAMING
041 2620 C22F26     JNZ     P10ERR       / BRANCH ON PARITY/FRAMING
042 2623 110727     LXI     D;MSGCHK     / [D,E] ← MESSAGE ADDR
043 2626 7E         MOV     A;M          / A ← STATUS BYTE
044 2627 E602       ANI     PPIICR       / CHECK FOR CHKSUM ERROR
045 2629 C22F26     JNZ     P10ERR       / BRANCH ON IT
046 262C 11A927     LXI     D;MSGRSP     / MUST BE TIME-OUT
047                /
048 262F CD7E05     P10ERR, CALL    ERROR  / DISPLAY MESSAGE
049 2632 3A7CFE     LDA     KSTATE       / SET P180 I/O ERROR BIT
050 2635 F640       ORI     KERROR       / X
051 2637 327CFE     STA     KSTATE       / (FORCES RST 0 BY USER)
052 263A D1         POP     D            / CLEAR COMMAND CODE
053 263B CD2004     CALL    PFINIT      / INIT THE PORT
054 263E C39026     JMP     P10130       / GO TO EXIT
055                EJECT

001                /
002 2641 AF         P10100, CLA         / RESPONSE RECEIVED
003 2642 3290FD     STA     TMRACK       / CLEAR TIMER
004 2645 3AAEFD     LDA     PPISTA       / A ← STATUS
005 2648 E6DF       ANI     -1-PPIIDUN   / CLEAR DONE FLAG
006 264A 32AEFD     STA     PPISTA       / STORE FLAG
007 264D 019CFD     LXI     B;PPIBLK     / [B,C] ← BUFFER BLK ADDR
008 2650 CD5601     CALL    UBFCB        / GET COMMAND
009 2653 21A8FE     LXI     H;RSFBUF     / [H,L] ← RESPONSE BLK
010 2656 77         MOV     M;H          / A ← RESPONSE
011 2657 E5         PUSH   H            / STACK IT
012 2658 CD5601     CALL    UBFLH        / GET COUNT
013 265B E1         POP     H           / RESTORE POINTER
014 265C D603       SUI     C03         / ADJUST COUNTER
015 265E 57         MOV     D;A          / D ← REMAINING COUNT

```



```

016 /
017 265F E5 P10110, PUSH H / SAVE POINTER
018 2660 CD5601 CALL UBFCB / GET NEXT BYTE
019 2663 E1 POP H / RESTORE POINTER
020 2664 23 INX H / BUMP POINTER
021 2665 77 MOV M;A / STORE INTO BUFFER
022 2666 15 DCR D / DECREMENT COUNT
023 2667 C25F26 JNZ P10110 / LOOP UNTIL DONE
024 /
025 266A D1 POP D / GET PARAMETERS
026 266B 21A8FE LXI H;RSPBUF / [H,L] ← RESPONSE BUFFER
027 266E 7E MOV A;M / A ← FUNCTION CODE
028 266F FED0 CPI ABCNAK / CHECK FOR NAK
029 2671 CA8826 JZ P10120 / BRANCH ON NAK
030 2674 BA CMP D / DO FUNCTION CHECK
031 2675 CA9226 JZ P10X / MATCH! I/O OKAY
032 / SO EXIT
033 EJECT
034 / HERE WHEN RESPONSE WAS NOT WHAT WE
035 / ASKED FOR, NOR WAS IT "NAK". ASSUME
036 / GARBAGE AND RETRY.
037
038 2678 21B2FD LXI H;RCOUNT / POINT TO RETRY COUNT
039 267B 34 INR M / STEP IT
040 267C 3E05 MVI A;MAXTRY+1 / CHECK TO MAX
041 267E BE CMP M / TRIED ALL?
042 267F C20326 JNZ P10087 / NO, GO REPEAT
043 2682 11F827 LXI D;MSGRES / YES, GET ERROR MGS
044 2685 C32F26 JMP P10ERR / AND GO DISPLAY IT
045 /
046 2688 23 P10120, INX H / BUMP POINTER TO
047 2689 7E MOV A;M / GET NAK CODE
048 268A 219326 LXI H;P10TAB / [H,L] ← TABLE ADDR
049 268D C0C726 CALL PU01 / DO TABLE CHECK
050 /
051 2690 AF P10130, CLA / A ← 0
052 2691 30 INR A / Z-BIT ← 0
053 /
054 2692 C9 P10X, RET / EXIT
055 EJECT

001 /
002 /***TABLE
003 /
004 2693 11 P10TAB, DB P10TAB /
005 2694 01 DB ERRPAR / PARITY/FRAMING
006 2695 EE26 DW MSGPAR /
007 2697 02 DB ERROVR / OVERRUN
008 2698 FB26 DW MSGOVR /
009 269A 03 DB ERRCHK / CHECKSUM
010 269B 0727 DW MSGCHK /
011 269D 07 DB ERRTIM / TIME OUT
012 269E 3527 DW MSGTIM /
013 26A0 04 DB ERRADR / ADDRESS ERROR
014 26A1 3D27 DW MSGADR /
015 26A3 05 DB ERRADI / INVALID ADDRESS
016 26A4 4A27 DW MSGADI /
017 26A6 06 DB ERRCMD / INVALID COMMAND CODE
018 26A7 B327 DW MSGCMD /
019 26A9 08 DB ERRMSK /
020 26AA 5727 DW MSGMSK /
021 26AC 09 DB ERRSEQ / BAD STEP NUMBER
022 26AD 9227 DW MSGSEQ /
023 26AF 0A DB ERRNOD / INVALID NODE
024 26B0 6427 DW MSGNOD /
025 26B2 0B DB ERRMEM / MEMORY PROTECT FAULT
026 26B3 1027 DW MSGMEM /
027 26B5 0C DB ERRSTP / NOT STOP STATE
028 26B6 2927 DW MSGSTP /
029 26B8 0D DB ERRLEN / BAD LENGTH
030 26B9 0528 DW MSGBDL /
031 26BB 0E DB ERRCON / BAD CONTACT
032 26BC BF27 DW MSGCON /
033 26BE 0F DB ERRNPD / NOT IN POWER DISPLAY
034 26BF 9D27 DW MSGNPD /
035 26C1 10 DB ERRSUP / NODE NOT SUPPORTED

```

```

036 2602 7127      DW      MSGSUP
037 2604 11        DB      ERRFUL      / MEMORY FULL
038 2605 7027      DW      MSGFUL
039
040      0011      FIOTBL-  -FIOTAB-123
041      EJECT

SUBJOB PERIPHERAL UTILITY : PU01 : NAK CODE SEAR 4
/
/ ***PERIPHERAL UTILITY : PU01 : NAK SEARCH
/
/ ***PARAMETERS:
/
/      Z-BIT. EQ. 0 => NO MATCH
/      Z-BIT. EQ. 1 => MATCH FOUND
/
/ ***TABLE FORMAT:
/
/      BYTE      CONTENTS
/      0          NUMBER OF ENTRIES
/      1          ENTRY 1 - NAK CODE
/      2          ENTRY 1 - MESSAGE ADDRLO
/      3          ENTRY 1 - MESSAGE ADDRHI
/      4          ENTRY 2 - NAK CODE
/      5          ENTRY 2 - MESSAGE ADDRLO
/      6          ENTRY 2 - MESSAGE ADDRHI
/      ETC
/
/ ***REGISTER USAGE:
/
/      A          - NAK CODE
/      (B,C)      - COUNT
/      (D,E)      - MESSAGE ADDR
/      (H,I)      - TABLE ADDRESS
/
/      EJECT .

001 2607 46        PU01,  MOV      B;M      / B <- ENTRY COUNT
002 2608 23        INX      H          / BUMP POINTER
003
004 2609 BE        PU0110, CMP      M          / CHECK FOR NAK CODE MATCH
005 260A C4D826    JZ        PU0120    / BRANCH ON MATCH
006 260D 23        INX      H          / BUMP POINTER
007 260E 23        INX      H          / TO NEXT
008 260F 23        INX      H          / TABLE ENTRY
009 26D0 05        DCR      B          / DECREMENT ENTRY COUNT
010 26D1 C2D926   JNZ      PU0110    / LOOP IF NOT DONE
011 26D4 04        INR      B          / Z <- 0
012 26D5 C3E026   JMP      PU01X     / GO TO EXIT
013
014 26D8 23        PU0120, INX      H          / BUMP POINTER
015 26D9 5E        MOV      E;M     / E <- MESSAGE ADDRLO
016 26DA 23        INX      H          / BUMP POINTER
017 26DB 56        MOV      D;M     / D <- MESSAGE ADDRHI
018 26DC CD7E05   CALL     ERROR     / SET ERROR STATE
019 26DF AF        CLA          / SET Z-BIT
020
021 26E0 C9        PU01X,  RET          / EXIT
022      EJECT

SUBJOB PERIPHERAL UTILITY : PU02 : PORT STATUS
/
/ ***PERIPHERAL UTILITY : PU02 : PORT STATUS
/
/ ***PARAMETERS:
/
/      Z-BIT. EQ. 0 => PORT NOT AVAILABLE
/      Z-BIT. EQ. 1 => PORT AVAILABLE
/
/ ***REGISTER USAGE:
/
/      A          - SCRATCH
/      (B,C)      - NOT USED
/      (D,E)      - NOT USED

```

```

015 / [H, L] - NOT USED
016 /
017 26E1 DB3A PU02, IN SP1STA / REAT STATUS
018 26E3 E680 ANI SP5DSR / CHECK DSR (-EIA)
019 26E5 C2EC26 UNZ PU0210 / BRANCH IF AVAILABLE
020 26E8 3C INR A / Z-BIT <- 0
021 26E9 C3ED26 JMP PU02X / GO TO EXIT
022 /
023 26EC AF PU0210, CLA / Z-BIT <- 1
024 /
025 26ED C9 PU02X, RET / EXIT
026 EJECT

```

```

001 SUBJOB MESSAGE AREA
002 /
003 /***THIS SECTION CONTAINS ALL THE SYSTEM MESSAGES
004 /
005 /***MESSAGE FORMAT:
006 /
007 / MSG, DB MSGEND
008 / DA /... TEXT...
009 / MSGEND= .-MSG-1
010 /
011 26EE 0C MSGPAR, DB MSGPAX
012 26EF 50415249 DA /PARITY ERROR
013 26F3 54592045
014 26F7 52524F52
015 0000 MSGPAX= .-MSGPAR-1
016 /
017 26FB 0B MSGOVR, DB MSGOVX
018 26FC 4F564552 DA /OVERRUN ERR
019 2700 52554E20
020 2704 455252
021 000B MSGOVX= .-MSGOVR-1
022 /
023 2707 0C MSGCHK, DB MSGCHX
024 2708 43484853 DA /CHKSUM ERROR
025 270C 554D2045
026 2710 52524F52
027 000C MSGCHX= .-MSGCHK-1
028 /
029 2714 08 MSGHI, DB MSGHIX
030 2715 52455620 DA /REV
031 2719 41 DB MAUREV / MASTER REV LEVEL
032 271A 583233 DB DVR1, DVR2, DVR3 / DEVELOPMENT LEVEL
033 0008 MSGHIX= .-MSGHI-1
034 /
035 271D 0B MSGMEM, DB MSGMEX
036 271E 4D454D20 DA /MEM PROTECT
037 2722 50524F54
038 2726 454354
039 000B MSGMEX= .-MSGMEM-1 / MESSAGE LENGTH
040 /
041 2729 0B MSGSTP, DB MSGSTA
042 272A 34383420 DA /484 RUNNING
043 272E 52554E4E
044 2732 494E47
045 000B MSGSTX= .-MSGSTP-1 / MESSAGE LENGTH
046 /
047 2735 07 MSGTIN, DB MSGTIX
048 2736 54494D45 DA /TIMEOUT
049 273A 4F5554
050 0007 MSGTIX= .-MSGTIN-1
051 /
052 273D 0C MSGADR, DB MSGADX
053 273E 42414420 DA /BAD ADR RNGE
054 2742 41445220
055 2746 524E4745
056 000C MSGADX= .-MSGADR-1
057 /
058 274A 0C MSGADI, DB MSGADY
059 274B 494C4C45 DA /ILLEGAL ADDR
060 274F 47414C20
061 2753 41444452
062 000C MSGADY= .-MSGADI-1

```


048		/		
049	2757	00	MSGMSK, DB	MSGMSX
050	2758	49404045	DA	'ILLEGAL MASK'
	275C	47414020		
	2760	4D41534B		
051		0000	MSGMSX=	.-MSGMSK-1
052			/	
053	2764	00	MSGNOD, DB	MSGNOX
054	2765	49404045	DA	'ILLEGAL NODE'
	2769	47414020		
	276D	4E4F4445		
055		0000	MSGNOX=	.-MSGNOD-1
056			/	
057	2771	0A	MSGSUP, DB	MSGSUX
058	2772	53555045	DA	'SUPERVISOR'
	2776	52564953		
	277A	4F52		
059		000A	MSGSUX=	.-MSGSUP-1
060			/	
061	2770	0B	MSGFUL, DB	MSGFULX
062	277D	4D454D4F	DA	'MEMORY FULL'
	2781	52592046		
	2785	554040		
063		000B	MSGFULX=	.-MSGFUL-1
064			/	
065	2788	09	MSGBSY, DB	MSGBSX
066	2789	434F4D4D	DA	'COMM BUSY'
	278D	20425553		
	2791	59		
067		0009	MSGBSX=	.-MSGBSY-1
068			/	
069	2792	0A	MSGSEQ, DB	MSGSEX
070	2793	42414420	DA	'BAD STEP #'
	2797	53544350		
	279B	2023		
071		000A	MSGSEX=	.-MSGSEQ-1
072			/	
073	279D	0B	MSGNPD, DB	MSGNPX
074	279E	42414420	DA	'BAD LED REQ'
	27A2	4C454420		
	27A6	524351		
075		000B	MSGNPX=	.-MSGNPD-1
076			/	
077	27A9	09	MSGRSP, DB	MSGRSX
078	27AA	4E4F2041	DA	'NO ANSWER'
	27AE	4E535745		
	27B2	52		
079		0009	MSGRSX=	.-MSGRSP-1
080			/	
081	27B3	0B	MSGCMD, DB	MSGCMX
082	27B4	42414420	DA	'BAD COMMAND'
	27B8	434F4D4D		
	27BC	414E44		
083		000B	MSGCMX=	.-MSGCMD-1
084			/	
085	27BF	0B	MSGCON, DB	MSGCOX
086	27C0	42414420	DA	'BAD CONTACT'
	27C4	434F4E54		
	27C8	414354		
087		000B	MSGCOX=	.-MSGCON-1
088			/	
089	27CB	0B	MSGSOL, DB	MSGSOX
090	27CC	53544152	DA	'START LOGIC'
	27D0	5420404F		
	27D4	474943		
091		000B	MSGSOX=	.-MSGSOL-1
092			/	
093	27D7	0C	MSGEOL, DB	MSGEOX
094	27D8	454E4420	DA	'END OF LOGIC'
	27DC	4F46204C		
	27E0	4F474943		
095		000C	MSGEOX=	.-MSGEOL-1
096			/	
097	27E4	0A	MSGNET, DB	MSGNEX
098	27E5	4E4F204E	DA	'NO NETWORK'
	27E9	4554574F		
	27ED	524B		

```

099      000A      MSGNEX= .-MSGNET-1
100
101 27EF 08      MSGSCH, DB      MSGSCX
102 27F0 4E4F204D DA      'NO MATCH'
      27F4 41544348
103      0008      MSGSCX= .-MSGSCH-1
104
105 27F8 0C      MSGRES, DB      MSGREX
106 27F9 42414420 DA      'BAD RESPONSE'
      27FD 52455350
      2801 4F4E5345
107      000C      MSGREX= .-MSGRES-1
108
109 2805 0A      MSGBDL, DB      MSGBDX
110 2806 42414420 DA      'BAD LENGTH'
      280A 4C454E47
      280E 5448
111      000A      MSGBDX= .-MSGBDL-1
112
      EJECT

001      JOB \LDV 180 MOD 01 REV AX21
002
003
004      /      COPYRIGHT, (C) 1978, GOULD INC., MODICON DIV.,
005      /      ALL RIGHTS RESERVED. NO PART OF THIS PROGRAM
006      /      MAY BE REPRODUCED IN ANY FORM WITHOUT THE
007      /      EXPRESS WRITTEN PERMISSION OF GOULD INC.
008
009
010      /      WRITTEN BY: R. SOLOMON
011
012      EJECT

001      SUBJOB \      REVISION HISTORY OF THIS FILE
002
003      /      THIS SECTION CONTAINS INFORMATION PERTAINING
004      /      TO ALL REVISIONS. THIS INFORMATION MUST
005      /      CONSIST OF AT LEAST:
006      /      1- NEW REVISION LETTER
007      /      2- WHAT OTHER FILES WERE AFFECTED
008      /      3- WHY REVISION WAS DONE.
009      /      4- ALL "ECO" #'S FOR THE REVISION.
010
011
012      \      REVISION A
013      /      REVISION A IS THE ORIGINAL PROGRAM RELEASE
014      /      ECO # = XXXX
015
016      EJECT

001      SUBJOB \      FILE DESCRIPTION OF LDV 180
002
003      /      THIS FILE CONTAINS SOURCE FOR THE
004      /      LOAD-DUMP-VERIFY FUNCTIONS OF
005      /      THE P180 PROGRAMMING PANEL FOR
006      /      THE 484 CONTROLLER.
007
008      EJECT
009      SUBJOB \      DATA FOR DUMP-LOAD-VER
010
011      0010      MAXBUF= 016      / MAX BUFFER THAT CAN BE
012      /      SENT TO 484
013      /      ALSO SIZE OF TAPE RECORD
014
015      3030      ASCZZ= :3030      / ASCII 00 FOR CREATION OF
016      /      INTEL RECORDS
017
018      3031      ASC01= :3031      / '01' RECORD FOR 484 TAPES
019
020      4646      ENDCHK= :4646      / 'FF' CHECKSUM OF END OF TAPE
021      /      RECORD FOR 484 DUMP
022
023      002F      LENCAS= 047      / LENGTH OF MAXIMUM CASSETTE REC RD
024
025      0D0A      CRLF= :0D0A      / ASCII CRLF
026
027      0003      TYPE1= 3      / USER LOGIC RAM FLAG

```

```

028
029      0002      TYPE2= 2          / COIL RAM FLAG
030
031      0001      TYPE3= 1          / REGISTER RAM FLAG
032
033      0001      TYPE01= 1         / RECORD TYPE FOR CASS.
034
035      0000      FIELD1= 0         / HI-ORDER NIBBLE - LOGIC RAM
036
037      0020      FIELD2= :20       / HI-ORDER NIBBLE - COIL RAM
038
039              EJECT

```

```

001      SUBJOB \          DUMP - DUMP 484 TO PORT 2
002
003      /              THIS ROUTINE WILL DUMP AN ENTIRE 484
004      /              CONTROLLER TO THE PRIPHERAL PORT # 2
005      /              IN INTEL FORMAT FOR A CASSETTE.
006
007      /              ENTRY IS FROM 'SUPERVISORY' STATE BY SELECTING
008      /              THE PROPER KEY.
009      /
010      /              EXIT IS TO 'EXEC'
011
012              EJECT

```

```

001      DUMP,
002
003      /              DISPLAY ADVISORY MESSAGE
004
005      2810 115C2D      LXI      D;MSGDPG/ "DUMPING"
006      2813 CD681F      CALL     KU04 / DISPLAY
007
008      /              INITIALIZE PORT 2
009      /              MUST ADVANCE PAST LEADER
010      /              APPROX 8 SEC. DELAY
011
012      2816 CDCC2A      CALL     P2INIT / INIT PERIPHERAL PORT 2
013      2819 0614        MVI     B;@20 / CTR FOR 10 SEC DELAY
014      281B 3E35        MVI     A;SPCRS+SPCER+SPCRE+SPCTE / TURN ON XMIT
015      281D D33C        OUT     SP2CTL /DONE
016
017      DUMP2,
018      281F CD5B2C      CALL     DELHLF / .5 SEC DELAY
019      2822 05          DCR     B / DONE YET?
020      2823 C21F28      JNZ     DUMP2 / 0 --> DONE
021      2826 3E15        MVI     A;:15 / STOP TAPE
022      2828 D33C        OUT     SP2CTL / DONE
023
024      282A 3E03        MVI     A;TYPE1 / FIND 1ST, LAST ADDR OF USER LOGIC
025      282C F5          PUSH    PSW / SAVE RAM TYPE
026
027      282D CD1529      CALL     CON484 / GET HI ADDRESS INTO EOUSEG
028      /              / 2 BYTES (HI,LO), LOEST->H/L
029      DUMP15,
030      2830 F1          POP     PSW / GET RAM TYPE
031      2831 F5          PUSH    PSW / SAVE RAM TYPE
032      2832 CD5B28      CALL     DUMP20 / GET SIZE OF READ
033      2835 DA4928      JC      DUMP30 / CY=1 --> END OF SEGMENT
034      2838 CD6228      CALL     DUMP25 / DO THE READ
035

```

WE SUCCESSFULLY READ THE DATA
PREPARE TO SEND IT TO PORT # 2

```

003
004      283B CDC928      CALL     CSFRMT / FORMAT RECORD FOR CASSETTE
005      283E CDFF2B      CALL     P2TID / OUTPUT THE RECORD
006
007      /              SEE IF THERE IS MORE DATA LEFT
008
009      2841 F1          POP     PSW / GET FIELD BIT
010      2842 F5          PUSH    PSW / SAVE IT AGAIN
011      2843 CD7928      CALL     NXTADR / GET START ADDR OF NEXT READ
012      2846 C33028      JMP     DUMP15 / & GET NEXT RECORD
013
014      /              COME HERE WHEN RAM SEGMENT IS COMPLETE

```



```

015          DUMP30.
016
017          /          SET LIMITS FOR NEXT RAM FIELD
018
019 2849 F1          POP          PSW          / GET # OF SEGS
020 284A 3D          DCR          A          / DECR & SEE IF DONE
021 284B F5          PUSH         PSW          / SAVE AGAIN
022 284C C22D28     JNZ          DUMP10 / GO PROCESS NEXT SEGMENT
023
\          ALL DONE WITH DUMP - SEND EOF
025
026 284F CDBB2C     CALL          EOF          / WRITE END RECORD TO PORT 2
027
028          /          DISPLAY ADVISORY MESSAGE: "DUMP O.K."
029
030 2852 11642D     LXI          D,MSGDOK/ DUMP OK
031 2855 CD681F     CALL          KU04         / DSPLY
032
033 2858 C3EF00     JMP          EXEC          / GO TO EXEC
034
035          EJECT

001          SUBJOB \          DUMP20 - CALC SIZE OF READ
002
003          /          THIS RTN WILL HAVE THE SIZE OF THE NEXT READ BUFFER
004
005          /          CALCULATED.
006
007          /          ** ENTRY
008          /          A=RAM TYPE
009          /          3=LOGIC RAM
010          /          2=COIL RAM
011          /          1=REGISTER RAM
012          /          H/L = ADDR OF 1ST BYTE TO DUMP
013
014          /          CALL DUMP20
015
016          /          ** EXIT
017          /          IF CY=1 --> END OF SEGMENT
018          /          IF CY=0 -->
019          /          B/C = # OF BYTES TO BE READ
020
021          DUMP20.
022 285B 011000     LXI          B,MAXBUF/ GET SIZE OF BUFFER FOR READ
023 285E CD8828     CALL          GETSIZ / IN B/C ON RET
024 2861 C9          RET          / IF CY=1 --> END OF RAM SEGMENT
025          EJECT

001          SUBJOB \          DUMP25 - READ DATA FROM 484
002
003          /          THIS RTN WILL FORMULATE A READ COMMAND TO
004          /          READ A SPECIFIED # OF BYTES FROM A
005          /          484 AND HAVE "PIO" DO THE READ.
006
007          /          ** ENTRY
008          /          B/C = # OF BYTES TO READ
009          /          H/L = ADDRESS OF 1ST BYTE
010          /          D/E AVAILABLE
011
012          /          CALL DUMP25
013
014          /          ** EXIT
015          /          ERROR --> UNCOND CALL TO ERROR (NO RET)
016          /          ELSE --> DATA IS STORED IN "RSPBUF"
017          /          "CMDDBUF" IS LEFT SET UP BY "PIO"
018          /          D/E = READ COMMAND FOR "PIO"
019
020
021          DUMP25.
022 2862 C5          PUSH         B          / SAVE REGS
023 2863 E5          PUSH         H
024
025 2864 79          MOV          A,C          / GET # OF BYTES TO READ

```

```

026 2865 0F      RRC          / DIVIDE # OF BYTES BY 2
027 2866 110610 LXI      D; CMDRED!; 100+LENRED / READ COMMAND
028 2869 82      ADD      D      / INDEX INTO COMMAND
029 286A 57      MOV      D; A    / ALL SET UP
030
031 /           / NOW STORE THE 484 ADDRESS IN CMDBUF AND LET
032 /           / PIQ TAKE IT FROM THERE
033
034 286B E5      PUSH     H      / MOVE ADDR TO B/C
035 286C C1      POP      B      / DONE
036 286D 2193FE LXI      H; CMDBUF+3 / BUFFER FOR PIO COMMAND
037 2870 D7      MOVBC
038 2871 D5      PUSH     D      / SAVE READ COMMAND
039 2872 CD052D CALL     TOPIO   / DO THE READ
040
041 2875 D1      POP      D      / RESTORE READ COMMAND
042 2876 E1      POP      H      / RESTORE REGS
043 2877 C1      POP      B
044 2878 C9      RET
045
046            EJECT

```

```

001 SUBJOB \      NXTADR - NEXT BUFFER ADDR
002
003
004 /           / THIS RTN WILL COMPUTE THE START ADDR OF
005 /           / THE NEXT BUFFER TO BE READ FROM THE 484.
006
007 /           / ** ENTRY
008
009 /           /      A=RAM SEG #
010 /           /      3=LOGIC
011 /           /      2=COILS
012 /           /      1=REGISTERS
013 /           /      B/C=SIZE OF LAST READ(BYTES)
014 /           /      H/L=LAST START ADDR FOR READ
015
016 /           / CALL NXTADR
017
018 /           / ** EXIT
019
020 /           /      H/L=START ADDR OF READ
021 /           /      A=?
022

```

```

023 NXTADR,
024 2879 C5      PUSH     B      / SAVE REG
025
026 287A FE01    CPI      TYPE3   / IS IT REG RAM?
027 287C CA8228 JZ       NXTREG  / 0--> REG RAM
028
029 /           / LOGIC OR COIL RAM
030 /           / INC BY PREVIOUS COUNT
031
032 287F 09      DAD      B      / INC BY LAST COUNT
033 2880 C1      POP      B      / RESTORE
034 2881 C9      RET          / ALL DONE
035
036 /           / REGISTER RAM - INC BY 1/2 LAST COUNT
037
038 NXTREG,
039 2882 79      MOV      A; C    / DIVIDE COUNT BY 2
040 2883 0F      RRC          / DONE
041 2884 4F      MOV      C; A    / PUT BACK
042 2885 09      DAD      B      / GET NEXT ADDR
043 2886 C1      POP      B      / RESTORE
044 2887 C9      RET          / ALL DONE
045
046            EJECT

```

```

001 SUBJOB \      GETSIZ - FIND MAX SIZE OF READ
002
003 /           / THIS SUBROUTINE WILL RETURN THE MAXIMUM
004 /           / # OF BYTES. THAT CAN BE READ FROM A 484
005 /           / SIZE IS LIMITED BY THE SIZE OF THE BUFFER
006 /           / AND THE HIGHEST RAM LOCATION (ALWAYS EVEN LOC)

```

```

007
008 / ** ENTRY
009
010 / A = RAM TYPE
011 / 3 = LOGIC
012 / 2 = COIL
013 / 1 = REGISTER
014 / B/C = MAX SIZE REQUESTED
015 / H/L = ADDR TO BEGIN AT
016 / EOUSEG= HIGHEST ADDRESS IN RAM
017 / SEGMENT - (2 BYTES. HI,LO)
018 / I. E LOGIC RAM, COIL RAM, REG RAM.
019
020 / CALL GETSIZ
021
022 / ** EXIT
023
024 / B/C= # OF BYTES TO BE READ
025 / IF CY=0 --> OK TO READ
026 / IF CY=1 --> END OF RAM SEGMENT
027
028 EJECT

001 GETSIZ,
002 2888 D5 PUSH D / SAVE REG
003 2889 E5 PUSH H / SAVE ADDR
004 288A F5 PUSH PSW / SAVE RAM TYPE
005 288B 2B DCX H / MAKE 0 REL
006
007 / IF REGISTER RAM- DIVIDE BYTE COUNT BY 2
008
009 289C FE01 CPI TYPE3 / IS IT REG RAM?
010 288E C29428 JNZ GETSZ2 / 0--> REG RAM
011 2891 79 MOV A,C / GET COUNT
012 2892 0F RRC / DIVIDE BY 2
013 2893 4F MOV C,A / PUT BACK IN A
014
015 GETSZ2,
016
017 2894 C5 PUSH B / MOVE SIZE TO D/E
018 2895 D1 POP D / DONE
019 2896 19 DAD D / SEE IF WITHIN RANGE
020 2897 2B DCX H / MAKE 0 REL
021
022 2898 EB XCHG / SAVE TEST ADDR IN D/E
023 2899 21C0FE LXI H,EOUSEG/ GET LAST VALID ADDR
024 289C E7 GETHL / DONE
025 289D EB XCHG / TEST ADDR BACK TO H/L
026
027 GETSZ5,
028 289E 78 MOV A,B / IF 0 --> END OF SEGMENT
029 289F B1 ORA C / TEST B/C FOR 0
030 28A0 CAB528 JZ GETEND / SET END OF SEG FLAG
031 28A3 F7 DCMP / ARE WE LEGAL?
032 28A4 D2BA28 JNC GETSXT / CY=0 --> LEGAL
033
034 / ,DECREMENT SIZE & TRY AGAIN
035
036 / IF REGISTER RAM --> DECR BY 1
037 / -ELSE DECR BY 2
038
039 28A7 F1 POP PSW / GET RAM TYPE
040 28A8 E1 POP H / GET START ADDR
041 28A9 E5 PUSH H / SAVE START ADDR
042 28AA F5 PUSH PSW / SAVE RAM TYPE AGAIN
043 28AB FE01 CPI TYPE3 / IS IT REGISTER?
044 28AD CAB128 JZ REGRAM / 0--> REGISTER
045
046 EJECT
047 / NOT REGISTER RAM
048 / DECR BY 2
049
050 28B0 0B DCX B / DECR SIZE BY 1
051
052 REGRAM,
053 28B1 0B DCX B / DECR SIZE

```



```

054 28B2 C39428      JMP      GETSZ2 / TRY AGAIN
055
056 /               AT END OF RAM SEGMENT
057 /               SET CY = 1 & RETURN
058
059 GETEND,
060 28B5 F1          POP      PSW      / REST STACK
061 28B6 E1          POP      H        / RESTORE STACK
062 28B7 D1          POP      D        /
063 28B8 37          STC          / CY=1 (END OF SEG)
064 28B9 C9          RET
065
066 /               ALL DONE - HAVE VALID SIZE IN B/C
067
068 GETSXT,
069 28BA F1          POP      PSW      / REST STACK
070
071 /               IF TYPE 3 - MULT COUNT BY 2
072
073 28BB FE01        CPI      TYPE3   / IS IT REG RAM?
074 28BD C2C528      JNZ     GETSX2  / 0--> REGISTER RAM
075 28C0 C5          PUSH     B        / MOVE COUNT TO H/L
076 28C1 E1          POP      H        / DONE
077 28C2 29          DAD     H        / DOUBLE COUNT
078 28C3 E5          PUSH     H        / MOVE BACK TO B/C
079 28C4 C1          POP      B        / DONE
080
081 GETSX2,
082 28C5 E1          POP      H        / RESTORE ADDR
083 28C6 D1          POP      D        / RESTORE D/E
084 28C7 A7          CLC          / CLEAR CARRY
085 28C8 C9          RET          / RETURN
086
087 EJECT
088 SUBJOB \         CSFRMT-- FORMAT RECORD TO INTEL
089
090 /               THIS SUBR WILL TAKE RSPBUF AND SET UP A BUFFER
091 /               IN INTEL FORMAT TO BE SENT OUT OF PORT #2
092 /               IT MUST CONVERT FROM BINARY TO ASCII HEX
093
094 /               ** ENTRY
095
096 /               E = READ CMND GIVEN TO "PIO"
097 /               RSPBUF SET UP AFTER CALL TO PIO (READ)
098
099 /               CALL CSFRMT
100
101 /               ** EXIT
102
103 /               CASBUF FORMATTED TO BE OUTPUT TO PORT 2
104
105 EJECT

001 CSFRMT,
002 28C9 C5          PUSH     B        / SAVE REGS
003 28CA E5          PUSH     H
004 28CB D5          PUSH     D
005
006 28CC 210A0D      LXI     H,CRLF  / GET CRLF
007 28CF EB          XCHG          / TO D/E
008 28D0 21C4FE      LXI     H,CASBUF / PTR TO OUTPUT BUFFER
009 28D3 EF          MOVDE         / STORE CRLF
010
011 28D4 3E3A        MVI     A,ASCCOL / COLON = 'START OF REC'
012 28D6 77          MOV     M,A     / STORE IT
013 28D7 23          INX     H       / BUMP PTR (DEST)
014
015 /               COMPUTE SIZE OF RESPONSE
016 /               E = READ CMND GIVEN TO PIO
017 /               SUBTRACT "CMDRED" & MULT BY 2 FOR # OF BYTES READ
018
019 28D8 C1          POP      B        / GET READ CMND SIZE TO A
020 28D9 C5          PUSH     B        / DOING
021 28DA 78          MOV     A,B     / DONE
022 28DB D610        SUI     CMDRED  / SUB BASIC CMND
023 28DD 07          RLC          / MULT BY 2

```

```

024
025 28DE 11A9FE LXI D;RSPBUF+1 / PTR TO HI ADDR RESP BUF
026 28E1 F5 PUSH PSW / SAVE BINARY SIZE OF DATA
027 28E2 F5 PUSH PSW
028 28E3 CD692C CALL BN2HX / CONVERT TO ASCII HEX
029 28E6 D7 MOVBC / STORE IN OUTPUT RECORD
030
031 28E7 1A LDAX D / GET HI ADDR
032 28E8 CD692C CALL BN2HX / CONVERT TO ASCII
033 28EB D7 MOVBC / STORE
034 28EC 13 INX D / LO ADDR
035 28ED 1A LDAX D / GET IT
036 28EE CD692C CALL BN2HX / CONVERT
037 28F1 D7 MOVBC / AND STORE
038 28F2 13 INX D / BUMP PTR TO DATA
039 28F3 D5 PUSH D / SAVE SOURCE PTR
040 28F4 EB XCHG / SAVE DEST. PTR IN D/E
041 28F5 213130 LXI H;ASC01 / GET RECORD TYPE '01'
042 28F8 EB XCHG / SWITCH
043 28F9 EF MOVDE / STORE '01'
044 28FA D1 POP D / GET BACK SOURCE PTR
045
046 EJECT
047 / NOW GET DATA BYTES, CONVERT TO ASC HEX
048 / & STORE IN OUTPUT RECORD BUFFER
049
050 CSFMT2,
051 28FB 1A LDAX D / GET A DATA BYTE
052 28FC CD692C CALL BN2HX / CONVERT TO ASCII HEX
053 28FF D7 MOVBC / STORE IN RECORD
054 2900 13 INX D / BUMP TO NEXT BYTE
055 2901 F1 POP PSW / GET COUNT
056 2902 3D DCR A / DONE YET?
057 2903 F5 PUSH PSW / SAVE AGAIN
058 2904 C2FB28 JNZ CSFMT2 / NOW 0--> GET NEXT BYTE
059
060 / ALL DATA IS IN RECORD - NOW ADD CHKSUM
061
062 2907 F1 POP PSW / GET COUNT
063 2908 F1 POP PSW / DONE
064 2909 C604 ADI 4 / ADD OVERHEAD
065 290B 21C7FE LXI H;CASBUF+3/ PTR TO 1ST CHAR IN RECORD
066 / TO BE CHECKSUMMED
067
068 290E CDE02C CALL CHEX80
069
070 / CLEAR STACK & RETURN
071
072 2911 D1 POP D / REST REGS
073 2912 E1 POP H / RESTORE REGS
074 2913 C1 POP B
075 2914 C9 RET
076
077 EJECT

```

```

001 SUBJOB \ CON484 - FIND HIGHEST LOC IN SEG
002
003 / THIS SUBR WILL STORE THE HIGHEST LOC IN
004 / A MEMORY SEGMENT (USER LOGIC, COIL RAM,
005 / OR REGS) IN 'EOUSEG'
006
007
008 / ** ENTRY
009
010 / A= 3 --> USER LOGIC
011 / 2 --> COIL RAM
012 / 1 --> REGISTERS
013 / SCONF1 & SCONF2 SET UP WITH
014 / 484 CONFIG AS PER SPEC SP-4810-002
015
016 / CALL CON484
017
018 / ** EXIT
019
020 / H/L = LOWEST ADDR OF SEGMENT
021 / EOUSEG= HIGHEST VALID ADDRESS
022 / IN RAM SEGMENT

```

```

023
024          CON484,
025 2915 C5      PUSH    B          /SAVE REGS
026 2916 D5      PUSH    D          /SAVE REGS
027
028 2917 010200  LXI     B;ADRUSE'    /SET UP FOR USER LOGIC RAM
029
030 291A FE03     CPI     TYPE1    /IS IN USER LOGIC?
031 291C CA3329  JZ      CONUSE    /0 --> USER LOGIC
032 291F FE02     CPI     TYPE2    /DISCREET , COIL RAM?
033 2921 CA5A29  JZ      CONCOL    /0--> COIL RAM
034

```

REGISTER RAM SEGMENT

```

036
037 2924 010240  LXI     B;4002 /LOWEST REG RAM ADDR
038 2927 21C0FE  LXI     H;E0USEG /SAME AS COIL RAM ADDR
039 292A E7      GETHL   /GET ADDR FOR PREV SEG
040 292B 7C      MOV     A;H      /SET FIELD BITS
041 292C C620    ADI     :20      /SET THAT BIT
042 292E 67      MOV     H;A      /PUT BACK IN H
043 292F 23      INX     H        /PREPARE TO BE OFFSET!
044 2930 C38E29  JMP     CON4A    /FINISHED
045

```

EJECT

USER LOGIC RAM SEGMENT

```

048
049          CONUSE,
050 2933 1184FE  LXI     D;SCONF1 /GET USER LOGIC CONFIG
051 2936 210001  LXI     H;@256  /ASSUME .25K
052
053 2939 1A      LDAX   D        /USER LOGIC CONFIG BYTE
054 293A FE08    CPI     SY0256 /IS IT .25K?
055 293C CA8E29  JZ     CON4A   /0--> .25K
056 293F 29      DAD    H        /TRY FOR .50K
057 2940 FE10    CPI     SY0512 /IS IT .50K?
058 2942 CA8E29  JZ     CON4A   /0--> .50K
059 2945 29      DAD    H        /TRY 1K
060 2946 FE20    CPI     SY1024 /IS IT 1K?
061 2948 CA8E29  JZ     CON4A   /0--> 1K
062 294B 29      DAD    H        /TRY FOR 2K
063 294C FE40    CPI     SY2048 /IS IT 2K?
064 294E CA8E29  JZ     CON4A   /0 --> 2K
065 2951 29      DAD    H        /SHOULD BE 4K
066 2952 FE80    CPI     SY4096 /IS IT 4K?
067 2954 CA8E29  JZ     CON4A   /0--> 4K
068
069          /      ERROR --> DON'T KNOW WHAT CONFIG IS
070
071 2957 C37E29  JMP     CON4ER  /GO DO ERROR CALL
072
073          EJECT

```

MEMORY TYPE = COIL RAM

```

002
003          CONCOL,
004 295A 014000  LXI     B;@64   /USE AS INCREMENTER
005 295D 1185FE  LXI     D;SCONF2 /GET COIL RAM CONFIG
006 2960 1A      LDAX   D        /--> AREG
007 2961 214000  LXI     H;@64   /ASSUME 1/16 K
008 2964 E610    ANI     SYS064  /IS IT 1/16 TH K?
009 2966 C28729  JNZ    CON4A0  /0 --> 1/16 K
010 2969 09      DAD    B        /TRY 1/8 K
011 296A 1A      LDAX   D        /GET BACK MASK
012 296B E620    ANI     SYS128  /IS IT 1/8 K?
013 296D C28729  JNZ    CON4A0  /0 --> 1/8 K
014 2970 09      DAD    B        /TRY .25K?
015 2971 1A      LDAX   D        /GET BACK MASK
016 2972 E640    ANI     SYS192  /IS IT .25K?
017 2974 C28729  JNZ    CON4A0  /0--> .25K
018 2977 09      DAD    B        /SHOULD BE .50K
019 2978 1A      LDAX   D        /GET BACK MASK
020 2979 E680    ANI     SYS256  /IS IT .50K?
021 297B C28729  JNZ    CON4A0  /0 --> .50K
022
023          CON4ER,

```



```

024 297E 11272D      LXI      D:MSGBDC/ ERROR --> CAN'T DETRMN CONFIG
025 2981 CD7E05      CALL     ERROR
026 2984 C3EF00      JMP      EXEC
027
028 /                SET UP FOR COIL RAM FIELD
029
030 CON4A0,
031 2987 7C           MOV      A:H      /SET FIELD BIT
032 2988 F620        ORI      :20      / DONE
033 298A 67          MOV      H:A      / MOVE BACK TO H/L
034 298B 010020      LXI      B:2000 / LOWEST VALID COIL ADDR
035
036 EJECT
037 /                HIGHEST VALID ADDRESS IS IN H/L
038 /                LOWEST VALID ADDRESS IS IN B/C
039
040 CON4A,
041 298E 2B          DCX      H        / OFFSET
042 298F EB          XCHG                     / STORE HI-LO
043 2990 21C0FE      LXI      H:EOUSEG/ TRICKY MNVR
044 2993 EF          MOVDE                     / DONE
045 2994 60          MOV      H:B      / NOW PUT LOW ADDR IN H/L
046 2995 69          MOV      L:C      / DONE
047
048 2996 D1          POP      D
049 2997 C1          POP      B
050 2998 C9          RET
051
052 EJECT
053 SUBJOB \          VALOAD - VALID LOAD?
054
055
056 /                THIS RTN WILL DETERMINE IF AN ADDRESS
057 /                IS VALID TO LOAD INTO A 484.
058 /                THIS IS USED IF THE NEXT ADDRESS TO BE
059 /                LOADED IS NOT THE NEXT CONTIGUOUS LOCATION.
060 /                IT IS VALID TO LOAD A SMALLER 484 INTO A LARGER
061 /                ONE. TO BE VALID --> THE ADDRESS MUST BE THE 15
062 /                LOC OF THE NEXT RAM SEGMENT.
063
064 /                ** ENTRY
065 /                A=RAM TYPE (SEE BELOW)
066 /                H/L=ADDR OF TAPE BLOCK
067
068 /                CALL VALOAD
069
070 /                ** EXIT
071 /                REG RET --> O.K. TO CONTINUE
072 /                "EOUSEG" IS UPDATED (SEE "CON484")
073 /                A=NEW RAM TYPE (SEE BELOW)
074 /                ERROR --> UNCOND. ERROR CALL (NO RET)
075
076 /                RAM TYPES:
077 /                3=LOGIC RAM
078 /                2=COIL RAM
079 /                1=REGISTER RAM
080
081
082 VALOAD,
083 2999 C5           PUSH     B        /SAVE
084 299A D5           PUSH     D
085
086 299B 47           MOV      B:A      / SAVE RAM TYPE
087 299C FE03        CPI      TYPE1    / IS IT PRESENTLY LOGIC RAM?
088 299E CAA929      JZ      LOGRAM   / 0-->LOGIC RAM
089 29A1 FE02        CPI      TYPE2    / IS IT COIL RAM?
090 29A3 CAAF29      JZ      COIRAM   / 0--> COIL RAM
091
092 /                THERE ARE NO OTHER VALID RAM TYPES --> ERROR!
093
094 29A6 C3C029      JMP      VALERR / DO AN ERROR CALL
095
096 EJECT
097 \                ** COME HERE FOR LOGIC RAM
098

```

```

099          LOGRAM,
100 29A9 110020      LXI      D: 2000 / 1ST VALID COIL RAM LOC
101 29AC C3B229      JMP      VALOD5 / GO VALIDATE
102
\          ** COME HERE FOR COIL RAM
104
105          COIRAM,
106 29AF 110240      LXI      D: 4002 / 1ST VALID REGISTER ADDRESS
107          VALOD5,
108 29B2 F7          DCMPL      / SAME AS OUR ADDRESS?
109 29B3 C2C029      JNZ      VALERR / .NE. 0--> ERROR
110
111          /          WE HAVE A VALID ADDRESS, UPDATE "EOUSEG"
112
113 29B6 78          MOV      A: B      / GET BACK RAM TYPE
114 29B7 3D          DCR      A          / UPDATE TO NEW TYPE
115 29B8 F5          PUSH     PSW      / SAVE RAM TYPE
116 29B9 CD1529      CALL     CON484 / UPDATE "EOUSEG"
117
118 29BC F1          POP      PSW      / RESTORE RAM TYPE TO A
119 29BD D1          POP      D          / RESTORE
120 29BE C1          POP      B
121 29BF C9          RET
122
\          ** COME HERE FOR ERROR
124
125          VALERR,
126 29C0 114A27      LXI      D: MSGADI / INVALID ADDRESS
127 29C3 CD7E05      CALL     ERROR / DO AN ERROR CALL
128 29C6 C3EF00      JMP      EXEC / GO TO EXEC
129
130          EJECT

001          SUBJOB \          LOAD - LOAD 484 FROM PORT 2
002
003          /          THIS ROUTINE WILL LOAD A 484 CONTROLLER
004          /          FROM AN INTEL FORMAT TAPE THROUGH PORT 2
005          /          IT WILL FIRST TRAP THE 484 THEN INIT
006          /          THE MEMORY.
007          /          ALL RECORDS MUST BE TYPE '01' AND
008          /          ADDRESSES MUST BE CONTIGUOUS
009          /          A TAPE OF A LARGER CONTROLLER MAY NOT BE
010          /          LOADED INTO A SMALLER 484 BUT THE
011          /          OPPOSITE IS QUITE LEGAL
012
013          /          ENTRY FROM SUPERVISORY MODE VIA
014          /          SELECTION OF 'LOAD' KEY
015
016          /          EXIT TO EXEC
017
018          /          IN CASE OF ERROR --> UNCOND ERROR CALL
019          /          BAD ADDRESS
020          /          MEMORY OVERFLOW
021          /          CHECKSUM ERROR
022          /          HARDWARE ERROR
023          /          LOADING WILL CEASE
024          /          BACK TO SUPER MODE
025
026          EJECT

001          LOAD,
002          /          DISPLAY MESSAGE: "LOADING"
003
004 29C9 114A2D      LXI      D: MSGLDG / MSSG ADDR
005 29CC CD681F      CALL     KU04 / DISPLAY ADVISORY
006
007
008          /          TRAP THE 484 & INIT MEMORY
009
010 29CF 110480      LXI      D: CMDSTP!: 100+LENSTP/ STOP 484 COMMAND
011          /          / LENGTH TO E
012 29D2 CD8125      CALL     PIO / ISSUE COMMAND
013 29D5 C2EF00      JNZ      EXEC / .NE. 0 --> ERROR, SO QUIT!
014
015 29D8 1104A0      LXI      D: CMDINI!: 100+LENINI/ INIT 484 CMND
016          /          / LENGTH TO E

```

```

017 29DB CD8125      CALL    PIO      / ISSUE COMMAND
018 29DE C2EF00      JNZ     EXEC     / .NE. 0 --> ERROR, SO QUIT!
019
020 /                / 484 IS TRAPPED AND INIT'D
021 /                / INIT PORT # 2
022
023 29E1 CDCC2A      CALL    P2INIT   / INIT PORT #2
024
025 29E4 3E03        MVI     /A;TYPE1 / GET HIGHEST ADDR IN SEG
026 LOAD05,
027 29E6 F5          PUSH    PSW      / SAVE MEM TYPE
028 29E7 CD1529      CALL    CON484   / .HIGHEST ADDR IN 'EOUSEG'
029 /                / ON RET, LOWEST IN H/L
030
031
032 LOAD10,
033
\                INPUT A RECORD FROM PORT 2
035
036 29EA CD292B      CALL    P2RIO    / READ A RECORD
037 29ED DA422A      JC      LOAD30   / CY=1 --> END OF TAPE
038 29F0 0192FE      LXI    B;CMBUF+2 / DEST PTR
039 29F3 CDA62B      CALL    UNFORM   / PLACE INTO CMBUF
040 /                / TO SHIP TO 484
041 29F6 EB          XCHG          / MOVE ADDR TO D/E
042 29F7 2193FE      LXI    H;CMBUF+3/ PICK UP ADDRESS & SEE IF VALID
043 29FA E7          GETHL        / DONE
044 29FB E5          PUSH    H       / SAVE ADDR (484)
045 LOAD20,
046 29FC F7          DCMPL        / ADDRESS SHOULD BE SAME
047 29FD CA072A      JZ      LOAD25   / 0--> CONTIG. ADDR, OK
048
049 /                / IF ADDRESS IS NOT THE NEXT CONTIGUOUS
050 /                / LOC, ALL IS NOT LOST ...YET.
051 /                / A SMALLER 484 MAY LOAD INTO A LARGER ONE.
052 /                / SEE IF ADDR IS THE 1ST VALID ADDR OF THE NEXT
053 /                / SEGMENT OF RAM.
054
055 2A00 E1          POP     H       / GOT TO GET TO RAM TYPE
056 2A01 F1          POP     PSW    / GOT RAM TYPE
057 2A02 CD9929      CALL    VALOAD   / VALIDATE THIS ADDRESS TO SEE
058 /                / IF IT IS IN NEXT SEGMENT
059
060 /                / RETURNED FROM "VALOAD" O.K. TO CONTINUE
061
062 2A05 F5          PUSH    PSW    / SAVE RAM TYPE
063 2A06 E5          PUSH    H     / SAVE ADDR (484)
064
065 LOAD25,
066
067 2A07 3A92FE      LDA     CMBUF+2/ LENGTH OF CMND
068 2A0A F5          PUSH    PSW    / SAVE COUNT
069 2A0B F5          PUSH    PSW    / AGAIN
070 2A0C 110820      LXI    D;CMDWRT!:100+LENWRT-2/WRITE COMMAND / LENG
071 2A0F 82          ADD     D      / INDEX PROPER COMMAND
072 2A10 57          MOV    D;A    / INIT D-REG
073 2A11 F1          POP     PSW    / CALC CORRECT LENGTH
074 2A12 07          RLC          / MULT BY 2 FOR BYTE COUNT
075 2A13 83          ADD     E     / ACCOUNT FOR DATA
076 2A14 5F          MOV    E;A    / BACK TO E
077 2A15 CD052D      CALL    TOPIO   / LOAD CONTROLLER W/BUFFER
078
079 /                / THE DATA WAS JUST SENT TO THE 484
080 /                / UPDATE THE ADDRESS FOR VERIFICATION
081
082 2A18 C1          POP     B     / GET COUNT BACK (LSB)
083 2A19 E1          POP     H     / GET ADDR BACK
084 2A1A 78          MOV    A;B    / MOVE COUNT TO
085 2A1B 07          RLC          / MULT BY 2 FOR
086 /                / BYTE COUNT
087 2A1C 4F          MOV    C;A    / MOVE TO C
088 2A1D 0600        MVI    B;0    / MSB OF COUNT
089 2A1F F1          POP     PSW    / GET RAM TYPE
090 2A20 F5          PUSH    PSW    / SAVE RAM TYPE
091 2A21 CD7928      CALL    NXTADR  / GET NEXT VALID ADDR

```



```

097
098 / SEE IF NEXT SEGMENT
099
100 2A24 EB XCHG / TAPE ADDR TO D/E
101 2A25 21C0FE LXI H,EDUSEB/ HIGHEST 484 ADDR
102 2A28 E7 GETHL / HI-LO TO H/L
103 2A29 EB XCHG / SWITCH
104 2A2A F7 DCMF / COMPARE
105 2A2B DA312A JC NXTSEG / CY=1 --> NEXT SEGMENT
106 2A2E C3EA29 JMP LOAD10 / CONTINUE
107
108 / GET HIGHEST ADDR OF NEXT SEG
109
110 NXTSEG,
111 2A31 F1 POP PSW / GET MEM TYPE
112 2A32 3D DCR A / NXT SEG #
113 2A33 C2E629 JNZ LOAD00 / 0--> END OR 484 RANGE
114
115 /
116 /
117 /
118 /
119 /
120 /
121 /
122 /
123 /
124 /
125 /
126 /
127 2A42 11523D LXI D,MSGEDD/ "LOAD O.K."
128 2A45 CD681F CALL R004 / DFLY
129
130 2A48 C3EF00 JMP EXEC / EXIT
131
132 /
133 /
134 /
135 2A4B 114A27 LXI D,MSGAD1/ BAD ADDRESS
136 2A4E CD7E05 CALL ERROR / DO ERROR CALL
137 2A51 C3EF00 JMP EXEC / EXIT
138
139 /
140 /
141
142 SUBJOB \ VERIFY = VERIFY TAPE AGAINST 484
143
144 / THIS ROUTINE WILL READ AN INTEL FORMAT
145 / TAPE FROM PORT 2 AND VERIFY IT AGAINST
146 / THE 484 CONTROLLER
147 / ANY DATA MISMATCH WILL CAUSE A
148 / MESSAGE TO BE DISPLAYED & THE VERIFY
149 / WILL STOP AND EXIT TO EXEC
150
151 /
152 /
153 /
154 /
155 /
156 /
157 /
158 /
159 /
160 /
161 /
162 /
163 /
164 /
165 /
166 /
167 /
168 /
169 /
170 /
171 /
172 /
173 /
174 /
175 /
176 /
177 /
178 /
179 /
180 /
181 /
182 /
183 /
184 /
185 /
186 /
187 /
188 /
189 /
190 /
191 /
192 /
193 /
194 /
195 /
196 /
197 /
198 /
199 /
200 /
201 /
202 /
203 /
204 /
205 /
206 /
207 /
208 /
209 /
210 /
211 /
212 /
213 /
214 /
215 /
216 /
217 /
218 /
219 /
220 /
221 /
222 /
223 /
224 /
225 /
226 /
227 /
228 /
229 /
230 /
231 /
232 /
233 /
234 /
235 /
236 /
237 /
238 /
239 /
240 /
241 /
242 /
243 /
244 /
245 /
246 /
247 /
248 /
249 /
250 /
251 /
252 /
253 /
254 /
255 /
256 /
257 /
258 /
259 /
260 /
261 /
262 /
263 /
264 /
265 /
266 /
267 /
268 /
269 /
270 /
271 /
272 /
273 /
274 /
275 /
276 /
277 /
278 /
279 /
280 /
281 /
282 /
283 /
284 /
285 /
286 /
287 /
288 /
289 /
290 /
291 /
292 /
293 /
294 /
295 /
296 /
297 /
298 /
299 /
300 /
301 /
302 /
303 /
304 /
305 /
306 /
307 /
308 /
309 /
310 /
311 /
312 /
313 /
314 /
315 /
316 /
317 /
318 /
319 /
320 /
321 /
322 /
323 /
324 /
325 /
326 /
327 /
328 /
329 /
330 /
331 /
332 /
333 /
334 /
335 /
336 /
337 /
338 /
339 /
340 /
341 /
342 /
343 /
344 /
345 /
346 /
347 /
348 /
349 /
350 /
351 /
352 /
353 /
354 /
355 /
356 /
357 /
358 /
359 /
360 /
361 /
362 /
363 /
364 /
365 /
366 /
367 /
368 /
369 /
370 /
371 /
372 /
373 /
374 /
375 /
376 /
377 /
378 /
379 /
380 /
381 /
382 /
383 /
384 /
385 /
386 /
387 /
388 /
389 /
390 /
391 /
392 /
393 /
394 /
395 /
396 /
397 /
398 /
399 /
400 /
401 /
402 /
403 /
404 /
405 /
406 /
407 /
408 /
409 /
410 /
411 /
412 /
413 /
414 /
415 /
416 /
417 /
418 /
419 /
420 /
421 /
422 /
423 /
424 /
425 /
426 /
427 /
428 /
429 /
430 /
431 /
432 /
433 /
434 /
435 /
436 /
437 /
438 /
439 /
440 /
441 /
442 /
443 /
444 /
445 /
446 /
447 /
448 /
449 /
450 /
451 /
452 /
453 /
454 /
455 /
456 /
457 /
458 /
459 /
460 /
461 /
462 /
463 /
464 /
465 /
466 /
467 /
468 /
469 /
470 /
471 /
472 /
473 /
474 /
475 /
476 /
477 /
478 /
479 /
480 /
481 /
482 /
483 /
484 /
485 /
486 /
487 /
488 /
489 /
490 /
491 /
492 /
493 /
494 /
495 /
496 /
497 /
498 /
499 /
500 /

```

NOTE. TAPE MUST HAVE BEEN MADE BY P180 OR EQUIV.
(16 BYTE RECORDS THRU N-1)

077

\ ***** GOOD DATA-MATCH**

079

080 VERCHK,

081 2AA7 119F2D LXI D;MSGVOK/ DATA MATCH GOOD

082 2AAA CD681F CALL KU04 / DISPLAY ADVISORY

083 2AAD C3EF00 JMP EXEC

084

085 EJECT

001

SUBJOB \ VERCHK - MIS-MATCH HNDLR

002

003

/ THIS RTN WILL DETERMINE WHICH RAM SEGMENT

004

/ A DATA MIS-MATCH OCCURED IN AND DISPLAY AN

005

/ APPROPRIATE MESSAGE.

006

\

NOTE: NOTE: NOTE: NOTE:

008

009

/ IF CONTROLLER IS RUNNING,

010

/ IT IS EMINENT THAT A MIS-MATCH WILL

011

/ OCCUR SINCE THE COIL RAM AND THE

012

/ REGISTER RAM IS CONSTANTLY

013

/ CHANGING.

014

015

/ ** ENTRY

016

/ H/L = 484 ADDRESS (NORMAL ENTRY)

017

/ @ VERCK5 DON'T NEED H/L SET!

018

019

020

/ JMP VERCHK (NORMAL ENTRY)

021

/ JMP VERCK5 (KNOWN REGISTER MIS-MATCH)

022

023

/ ** EXIT

024

/ DISPLAY APPROPRIATE MESSAGE IN

025

/ ADVISORY AREA AND JMP TO EXEC.

026

027

VERCHK,

028 2AB0 7C MOV A,H / GET FIELD BITS

029 2AB1 E6F0 ANI :OF0 / MASK

030 2AB3 11852D LXI D;MSGLNM / LOGIC NO MATCH

031 2AB4 FE00 CPI FIELD1 / IS IT LOGIC RAM?

032 2AB8 CAC62A JZ VEREXT / 0--> LOGIC RAM MIS-MATCH

033 2ABB 11922D LXI D;MSGCNM / COIL NO MATCH

034 2ABE FE20 CPI FIELD2 / IS IT COIL RAM?

035 2AC0 CAC62A JZ VEREXT / COIL RAM MIS-MATCH

036

\

ENTRY POINT FOR KNOWN REGISTER MIS

038

039

VERCK5,

040 2AC3 11782D LXI D;MSGRNM / REGISTER RAM MIS-MATCH

041

042

VEREXT,

043 2AC6 CD7E05 CALL ERROR / DO AN ERROR CALL

044 2AC9 C3EF00 JMP EXEC

045

046

EJECT

001

SUBJOB \ P2INIT - INIT PORT 2

002

003

/ THIS ROUTINE WILL INITIALIZE PRIRPHERAL

004

/ PORT 2 IN THE P180

005

006

007

/ ** ENTRY

008

009

/ NO ENTRY REQUIREMENTS

010

011

/ CALL P2INIT

012

013

/ ** EXIT

014

015

/ PORT 2 INITIALIZED

016

017

018

P2INIT,

019

020 2ACC 3E81 MVI A;PNULL


```

021 2ACE D33C      OUT      SP2CTL / LOAD NULL INSTRUCTION
022 2AD0 00        NOP                / PRECAUTIONARY WAIT
023 2AD1 D33C      OUT      SP2CTL / LOAD SECOND NULL
024
025 2AD3 3E50      MVI      A,SPC1R+SPCER / RESET COMMAND
026 2AD5 D33C      OUT      SP2CTL / RESET INTERFACE
027 2AD7 3E79      MVI      A,P2MODE / INTERFACE MODE
028 2AD9 D33C      OUT      SP2CTL / SET INTERFACE MODE
029 2ADB 3E10      MVI      A,SPCER/ INTERFACE STATE
030 2ADD D33C      OUT      SP2CTL / LOAD STATE
031

```

IF NO DEVICE IS ATTACHED TO PORT
ALLOW NO MORE FORWARD PROGRESS

```

034
035 2ADF DB3C      IN       SP2CTL / CHECK CONTROL PORT
036 2AE1 E680      ANI      SPSDSR / IS THERE A "DSR"?
037 2AE3 C0        RNZ                / OK IF .NE. 0
038
039 /              PORT 2 IS NOT ATTACHED - ERROR
040
041 2AE4 11322D     LXI      D;MSGN02 / PORT 2 EMPTY
042 2AE7 C17E05     CALL     ERROR
043 2AEA C3EF00     JMP      EXEC / QUIT RIGHT NOW!
044
EJECT

```

```

001 SUBJOB \        P2RDCH - READ CHAR FROM PORT 2
002
003 /              THIS 'SUBR' WILL INPUT ONE CHAR FROM
004 /              PORT 2 IN CASE OF ANY HADWARE OR
005 /              CHECKSUM ERRORS - AN ERROR MESSAGE
006 /              WILL BE DISPLAYED AND CY SET = 1.
007
008
009 /              * *ENTRY
010
011 /              NO ENTRY REQUIREMENTS
012
013 /              CALL P2RDCH
014
015 /              * *EXIT
016
017 /              CHAR IN A REG
018 /              IF ERROR --> UNCOND ERROR CALL
019 /              DISPLAY ERROR MESSAGE, GO TO EXEC
020
021
EJECT

```

```

001 P2RDCH,
002 2AED C5        PUSH     B / SAVE REGS
003 2AEE D5        PUSH     D
004 2AEF E5        PUSH     H
005
006 2AF0 11322D     LXI      D;MSGN02/ NOT CONNECTED TO PORT
007 P2RD05,
008 2AF3 DB3C      IN       SP2STA / GET PORT STATUS IN AREG
009 2AF5 47        MOV      B,A / AND B
010 2AF6 E680      ANI      SPSDSR / CHECK IF DATA SET RDY
011 2AF8 CA1F2B     JZ      P2RDOE / 0 --> ERROR
012 2AFB 78        MOV      A,B / GET BACK STATUS
013 2AFC E602      ANI      SPSRRY / CHECK FOR RCVR RDY
014 2AFE CAF32A     JZ      P2RD05 / 0 --> NOT READY
015

```

READY TO RECEIVE A CHAR

```

017
018 2B01 78        MOV      A,B / GET STATUS
019 2B02 E628      ANI      SPSFE+SPSPE/ CHECK PARITY/FRAMING ERROR
020 2B04 CA0D2B     JZ      P2RDO2 / 0 --> NO ERROR
021
022 /              *** PARITY/ FRAMING ERROR
023
024 2B07 110C00     LXI      D;MSGPAX/ ERROR MESSAGE
025 2B0A C31F2B     JMP      P2RDOE / GO DISPLAY
026
027 P2RDO2,
028 2B0D 78        MOV      A,B / GET STATUS BACK

```

```

029 2B0E E610      ANI     SPSOE   / CHECK FOR OVERRUN
030 2B10 CA192B     JZ      PZRDO3  / 0 --> NO ERROR
031
032 /              *** OVERRUN ERROR
033
034 2B13 110B00     LXI     D,MSGDVX / DISPLAY MESSG
035 2B16 C31F2B     JMF     PZRDOE  / GO DO IT
036
037 PZRDO3,
038 2B19 DB3D       IN      SP2IN   / GET CHARACTER
039 2B1B E1         POP     H       / RESTORE REGS
040 2B1C D1         POP     D
041 2B1D C1         POP     B
042 2B1E C9        RET                    / CHAR IN A-REG
043
044 /              COME HERE IN CASE OF ERROR
045 /              ADDR OF ERROR MESSG IN D/E
046
047 PZRDOE,
048 2B1F CD7E05     CALL    ERROR   / DISPLAY ERROR & SET STATE
049 2B22 3E10       MVI     A,SPUER / TURN OFF TAPE
050 2B24 D33C       OUT     SP2CTL  / DONE
051 2B26 C3EF00     JMP     EXEC    / GO TO EXEC
052
053 EJECT

001 SUBJOB \        PZR10 - READ RECORD PORT 2
002
003 /              THIS RIN WILL INPUT AN INTEL FORMAT
004 /              RECORD FROM PORT 2 OF THE P180 AND
005 /              VALIDATE THE CHECKSUM.
006
007
008 /              ** ENTRY
009
010 /              NO ENTRY REQUIREMENTS
011
012 /              CALL PZR10
013
014 /              ** EXIT
015
016 /              RECORD INPUT TO 'CASBUF'
017 /              ERROR --> MESSG DISPLAYED
018 /              UNCOND ERROR CALL
019 /              IF END RECORD --> CY=1
020 /              ELSE CY=0
021
022 PZR10,
023 2B29 C5         PUSH    B       / SAVE REGS
024 2B2A D5         PUSH    D
025 2B2B E5         PUSH    H
026
027 2B2C 01C4FE     LXI     B,CASBUF / PTR TO STORAGE BUFFER
028
029 /              TURN ON RECEIVER
030
031 2B2F 3E16       MVI     A,SPCDTR+SPCER+SPCRE / RECEIVER ON
032 2B31 D33C       OUT     SP2CTL  / O.K.
033
034 PZR102,
035 2B33 CDDE2A     CALL    PZRDOH  / GET A CHAR
036 2B36 FE3A       CPI     A$COLL  / IS IT A COLON?
037 2B38 C2332B     JNZ     PZR102  / NOT 0--> TRY AGAIN
038
039 /              ** GOT A : BEGINNING OF INTEL RECORD
040 /              GET THE 2 SIZE OF RECORD CHARS
041
042 2B3B 07         STAX    B       / STORE CHAR
043 2B3C 03         INX     B       / BUMP PTR
044 2B3D CDDE2A     CALL    PZRDOH  / 1ST CHAR
045 2B40 07         STAX    B       / STORE CHAR
046 2B41 03         INX     B       / BUMP PTR
047 2B42 CDDE2A     CALL    PZRDOH  / 2ND
048 2B45 07         STAX    B       / STORE CHAR

```

```

049 2B46 03      INX      B      / BUMP PTR
050
051      /      CONVERT LENGTH TO BINARY FOR COUNTER
052
053 2B47 1105FE  LXI      D;CASBUF+1/ PTR TO BCD LENGTH
054 2B4A CD982C  CALL     H2BN2   / CONVERT TO BINARY (H/L)
055
\      SEE IF END RECORD
057
058 2B4D 7D      MOV      A,L     / TEST FOR END RECORD
059 2B4E B7      TST                     / DONE
060 2B4F CA572B  JZ       P2RIAA  / 0 --> END RECORD
061 2B52 A7      CLC                     / NOT END RECORD
062 2B53 F5      PUSH     PSW     / SAVE CY
063 2B54 C3592B  JMP      P2RIOA  / NOT END RECORD
064
065      /      IT IS THE END RECORD
066
067      P2RIAA,
068 2B57 37      STC                     / SIGNAL END RECORD
069 2B58 F5      PUSH     PSW     / SAVE CY
070
071      /      CONTINUE TO PROCESS RECORD
072
073      P2RIOA,
074 2B59 7D      MOV      A,L     / ADD OVERHEAD BYTES TO LENGTH
075 2B5A 65      MOV      H,L     / SAVE IN H
076 2B5B 07      RLC                     / DOUBLE FOR ASCII COUNT
077 2B5C C604  ADI      4       / DON'T INCLUDE CHECKSUM
078 2B5E 6F      MOV      L,A     / BACK TO L
079      P2RIOB,
080 2B5F CD2ED2A CALL     P2RDCH  / GET A CHAR
081 2B62 02      STAX     B       / STORE IT
082 2B63 03      INX      B       / BUMP POINTER
083 2B64 2D      DCR      L       / DONE YET?
084 2B65 C25F2B  JNZ     P2RIOB  / NOT 0--> GET ANOTHER
085
086      /      ** RECORD HAS BEEN INPUT EXCEPT FOR CHKSUM
087
088 2B68 CD2ED2A CALL     P2RDCH  / GET HI CHKSUM
089 2B6B 02      STAX     B       / STORE
090 2B6C 03      INX      B       / BUMP PTR
091 2B6D CD2ED2A CALL     P2RDCH  / LO CHKSUM
092 2B70 02      STAX     B
093
\      ** GET CHECKSUM FROM TAPE
095
096 2B71 CD2ED2A CALL     P2RDCH  / HI ORDER
097 2B74 57      MOV      D,A     / TO D
098 2B75 CD2ED2A CALL     P2RDCH  / LO ORDER
099 2B78 5F      MOV      E,A     / TO E
100
101      /      ** TURN OFF RECEIVER
102
103 2B79 3E10  MVI     A;SPCER / OFF RCVR
104 2B7B D33C  OUT     SP2CTL  / DONE
105
\      ** NOW VERIFY CHECKSUM
107
108 2B7D 7C      MOV      A,H     / CTR TO A
109 2B7E 07      RLC                     / MULT BY 2 FOR ASCII HEX
110 2B7F C608  ADI     @8      / ADD OVERHEAD
111 2B81 0105FE LXI     B;CASBUF+1 / 1ST CHAR TO CHKSUM
112 2B84 210000 LXI     H;0      / ADDR OF CHKSUM TO B/C
113 2B87 6F      MOV      L,A     / COUNT
114 2B88 09      DAD     B       / COMPUTE ADDR OF CHKSUM
115 2B89 05      PUSH     B       / SWITCH B/C H/L
116 2B8A E5      PUSH     H
117 2B8B C1      POP      B
118 2B8C E1      POP      H       / DONE
119 2B8D 0F      RRC                     / DIVIDE BY 2 FOR BINARY COUNT
120 2B8E CDE02C CALL     CHEX80  / COMPUTE CHECKSUM
121
122      /      NOW VERIFY 2 CHKSUMS TO BE SAME
123      /      TAPE CHKSUM IN D/E
124      /      COMPUTED CHECKSUM (B/C --> ADDR)

```



```

125
126 2B91 C5      PUSH   B      / GET ADDR TO H/L
127 2B92 E1      POP     H      / DONE
128 2B93 E7      GETHL             / GET CHKSUM TO H/L
129 2B94 F7      DCOMP          / COMPARE
130 2B95 C29D2E  JNZ     PZCKER  / .NE. 0--> CHKSUM ERROR
131
132
133      /      DONE-CLEAN STACK & RET
134
135 2B98 F1      POP     PSW    / RESTORE CY FLAG
136 2B99 E1      POP     H
137 2B9A D1      POP     D
138 2B9B C1      POP     B
139 2B9C C9      RET

```

```

140 \ ### COME HERE IF CHKSUM ERROR

```

```

142
143      PZCKER,
144 2B9D 110727  LXI    D;MSGCHK / CHECKSUM ERROR
145 2BA0 CD7E05  CALL   ERROR
146 2BA3 C3EF00  JMP    EXEC
147
148      EJECT

```

```

001      SUBJOB \      UNIFORM - SET UP 484 CMD
002
003      /      THIS RTN WILL TAKE A RECORD IN 'CASBUF'
004      /      CASLII? & CONVERT IT TO A 484 MESSAGE
005      /      COMMAND (BINARY) ( FOR A WRITE COMMAND)
006
007
008      /      ** ENTRY
009
010      /      B/C --> PLACE IN BUFFER FOR 1ST
011      /      CHAR TO BE STORED
012      /      SHOULD POINT TO LENGTH BYTE
013
014      /      CALL UNIFORM
015
016      /      ** EXIT
017
018      /      SET UP IN 484 CMD FORMAT
019      /      --> XXCAADDDDD...MM
020      /      WHERE C = BINARY BYTE COUNT, A= ADDRESS
021      /      HI,LO, D= DATA HI,LO, M= MASK HI,LO.
022      /      IF RECORD TYPE. NE. 01 --> ERROR
023      /      UNCOND CALL TO ERROR (NO RET)
024
025

```

```

\      NOTE:      NOTE:      NOTE:      NOTE:

```

```

027
028      /      THE LENGTH STORED IN DEST WILL
029      /      = THE BINARY # OF DATA BYTES IN
030      /      CASBUF DIVIDED BY 2
031      /      I. E. IF 16 BYTES WERE READ,
032      /      8 WILL BE STORED!!
033

```

```

001      UNIFORM,
002 2BA6 D5      PUSH   D
003 2BA7 E5      PUSH   H
004 2BA8 C5      PUSH   B
005 2BA9 C5      PUSH   B      / ONE MORE TIME
006
007 2BAA 11C5FE  LXI    D;CASBUF+1/ CONVERT LENGTH TO BINARY
008 2BAD CD982C  CALL   H2BN2
009 2BB0 7D      MOV    A;L      / DIVIDE BY 2 FOR BINARY COUNT
010 2BB1 0F      RRC             / DONE
011 2BB2 6F      MOV    L;A      / PUT BACK IN L
012 2BB3 65      MOV    H;L      / SAVE IN H TOO
013 2BB4 EB      XCHG          / COUNT TO D/E
014 2BB5 E1      POP     H      / GET DEST ADDR
015 2BB6 73      MOV    M;E      / STORE COUNT
016 2BB7 23      INX    H      / BUMP DEST PTR
017 2BB8 D5      PUSH   D      / SAVE COUNT
018 2BB9 E5      PUSH   H      / DEST PTR-SAVE

```

```

019
020 / ** PROCESS ADDRESS
021
022 2BBA 11C7FE LXI D,CASBUF+3/ PTR TO BCD ADDR
023 2BBD CD842C CALL H2BN4 / CONVERT TO BINARY
024 2BC0 D1 POP D / GET DEST ADDR
025 2BC1 C1 POP B / GET COUNT OFF STACK
026 2BC2 EB XCHG / BINARY TO D/E
027 2BC3 EF MOVDE / STORE
028 2BC4 C5 PUSH B / STORE COUNT AGAIN
029 2BC5 E5 PUSH H / SAVE DEST (NEW)
030
031 / *** NOW VERIFY RECORD TYPE
032
033 2BC6 11CBFE LXI D,CASBUF+7/ PTR TO RECORD TYPE
034 2BC9 CD982C CALL H2BN2 / SEE IF TYPE '01'
035 2BCC 7D MOV A,L / TYPE TO A
036 2BCD FE01 CRI TYPE01 / IS IT VALID?
037 2BCF C2F62E JNZ BADREC / NOT 0--> INVALID REC
038
039 EJECT
040 / *** NOW PROCESS DATA
041
042 2BD2 11CDFE LXI D,CASBUF+09/ PTR TO 1ST DATA BYTE
043 2BD5 D5 PUSH D / SAVE PTR (SOURCE)
044 UNFRM2,
045 2BD6 CD842C CALL H2BN4 / CONVERT 21 BYTES (HI,LO)
046 2BD9 C1 POP B / SOURCE ADDR TO B/C
047 2BDA D1 POP D / DEST ADDR TO D/E
048 2BDB EB XCHG / SWITCH
049 2BDC EF MOVDE / PLACE BYTES AT DEST
050 2BDD D1 POP D / GET BACK COUNT
051 2BDE 15 DCR D / DCR & SEE IF DONE
052 2BDF CAEE2E JZ UNFRM5 / 0 --> DONE WITH DATA
053
054 / *** SET UP FOR 2 MORE DATA BYTES
055
056 2BE2 D5 PUSH D / SAVE COUNT
057 2BE3 E5 PUSH H / SAVE DEST
058 2BE4 03 INX B / GET TO NEXT WORD
059 2BE5 03 INX B / SKIP OVER NEXT
060 2BE6 03 INX B / 3 CHARS
061 2BE7 03 INX B
062 2BE8 C5 PUSH B / SAVE SOURCE ADDR
063 2BE9 D1 POP D / MOVE SRC ADDR TO D/E
064 2BEA D5 PUSH D / & REST. STACK
065 2BEB C3D62E JMP UNFRM2 / PROCESS NEXT WORD
066
067 UNFRM5,
068
069 / *** ALL DONE WITH DATA
070
071 / STORE MASK IN NEXT 2 BYTES FOR WRITE
072
073 2BEE 010000 LXI B,0 / MASK = 0
074 2BF1 D7 MOVBC / STORE
075
076 2BF2 C1 POP B / REST REGS
077 2BF3 E1 POP H
078 2BF4 D1 POP D
079 2BF5 C9 RET
080
081 BADREC,
082 2BF6 113F2D LXI D,MSGBDR/ MSSG FOR RECORD TYPE ERROR
083 2BF9 CD7E05 CALL ERROR
084 2BFC C3EF00 JMP EXEC / FATAL ERROR - GO TO EXEC
085
086 EJECT

SUBJOB \ P2TIO - TRANSMIT REC TO PORT 2
001
002 /
003 / THIS RTN WILL OUTPUT AN INTEL FORMAT
004 / RECORD STORED IN 'CASBUF' OVER PORT 2
005 / THE FORMAT MUST BE AS FOLLOWS:
006
007 / CRLF:NNADDR01DDDDDDCK
008

```

```

009 / WHERE NN= THE # OF DATA BYTES IN THE REC
010 / ADDR = THE ADDRESS (484 - HI/LO)
011 / DD IS THE DATA
012 / CK IS THE CHECKSUM OF ALL BYTES AFTER THE
013 / 16->8 BIT ADDITIVE W/O END AROUND CARRY
014
015 / ** ENTRY
016
017 / CASBUF SET UP AS DESCRIBED
018
019 / CALL P2T10
020
021 / ** EXIT
022
023 / BUFFER WILL HAVE BEEN SENT OUT PORT 2
024 / IN CASE OF ERROR --> DISPLY MSSG &
025 / EXIT TO EXEC
026
027 / EJECT

001 P2T10,
002 2BFF C5 PUSH B / SAVE REGS
003 2C00 D5 PUSH D
004 2C01 E5 PUSH H
005
006 2C02 11C7FE LXI D,CASBUF+3 / PTR TO LENGTH
007 2C05 CD982C CALL H2BN2 / CONVERT TO BINARY
008 2C08 7D MOV A,L / ADD IN OVERHEAD BYTES
009 2C09 07 RLC / DOUBLE FOR BCD COUNT
010 2C0A C60D ADI @13 / OVERHEAD
011 2C0C 5F MOV E,A / SAVE IN E
012
013 2C0D 21C4FE LXI H,CASBUF/ SOURCE PTR
014
015
016 / TURN ON PORT 2 XMIT
017
018 2C10 3E35 MVI A,SPCRS+SPCR+SPCR+SPCR/ SENT RTS TO DR
019 2C12 D33C OUT SP2CTL / DONE
020
021 / MUST DELAY APPROX .5 SEC
022 / TO ALLOW TAPE TO GET UP TO SPEED
023
024 2C14 CD5B2C CALL DELHLF/ DELAY .5 SEC
025 / SEND 4 NULLS TO PORT 2 TO AVOID
026 / LOST CHARACTERS
027
028 2C17 0604 MVI B,4 / COUNTER
029 P2T107,
030 2C19 AF CLA / NULL CHAR
031 2C1A CD352C CALL P2TCH / XMIT
032 2C1D 05 DCR B / DONE YET?
033 2C1E C2192C JNZ P2T107 / 0--> DONE
034
035
036 / OUTPUT A CHAR
037
038 P2T105,
039 2C21 7F MOV A,M / GET CHAR
040 2C22 CD352C CALL P2TCH / TRANSMIT
041 2C25 23 INX H / BUMP PTR
042 2C26 1D DCR E / DONE?
043 2C27 C2212C JNZ P2T105 / NEG--> DONE
044
045 EJECT
046 / RECORD HAS BEEN TRANSMITTED
047
048 / DELAY BEFORE SHUTTING TAPE OFF TO ALLOW
049 / DATA TO BE WRITTEN
050
051 2C2A CD5B2C CALL DELHLF / DONE
052
053 / TURN OFF PORT 2 XMIT
054

```



```

055 2C2D 3E10          MVI    A,SPCER / HALT XMIT
056 2C2F D33C          OUT    SP2CTL / DONE
057
058
059 2C31 E1            POP    H
060 2C32 D1            POP    D
061 2C33 C1            POP    B
062 2C34 C9            RET
063
064                    EJECT

001                    SUBJOB \      P2TCH - TRANSMIT CHAR PORT 2
002
003                    /          THIS RTN WILL TRANSMIT A CHAR OVER PORT 2
004                    /          IN CASE OF ERROR --> DISPL MSSG & GO TO EXEC
005
006                    /          ** ENTRY
007
008                    /          A= CHAR
009
010                    /          CALL P2TCH
011
012                    /          ** EXIT
013
014                    /          CHAR TRANSMITTED
015                    /          IF ERROR --> DISPL MSSG & GO TO EXEC
016
017                    P2TCH,
018 2C35 C5            PUSH   B      / SAVE REGS
019 2C36 D5            PUSH   D
020 2C37 E5            PUSH   H
021
022 2C38 F5            PUSH   PSW   / SAVE CHAR
023
024                    P2TCH2,
025 2C39 11322D        LXI    D,MSGND2 / MSG IF NOT CONNECTED
026 2C3C DB3C          IN     SP2STA / GET STATUS OF PORT 2
027 2C3E 47            MOV    B,A    / SAVE IN B
028 2C3F F680          ANI   SPDSR  / SEE IF CONNECTED
029 2C41 CA512C        JZ    P2TER  / 0 --> NOTHING CONNECTED TO PORT
030
031 2C44 78            MOV    A,B    / GET STATUS BACK
032 2C45 E601          ANI   SPSTRY / SEE IF XMTR READY
033 2C47 CA392C        JZ    P2TCH2 / 0 --> NOT READY
034
035                    /          SEND THE CHARACTER
036
037 2C4A F1            POP    PSW   / GET CHAR BACK
038 2C4B D33D          OUT    SP2OUT / XMIT IT
039
040 2C4D E1            POP    H
041 2C4E D1            POP    D
042 2C4F C1            POP    B
043 2C50 C9            RET
044
045                    EJECT

\          ERROR IN PORT 2
047
048                    P2TER,
049 2C51 CD7E05        CALL   ERROR
050 2C54 3E10          MVI    A,SPCER / TURN OFF TAPE
051 2C56 D33C          OUT    SP2CTL / DONE
052 2C58 C3EF00        JMP    EXEC   / EXIT
053
054                    EJECT
055                    SUBJOB \      DELHLF - .5 SEC DELAY
056
057
058                    /          THIS RTN WILL CAUSE A DELAY
059                    /          FOR APPROX .5 SEC. AND RETURN
060
061
062                    /          ** ENTRY
063                    /          NO ENTRY REQUIREMENTS
064
065                    /          CALL DELHLF

```

```

066
067
068
069
070
071
072 2C5B C5          DELHLF,      PUSH      B          / SAVE
073 2C5C 01AA3A      LXI        B, 2AAA / SET UP .5 SEC CTR
074
075
076 2C5F E3          DELHF2,      XTHL           / WASTE SOME TIME
077 2C60 E3          XTHL           / LEAVE AS FOUND
078 2C61 0B          DCX        B          / SEE IF DONE
079 2C62 79          MOV        A,C        / DONE YET?
080 2C63 B0          ORA        B          / TEST B/C FOR 0
081 2C64 C25F2C      JNZ        DELHF2    / 0--> DONE
082
083 2C67 C1          POP        B          / ALL DONE
084 2C68 C9          RET
085
086          EJECT

001          SUBJOB \      BN2HX - BIN TO ASCII HEX
002
003
004          /      THIS SUBR WILL CONVERT A BINARY BYTE INTO
005          /      2 ASCII HEX BYTES
006
007          /      ** ENTRY
008
009          /      A=BINARY BYTE TO BE CONVERTED
010
011          /      CALL BN2HX
012
013          /      ** EXIT
014
015          /      A = ?
016          /      B/C = 2 ASCII HEX BYTES
017
018          BN2HX,
019 2C69 F5          PUSH      PSW        / SAVE DIGIT
020 2C6A E6F0      ANI        :0F0      / MASK FOR M-S-NIBBLE
021 2C6C CF          NSWP           / GET M-S-NIBBLE
022 2C6D CD792C    CALL      CONVRT    / CONVERT TO ASCII
023 2C70 47          MOV        B,A        / STORE IN B
024
025 2C71 F1          POP        PSW        / GET L-S-NIBBLE
026 2C72 E60F      ANI        :0F      / MASK
027 2C74 CD792C    CALL      CONVRT    / CONVERT TO ASCII
028 2C77 4F          MOV        C,A        / STORE IN C
029 2C78 C9          RET
030
031          EJECT
032          SUBJOB \      CONVRT.- BINARY TO ASCII HEX
033
034
035          /      THIS RTN WILL CONVERT A VALUE FROM
036          /      BINARY TO HEX ASCII
037
038          /      ** ENTRY
039
040          /      A = VALUE (BINARY)
041          /      1 NIBBLE RIGHT JUSTIFIED
042
043          /      CALL CONVRT
044
045          /      ** EXIT
046
047          /      A = HEX ASCII DIGIT
048
049
050
051          CONVRT,
052 2C79 FE0A      CPI        :0A      / 09 ?
053 2C7B D2812C    JNC        CNVHEX   / CY = 0 --> HEX DIGIT

```

```

054
055 / CONVERT NUMBER TO HEX ASCII
056
057 207E C630 ADI :30 / DONE
058 2080 C9 RET
059
060 / CONVERT LETTER TO HEX ASCII
061
062 CNVHEX,
063 2081 C637 ADI :37 / DONE
064 2083 C9 RET
065
066 EJECT
067 SUBJOB \ H2BN4, H2BN2
068
069 / H2BN4 AND H2BN2 ARE SUBROUTINES TO CONVERT
070 / FOUR (TWO) HEX DIGITS TO ONE 16 BIT BINARY
071 / VALUE.
072 / THE RTN GETS A POINTER TO A 4 (2) BYTE
073 / STRING AND CONVERTS.
074 / 1ST CHAR TO MOST SIG 4 BITS OF RESULT
075 / 2ND CHAR TO NEXT 4 BITS OF RESULT
076 / 3RD CHAR TO NEXT 4 BITS OF RESULT
077 / 4TH CHAR TO LEAST SIG 4 BITS OF RESULT
078
079 / THERE IS NO CHECKING FOR BAD CHARS, EXCEPT
080 / SPACES ARE CONVERTED TO ZEROES.
081
082 / ** ENTRY
083 / D/E = ADDR OF STRING, 1ST CHAR
084
085 / CALL H2BN4 (H2BN2)
086
087 / ** EXIT
088 / H/L = 16 BIT VALUE
089
090 H2BN4,
091 2084 C5 PUSH B / SAVE REG
092 2085 D5 PUSH D / MOVE D/E TO B/C
093 2086 C1 POP B / DONE
094
095 / 1ST CHAR
096
097 2087 0A LDAX B / GET IT
098 2088 CDAA2C CALL CHOPIT / GET THE 4-BIT VERSION
099 208B CF NSWF / SHIFT IT TO HIGH 4 POS.
100 208C 67 MOV H,A / SET H=11110000
101 208D 03 INX B / TO 2CND
102
103 / 2ND CHAR
104
105 208E 0A LDAX B / GET IT
106 208F CDAA2C CALL CHOPIT / GET THE 4 -BIT VERSION
107 2092 B4 ORA H / CONCAT WITH A
108 2093 67 MOV H,A / SET H=11112222 (H DONE)
109 2094 03 INX B / TO 3RD
110 2095 C39B2C JMP H2BN2 / JMP AROUND 2ND ENTRY
111
112 / 3RD CHAR
113
114 EJECT
\ ENTER HERE FOR "H2BN2"
116
117 H2BN2,
118 2098 C5 PUSH B / SAVE REGS
119 2099 D5 PUSH D / MOVE TO B/C
120 209A C1 POP B / DONE
121
122 H2BNZ,
123 209B 0A LDAX B / GET IT
124 209C CDAA2C CALL CHOPIT / GET 4 BITS
125 209F CF NSWF / SHIFT TO HIGH 4 BIT POS
126 20A0 6F MOV L,A / SAVE IN LOW REG, L=33330000
127 20A1 03 INX B / TO 4TH
128
129 / 4TH CHAR
130

```



```

131 2CA2 0A          LDAX   B          / GET IT
132 2CA3 CDAA2C     CALL   CHOPIT    / GET THE 4 BIT VERSION
133 2CA6 B5         ORA    L          / CONCAT WITH A
134 2CA7 6F         MOV    L,A       / L=33334444 (DONE)
135
136 2CA8 C1         POP    B         / REST REG
137 2CA9 C9         RET
138
139
140
141 /
142 /
143 /
144 /
145 CHOPIT,
146 2CAA FE20       CPI    :20       / IF SPACE, SUBS 0
147 2CAC D2B12C     JNZ   CHOPGO    / NO , NOT 0
148 2CAF 3E30       MVI   A,:30     / YES, FORCE 0
149 CHOPGO,
150 2CB1 D630       SUI   :30       / MAKE CHAR 0-REL
151 2CB3 FE04       CPI   @10      / IS THIS AN A-F CHAR?
152 2CB5 D8         RC      /NO, 0-9 ALL SET
153 2CB6 DA07       SUI   @7        / YES, SUB HOLE IN ASCII SET
154 2CB8 E60F       ANI   :0F      / CLEAR TOP 4 BITS
155 2CBA C9         RET
156
157 EJECT

001 SUBJOB \      EOF - SEND EOF TO TAPE
002
003 /
004 /
005 /
006 /
007 /
008 /
009 /
010 /
011 /
012 /
013 /
014 /
015 /
016 /
017 /
018 /
019 /
020 EOF,
021
022 2CBB C5         PUSH  B         / SAVE REGS
023 2CBC D5         PUSH  D
024 2CBD F5         PUSH  H
025
026 2CBE 21C4FE     LXI   H,CASBUF / PTR TO CASSETTE BUFF
027 2CC1 110A0D     LXI   D,CRLF   / GET 'CRLF'
028 2CC4 EF         MOVDE          / STORE CRLF
029 2CC5 3E3A       MVI   A,ASDDU  / START OF REC
030 2CC7 77         MOV   M,A      / STORE IT
031 2CC8 23         INX   H        / BUMP PTR
032
033 /
034 /
035 2CC9 EB         XCHG          / SAVE ADDR IN D/E
036 2CCA 213030     LXI   H,ASL22 / PTR TO '00'
037 2CCD EB         XCHG          / SWITHC
038 2CCE EF         MOVDE          / LENGTH
039 2CCF EF         MOVDE          / ADDR HI
040 2CD0 EF         MOVDE          / ADDR LO
041
042 /
043 /
044 2CD1 113130     LXI   D,ASDD1 / '01'
045 2CD4 EF         MOVDE          / STORE
046
047 EJECT

```

```

048 / FINALLY STORE THE END RECORD CHKSM
049
050 2CD5 114646 LXI D,ENDCHK/ ADDR OF CHECKSUM
051 2CD8 EF MOVDE / STORE
052
053 2CD9 CDF2B CALL P2T10 / TRANSMIT THE RECORD
054
055 2CDD E1 POP H / RESTORE REGS
056 2CDD D1 POP D
057 2CDE C1 POP B
058 2CDF C9 RET
059
060 EJECT

001 SUBJOB \ CHEX80 - COMPUT INTEL CHKSM
002
003 / THIS RTN WILL COMPUTE & STORE A
004 / CHECKSUM FOR AN INTEL RECORD.
005
006 / ** ENTRY
007
008 / H/L --> 1ST BYTE TO CHECKSUM
009 / A = # OF BYTES
010
011 / CALL CHEX80
012
013 / **EXIT
014
015 / CHECKSUM STORED AFTER LAST BYTE
016 / OF RECORD
017 / A = ?
018
019 CHEX80,
020 2CE0 C5 PUSH B / SAVE REGS
021 2CE1 D5 PUSH D
022 2CE2 E5 PUSH H
023
024 2CE3 47 MOV B,A / COUNTER IN B
025 2CE4 AF CLA / INIT CHKSUM
026 2CE5 F5 PUSH PSW / SAVE CHKSUM
027 2CE6 C5 PUSH B / SAVE COUNTS
028 2CE7 EB XCHG / ADDRESS TO D/E
029 CHEX8A,
030 2CE8 D5 PUSH D / SAVE ADDR
031 2CF9 CD982C CALL H2BN2 / CONVERT TO BIN
032 2CEC D1 POP D / GET BACK ADDR
033 2CED 13 INX D / BUMP TO NEXT CHAR
034 2CEF 13 INX D / DONE
035 2CFE C1 POP B / GET COUNTER
036 2CF0 F1 POP PSW / GET CHKSUM
037 2CF1 85 ADD L / UPDATE CHKSUM
038 2CF2 F5 PUSH PSW / SAVE CHKSUM AGAIN
039 2CF3 C5 POP B / SEE IF DONE
040 2CF4 C5 PUSH B / SAVE
041 2CF5 CD982C JNZ CHEX80 / NOT 0--> NOT DONE
042
043
044 EJECT
045 / ** CHECKSUM IS IN A REG
046 / STORE IN RECORD AFTER 2'S CMP
047
048 2CF8 2F CMA / 1'S COMPL
049 2CF9 3C INR A / 2'S COMPL
050 2CFA CD692C CALL BN2HX / CONVERT TO HEX ASCII
051 2CFD FB XCHG / ADDR TO H/L
052 2CFE D7 MOVBC / STORE CHKSUM
053
054 2CFF C1 POP B / CLEAN STACK
055 2D00 F1 POP PSW
056 2D01 E1 POP H
057 2D02 D1 POP D
058 2D03 C1 POP B
059 2D04 C9 RET
060
061 EJECT

```

```

062 SUBJOB \ TOPIO - INDIRECT CALL
063
064 / THIS SUBR WILL HOPEFULLY SOLVE THE
065 / COMMUNICATIONS PROBLEMS BETWEEN PORT 1 OF THE
066 / P180 AND THE 484. IT APPEARS THAT THE P180
067 / AND THE 484 CAN GET OUT OF SYNC. THE 484
068 / DETECTS AN ERROR AND INTERPRETS THE NEXT '02' AS
069 / A 'STX' WHICH CAN LEAD TO ALL SORTS OF PROBLEMS.
070 / THIS RTN WILL HOPEFULLY GIVE THE UNITS ENOUGH TIME
071 / TO GET BACK INTO SYNC AND RETRY THE COMM. A
072 / FEW TIMES, BEFORE CALLING IT A HARD ERROR.
073
074
075 / ** ENTRY
076 / D/E SET UP FOR CALL TO "PIO"
077 / "UMDFUF" SET UP FOR CALL TO "PIO"
078
079 / CALL TOPIO
080
081 / ** EXIT
082 / HARD ERROR --> ERROR CALL
083 / GO TO "EXEC"
084 / SUCCESS --> NORMAL RET
085
086
087 TOPIO,
088 2D05 E5 PUSH H / SAVE REGS
089 2D06 E5 PUSH B
090 2D07 D5 PUSH D / CERTAINLY SAVE COMMAND!
091
092 2D08 0A04 MVI B,4 / # OF RETRIES
093 2D0A C5 PUSH B / SAVE RETRY COUNTER
094
095 TOPIO2,
096 2D0B C78125 CALL PIO / ISSUE COMMAND
097 2D0E CA1F3D JZ TOPIOX / 0 --> NO ERROR
098 2D11 CD5B2C CALL DELHLF / DELAY APPROX .5 SEC
099 2D14 C1 POP B / GET RETRY COUNTER
100 2D15 D1 POP D / GET COMMAND
101 2D16 D5 PUSH D / SAVE COMMAND
102 2D17 05 DCR B / DONE ?
103 2D18 C5 PUSH B / SAVE JUST IN CASE
104 2D19 C20B2D JNZ TOPIO2 / 0 --> TIMED OUT
105
106 EJECT
107
108 \ ** COME HERE ON HARD ERROR
109 2D1C C8EF00 JMP EXEC / QUIT!
110
111 \ ** SUCCESSFUL
112
113 TOPIOX,
114 2D1F CD9F1B CALL CLRERR / CLEAR ERROR CONDITION
115 2D22 C1 POP B / CLEAN STACK
116 2D23 D1 POP D
117 2D24 C1 POP B
118 2D25 E1 POP H
119 2D26 C9 RET
120
121 EJECT

001 SUBJOB \ MESSAGES FOR L-D-V
002
003 MSGBDC,
004 2D27 0A DB MSGBCX / CAN'T DETERMINE CONFIG
005 2D28 42414420 DA 'BAD CONFIG'
006 2D2C 434F4E46
007 2D30 4947
008 MSGBCX= .-MSGBDC-1
009
010 MSGND2,
009 2D32 0C DB MSGNX2 / NOTHING CONNECTED TO PORT 2
010 2D33 504F5254 DA 'PORT 2 EMPTY'
011 2D37 20322045
012 2D3B 4D505459
013 MSGNX2= .-MSGND2-1

```


012				
013			MSGBDR,	
014	2D3F	0A	DB	MSGBRX / ILLEGAL RECORD TYPE
015	2D40	42414420	DA	'BAD RECORD'
	2D44	5245434F		
	2D48	5244		
016		000A	MSGBRX=	.-MSGBDR-1
017				
018			MSGLDG,	
019	2D4A	07	DB	MSGLDX / DISPLAY @ START OF LOAD
020	2D4B	4C4F4144	DA	'LOADING'
	2D4F	494E47		
021		0007	MSGLDX=	.-MSGLDG-1
022				
023			MSGLDD,	
024	2D52	09	DB	MSGLDZ / @ END OF LOAD
025	2D53	4C4F4144	DA	'LOAD O.K.'
	2D57	204F2E4B		
	2D5B	2E		
026		0009	MSGLDZ=	.-MSGLDD-1
027				
028			MSGDPG,	
029	2D5C	07	DB	MSGDPX / @ START OF DUMP
030	2D5D	41554D50	DA	'DUMPING'
	2D61	494E47		
031		0007	MSGDPX=	.-MSGDPG-1
032				
033			MSGDOK,	
034	2D64	09	DB	MSGDOX / @ END OF DUMP
035	2D65	44554D50	DA	'DUMP O.K.'
	2D69	204F2E4B		
	2D6D	2E		
036		0009	MSGDOX=	.-MSGDOK-1
037				
038			EJECT	
039			MSGVFG,	
040	2D6E	09	DB	MSGVFX / @ START OF VERIFY
041	2D6F	56455249	DA	'VERIFYING'
	2D73	4659494E		
	2D77	47		
042		0009	MSGVFX=	.-MSGVFG-1
043				
044			MSGRNM,	
045	2D7B	0C	DB	MSGRNX / REG-RAM MIS-MATCH
046	2D79	52454720	DA	'REG NO-MATCH'
	2D7D	4E4F2D4D		
	2D81	41544348		
047		000C	MSGRNX=	.-MSGRNM-1
048				
049			MSGLNM,	
050	2D85	0C	DB	MSGLNX / LOGIC RAM MIS-MATCH
051	2D86	4C434720	DA	'LCG NO-MATCH'
	2D8A	4E4F2D4D		
	2D8E	41544348		
052		000C	MSGLNX=	.-MSGLNM-1
053				
054			MSGCNM,	
055	2D92	0C	DB	MSGCNX / COIL RAM MIS-MATCH
056	2D93	434F4920	DA	'COI NO-MATCH'
	2D97	4E4F2D4D		
	2D9B	41544348		
057		000C	MSGCNX=	.-MSGCNM-1
058				
059			MSGVOK,	
060	2D9F	09	DB	MSGVKX / VERIFY OK
061	2DA0	56455249	DA	'VERIFY OK'
	2DA4	4659204F		
	2DA8	4B		
062		0009	MSGVKX=	.-MSGVOK-1
063				
064			EJECT	
065			SUBJOB \	CROSS REFERENCE
066				
067				
068				

069 / END OF FILE: LDV 180 MOD 01

070

071 2DA9 EOLDV=.

072

073

* FOLLOWS BUT DOES NOT PRINT!!!

ACKTMR	0030	#024-031	345-001				
ADDFLG	0000	#031-045	174-031	259-035			
ADRCOON	60BE	#031-021	119-005				
ADREON	FESE	#031-013	035-005	121-009	160-021	163-010	179-041
			179-046	182-012	215-090	226-011	227-038
			305-013	306-004	336-077	341-017	341-021
ADRSON	FE8C	#031-011	031-013	121-007	160-017	163-008	212-018
			213-019	215-089	227-043		
ADRSYS	60BD	#031-019	275-020				
ADRUSE	0002	#031-023	119-037	182-009	213-046	373-028	
ADVFLD	000B	#014-013	091-025	291-014			
ASCADN	001B	#038-032	177-005	253-034			
ASCAST	002A	#038-047	174-036				
ASCBAR	0074	#040-121	322-008				
ASCBLK	0020	#038-037	092-013	092-015	092-015	092-021	092-021
			132-071	140-101	143-034	149-011	150-019
			154-047	177-005	177-005	197-027	217-030
			224-051	245-090	245-096	246-028	248-012
			248-048	264-016	266-022	290-012	295-023
			305-010	307-011	310-008	329-042	330-058
			331-106	333-117	333-129	333-141	331-069
ASCC	0043	#039-072	258-020	258-032			
ASCCBK	001F	#038-036	092-013	092-014	092-014	092-014	092-014
			183-036	185-124	193-024	232-014	240-030
ASCCOL	003A	#039-063	371-011	391-036	408-029		244-059
ASCD	0044	#039-073	132-050	132-067	154-042	266-024	
ASCDIS	0066	#040-107	156-007				
ASCDIV	000A	#038-015	259-044				
ASCDOT	002E	#039-051	258-023	258-026	258-029		
ASCDSH	0072	#040-119	142-151	166-045	218-058	218-077	224-058
			234-060	318-038	329-036		
ASCLB	0005	#038-010	172-027	174-006	174-014	177-005	238-016
			252-064	253-003	253-017	253-031	258-009
			258-032	320-016			258-020
ASCLRE	0060	#040-101	284-005	325-021			
ASCMIN	002D	#039-050	174-033	259-038			
ASCOMPX	000B	#038-016	259-041				
ASCN	004E	#039-083	258-032				
ASCNAK	00D0	#041-170	107-028	347-028			
ASCNBK	001D	#038-034	092-019	092-020	092-020	092-020	092-020
			184-052	186-131	193-028	232-011	242-004
							245-075
ASCO	004F	#039-084	258-032				
ASCPLS	002B	#038-048	174-030	259-035			
ASCR	0052	#039-087	258-020				
ASCRB	0009	#038-014	174-009	177-005	253-011	253-037	258-020
			258-023	258-026	258-029	258-032	
ASCS	0053	#039-088	153-013				
ASCSLH	002F	#039-052	174-039				
ASCSTX	0002	#041-169	106-013	107-023	344-005		
ASCT	0054	#039-089	258-020				
ASCTL	0002	#038-007	258-005				
ASCTMR	0007	#038-012	258-023	258-026	258-029		
ASCTR	0004	#038-009	258-005				
ASCUB	0003	#038-008	258-005	258-005	258-005		
ASCVBK	001E	#038-035	092-014	092-020	232-017	246-004	
ASCZZ	3030	#360-015	408-036				
ASCO	0030	#039-053	074-002	075-019	075-032	075-045	076-058
			118-017	150-021	152-004	167-017	173-063
			242-027	242-034	245-102	245-109	251-016
			258-023	258-026	258-029	266-013	295-013
			298-059	299-008			297-021
ASCOUN	0010	#038-021	173-063	173-070	245-109	258-009	258-009
			258-009				
ASC01	3031	#360-018	371-041	408-044			
ASC1	0031	#039-054	131-027	152-005	154-033	167-024	168-052
			175-018	236-013	258-023	258-026	258-029
							295-015
ASC2	0032	#039-055	152-006	158-035	167-043	295-017	
ASC3	0033	#039-056	152-007	171-046	187-153	264-045	295-019
ASC3UN	0013	#038-024	173-052				

4,292,666

533

534

ASC4	0034	#039-057	152-008	171-038	175-022	191-006	255-007
		264-043	295-021				
ASC4UN	0014	#038-025	172-044	173-076	258-009		
ASC5	0035	#039-058	152-009				
ASC6	0036	#039-059	152-010				
ASC7	0037	#039-060	152-011				
ASC8	0038	#039-061	152-012				
ASC9	0039	#039-062	152-013				
ASMCOL	0006	#014-021	121-014	264-023	267-037		
ASMCON	FE7F	#030-034	030-036	143-048	183-043	196-026	229-028
		232-039	233-024	233-033	233-044	235-015	235-036
		238-026	240-004	241-054	241-062	241-069	243-004
		244-046					
ASMNUM	0006	#030-032	264-007				
ASMROW	0080	#030-026	112-006	131-019	135-040	136-064	136-078
		137-099	137-111	154-026	158-025	197-019	228-020
		264-011	285-006				
BADREC	2BF6	396-037	#397-081				
BCDBN1	01A2 *	#073-042					
BCDBN2	0198	#073-037	299-020				
BCDBN3	018E	#073-032	131-037	155-061	156-014	236-018	255-013
		264-036	297-007	300-009	301-005		
BCDBN4	0181	#073-026	255-029	302-006			
BCDSUB	01A9	073-028	073-033	073-038	073-043	#074-001	
BCDSX	01C1	074-008	#074-016				
BCDS10	01B5	#074-008	074-011				
BCDS20	01BD	074-003	074-005	#074-013			
BCDX	01A8	073-029	073-034	073-039	#073-045		
BEEP	010F	059-005	059-008	#065-024			
BEEP10	0114	#065-027	123-038				
BEEP20	0116	065-025	#065-029				
BFBASE	0000	#025-008	025-009				
BFBLKL	0006	#025-013	025-019	025-021	025-023	026-003	
BFCH	012E	#070-001	079-018	108-075	123-034	344-027	
BFCHX	0155	070-012	#070-035				
BFCH10	0140	070-007	#070-014				
BFINIT	0124	#068-001	077-010	103-004	103-009	107-042	109-103
		118-004	126-020				
BFIPTR	0002	#025-009	025-010				
BFLEN	0004	#025-011	025-012	070-002	072-024		
BFOPTR	0003	#025-010	025-011				
BFUSE	0005	#025-012	025-013	063-007	063-010	072-002	079-004
		110-007					
BNBCD1	01FB *	#076-058					
BNBCD2	01E8	#075-045	168-061				
BNBCD3	01D5	#075-032	167-040	171-050	173-057	175-026	
BNBCD4	01C2	#075-019	120-066	170-022	194-056	214-060	253-021
		267-029	303-025	324-023			
BN010	01C7	#075-022	075-025				
BN020	01CF	075-023	#075-027				
BN030	01DA	#075-035	075-038				
BN040	01E2	075-036	#075-040				
BN050	01ED	#075-048	076-051				
BN060	01F5	076-049	#076-053				
BN2HX	2C69	371-028	371-032	371-036	372-052	#404-018	411-050
BUFFER	FF70 *	#025-039					
BURST	000B	#005-044	086-027				
CASBUF	FEC4	#036-034	371-008	372-065	391-027	392-053	393-111
		396-007	396-022	396-033	397-042	399-006	399-013
		408-026					
CATHI	0001	#041-165	142-150	156-006	166-044	218-057	218-076
		224-057	224-073	284-013	284-016	322-008	322-010
		322-013	322-017	322-019	322-024	326-024	326-032
		326-035	326-040	329-035	331-090		
CA0011	00E4	#041-154	149-008	166-055	224-074	229-035	231-097
		322-013	322-019	326-022	326-024		
CA0101	00C0	#041-145	284-009	325-025			
CA0111	00D8	#041-151	322-017				
CA1010	00CC *	#041-148					
CA1100	00E0	#041-153	149-011	172-023	174-011	177-005	238-045
		248-043	253-013	253-039	319-058	320-012	326-030
		326-032	331-091	333-143			
CA1101	00D0	#041-149	149-008	165-032	322-010	333-131	
CA1110	00DC	#041-152	166-047	217-025	217-036	224-078	322-024
		326-036	330-052	331-097	333-119		

4,292,666

535

536

CA1111	00E8	#041-155	186-052	217-023	326-041	331-094	331-108
CHEX8A	2CE8	#410-029	410-041				
CHEX80	2CE0	372-058	393-120	#410-019			
CHDFGO	2CB1	407-147	#407-149				
CHDFIT	2CAA	406-098	406-106	407-124	407-132	#407-145	
CKDASH	09SD	138-018	139-077	#142-146			
CLKINI	039E	058-006	#095-001				
CLKINT	03AB	083-032	#097-001				
CLKI10	03A4	#095-005	095-008				
CLK010	03AF	#097-004	097-030				
CLK020	03C7	#097-022	099-008	099-012	099-024	099-038	
CLK030	03C8	097-008	097-010	#097-024			
CLK100	03DE	098-010	#099-004				
CLK200	03EE	098-011	#099-012				
CLK300	03EE	098-012	#099-016				
CLK400	03F4	098-013	#099-021				
CLK410	03F7	099-017	#099-023	099-043			
CLK500	03FD	098-014	#099-028				
CLK510	0408	099-031	#099-034				
CLK520	0409	* #099-036					
CLK600	0411	098-015	#099-042				
CLRERR	1B9F	119-011	263-006	#263-021	413-114		
CMDBFL	0018	#035-007	035-009				
CMDBUF	FE90	#035-005	035-009	119-004	119-036	131-041	132-054
		133-033	155-052	155-065	156-019	156-041	161-035
		163-022	182-018	183-039	183-041	183-046	184-060
		192-023	195-083	203-021	207-035	207-053	209-085
		210-032	212-021	213-023	226-017	233-039	235-034
		235-040	235-042	236-046	237-049	237-058	237-061
		240-017	241-072	241-075	242-042	242-044	243-024
		246-041	248-024	250-056	250-068	250-078	251-024
		251-028	251-046	255-017	255-032	265-050	267-015
		275-019	278-015	286-042	305-018	305-022	306-009
		307-022	308-010	308-024	310-012	311-005	312-052
		313-013	314-007	314-014	344-004	344-008	344-019
		365-036	380-038	380-042	381-067		
CMDCUR	0080	#005-010	086-013				
CMDDEC	00C0	#042-016	203-035	206-012	248-027		
CMDDEL	0060	#042-010	205-033	209-095	211-077	212-026	221-045
		313-016					
CMDGO	0090	#042-013	273-028				
CMDINC	00B0	#042-015	246-046	307-037	308-039		
CMDINI	00A0	#042-014	274-051	380-015			
CMDINS	0050	#042-009	226-022	305-033	306-027	310-015	311-025
CMDLED	0070	#042-011	286-044				
CMDNAK	00D0	* #042-017					
CMDPRE	00E0	#005-013	086-024				
CMDPWR	0040	#042-008	278-018				
CMDREN	0010	#042-005	119-008	156-021	161-038	163-026	213-025
		266-001	267-018	275-023	365-027	371-022	384-039
CMDRST	0000	#005-006	086-001				
CMDSCH	0030	#042-007	119-044	185-084			
CMDST	0020	#005-007	086-027				
CMDSTP	0080	#042-012	273-007	380-010			
CMDWRT	0020	#042-006	132-060	133-047	155-067	156-045	209-070
		244-039	250-071	255-038	312-061	381-070	
CMD02	0001	#042-023	119-008	132-060	133-047	155-067	156-021
		156-045	161-038	163-026	209-070	212-026	213-025
		226-022	244-039	246-046	247-057	248-027	248-038
		250-071	255-038	266-001	267-018	275-023	312-061
CNVHEX	2081	405-053	#405-062				
COIRAM	29AF	377-090	#378-105				
COLBKL	0006	#028-019	028-024	028-025	028-026	028-027	028-028
		028-029	028-030	028-031	028-032	028-033	028-035
		180-054	180-055	278-039	278-040	328-014	328-015
		339-020					
COLDEC	2500	219-104	248-039	313-028	#337-015		
COLD10	2524	#338-066	338-070				
COLD20	2530	338-059	#338-075				
COLD99	2538	338-068	#338-085				
COLEHI	0002	#028-014	028-015	207-046	211-048	248-020	250-064
		307-016	311-046	313-007	335-046	337-044	
COLELO	0003	#028-015	028-016				
COLINC	24B8	247-058	305-038	306-039	307-042	309-053	310-020
		311-032	#334-015				
COLI10	24DA	#335-060	335-068				

4,292,666

537

538

COLI20	24E6	334-037	#336-072				
COLI99	24F4	335-063	#336-095				
COLMSK	000F	#030-028	112-020	135-035	136-067	137-088	138-009
		138-037	139-052	165-012	180-053	180-081	201-025
		202-013	224-088	230-063	231-089	234-053	241-048
		286-040	318-030	323-040	325-013	328-013	
COLSHT	0000	#028-012	028-013				
COLSLD	0001	#028-013	028-014				
COLTAB	FDED	#028-021	028-023	028-037	180-055	278-039	293-016
		328-014					
COLTBA	FE23	#028-032	028-033				
COLTBB	FE29	#028-033	028-035				
COLTBL	0042	#028-037	293-017				
COLTRX	FE2F	#028-035	028-037	029-006	339-022		
COLTB1	FDED	#028-023	028-024				
COLTB2	FDF3	#028-024	028-025				
COLTB3	FDF9	#028-025	028-026				
COLTB4	FDFE	#028-026	028-027				
COLTB5	FE05	#028-027	028-028				
COLTB6	FE0B	#028-028	028-029				
COLTB7	FE11	#028-029	028-030				
COLTB8	FE17	#028-030	028-031				
COLTB9	FE1D	#028-031	028-032				
COMPB1	0043	#005-017	086-004				
COMPB2	0014	#005-019	086-006				
COMPB3	007B	#005-021	086-008				
COMPB4	0036	#005-023	086-010				
COMPHI	1164	188-026	189-005	190-034	#195-079		
CONCOL	295A	373-033	#375-003				
CONUSE	2933	373-031	#374-049				
CONVRT	2C79	404-022	404-027	#405-051			
CON4A	298E	373-044	374-055	374-058	374-061	374-064	374-067
		#376-040					
CON4A0	2987	375-009	375-013	375-017	375-021	#375-030	
CON4ER	297E	374-071	#375-023				
CON484	2915	362-027	#373-024	378-116	380-028	384-015	
CRLF	0D0A	#360-025	371-006	408-027			
CRTCTL	0038	#002-017	054-013	086-002	086-014	086-025	086-028
CRTDAT	0039	#002-018	086-005	086-007	086-009	086-011	086-016
		086-018					
CRTINI	026B	058-007	#086-001				
CRTRFH	F804	#012-012	012-017	012-019	013-038	013-039	014-004
		015-006	089-004				
CRTRFX	FD4E	#012-025	023-007				
CRTSIR	0020	#005-030	083-024	086-033	086-037		
CRTSTA	0038	#002-016	083-023	086-032	086-036		
CRTSVE	0004 *	#005-033					
CRTTMP	0544	#012-015	012-021	012-023	015-006		
CRT010	0296	#086-032	086-034				
CRT020	029D	#086-036	086-038				
CRT50E	FD4E	#012-021	012-025	013-038			
CRT50S	F804	#012-017	013-047	013-048			
CRT60E	FD4C	#012-023	013-039				
CRT60S	F804	#012-019	012-021	012-023	013-050	013-051	
CSFMT2	28FB	#372-050	372-058				
CSFRMT	28C9	363-004	#371-001				
CURACT	FE7E	#030-020	030-034	137-107	138-008	138-016	139-072
		140-096	140-109	140-112	142-138	158-023	165-011
		178-012	180-052	180-079	180-087	181-100	197-016
		199-052	200-081	201-024	202-011	210-012	212-005
		212-012	217-008	219-085	223-030	228-017	229-042
		229-049	232-003	232-020	233-048	286-039	293-043
		305-004	316-005	318-025	323-014	328-012	
CURCON	FE80	#030-036	030-038	137-115	140-086	141-127	178-018
		198-012	222-067	242-015	242-039	243-008	244-049
		314-011	321-006				
CURDSP	FE7D	#030-018	030-020	137-108	139-067	139-071	142-136
		178-008	178-013	178-026	178-031	180-088	180-092
		223-027	223-038	231-080	234-074	238-036	238-038
		239-054	239-056	241-047	252-073	252-075	253-025
		253-027	253-048	254-050	293-039	317-008	319-050
		325-012					
CURHOZ	007F	#005-040	086-015				
CURSOR	052B	#111-028	137-105	139-069	178-033	180-090	223-040
		234-056	293-045	318-033			
CURVER	0000	#005-039	086-017				

4,292,666

539

540

CUR010	0588	111-030	#111-037				
CUR100	0517	111-032	111-040	#112-004	140-098	142-147	165-015
		197-024	217-010	223-047	234-057	255-004	264-013
		317-010	318-035	320-006			
CUR110	0561	#112-020	112-031				
CUR120	0568	112-007	#112-025				
CUR200	0578	112-014	112-021	#112-033	112-036		
DCMP	0057	#052-012	120-054	185-108	213-047	334-036	338-058
		339-023	339-034	368-031	378-108	380-046	382-099
		393-129					
DELDEF	140E	223-032	#225-103				
DELDSH	1478	223-071	#223-013				
DEL D10	148E	#223-045	224-090				
DEL D20	1487	#224-055	224-052				
DEL D25	1490	224-059	#224-061				
DEL D30	14E2	224-077	#224-079				
DELHE2	205E	#403-075	403-081				
DELHLE	205E	273-013	273-034	362-018	399-024	400-051	#403-071
		412-092					
DELTEX	1474	222-057	#222-075				
DEL T10	144F	203-036	205-035	206-013	209-096	#221-024	
DEL T10	1450	221-043	#222-051				
DTNFLG	0002	#032-070	173-073	175-006	237-060		
DTSMSE	57FF	#032-061	132-057				
DTSPTR	FE81	#030-038	030-040	121-015	264-010	264-019	264-025
		237-031					
DTSTMR	0001	#014-033	121-012	267-039			
DTVFLG	0003	#001-018	1-4-029	259-044			
DMARLY	0020	#012-032	090-036	090-028	091-038	092-019	
DMACMD	008E	#006-017	056-040				
DMAEOR	00E1	#012-031	090-004				
DMAFAN	0090	#012-033	090-025	091-013	091-035	092-013	092-019
		111-033	111-035	269-037	269-039	276-002	276-004
		276-008	276-010	276-014	276-016	276-020	276-022
		276-024	276-028	276-030	276-034	276-036	276-040
		277-042					
DMATNI	02B0	086-020	#088-001				
DMAMA1	0020	#006-008	006-017				
DMAME2	0004	#006-013	006-017				
DMAME3	000E	#006-012	006-017				
DMAMOD	0038	#002-014	086-041	088-002			
DMAREB	0030	#006-006	088-009	088-018			
DMAST1	036E	091-015	#092-004	092-006			
DMAST2	0380	091-029	#092-003	092-010			
DMAST3	0388	091-009	#092-012	092-016	196-016		
DMAST4	0393	091-021	#092-018	092-022	196-019		
DMAS1X	0011	092-004	#092-006				
DMAS2X	0007	092-008	#092-010				
DMAS3X	000A	092-012	#092-016				
DMAS4X	000A	092-018	#092-022				
DMA0AD	0020	* #002-005					
DMA010	02DE	#089-009	089-011				
DMA020	02F6	#089-020	089-022				
DMA2AD	0024	#002-009	088-004	088-006			
DMA2TC	0025	#002-010	088-008	088-010			
DMA3AD	0026	#002-011	088-013	088-015			
DMA3TC	0027	#002-012	088-017	088-019			
DM50AH	00F8	#013-047	013-048				
DM50AL	0004	* #013-048					
DM50TC	0549	#013-038	013-041	013-042			
DM50TH	0005	#013-041	013-042				
DM50TL	0049	* #013-042					
DM60AH	00F8	#013-050	013-051	088-005	088-014		
DM60AL	0006	#013-051	088-003	088-012			
DM60TC	0547	#013-039	013-044	013-045			
DM60TH	0005	#013-044	013-045	088-009	088-018		
DM60TL	0047	#013-045	088-007	088-016			
DRGFLG	0003	#032-071	173-072	175-005	175-019	236-048	
DSPADV	FD0A	#015-020	015-022	291-013	291-017		
DSPASM	FCB1	#015-006	015-008	015-016	196-015	269-041	
DSPBSY	FC6C	#014-017	344-038	344-045			
DSPCON	FCB3	#015-008	015-010	143-043	183-035	185-123	193-023
		232-013	234-078	240-029	244-058	244-067	251-032
DSPERR	FCBB	#015-012	015-014	099-028	113-017	113-021	263-029
DSPLOG	F808	#014-004	014-017	016-004	112-008	293-032	322-007

4,292,666

541

542

DSPNOD	0007	#014-006	090-023	112-017	112-028	134-069	143-044
		165-016	172-021	174-012	217-017	217-045	218-051
		218-063	218-070	224-050	225-094	225-096	225-098
		234-059	234-079	234-081	234-084	238-010	238-014
		238-043	238-046	239-049	244-068	248-041	250-083
		252-057	252-062	253-015	253-040	282-015	282-024
		318-037	319-055	320-009	320-014	326-019	
DSPNUM	FCFE	#015-016	015-018	143-035	150-018	150-033	151-056
		158-033	158-045	184-051	186-130	191-005	193-027
		196-018	232-010	234-083	235-025	236-012	236-016
		238-018	242-003	242-026	242-033	245-074	245-088
		251-015	252-066	255-020	295-012	297-006	297-020
		298-058	299-007	299-019	300-008	301-004	302-004
		302-011					
DSPPOW	0003	#014-008	016-004	112-008			
DSPREF	FCDA	#015-014	112-025				
DSPSHT	FD02	#015-018	015-020	153-021	290-013	290-016	
DSPSTP	FD18	#015-022	015-024	118-019	194-055	214-057	303-024
DSPST0	1D7F	#276-002	276-006				
DSPST1	1D8A	#276-008	276-012				
DSPST2	1D95	#276-014	276-018				
DSPST3	1DA1	#276-020	276-026				
DSPST4	1DB3	#276-028	276-032				
DSPST5	1DBE	#276-034	276-038				
DSPST6	1DC9	#276-040	277-044				
DSPSUP	1D7F	270-049	#276-001				
DSPSOX	000A	276-002	#276-006				
DSPS1X	000A	276-008	#276-012				
DSPS2X	000B	276-014	#276-018				
DSPS3X	0011	276-020	#276-026				
DSPS4X	000A	276-028	#276-032				
DSPS5X	000A	276-034	#276-038				
DSPS6X	000C	276-040	#277-044				
DSPTAB	00DF	#049-012	097-020	163-039	271-020		
DSPUSE	FD1D	#015-024	015-026	120-065	324-022		
DSPVAL	FD28 *	#015-026					
DSPVER	FCB3	#015-010	015-012	149-014	149-016	229-034	231-096
		232-016	238-012	238-021	246-003	246-027	248-011
		250-085	250-089	252-060	305-009	307-010	310-007
DUMFLG	0002	#031-043	171-041	172-047	235-039	251-027	
DUMP	2810	272-015	#362-001				
DUMP10	282D	#362-026	363-022				
DUMP15	2830	#362-029	363-012				
DUMP2	281F	#362-017	362-020				
DUMP20	285B	362-032	#364-020	384-020			
DUMP25	2862	362-034	#365-021	384-023			
DUMP30	2849	362-033	#363-015				
DVR1	0058	#001-019	353-026				
DVR2	0032	#001-020	353-026				
DVR3	0033	#001-021	353-026				
ENDCHK	4646	#360-020	409-050				
EOCFLG	0080	#031-034	170-020	180-076	242-043	305-030	307-034
EOCHI	0004	#028-016	028-017	165-005	203-027	246-020	247-053
		248-033	306-034	307-004	308-046		
EOCLO	0005	#028-017	028-019	205-017	207-046	248-020	250-076
		278-046	311-010	312-068			
EOF	2CBB	363-026	#408-020				
EOLDV	2DA9 *	#417-071					
EOUSEG	FECO	#036-022	368-023	373-038	376-043	381-096	
ERRADI	0005	#043-059	349-015				
ERRADR	0004	#043-058	349-013				
ERRCHK	0003	#043-057	349-009				
ERRCMD	0006	#043-060	349-017				
ERRCDN	000E	#043-068	349-031				
ERRFLD	000D	#014-015	091-011	113-018	263-030		
ERRFUL	0011	#043-071	349-037				
ERRLEN	000D	#043-067	349-029				
ERRMEM	000B	#043-065	349-025				
ERRMSK	0008	#043-062	349-019				
ERRNOD	000A	#043-064	349-023				
ERRNPD	000F	#043-069	349-033				
ERRDR	057E	#113-016	121-021	132-078	134-078	143-024	157-006
		158-030	162-054	176-005	195-066	198-032	201-037
		216-005	256-003	261-008	262-006	271-016	274-066
		292-006	295-027	346-048	351-018	375-025	378-127
		382-137	386-043	387-042	389-048	394-145	397-083
		402-049					

ERRDVR	0002	#043-056	349-007				
ERRPAR	0001	#043-055	349-005				
ERRSED	0009	#043-063	349-021				
ERRSTP	0000	#043-066	349-027				
ERRSUP	0010	#043-070	349-035				
ERRTIM	0007	#043-061	349-011				
ERRTMR	001E	#024-025	099-036	113-023			
EXEC	00EF	058-020	#063-005	363-033	375-026	378-128	380-013
		380-018	382-130	382-138	385-083	386-044	387-043
		390-051	394-146	397-084	402-052	413-109	
EXEC10	00F2	#063-007	063-013				
FACNDR	0080	#040-136	099-029	153-018	284-007	290-015	325-023
FACREV	0090	#040-137	099-032	111-041	111-043	153-013	
FIELD1	0000	#360-035	386-031				
FIELD2	0020	#360-037	386-034				
FIXCDL	255A	335-067	338-069	#340-014			
FIXEDN	256B	336-096	338-086	#341-013			
FIXVER	2462	234-088	249-049	250-092	#329-023		
FIXV05	246E	329-037	#329-041				
FIXV10	2473	#329-047	331-070				
FIXV15	247B	330-050	#330-057				
FIXV20	247E	329-043	#331-065				
FIXV30	24A0	331-092	#333-114				
FIXV40	24A8	331-095	#333-126				
FIXV50	24B0	331-098	#333-138				
GETEND	28B5	368-030	#369-059				
GETHL	00E7	#050-009	070-028	072-019	120-052	133-023	160-025
		170-018	171-032	172-032	175-012	179-042	182-013
		185-092	185-096	185-105	193-014	194-040	203-010
		207-050	210-024	211-050	212-019	213-020	227-039
		240-009	246-036	248-022	250-066	267-028	278-013
		303-018	305-014	306-005	307-018	308-006	310-006
		311-048	313-009	324-016	334-035	335-049	336-078
		337-047	338-057	339-033	340-018	340-026	341-018
		368-024	373-039	380-043	382-097	393-128	
GETNET	0098	160-029	#161-009	162-050	194-036	215-096	
GETSIZ	2888	364-022	#368-001				
GETSXT	28BA	368-032	#369-068				
GETSX2	28C5	369-074	#369-081				
GETSZ2	2894	368-010	#368-015	369-054			
GETSZ5	289E *	#368-027					
GETYPE	1446	199-054	200-089	201-027	202-022	#220-014	
HLDFLG	0000	#031-041	171-039	172-045			
H2BNZ	209B	406-110	#407-122				
H2BNZ	2098	392-054	396-008	396-034	399-007	#407-117	410-031
H2BN4	2084	396-023	397-045	#406-090			
INPBAS	0000	#031-017	267-011				
INPDIS	0008	#032-054	032-061	131-048	266-020		
INPFLG	0001	#031-042	300-018				
INPSTA	0004	#032-053	154-048	155-055	266-015		
INTFLG	0002	#031-036	167-020	167-036			
INTREX	0265	083-017	083-025	#083-036			
INTRF	0245	053-008	#083-001				
INTR10	0257	083-013	#083-022				
INTSTA	0001	#032-051	156-034	156-035	266-007		
INTVCL	0004	#010-015	012-012				
INTVEC	F800	#010-011	012-012	053-007	058-013		
IOFLD	0020	#031-028	131-040	155-064	156-018	265-049	
ISCOIL	257A	178-020	229-029	233-045	241-055	241-063	318-022
		#342-016					
KBDBFL	0010	#025-030	025-037	118-002	126-018		
KBDBLK	FDAB	#025-023	026-003	063-010	118-003	123-033	126-001
		126-019					
KBDBUF	FF70	#025-037	025-039	118-001	126-017		
KBDCMD	06CD	063-012	#126-001				
KBDCMX	06E5	126-003	#126-017				
KBDIM3	065B	121-020	#121-027	121-029			
KBDIMX	0008	121-027	#121-029				
KBINI	059F	058-016	#118-001				
KBINT	0664	083-030	#123-001				
KBIX	065A	121-018	#121-023				
KBID10	0508	#118-021	118-024				
KBID15	050E	#119-004	119-010				
KBID20	05FA	#119-029	119-031				
KBID25	0613	#119-044	119-046				

KBDI50	0654	120-055	#121-020				
KBDTAB	06F2	126-004	#127-004				
KBDX	06CC	123-041	124-050	#124-060			
KBD010	066F	#123-007	124-056				
KBD020	067C	#123-015	124-049				
KBD021	068F	123-020	#123-026				
KBD025	06AE	123-024	123-028	123-037	#123-040		
KBD030	06B2	123-017	#123-043				
KBD040	06BE	123-011	#124-052				
KCLADV	0004	#030-014	118-014				
KCLEAR	0010	#030-012	118-014	150-016	150-029	196-021	227-036
		242-010	295-007				
KERROR	0040	#030-010	263-011	346-050			
KEYCLR	0018	* #129-027					
KEYCOL	0004	#129-007	146-047				
KEYDEL	0020	* #130-051					
KEYDIS	0024	* #130-055					
KEYDWN	0008	#129-011	135-011				
KEYENT	0000	* #129-003					
KEYERR	0002	* #129-005					
KEYFOR	002D	* #129-048					
KEYGET	0017	* #129-026					
KEYHT	000A	* #129-013					
KEYHZ0	0022	#129-037	143-029	146-059			
KEYHZS	0021	#129-036	143-031	146-065			
KEYLAT	0003	#129-006	146-053				
KEYLFT	0009	#129-012	135-013	138-030			
KEYNCR	0029	#129-044	145-029				
KEYNEG	0026	#129-041	145-041				
KEYNOR	0028	#129-043	145-023				
KEYNUA	0021	* #130-052					
KEYNUB	0022	* #130-053					
KEYNUC	0023	* #130-054					
KEYNU1	001C	* #129-031					
KEYNU2	001D	* #129-032					
KEYNU3	001E	* #129-033					
KEYNU4	0023	* #129-038					
KEYNU5	0024	* #129-039					
KEYNU6	0025	* #129-040					
KEYNU7	002A	* #129-045					
KEYNU8	002B	* #129-046					
KEYNU9	002C	* #129-047					
KEYNXT	0016	* #129-025					
KEYPOS	0027	#129-042	145-035				
KEYPRE	0015	#129-024	160-011				
KEYRGT	0007	* #129-010					
KEYSCH	000E	* #129-017					
KEYSP1	0027	* #130-058					
KEYSP2	0025	* #130-056					
KEYSP3	002E	* #130-049					
KEYSP4	0020	* #129-035					
KEYSP5	001F	* #129-034					
KEYSP6	0010	* #129-019					
KEYSP7	000F	* #129-018					
KEYSTR	0026	* #130-057					
KEYSUP	002F	#130-050	269-029				
KEYUP	0001	#129-004	135-009				
KEYVOP	0006	#129-009	149-009				
KEYVSH	0005	* #129-008					
KEY0	0011	#129-020	146-083	152-004	258-019	272-004	
KEY1	000B	#129-014	147-102	152-005	258-028	272-006	
KEY2	0012	#129-021	147-095	152-006	258-025	272-008	
KEY3	0019	#129-028	147-089	152-007	258-022	272-010	
KEY4	000D	#129-016	147-116	152-008	272-012		
KEY5	0014	#129-023	152-009	272-014			
KEY6	001B	#129-030	148-130	152-010	272-016		
KEY7	000C	#129-015	147-109	152-011			
KEY8	0013	#129-022	148-136	152-012	258-031		
KEY9	001A	#129-029	147-123	152-013			
KF01	0762	128-056	#131-010				
KF01A0	0839	134-059	#134-063				
KF01ER	07D1	131-030	#132-077				
KF01NC	0850	134-077	#134-081	134-083			
KF01NX	0008	134-081	#134-083				
KF01RR	0849	133-017	#134-076				
KF01X	0848	131-014	132-063	132-072	132-075	132-079	134-051
		#134-072					

KF0110	0771	131-012	#131-016				
KF0120	0788	131-028	#131-032				
KF0130	07AC	132-051	#132-054				
KF0140	07CD	132-069	#132-074				
KF0150	07DA	131-020	#133-004				
KF0175	07FA	133-008	133-011	133-014	#133-019		
KF0185	0803	#133-026	133-030				
KF0190	080C	133-027	#133-032				
KF0195	0831	#134-058	134-061				
KF02	0859	127-005	127-011	127-012	127-013	#135-006	
KF02CL	094E	138-011	#140-083				
KF02LP	0913	#138-023	139-079				
KF02X	0980	136-056	136-079	137-117	140-091	140-102	#141-129
KF0205	0875	135-024	#135-032				
KF0210	088C	135-041	#135-045				
KF0220	0891	135-010	#136-051				
KF0230	089F	135-012	#136-060				
KF0240	08BC	136-065	#136-078				
KF0250	08C4	135-014	#137-085				
KF0260	08D7	137-089	#137-095				
KF0270	08E6	135-043	135-046	136-076	137-093	#137-103	
KF0280	08E7	135-037	136-055	136-069	136-080	137-096	137-100
		#137-105					
KF0285	08FF	137-112	#138-003	139-078			
KF0290	0928	138-031	#139-049				
KF0292	0936	138-038	139-045	139-054	#139-066		
KF0295	0967	#140-108	140-117				
KF0296	0979	138-019	#141-123				
KF0297	097C	140-118	#141-125				
KF03	0997	127-007	127-008	127-037	127-038	127-042	127-043
		127-044	127-045	#143-005			
KF03MX	000B	144-055	#144-057				
KF03M1	09EA	143-023	#144-055	144-057	176-004		
KF03X	09E6	143-009	143-025	#144-050			
KF0310	09A6	143-007	#143-013	150-013			
KF0320	09AE	#143-017	143-021				
KF0330	09C0	143-018	#143-027				
KF0340	09CC	143-030	#143-034				
KF0350	09D3	#143-038	143-041				
KF0360	09D9	143-032	#143-043				
KF04	0AB3	127-009	127-010	#149-005			
KF04X	0AD5	149-017	#149-021				
KF0410	0AC7	149-010	#149-013				
KF0420	0AD2	149-007	#149-019				
KF05	0AD6	127-015	127-016	127-017	127-021	127-022	127-023
		127-024	127-029	127-030	127-031	#150-011	
KF05TB	0B21	150-046	#152-004				
KF0510	0AF2	#150-024	150-027				
KF0520	0B00	150-017	#150-032				
KF0530	0E06	#150-036	150-041				
KF0540	0B12	#150-048	151-052				
KF0550	0B1B	151-049	#151-054				
KF06	0E35	127-014	#153-006				
KF06X	0B50	153-014	#153-020				
KF0610	0B4A	153-010	#153-016				
KF07	0E56	128-049	#154-016				
KF07ER	0BFE	154-043	156-008	#157-004			
KF07MS	0C05	157-004	#157-012	157-014			
KF07MX	000C	157-012	#157-014				
KF07NX	0008	157-016	#157-018				
KF07N1	0C12	132-077	154-035	#157-016	157-018		
KF07RR	0C01	154-036	#157-005				
KF07X	0C04	154-020	155-069	156-024	156-047	#157-008	
KF0705	0B65	154-018	#154-022				
KF0707	0B7D	154-034	#154-038				
KF0710	0B92	155-049	#155-052				
KF0725	0BB2	154-027	#156-003				
KF0730	0BE4	156-028	#156-034				
KF0735	0BE9	156-032	#156-037				
KF08	0C1E	127-027	#158-017				
KF08ER	0C37	#158-030	158-036				
KF08MX	0C0B	159-057	#159-059				
KF08M1	0C66	158-028	#159-057	159-059			
KF08X	0C65	158-021	158-031	158-040	#159-053		
KF0805	0C2A	158-019	#158-023				
KF0810	0C3D	158-026	#158-033				

KF09	0C72	127-025	127-026	#160-005				
KF09ER	0CCA	161-025	#162-054					
KF09MS	0F56	174-043	#177-004	177-006				
KF09MX	0006	177-004	#177-006					
KF09TB	0D0A	163-037	#164-004					
KF0905	0C7F	160-007	#160-011					
KF0915	0C8D	160-012	#160-020					
KF0920	0C93	160-018	#160-025					
KF0927	0CA8	161-016	161-020	#161-029				
KF0930	0CDE	161-046	#163-004					
KF0935	0CE9	#163-019	166-066	168-064				
KF099A	0F7E	178-021	#178-030					
KF0999	0F5D	163-028	164-004	164-005	176-006	#178-003		
KF10	1001	127-018	#182-005					
KF10RB	10E3	187-154	#190-004					
KF10X	1157	184-057	185-086	194-043	#194-058	195-067		
KF1005	1010	182-007	#182-012					
KF1010	1017	182-010	#182-018					
KF1015	1041	183-037	#184-051					
KF1017	1064	184-072	#184-077					
KF1020	1067	184-053	#185-084	192-025				
KF1025	10C8	188-014	#189-004					
KF1030	10D5	189-007	#189-015					
KF1040	1107	190-018	#191-004					
KF1070	110F	188-019	188-021	188-027	189-010	189-017	189-021	
		190-023	190-028	190-030	190-035	#192-013		
KF1080	1119	185-125	186-132	188-028	189-009	189-022	190-036	
		191-007	#193-004					
KF1090	1134	193-025	193-029	#194-035				
KF1095	114B	194-046	#194-050					
KF11	1171	127-028	#196-005					
KF1110	117B	196-007	#196-015					
KF12	119D	128-052	#197-010					
KF12ER	13DA	198-006	210-006	#216-004				
KF12NL	1247	#201-035	202-024					
KF12NM	115B	185-109	185-117	#195-064				
KF12NV	11E9	198-014	#198-030	199-059	199-068	199-071	200-086	
		200-091						
KF1205	11B9	#197-029	197-036					
KF1210	11C8	197-020	#198-003					
KF1212	11F0	198-020	#198-038					
KF1214	11F5	198-023	#199-045					
KF1216	1217	198-026	#200-076					
KF1220	122F	198-040	199-061	199-070	#201-008			
KF1225	1257	201-015	201-029	#202-004				
KF1230	1273	202-007	202-015	#203-006				
KF1240	1296	203-031	#204-013					
KF1242	129C	#204-024	204-027					
KF1244	12A4	#204-036	204-048					
KF1245	12B0	204-038	#205-005					
KF1250	12CA	205-024	#206-004					
KF1260	12D6	205-029	#207-019					
KF1264	130B	209-073	#209-102					
KF1265	1310	197-012	#210-003					
KF1270	1320	#210-016	212-013					
KF1275	135A	210-028	#212-004					
KF1280	1369	212-008	#212-017					
KF1295	1378	211-083	#213-007					
KF1297	13BF	213-036	#215-074					
KF1298	13CB	213-048	214-068	#215-085				
KF13	14D3	128-058	#226-005					
KF13MS	1528	226-031	#227-054	227-056				
KF13MX	0005	227-054	#227-056					
KF13X	1527	226-009	226-025	#227-050				
KF1305	14E2	226-007	#226-011					
KF14	152E	127-004	#228-005					
KF14ER	1AFE	228-015	230-074	231-104	232-012	232-015	232-018	
		232-035	235-007	236-042	255-026	#256-003		
KF14IV	1AFB	233-026	233-028	236-009	240-006	241-038	241-057	
		241-064	251-008	255-008	#256-001			
KF14M1	1B41	158-034	198-031	232-034	236-041	256-002	#260-004	
		260-006	271-015	295-026				
KF14M2	1B49	232-009	255-025	#260-008	260-010			
KF14M3	1B52	235-006	#260-012	260-014				
KF14M4	1B5F	231-103	#260-016	260-018				
KF14M5	1B67	230-073	#260-020	260-022				
KF14NR	1587	229-058	#230-072					

4,292,666

551

552

KF14NV	15A8	229-036	#231-102				
KF14VT	15A0	231-084	#231-095				
KF14X	1B01	228-009	233-013	233-042	234-091	235-023	237-054
		238-005	239-057	243-021	244-041	246-005	246-031
		247-050	248-030	249-050	250-074	250-094	251-013
		252-051	254-051	255-040	#256-005		
KF14Z	1B02	229-043	229-057	232-004	232-033	#257-011	
KF15	1B73	127-019	127-020	127-035	127-036	128-050	128-057
		128-059	#261-007				
KF15MS	1B7A	261-007	#261-013	261-015			
KF15MX	0009	261-013	#261-015				
KF16	1B84	127-032	127-033	127-034	127-039	127-040	127-041
		127-046	127-047	127-048	128-053	128-054	128-055
		#262-005					
KF16MS	1B8B	262-005	#262-011	262-013			
KF16MX	000B	262-011	#262-013				
KF17	1B97	127-006	#263-005				
KF18	1B84	100-017	#264-007				
KF18M1	1C7E	266-032	#268-046	268-048			
KF18M2	1C82	266-034	#268-050	268-052			
KF18X	1C78	264-029	266-005	267-021	267-035	#267-039	
KF1805	1B86	#264-009	264-028				
KF1810	1BD2	264-022	#264-025				
KF1815	1BDD	264-017	#264-031				
KF1816	1C2B	266-023	#266-026				
KF1820	1C2C	266-010	266-014	#266-028			
KF1825	1C3D	266-033	#266-036				
KF1830	1C43	264-044	#267-004				
KF1835	1C4A	264-046	#267-009				
KF1840	1C51	267-007	#267-015				
KF1899	1C6C	266-037	#267-031				
KF19	1C86	128-051	#269-016				
KF19TB	1CFE	271-004	#272-004				
KF19X	1D13	269-020	270-062	271-017	#272-021	273-024	273-046
		274-059					
KF1905	1C95	269-018	#269-022				
KF1910	1CD1	#270-052	270-060				
KF1920	1CE3	#271-004	288-021				
KF1925	1CE8	#271-007	271-013				
KF1930	1CFC	271-008	#271-019				
KF20	1DFA	100-011	#278-008				
KF20X	1E9A	278-010	278-020	#282-029			
KF2005	1E1C	#278-026	282-027				
KF2010	1E33	#278-042	278-044				
KF2020	1E52	#280-017	280-044				
KF2025	1E59	#280-022	280-028				
KF2030	1E6C	280-019	280-034	#280-040			
KF2035	1E7D	279-050	280-014	#282-004			
KF2040	1E8C	278-038	#282-018				
KF21	1EE1	100-005	#285-005				
KF21X	1F22	285-007	285-029	#286-047			
KF2120	1EFO	#285-013	285-015				
KF2130	1EF5	* #285-017					
KF2140	1EFA	#285-020	285-026				
KF2150	1F0B	285-022	#286-032				
KM12NL	124E	201-036	#201-040	201-042			
KM12X1	0008	201-040	#201-042				
KNET	0008	#030-013	163-006	198-004	210-005	214-063	227-036
		228-012	278-009				
KRESET	0020	#030-011	113-026	123-036	194-042	263-025	288-017
		288-024					
KSHIFT	0080	#030-009	153-009	153-012	153-017	263-025	289-013
		290-009					
KSTATE	FE7C	#030-005	030-018	113-025	113-027	118-015	123-035
		150-015	150-028	150-030	153-007	163-005	163-007
		194-041	196-020	196-022	198-003	210-004	214-062
		214-064	227-037	228-011	242-009	242-011	263-022
		269-034	278-008	288-012	288-016	288-023	289-012
		290-008	290-010	295-006	295-008	346-049	346-051
KSUPER	0002	#030-015	269-033	288-013			
KU01	1F28	131-010	135-006	143-005	149-005	150-011	153-006
		154-016	158-017	160-005	182-005	196-005	197-010
		226-005	228-005	269-016	#288-011		
KU0105	1F3E	288-014	#288-023				
KU0110	1F46	288-018	#288-027				
KU0115	1F47	288-025	#288-029				

4,292,666

553

554

KU02	1F49	131-011	143-006	149-006	150-012	154-017	158-018
		160-006	182-006	194-045	196-006	197-011	226-006
		228-006	269-017	#289-010			
KU03	1F53	113-028	141-130	144-050	194-058	196-024	213-009
		#290-007					
KU04	1F68	118-012	226-032	269-046	273-023	273-045	274-058
		#291-012	362-006	363-031	380-005	382-128	384-006
		385-082					
KU05	1F79	131-013	143-008	149-019	154-019	158-020	160-008
		226-008	228-008	269-019	#292-005		
KU05MS	1F80	292-005	#292-011	292-013			
KU05MX	0009	292-011	#292-013				
KU06	1F8A	118-009	163-015	213-008	226-026	269-028	#293-007
KU0610	1F90	#293-011	293-014				
KU0620	1F9B	#293-019	293-022				
KU0630	1FA6	#293-027	293-030				
KU0640	1FB1	#293-035	293-037				
KU07	1FC7	158-039	184-056	233-012	235-022	237-053	243-020
		251-012	#295-001				
KU07ER	1FF1	#295-026	296-010	296-017	296-024	296-031	296-038
		296-045	297-016	297-022	298-053	299-006	299-009
		299-011	299-027	300-005	300-014	300-017	301-012
		301-039	302-016				
KU07X	2120	295-030	#302-025				
KU0700	1FFC	295-014	#296-004				
KU0701	200B	295-016	#296-014				
KU0702	2014	295-018	#296-021				
KU0703	201D	295-020	#296-028				
KU0704	2026	295-022	#296-035				
KU0705	202F	295-024	#296-042				
KU0710	203B	296-006	296-016	#297-005			
KU0713	2053	297-014	#297-024				
KU0715	205D	#297-033	297-037				
KU0717	2065	297-035	#297-042				
KU0719	2070	297-046	#298-057				
KU0730	207C	296-023	#299-004				
KU0735	20AA	299-025	#299-029				
KU0740	20B2	296-030	#300-003				
KU0745	20D4	296-037	#301-003				
KU0750	20ED	#301-021	301-025				
KU0753	20F5	301-023	#301-032				
KU0770	2100	296-009	#302-004				
KU0780	210C	296-044	#302-011				
KU0785	2113	#302-015	302-019				
KU0799	211F	298-060	298-062	299-033	300-019	301-040	302-007
		#302-023					
KU08	2121	163-013	215-076	226-029	#303-017		
KU09	2133	233-041	238-004	252-049	#305-001		
KU09ER	2176	305-037	#305-042	306-031	307-041	308-043	310-019
KU09UP	22CF	247-059	305-039	306-040	307-043	309-054	310-021
		311-037	#314-006				
KU0910	215C	#305-025	305-027				
KU0915	2179	305-011	#306-004				
KU0920	2185	#306-014	306-017				
KU0930	21AF	305-007	#307-004				
KU0931	21CF	#307-027	307-031				
KU0932	21D8	307-028	#307-033				
KU0935	21ED	307-012	#308-004				
KU0936	21FD	#308-016	308-018				
KU0937	2209	#308-026	308-029				
KU0940	2233	307-008	#310-004				
KU0950	2254	310-009	#311-004				
KU0955	2265	#311-017	311-019				
KU0960	22AC	312-066	#313-004				
KU099A	22DF	#314-016	314-031				
KU099B	22EA	314-021	#314-024				
KU11	2304	133-024	201-012	202-005	203-012	204-019	221-041
		235-003	240-010	246-007	285-005	308-012	311-013
		#316-005	326-014				
KU12	230B	131-016	134-066	154-022	158-042	167-009	172-017
		174-004	174-025	234-076	238-007	238-039	244-065
		245-080	248-040	250-082	252-054	253-001	253-029
		269-036	#317-008	319-052	320-005	325-016	326-018
KU13	2313	167-007	169-008	#318-010			
KU1305	2319	#318-013	318-016				

KU1310	2321	318-014	#318-018				
KU1320	2331	#318-029	318-047				
KU1325	2345	#318-040	318-043				
KU1330	234B *	#318-045					
KU1335	2350	318-032	#319-049				
KU1340	2354	318-023	#319-052				
KU14	2364	170-014	171-028	175-009	#320-005		
KU15	237D	241-037	244-062	#321-006			
KU15A	2380	240-005	#321-007				
KU15X	238B	321-012	#321-016				
KU1505	238A	321-009	#321-014				
KU16	238C	163-017	226-034	#322-007			
KU1610	239C	#322-016	322-021				
KU17	23AA	133-004	140-084	140-113	141-124	178-015	179-024
		244-052	314-013	#323-012			
KU17A	23B1	219-091	220-018	257-012	#323-019		
KU1705	23B2	323-015	#323-024				
KU1710	23BC	#323-035	323-037				
KU18	23CB	211-087	227-048	#324-010	336-098	338-088	
KU19	23E3	234-089	238-023	239-051	244-070	252-070	253-022
		253-045	319-059	#325-012			
KU19X	2409	325-015	325-022	325-026	#325-033		
KU1905	23F3	#325-020	325-031				
KU1910	2403	325-024	#325-028				
KU20	240A	234-090	238-024	239-052	250-093	252-071	253-023
		253-046	#326-012				
KU20X	2442	326-016	326-025	326-038	#326-044		
KU2005	2426	326-023	#326-027				
KU2010	2435	326-031	#326-035				
KU2015	243D	326-033	#326-040				
KU21	2445	121-017	178-035	#327-012			
KU22	2451	133-022	165-004	203-007	205-015	210-020	219-102
		240-008	246-018	305-002	#328-012		
KU2210	245C	#328-017	328-019				
KU9SUB	22F8	305-034	306-028	307-038	308-040	310-016	311-026
		#315-013					
K02SUB	0985	135-018	136-051	136-060	137-085	#142-136	
K09A00	0F4D	164-027	164-028	164-029	164-030	164-031	164-032
		164-033	164-034	#176-004			
K09Z	0F87	168-063	#179-019				
K09ZX	1000	180-083	180-093	181-104	#181-109		
K09ZZ	0F92	166-065	#179-034				
K09Z10	0FAB	#180-057	180-059				
K09Z20	0FBB	180-062	180-066	#180-070			
K09Z25	0FBD	180-068	#180-073				
K09Z30	0FC6	#180-079	181-098				
K09Z35	0FD3	#180-086	181-107				
K09Z40	0FE5	180-077	#180-095				
K09100	0D48	164-006	#165-004				
K09120	0D6D	#165-023	166-061				
K09130	0D7D	165-028	#165-032				
K09140	0D82	165-025	#166-035				
K09145	0D92	166-046	#166-048				
K09150	0D96	166-037	#166-052				
K09160	0D98	165-033	#166-054				
K09170	0D9B	165-030	166-050	#166-057			
K09200	0DAA	164-007	164-008	164-009	164-010	164-011	164-012
		164-013	164-014	#167-003			
K09210	0DD5	167-019	167-021	#167-026			
K09220	0DES	167-037	#167-040				
K09230	0DEE	167-023	#167-043				
K09240	0E09	167-041	#168-063	169-009	170-023	171-051	173-068
		175-027					
K09300	0E0F	164-015	164-016	#169-004			
K09400	0E1C	164-017	#170-014				
K09405	0E1F	#170-016	174-020				
K09500	0E2E	164-018	#171-028				
K09505	0E31	#171-030	174-021				
K09510	0E4F	171-042	#171-046				
K09515	0E51	171-040	171-044	#171-048			
K09600	0E59	164-019	164-020	164-021	164-022	#172-004	175-007
K09610	0E66	#172-011	172-014				
K09620	0E6E	172-012	#172-016				
K09625	0E7C	#172-025	174-045				
K09630	0EAA	172-048	#173-032				
K09640	0EAC	172-046	173-050	#173-054	173-074	173-077	
K09650	0EB5	#173-062	173-067				

K09660	0EC1	172-040	#173-070	
K09700	0ED0	164-023	164-024	#174-004
K09800	0EF3	164-025	#174-025	
K09810	0F16	174-032	174-035	174-038 #174-041
K09900	0F23	164-026	#175-004	
K09905	0F45	175-021	#175-024	
K12SUB	13DE	#217-003	222-064	
K12S02	13FE	217-024	217-026	#217-035
K12S04	1400	217-031	#217-041	
K12S10	1406	#218-051	218-066	
K12S15	142B	218-059	218-078	#219-081
K12S20	1431	#219-090	219-098	
K14M1X	0007	260-004	#260-006	
K14M2X	0008	260-008	#260-010	
K14M3X	000C	260-012	#260-014	
K14M4X	0007	260-016	#260-018	
K14M5X	000B	260-020	#260-022	
K14TAB	1B35	251-030	#259-034	
K14005	153D	228-007	#228-011	
K14010	154B	228-013	#228-017	
K14012	1574	#229-055	230-067	
K14014	158D	229-030	#231-079	
K14015	15AE	230-065	231-091	231-098 #232-003
K14016	15E7	232-027	#232-031	
K14020	15E9	232-029	#232-033	
K14025	15F2	232-022	#232-037	
K14030	15FB	#232-041	232-044	
K14035	1603	232-042	#233-001	
K14037	1649	#234-052	234-070	
K14038	165C	#234-062	234-066	
K14039	1667	234-055	#234-072	
K14040	166C	233-046	#234-076	
K14065	160D *	#233-012		
K14070	1628	233-020	#233-032	
K14075	1694	233-004	#235-003	
K14080	16C2	235-028	#235-031	
K14085	16D9	235-017	#236-006	
K14090	1700	#236-031	236-034	
K14095	1708	236-032	#236-036	
K14100	1723	236-014	#237-052	
K14110	1738	235-043	237-050	#238-003
K14115	1772	#238-030	238-033	
K14120	177A	238-031	#238-035	
K14135	17AD	229-044	232-005	#240-003
K14145	17DE *	#241-040		
K14146	17F4	241-050	#241-061	
K14147	17FD	241-056	#241-068	
K14150	180A	240-031	#242-003	
K14151	1831	242-021	#242-025	
K14152	183E	242-017	242-019	#242-032
K14153	1848	242-030	#242-038	
K14154	1856	242-023	242-028	242-035 #243-004
K14155	1860	243-006	#243-010	
K14156	1866	#243-013	243-016	
K14157	186E	243-014	#243-018	
K14158	1887	242-005	#244-039	
K14159	189A	244-048	#244-050	
K14160	18BD	244-060	244-063	#245-074
K14164	18E1	245-091	245-097	#245-104
K14165	18E7	245-103	#245-108	245-114
K14170	18F1	245-076	245-106	#246-003
K14172	18FF	#246-011	246-013	
K14174	191F	246-029	#246-033	
K14175	194C	246-025	#248-003	
K14180	199E	248-013	248-018	#250-056
K14190	19C4	247-060	#250-081	
K14200	19E1	255-011	#251-005	
K14201	19FD	251-018	#251-021	
K14205	1A13	#251-034	251-037	
K14210	1A1B	251-033	#251-039	
K14225	1AAF *	#253-048		
K14900	1ABA	228-021	#255-004	
K14905	1AE3	255-023	#255-028	
K18M1X	0003	268-046	#268-048	
K18M2X	0003	268-050	#268-052	

K19NG	1D5D	273-018	273-040	#274-064				
K19000	1D14	272-005	#273-003					
K191ME	0007	277-052	#277-054					
K191MS	1DD6	273-022	#277-051	277-054				
K19100	1D15	272-007	#273-007					
K192ME	0008	277-057	#277-059					
K192MS	1DDF	273-044	#277-056	277-059				
K19200	1D31	272-009	#273-028					
K193ME	0007	277-062	#277-064					
K193MS	1DE7	274-057	#277-061	277-064				
K19300	1D4D	272-011	#274-051					
K194ME	000A	277-067	#277-069					
K194MS	1DEF	274-065	#277-066	277-069				
K20SSR	1EC7	283-012	#284-004					
K20SSX	1EE0	284-006	284-010	#284-020				
K20SS1	1ED7	284-008	#284-012					
K20SUB	1EA0	278-033	282-016	#283-004				
K20SX	1EC6	283-027	#283-033					
K20S05	1EA2	#283-006	283-031					
K20S10	1EA5	#283-010	283-030					
K20S15	1EA6	#283-012	283-015					
K20S20	1EB4	#283-022	283-024					
LASTKY	FE83	#030-042	031-003	118-006	123-026	123-032	124-058	
		269-030						
LEDTMR	001E	#024-027	286-047	327-018				
LENCAS	002F	* #360-023						
LENDEC	0006	#042-045	222-052	248-027				
LENDEL	0006	#042-039	211-079	212-026	313-022			
LENGD	0004	#042-042	273-028					
LENINC	0008	#042-044	246-046					
LENINI	0004	#042-043	274-051	380-015				
LENINS	0008	#042-038	226-022	315-018				
LENLED	0005	#042-040	286-044					
LENNAK	0005	* #042-046						
LENPWR	0006	#042-037	278-018					
LENRED	0006	#042-034	119-008	156-021	161-038	163-026	213-025	
		266-001	267-018	275-023	365-027			
LENSCH	000A	#042-036	119-044	185-084				
LENSTP	0004	#042-041	273-007	380-010				
LENWRT	000A	#042-035	132-060	133-047	155-067	156-045	209-070	
		244-039	250-071	255-038	312-061	381-070		
LOAD	29D9	272-013	#380-001					
LOADER	2A4B	* #382-134						
LOADR2	2A4E	382-119	#382-136					
LOAD05	29E6	#380-026	382-108					
LOAD10	29EA	#380-032	382-101					
LOAD20	29FC	* #380-045						
LOAD25	2A07	380-047	#381-065					
LOAD30	2A42	380-037	382-114	#382-122				
LOGFLD	0000	* #031-027						
LOGRAM	29A9	377-088	#378-099					
L1C01L	F85B	#016-005	017-004					
L1C01U	F80B	#016-004	016-005	278-022				
L2C01L	F8FB	* #017-005						
L2C01U	F8AB	#017-004	017-005	269-048				
MAJREV	0041	#001-012	353-025					
MATROL	004D	#029-018	293-025					
MATROW	FE2F	#029-006	029-008	029-018	293-024	323-032		
MATROX	FE7C	#029-016	029-018	030-005				
MATRW1	FE2F	#029-008	029-009					
MATRW2	FE3A	#029-009	029-010					
MATRW3	FE45	#029-010	029-011					
MATRW4	FE50	#029-011	029-012					
MATRW5	FE5B	#029-012	029-013					
MATRW6	FE66	#029-013	029-014					
MATRW7	FE71	#029-014	029-016					
MAXBUF	0010	#360-011	364-021					
MAXCOL	000B	#030-022	090-012	135-036	138-010	139-043	139-053	
		178-025	180-082	202-014	210-011	224-089	230-064	
		231-090	234-054	241-049	264-021	267-034	278-037	
		282-026	314-027	318-031	323-032	323-033		
MAXROW	0007	#030-024	165-021	181-103	200-085	201-014	231-083	
		235-005	283-004	283-004	286-032	322-014		
MAXTRY	0004	#026-012	345-022	348-040				
MEMSIZ	FE86	#031-006	031-007	119-033				
MEMUSE	FE88	#031-007	031-009	120-062	324-015	324-019		

4,292,666

561

562

MOVBC	00D7	#048-010	068-001	121-006	121-008	121-010	155-053
		155-056	156-042	226-018	226-020	246-042	248-036
		312-054	312-059	365-037	371-029	371-033	371-037
		372-053	397-074	411-052			
MOVDE	00EF	#051-008	119-006	119-034	119-038	119-040	119-042
		120-063	131-042	132-055	132-058	133-034	133-041
		133-045	153-022	156-043	161-036	163-009	163-011
		163-023	179-047	180-067	180-073	182-019	182-022
		182-025	195-100	192-024	194-052	203-022	206-010
		207-055	208-063	208-066	210-033	212-022	213-024
		227-044	227-046	240-018	240-021	240-024	246-043
		247-055	248-025	250-057	250-060	250-069	255-018
		255-033	255-036	265-051	267-016	275-021	278-016
		303-022	305-019	306-010	306-020	306-037	307-023
		308-011	308-030	310-013	311-006	312-053	313-014
		324-020	335-053	336-084	336-091	338-051	338-079
		338-080	338-081	340-023	340-030	341-022	344-007
		371-009	371-043	376-044	396-027	397-049	408-028
		408-038	408-039	408-040	408-045	409-051	
MOVSTR	0103	#064-030	091-010	091-016	091-022	091-030	113-022
		174-041	196-017	196-023	266-036	270-054	291-018
		344-039					
MOVSI0	0106	#064-034	064-039	134-070	143-045	158-046	172-022
		234-080	234-085	238-011	238-020	238-044	239-050
		244-069	245-105	252-058	252-068	253-044	319-056
		320-010					
MPXFLG	0002	#031-047	174-037	259-041			
MSGADI	274A	349-016	#354-045	354-047	378-126	382-135	
MSGADR	273D	349-014	#354-041	354-043			
MSGADX	000C	354-041	#354-043				
MSGADY	000C	354-045	#354-047				
MSGBCX	000A	414-004	#414-006				
MSGBDC	2D27	375-024	#414-003	414-006			
MSGBDL	2805	349-030	#356-109	356-111			
MSGBDR	2D3F	382-118	397-082	#414-013	414-016		
MSGBDX	000A	356-109	#356-111				
MSGBRX	000A	414-014	#414-016				
MSGBSX	0009	354-065	#354-067				
MSGBSY	2788	344-037	344-044	#354-065	354-067		
MSGCHK	2707	346-042	349-010	#353-019	353-021	393-144	
MSGCHX	000C	353-019	#353-021				
MSGCMD	27B3	349-018	#355-081	355-083			
MSGCMX	000B	355-081	#355-083				
MSGCNM	2D92	386-033	#416-054	416-057			
MSGCNX	000C	416-055	#416-057				
MSGCON	27BF	349-032	#355-085	355-087			
MSGCOX	000B	355-085	#355-087				
MSGDOK	2D64	363-030	#414-033	414-036			
MSGDOX	0009	414-034	#414-036				
MSGDPG	2D5C	362-005	#414-028	414-031			
MSGDPX	0007	414-029	#414-031				
MSGEDL	27D7	162-052	#355-093	355-095			
MSGEDX	000C	355-093	#355-095				
MSGFUL	277C	349-038	#354-061	354-063			
MSGFLX	000B	354-061	#354-063				
MSGHI	2714	118-011	#353-023	353-027			
MSGHIX	0008	353-023	#353-027				
MSGLDD	2D52	382-127	#414-023	414-026			
MSGLDG	2D4A	380-004	#414-018	414-021			
MSGLDX	0007	414-019	#414-021				
MSGLDZ	0009	414-024	#414-026				
MSGLEN	FDR0	#026-005	026-006	108-058	108-065	108-069	
MSGLNM	2D85	386-030	#416-049	416-052			
MSGLNX	000C	416-050	#416-052				
MSGMEM	271D	349-026	#353-029	353-031			
MSGMEY	0007	303-029	#353-031				
MSGMSK	0007	344-020	#354-049	354-051			
MSGMSX	0007	344-049	#354-051				
MSGNET	27E4	198-005	210-003	226-014	#355-077	355-099	
MSGNEF	000A	355-097	#355-099				
MSGNOD	2764	349-024	#354-053	354-055			
MSGNOX	000C	354-053	#354-055				
MSGNOZ	2D32	387-041	389-006	401-025	#414-008	414-011	
MSGNPD	279D	349-034	#355-073	355-075			
MSGNFX	000B	355-073	#355-075				

MSGNXZ	000C	414-009	#414-011					
MSGDVR	26FB	346-034	349-008	#353-015	353-017			
MSGDVX	000B	353-015	#353-017	389-034				
MSGPAR	26EE	346-038	349-006	#353-011	353-013			
MSGPAY	000C	353-011	#353-013	389-024				
MSGRES	27FB	348-043	#356-105	356-107				
MSGREX	000C	356-105	#356-107					
MSGRNM	2D78	386-040	#416-044	416-047				
MSGRNX	000C	416-045	#416-047					
MSGRSP	27A9	346-046	#355-077	355-079				
MSGRSX	0009	355-077	#355-079					
MSGSCH	27EF	195-065	#355-101	355-103				
MSGSCX	0008	355-101	#355-103					
MSGSEQ	2792	349-022	#354-069	355-071				
MSGSEX	000A	354-069	#355-071					
MSGSQL	270B	161-024	#355-089	355-091				
MSGSOX	000B	355-089	#355-091					
MSGSTP	2729	349-028	#353-033	353-035				
MSGSTX	000B	353-033	#353-035					
MSGSUP	2771	269-045	349-036	#354-057	354-059			
MSGSUX	000A	354-057	#354-059					
MSGTIM	2735	349-012	#353-037	354-039				
MSGTIX	0007	353-037	#354-039					
MSGVFB	2DAE	384-005	#416-039	416-042				
MSGVFX	0009	416-040	#416-042					
MSGVKX	0009	416-060	#416-062					
MSGVOK	2D9F	385-081	#416-059	416-062				
MULDIS	000Z	#258-015	258-017					
MULKEY	0000	#258-013	258-014					
MULLN1	1B08	238-009	252-056	#258-005	320-008			
MULLN2	1B0D	238-048	253-042	#258-009				
MULNOD	0001	#258-014	258-015					
MULRCL	0007	172-009	238-027	#258-017				
MULTAB	1B12	172-008	238-028	#258-019				
NEWKEY	FE82	#030-040	030-042	118-007	123-018	123-023	123-031	
		269-031						
NDCALC	0016	#008-026	147-112	147-119	147-126	148-133	190-017	
		199-069	251-043					
NDCCON	0014	#008-024	174-019	190-022	199-067	242-018		
NDCOIL	0007	#008-011	133-006	133-012	146-050	342-017		
NDCON	0013	#008-023	148-139	172-039	188-013	198-025	199-060	
		200-090	235-016	236-045	236-048	237-057	237-060	
		258-031						
NDCPRE	000D	#008-017	148-144	198-019	235-027	242-016	251-017	
		321-008						
NDCREB	0015	#008-025	190-029	242-022	251-027			
NDCREL	0004	#008-008	145-032					
NDCTR	000F	#008-019	147-086	198-022	199-058	258-019		
NDCBLK	0040	#145-020	146-063	146-069	296-043			
NDCOOL	0009	#008-013	133-006	133-012	146-074			
NDCOON	0007	134-035	134-063	#145-007	145-008	232-037	243-010	
		318-010	318-018					
NDCOST	0020	#145-017	148-145	235-021	251-011	296-008		
NCDIS	0001	134-063	#145-005	145-006	318-018			
NCDHRG	0010	#145-018	147-087	147-093	147-099	147-106	147-113	
		147-120	147-127	148-134	148-140	148-150	158-038	
		235-021	237-052	251-011	296-036			
NCDJNF	0002	#145-015	145-027	145-033	145-039	146-045	148-140	
		158-038	296-015					
NCDIRG	0008	#145-017	148-150	158-038	235-021	251-011	296-029	
NCDKEY	0000	#145-004	145-005					
NCDLAT	000A	#008-014	133-009	133-015	146-080	188-020	342-020	
NCDMSK	007C	#031-039	133-044	163-031	167-004	169-005	170-020	
		172-005	172-038	174-018	179-026	180-096	183-040	
		185-115	188-007	190-011	241-074	314-008	314-017	
NCDOUT	0001	#145-014	145-027	145-033	145-039	146-045	146-051	
		146-057	146-075	146-081	148-140	158-038	296-005	
NCDRCL	0009	134-056	143-014	#145-010	148-153	232-038	243-011	
		318-011						
NCDSEQ	0004	#145-016	145-027	145-033	145-039	146-045	296-022	
NCDTAB	09F6	134-005	145-013	#145-023	148-153	232-037	243-010	
		318-010						
NCDTEL	0015	143-015	#148-153					
NCDTYP	0006	#145-006	145-007					
NCDVAL	0008	#145-008	145-010					

4,292,666

565

566

NOECC	0002	#008-006	165-007	181-097	246-033	306-019	306-036
		308-020	308-048	314-020			
NOEDL	0001	#008-005	119-039	162-049	183-116	213-035	
NOH070	000B	#008-015	146-062				
NOH075	000C	#008-016	146-068				
NOLATC	0008	#008-012	133-009	133-015	146-056		
NDNEGT	0004	#008-010	145-044	233-027			
NDNULL	0017	* #008-027					
NDQREL	0003	#008-007	145-026	186-018			
NDP0ST	0005	#008-009	145-038	233-025			
NDRPRE	000E	#008-018	148-149	190-027	235-029	242-020	251-019
NDSDN	0000	#008-004	161-045	226-019			
NDT001	0012	#008-022	147-105	258-028			
NDT010	0011	#008-021	147-098	258-025			
NDT100	0010	#008-020	147-092	258-022			
NSWP	000F	#047-015	112-010	201-021	236-024	316-007	323-028
		404-021	406-099	407-125			
NUM1K	000E	#009-007	061-013	061-017			
NXTADR	2879	363-011	#366-023	381-091	385-060		
NXTREG	2882	366-027	#366-038				
NXTSEG	2A31	382-100	#382-105				
OUTFLG	0001	#031-035	167-018				
OUTHIS	0020	* #032-057					
OUTSTA	0002	#032-052	156-030	156-031	266-011		
PADCNT	0001	#012-030	089-019				
PARIN	003E	#002-027	123-009				
PAROUT	003E	#002-028	054-011	065-032	086-043	099-006	123-008
PDIA	3300	#010-023	057-014				
PIO	2581	119-009	119-045	132-061	133-048	155-068	156-022
		156-046	161-039	163-027	185-085	209-071	211-080
		212-027	213-026	222-053	226-023	244-040	246-047
		248-028	250-072	255-039	266-002	267-019	273-008
		273-029	274-052	275-024	276-019	286-045	312-062
		313-024	315-020	#344-001	380-012	380-017	412-096
PIOERR	262F	346-037	346-041	346-043	#346-048	348-044	
PIOTAB	2693	348-048	#349-004	349-040			
PIOTBL	0011	349-004	#349-040				
PIOX	2692	347-031	#348-054				
PI0010	2593	#344-013	344-016				
PI0020	259A	#344-019	344-048	345-029			
PI0030	25A2	#344-024	344-032				
PI0040	25B5	#344-037	345-007				
PI0050	25C1	#344-042	344-043				
PI0060	25D4	344-035	#345-001				
PI0070	25DD	#345-006	345-018				
PI0080	25EA	* #345-012					
PI0085	25F9	345-010	#345-020				
PI0087	2603	#345-025	348-042				
PI0090	260E	345-024	#346-033				
PI0100	2641	345-015	#347-002				
PI0110	265F	#347-017	347-023				
PI0120	2658	347-029	#348-046				
PI0130	2690	346-054	#348-031				
PISTAT	0080	* #003-039					
POBEEP	0040	#003-007	065-031	099-005	123-002		
POPENB	0000	* #003-026					
POPEVN	0001	* #003-027					
POPWR	0080	#003-006	054-010	066-042	123-002		
POSAVE	FDB4	#026-010	027-030	063-030	063-033	086-044	099-004
		099-007	123-001				
PO2STP	0007	* #003-028					
PFCMD	0025	#007-039	103-024	107-047	110-015	345-003	
PPIRFL	0020	#025-028	025-035	103-002	109-101		
PPIBLK	FD9C	#025-019	025-021	103-003	108-074	109-102	347-007
PPIRUF	FFA0	#025-035	025-036	103-001	109-100		
PPICFR	0002	#026-023	109-093	345-027	346-044		
PPICHK	FDBJ	#026-006	026-007	107-024	108-076	108-083	
PPICNT	0061	4026-024	106-011	107-025			
PPIDON	0020	#026-019	108-066	345-014	347-005		
PPIFCN	0040	#026-018	106-008	106-016			
PPIMSG	0080	#026-017	106-005	106-037			
PPINIT	0470	058-008	#103-001	344-040	346-033		
PPINT	0453	083-015	#105-001				
PPIOVR	0008	#026-021	103-025	345-027	346-036		
PPIFAR	0010	#026-020	103-016	345-027	346-040		

4,292,666

567

568

PPIBET	0304	#031-022	107-097	345-017	345-027		
PPISTA	FD0E	#031-005	026-004	105-012	106-003	106-017	107-026
		108-060	108-087	109-098	345-012	345-026	345-028
		316-093	347-004	347-006			
PPTX	057A	110-006	110-013	#110-018			
PPI010	0465	105-012	#105-019				
PPI030	0470	105-021	#106-001				
PPI040	0494	106-009	#107-023				
PPI045	0400	104-012	#108-056				
PPI050	0400	106-008	#108-065				
PPI060	048E	107-027	107-051	108-061	#108-073		
PPI070	04EA	108-067	#108-083				
PPI080	04E9	108-045	#109-093				
PPI090	04EF	105-017	105-026	#109-097			
PPI100	050E	105-005	106-015	106-018	108-079	108-088	#110-004
PPI110	0534	110-009	#110-010				
PPMODE	00FE	#107-005	108-022				
PPNULL	0081	#007-040	108-015	387-020			
PPDBFI	0020	#025-019	025-036	108-007	107-040		
PPDBLK	FD07	#025-021	025-025	108-008	107-041	110-007	110-010
		344-020					
PPDBUF	FF80	#025-036	025-037	108-008	107-039		
PPDCHK	FD83	#037-008	026-010				
PPDSTA	FD0F	#036-004	026-005	108-013	107-045		
PU01	2407	345-049	#351-001				
PU01X	24E0	351-012	#351-021				
PU0110	2409	#351-004	351-010				
PU0120	2488	351-005	#351-014				
PU02	24E1	344-034	344-042	345-006	#352-017		
PU02X	24ED	352-021	#352-025				
PU0210	24FC	352-019	#352-023				
PURE10	0049	#057-005	059-006	062-057			
PURE20	004F	058-022	#059-008	059-015			
PURE21	0085	#079-010	059-014				
PWRTMR	0002	#024-027	262-027	327-016			
PWRUP	003E	048-006	#054-006				
PWR010	0040	#054-018	054-021				
PWR020	0055	#056-004					
PWR030	005E	#056-009	056-013	056-015			
PWR040	006P	#056-018	056-036				
PWR050	0071	#056-020	056-029	056-031			
P2CKER	2B9D	393-130	#393-143				
P2INIT	2AFC	362-012	380-023	384-008	#387-018		
P2MODE	0079	#007-037	387-027				
P2RDCH	2AED	#389-001	391-035	391-044	391-047	392-080	392-088
		392-091	392-096	393-098			
P2RD0F	2B1F	389-011	389-025	389-035	#389-047		
P2RD02	2B0D	389-030	#389-027				
P2RD03	2B19	389-030	#389-037				
P2RD05	2AF3	#389-007	389-014				
P2RIAA	2B57	392-060	#392-037				
P2RID	2B79	380-036	382-113	384-027	385-074	#391-022	
P2RID0A	2B59	392-063	#392-073				
P2RID2	2B33	#391-034	391-037				
P2RID3	2B5F	#392-079	392-084				
P2TCH	2C35	397-031	399-040	#401-017			
P2TCH2	2C39	#401-024	401-033				
P2TER	2C51	401-027	#402-046				
P2TID	2BFF	363-000	#399-001	409-053			
P2TID5	2C21	#397-038	397-043				
P2TID7	2C19	#399-029	399-035				
RAMHI	FFFF	#009-010	009-013	020-034			
RAMLO	FR00	#009-009	009-013	010-011	056-004	056-018	
RANSI7	0800	#009-013	036-005	056-019			
RCOUNT	FD82	#026-007	026-008	344-002	345-020	348-038	
RDSYS	1D64	273-014	273-035	#273-014			
RDSYSX	1D7F	275-025	#275-030				
REFLEN	0004	#014-019	158-044	158-048	159-050	197-026	302-013
REGFLD	0040	#031-029	255-015	267-006			
REGMSK	00E0	#034-051	167-048	299-023			
REGHAM	28R1	368-044	#369-052				
ROMCHK	008E	005-008	#061-012				
ROMHI	37FF	#009-006	009-007	009-012	061-013		
ROMLO	0000	#009-005	009-007	009-012	061-016		

4,292,666

569

570

ROMSTZ	3800	*	#009-012					
ROMTES	0007		#081-021	062-066				
ROMTS1	0000		#061-026	061-058				
ROMTS2	0051		061-044	#062-054				
ROWA	0007		#012-002	012-015	012-015	012-019	012-021	012-021
			012-023	014-004	014-004	014-017	014-017	015-006
ROWB	0050		#012-057	012-015	014-017	016-009	017-004	017-005
			112-009	112-009	112-015	165-018	167-010	174-012
			174-026	174-026	174-041	174-041	217-045	218-063
			214-070	225-096	234-081	234-086	238-014	238-046
			248-044	250-087	252-062	253-015	253-040	270-057
			283-017	320-014	322-011	326-019	326-027	
ROWBEG	0317		090-013	#090-020				
ROWBFI	0008		#027-015	027-038	027-034	027-035	027-036	027-037
			027-038	027-039	027-041	285-010	285-011	285-017
ROWBFI	0374		083-015	#091-004				
ROWC	0045		#012-008	012-015	091-004			
ROWENT	000E		#012-029	089-007	293-033			
ROWCUR	0005		#027-013	027-014				
ROWD	004D		#012-009	012-015	015-006	015-016	112-026	131-046
			132-065	149-016	158-048	172-025	197-030	197-033
			229-034	231-096	238-021	246-027	248-011	250-089
			266-017	266-030	267-023	305-009	307-010	310-007
ROWE	004E		#012-010	012-015	015-006			
ROWFBK	0010	*	#027-022					
ROWFEN	0040	*	#027-020					
ROWFLG	0000		#027-010	027-011				
ROWFMA	0001		#027-011	027-012				
ROWFMN	0020	*	#027-021					
ROWFSN	0080		#027-019	285-021				
ROWLMA	0003		#027-012	027-013				
ROWLOG	0305		089-009	#090-012	293-035			
ROWL10	030A		#090-015	090-017				
ROWMSK	00F0		#030-030	112-005	131-018	135-028	135-039	136-054
			136-063	136-074	137-091	137-098	137-110	139-059
			154-025	158-024	181-102	197-018	200-084	228-019
			231-082	232-024	305-005	316-006	323-027	
ROWNOB	0317		090-015	#090-023				
ROWN10	0319		090-021	#090-025	091-005	091-012	091-024	091-026
			091-028	113-019	263-031	291-015	344-047	
ROWN20	031B		#090-027	090-030	159-051	217-018	218-062	225-095
			225-099					
ROWPAD	02FE		089-005	089-006	089-013	089-014	089-020	#090-004
ROWSEQ	0006		#027-014	027-015				
ROWST1	0329		089-016	#091-009	269-042			
ROWST2	0340		089-017	#091-021	269-043			
ROWST3	035B		091-017	#091-032				
ROWST4	035D		#091-034	091-043				
ROWST5	0362		#091-038	091-041				
ROWTAB	FDB5		#027-030	027-032	027-041	285-011	293-007	
ROWTBL	0038		#027-041	293-008				
ROWTBX	FDED		#027-039	028-021				
ROWTB1	FDB5		#027-032	027-033				
ROWTB2	FDBD		#027-033	027-034				
ROWTB3	FDC5		#027-034	027-035				
ROWTB4	FDCD		#027-035	027-036				
ROWTB5	FDD5		#027-036	027-037				
ROWTB6	FDDD		#027-037	027-038				
ROWTB7	FDE5		#027-038	027-039	027-041			
RSPBFL	0018		#035-011	036-015				
RSPBLIF	FEA8		#035-009	036-015	119-012	120-051	156-037	161-044
			163-030	165-009	165-020	167-003	167-014	167-030
			167-046	169-004	170-017	171-031	172-004	172-031
			174-017	174-028	175-004	175-011	179-025	180-075
			180-095	185-091	185-104	185-114	188-006	190-010
			195-080	213-034	266-019	266-028	267-027	275-028
			278-023	347-009	347-026	371-025	384-037	
SCONF1	FE84		#031-003	031-004	119-013	374-050		
SCONF2	FE85		#031-004	031-006	233-018	236-007	236-022	251-006
			297-030	299-004	300-003	301-018	375-005	
SEQBAS	0033	*	#034-050					
SEQFLG	0003		#031-037	167-016	167-022	167-035	171-037	172-043
			184-075	243-033	299-032			
SINFLG	0000		#032-068	175-020	236-045			
SPCBRK	0008	*	#007-021					

4,292,666

571

572

SPCDTR	0002	#007-023	345-003	391-031			
SPCEH	0080	#007-017	007-042				
SPCER	0010	#007-020	362-014	387-025	387-029	390-049	391-031
		393-103	399-018	400-055	402-050		
SPCIR	0040	#007-018	103-020	387-025			
SPCRE	0004	#007-022	007-039	362-014	391-031	399-018	
SPORTS	0020	#007-019	007-039	362-014	399-018		
SPOTE	0001	#007-024	007-039	007-042	362-014	399-018	
SPDFLD	0050	#031-030	267-013				
SPLRFL	0040	#025-027	025-034	077-008	079-006		
SPLBLK	FD26	#025-017	025-019	063-007	077-009	079-004	079-017
		081-001					
SPLBUF	FF00	#025-034	025-035	077-007			
SPLDIS	041D	099-042	#100-016	100-018			
SPLDIX	0002	100-016	#100-018				
SPLINI	0200	058-005	#077-007	267-022			
SPLIX	022E	079-009	#079-024				
SPLI10	021D	079-007	#079-011				
SPLI15	021E	#079-013	079-022				
SPLLED	0417	099-016	#100-004	100-006			
SPLLEX	0002	100-004	#100-006				
SPLPWR	041A	099-021	#100-010	100-012			
SPLPWX	0002	100-010	#100-012				
SPLRX	0244	081-003	#081-014				
SPLR10	0243	081-006	#081-012				
SPMBRF	0002	#007-032	007-036				
SPMEVN	0020	#007-029	007-036				
SPMLEN	0000	#007-031	007-036				
SPMPAR	0010	#007-030	007-036				
SPMSTP	0000	#007-028	007-036				
SPDOLI	020D	#079-001	099-028				
SPDOLR	022F	063-009	#081-001				
SPSDSR	0080	#007-006	057-013	352-018	387-036	389-010	401-028
SPSFE	0020	#007-008	105-010	389-019			
SPSDE	0010	#007-009	105-020	389-029			
SPSPE	0008	#007-010	105-010	389-019			
SPSRRY	0002	#007-012	083-012	105-004	389-013		
SPSSYN	0040 *	#007-007					
SPSTE	0004 *	#007-011					
SPSTRY	0001	#007-013	083-012	110-005	401-032		
SP1CTL	003A	#002-020	103-016	103-018	103-021	103-023	103-025
		107-048	110-016	345-004			
SP1IN	003B	#002-021	106-001	109-099			
SP1OUT	003B	#002-022	110-012				
SP1STA	003A	#002-019	083-009	352-017			
SP2CTL	003C	#002-024	057-012	362-015	362-022	387-021	387-023
		387-026	387-028	387-030	387-035	390-050	391-032
		393-104	399-019	400-056	402-051		
SP2IN	003D	#002-025	389-038				
SP2OUT	003D	#002-026	401-038				
SP2STA	003C	#002-023	389-008	401-026			
SRCHST	FF0B	#037-051	182-030	185-095	185-099	194-039	
SRGFLR	0001	#032-069	237-057				
STACK	FD8F	#023-007	024-005	054-006	063-005		
STACKL	0040	#023-005	023-007				
STEPCL	2542	335-061	338-067	#339-017			
STPMSK	001F	#034-052	168-057				
STPNUM	FE8A	#031-009	031-011	121-005	175-013	194-051	214-059
		278-012	303-017	303-021			
SUBFLG	0001	#031-046	174-034	259-038			
SUPOPT	0007	270-050	271-005	#277-046			
SYSENH	0002	#033-022	233-019	236-008	251-007	299-005	300-004
SYSEQL	0000 *	#033-046					
SYSRUN	0080	#033-027	273-038				
SYSSTP	0010	#033-030	273-017				
SYS064	0010	#033-019	236-023	375-008			
SYS128	0020	#033-018	236-023	375-012			
SYS192	0040	#033-017	236-023	375-016			
SYS256	0080	#033-016	236-023	375-020			
SY0256	0009	#033-009	374-054				
SY0512	0010	#033-008	374-050				
SY1024	0020	#033-007	374-060				
SY2048	0040	#033-006	374-066				
SY4096	0080	#033-005	374-066				

TEMP	FEC2 *	#036-027					
TMRACK	FD90	#024-007	024-008	345-002	345-008	347-003	
TMRBEP	FD8F	#024-006	024-007	065-029			
TMRONT	0006	#024-016	095-002	097-029			
TMRDIS	FD94	#024-011	024-014	024-016	121-013	267-040	269-024
TMRDSE	03D2	097-016	#098-010				
TMRERR	FD93	#024-010	024-011	099-037	113-024	263-028	
TMRLED	FD91	#024-008	024-009	269-026	286-048	327-019	
TMRPWR	FD92	#024-009	024-010	269-025	282-030	327-017	
TMRTAB	FD8F	#024-005	024-006	024-016	095-003	097-001	
TMRTBX	FD95	#024-014	025-017				
TOPID	2D05	365-039	381-077	#412-087			
TOPIDX	2D1F	412-097	#413-113				
TOPIDZ	2D0B	#412-095	412-104				
TYPE01	0001	#360-033	396-036				
TYPE1	0003	#360-027	362-024	373-030	377-087	380-025	384-011
TYPE2	0002	#360-029	373-032	377-089			
TYPE3	0001	#360-031	366-026	368-009	368-043	369-073	
UBFCH	0156	#072-001	081-002	081-005	110-011	126-002	347-008
		347-012	347-018				
UBFCHX	0180	072-009	#072-035				
UBFCH1	0166	072-006	#072-011				
UNFORM	2BA6	380-039	384-032	#396-001			
UNFRM2	2BD6	#397-044	397-065				
UNFRM5	2BEE	397-052	#397-067				
VALERR	29C0	377-094	378-109	#378-125			
VALOAD	2999	#377-082	381-057				
VALOD5	29B2	378-101	#378-107				
VERBUF	FEF3	#036-041	384-031				
VERCHK	2AB0	384-030	385-050	#386-027			
VERCK5	2AC3	385-075	#386-039				
VERCXT	2AC6	386-032	386-035	#386-042			
VERIFY	2A54	272-017	#384-001				
VEROK	2AA7 *	#385-080					
VER03	2A60	#384-014	385-069				
VER05	2A63	#384-017	385-061				
VER10	2A86	#385-047	385-054				
VER30	2A9B	384-021	#385-065				

000 ERRORS

It is also to be understood that the following claims are intended to cover all of the generic and specific features of the invention herein described, and all statements of the scope of the invention which, as a matter of language, might be said to fall therebetween.

Having described the invention, what is claimed is:

1. An improved programmable controller of the type having a mainframe including a central processing unit, associated electronics and memory for solving a user control program, an input/output system for communicating input data from external devices to the mainframe and for transferring output data from the mainframe to the external devices for control thereof in response to the control program, and a programming panel for programming, monitoring and displaying the user selected control program, wherein the improvement comprises:

(A) an improved programming panel having first means for allowing the user to generate a control program in a network format, each network comprising up to I rows and J columns, where I and J are positive integers each greater than one, each network comprising up to $I \times J$ nodes where the location of each node is N_{ij} , where $i=1, 2, \dots, I, j=1, 2, \dots, J$, and where each of at least some of these nodes are designatable by the user as representing an electrical circuit element that can reference other nodes, and having a power output status that is a function of the input power status to the node in combination with the conductivity status of the element, and having second means for allowing the user to generate vertical interconnections between the output of node N_{ij} with the output

of node $N_{i-1,j}$ or with output of node $N_{i+1,j}$, for all existent nodes $N_{i-1,j}$ and $N_{i+1,j}$, where "or" is used in the inclusive sense, and wherein the power input status from node N_{ij} to node N_{ij+1} is represented by the following Boolean equation:

$$P_{IN_{ij}} = P_{OUT_{i,j}} + P_{VU_{i,j}} + P_{VD_{i,j}} \quad (1)$$

where

$$P_{OUT_{i,j}} = P_{IN_{i,j-1}} \cdot C_{i,j} \quad (2)$$

where $C_{i,j}$ is the conductivity state of node $N_{i,j}$, where

$$P_{VU_{i,j}} = P_{IN_{i+1,j}} \cdot C_{U_{i,j}} \quad (3)$$

where $C_{U_{i,j}}$ is the connectivity state between the output of node $N_{i,j}$ and node $N_{i+1,j}$, where

$$P_{VD_{i,j}} = P_{IN_{i-1,j}} \cdot C_{D_{i,j}} \quad (4)$$

where $C_{D_{i,j}}$ is the connectivity state between the output of node $N_{i,j}$ and node $N_{i-1,j}$, and where $P_{IN_{i,0}}$ is equal to logic true; and

(B) an improved mainframe having means for simultaneously solving equation (1) for each node in each column of the user generated network on a column by column basis.

2. An improved programmable controller of the type having a mainframe including a central processing unit, associated electronics and memory for solving a user control program, and input/output system for commu-

nicating input data from external devices to the mainframe and for transferring output data from the mainframe to the external devices for control thereof in response to the control program, and a programming panel for programming, monitoring and displaying the user selected control program, wherein the improvement comprises:

(A) an improved programming panel having first means for allowing the user to generate a control program in a network format, each network comprising up to I rows and J columns, where I and J are positive integers each greater than one, each network comprising up to $I \times J$ nodes where the location of each node is N_{ij} , where $i=1,2,\dots,I$, $j=1,2,\dots,J$, and where each of at least some of these nodes are designatable by the user as representing an electrical circuit element that can reference other nodes, and having a power output status that is a function of the input power status to the node in combination with the conductivity status of the element, and having second means for allowing the user to generate vertical interconnections between the output of node N_{ij} with the output of node $N_{i-1,j}$ or with output of node $N_{i+1,j}$, for all existent nodes $N_{i-1,j}$ and $N_{i+1,j}$, where "or" is used in the inclusive sense, and wherein the power input status from node N_{ij} is represented by the following Boolean equation:

$$P_{IN_{ij}} = P_{OUT_{i,j}} + P_{OUT_{i-1,j}} C_{V_{i,j}} + P_{OUT_{i-2,j}} C_{V_{i-1,j}} C_{V_{i,j}} + \dots + P_{OUT_{i,j}} C_{V_{2,j}} C_{V_{3,j}} \dots C_{V_{i,j}} + P_{OUT_{i+1,j}} C_{V_{i+1,j}} + P_{OUT_{i+2,j}} C_{V_{i+2,j}} C_{V_{i+1,j}} + \dots + P_{OUT_{i,j}} C_{V_{i,j}} C_{V_{i-1,j}} \dots C_{V_{i+1,j}} \quad (1)$$

where

$$P_{OUT_{i,j}} = P_{IN_{i-1,j-1}} C_{i,j} \quad (2)$$

where $C_{i,j}$ is the conductivity state of node $N_{i,j}$, where $C_{V_{i,j}}$ is the connectivity state between node $N_{i,j}$ and node $N_{i-1,j}$, and where $P_{IN_{i,0}}$ is equal to logic true; and

(B) an improved mainframe having means for simultaneously solving equation (1) for each node in each column of the user generated network on a column by column basis.

3. An improved programmable controller of the type having a mainframe including a central processing unit, associated electronics and memory for solving a user control program comprising a plurality of nodes in a network format, the nodes representing user selectable circuit elements that can reference other nodes, an input/output system for communicating input data from external devices to the mainframe and for transferring output data from the mainframe to the external devices for control thereof in response to the control program, and a programming panel for programming, monitoring and displaying on a cathode ray tube (CRT) at least a portion of the user selected control program, wherein the improvement comprises an improved programming panel and mainframe each having interacting means for performing one of a plurality of search functions containing one or more search parameters as designated by the user so as to display on the programming panel CRT the control program network satisfying the search parameters, wherein the search parameters that can be designated by the user include the searching for the first node of a network, the searching for the first occurrence of a particular contact type of a particular circuit

element, the searching for the first occurrence of a particular reference number, the searching for the first occurrence of a particular circuit element having a particular reference number, and the searching for the first occurrence of a particular node.

4. An improved programmable controller as defined in claim 3, wherein the improved programming panel has means allowing the user to generate vertical interconnections between the output of two nodes in adjacent rows of the network, and wherein the search parameters that can be designated by the user include the searching for the first occurrence of a vertical connector, the searching for the first occurrence of a particular contact type of a particular circuit element having a vertical connector and the searching for the first occurrence of a particular reference number having a vertical connector.

5. An improved controller of the type having a mainframe including a central processing unit, associated electronics and memory for solving a user control program comprising a plurality of nodes, an input/output system for communicating input data from external devices to the mainframe and for transferring output data from the mainframe to the external devices for control thereof in response to the control program, and a programming panel, including a screen display with a refresh rate for the screen information, for programming, monitoring and displaying at least a portion of the user selected control program, wherein the improvement comprises an improved programming panel and mainframe wherein the programming panel has a visual screen display and a cursor which can be moved from node to node of the control program as displayed on the visual display, and wherein the programming panel further includes means for indicating to a user the real-time power status of the node upon which the cursor is placed independent of the screen refresh rate.

6. An improved programmable controller as defined in claim 5, wherein the real-time power status indicating means is a light and means for energizing the light in response to the real-time power status of the node.

7. An improved programmable controller as defined in claim 1, wherein the mainframe means for solving equation No. 1 comprises (a) a first series of logic gates, the output of the first logic gates representing the Boolean state of P_{IN} simultaneously for each row i and sequentially for each column j , (b) a second series of logic gates, each second series gate receiving as one input the P_{OUT} state for the corresponding node in the network, the output of each second series gate connected as a first input to the corresponding first series gate, (c) a third series of logic gates, each third series gate receiving as one input the connectivity state between the present node and the node in the previous row and the same column, and as a second input the output of the second series gate in the previous row, if present, the output of each third series gate connected as a second input to the next row second series gate, (d) a fourth series of logic gates, each fourth series gate receiving as one input the P_{OUT} state for the corresponding node in the network, the output of each fourth series gate connected as a second input to the corresponding first series gate, and (e) a fifth series of logic gates, each fifth series gate receiving as one input the connectivity state between the present node and the node in the next row and the same column, and as a second input the output of the fourth series gate for the next row, the output of each fifth series gate connected to the corresponding fourth series gate as a second input

8. An improved programmable controller as defined in claim 2, wherein the mainframe means for solving equation 1 is performed by a group of logic gates which solve the P_{IN} status simultaneously for each row i and in a sequential manner for each column j .

9. An improved programmable controller of the type having a mainframe including a central processing unit, associated electronics and memory for solving a user control program comprising a plurality of nodes in a network format, the nodes representing user selectable circuit elements that can reference other nodes, an input/output system for communicating input data from external devices to the mainframe and for transferring output data from the mainframe to the external devices for control thereof in response to the control program, and a programming panel for programming, monitoring and displaying on a screen at least a portion of the user selected control program, wherein the improvement comprises an improved programming panel and mainframe each having interacting means for performing one or more search functions containing one or more search parameters as designated by the user so as to display on the programming panel screen the control program network satisfying the search parameters, wherein the search parameters that can be designated by the user include one or more of the following: the searching for the first node of a network, the searching for the first occurrence of a particular contact type of a particular circuit element, the searching for the first occurrence of a particular reference number, the searching for the first occurrence of a particular circuit element having a particular reference number, and the searching for the first occurrence of a particular node.

10. An improved programmable controller as defined in claim 9, wherein the improved programming panel has means allowing the user to generate vertical interconnections between the output of two nodes in adjacent rows of the network, and wherein the search parameters that can be designated by the user include one or more of the following: the searching for the first occurrence of a vertical connector, the searching for the first occurrence of a particular contact type of a particular circuit element having a vertical connector, and the searching for the first occurrence of a particular reference number having a vertical connector.

11. An improved programmable controller as defined in claims 9 or 10, wherein the means for performing one or more search functions examines the entire user control program to find whether the search parameters are satisfied and, if the search parameters are satisfied, displaying on the screen the plurality of nodes in the network format in which a specific node satisfies the search parameters and, if the search parameters are not satisfied displaying information on the screen indicating the same.

12. An improved programmable controller as defined in claim 11, wherein the means for performing the search function further generate a cursor on the screen at the specific node satisfying the search parameters.

13. An improved programmable controller as defined in claim 12, wherein the means for performing the search function if the search parameters are satisfied further displays on the screen identifying data of the specific plurality of nodes in a network format in which the search parameters have been satisfied.

14. An improved programmable controller of the type having a mainframe including a central processing unit, associated electronics and memory for solving a

user control program, an input/output system for communicating input data from external devices to the mainframe and for transferring output data from the mainframe to the external devices for control thereof in response to the control program, and a programming panel for programming, monitoring and displaying the user selected control program, wherein the improvement comprises:

(A) an improved mainframe having means for solving calculate functions in the user control program wherein at least some of the calculate functions have more than one discrete output designating information concerning the result of the calculate function; and

(B) An improved programming panel comprising:

(1) a screen for viewing at least a portion of the control program; and

(2) means for allowing the user to generate the control program in a network format, each network comprising up to I rows and J columns, where I and J are positive integers each greater than one, each network comprising up to $I \times J$ nodes where the location of each node is N_{ij} , where $i=1, 2, \dots, I, j=1, 2, \dots, J$, and where each of at least some of these nodes are designatable by the user as representing an electrical circuit element that can reference other nodes, and having a power output status that is a function of the input power status to the node in combination with the conductivity status of the element, and wherein the calculate functions can be displayed in nodes on the screen.

15. An improved programmable controller as defined in claim 14, wherein the means for solving calculate functions includes means for solving the subtract function with three discrete outputs, a first of the outputs having a first state indicating when the minuend is less than or equal to the subtrahend and a second state indicating when the minuend is greater than the subtrahend, a second output having a first state indicating when the minuend is not equal to the subtrahend and a second state indicating when the minuend is equal to the subtrahend, and a third output having a first state indicating when the minuend is greater than or equal to the subtrahend and a second state when the minuend is less than the subtrahend.

16. An improved programmable controller as defined in claim 14, wherein the means for solving calculate functions includes means for solving the divide function with three discrete outputs, a first of the outputs having a first state indicating when the division to be performed is not possible and a second state indicating when the division to be performed is possible, a second output having a first state indicating when the divisor if multiplied by a number equal to the total field of the answer is less than the dividend, and a second state indicating when the dividend is an overflow, and a third output having a first state indicating when the divisor is equal to zero and a second state indicating when the divisor is not equal to zero.

17. An improved programmable controller as defined in claim 14, 15, or 16 wherein the mainframe has means for allowing any or all calculate function discrete outputs to be referenced by electrical circuit elements in other nodes.

18. An improved programmable controller as defined in claim 14, 15, or 16 wherein the mainframe has means for allowing any or all calculate function discrete outputs to represent output data from the mainframe.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 4,292,666

Page 1 of 2

DATED : Sept. 29, 1981

INVENTOR(S) : Lawrence W. Hill et al.

It is certified that error appears in the above-identified patent and that said Letters Patent are hereby corrected as shown below:

Column 45 Line 5 insert the following:

where

$$(2) \quad P_{OUT_{i,j}} = P_{IN_{i,j-1}} \cdot C_{i,j}$$

where $C_{i,j}$ is the conductivity state
of node $N_{i,j}$,

and where $C_{V_{i,j}}$ is the connectivity

state between node $N_{i,j}$ and node $N_{i-1,j}$.

Column 575 Line 36 remove the following:

$$P_{OUT_{i,j}} = P_{IN_{i[-1],j-1}} \cdot C_{i,j}$$

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 4,292,666

Page 2 of 2

DATED : September 29, 1981

INVENTOR(S) : Lawrence W. Hill et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Insert the following:

$$P_{OUT_{i,j}} = P_{IN_{i,j-1}} \cdot C_{i,j}$$

Signed and Sealed this

Twenty-eighth **Day of** *September 1982*

[SEAL]

Attest:

Attesting Officer

GERALD J. MOSSINGHOFF

Commissioner of Patents and Trademarks