(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 193 576 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.04.2002 Bulletin 2002/14

(51) Int Cl.[7]: **G05B 9/03**, G06F 11/20

(21) Application number: **01125808.4**

(22) Date of filing: **15.03.1993**

(84) Designated Contracting States:
BE DE ES FR GB IT NL

(30) Priority: **31.03.1992 US 864931**

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
98109082.2 / 0 869 415
93908318.4 / 0 631 673

(71) Applicant: **THE DOW CHEMICAL COMPANY**
Midland, Michigan 48674 (US)

(72) Inventors:
- **Glaser, Robert S.**
  Midland, MI 48640 (US)
- **Hoy, Robert S.**
  Midland, MI 48640 (US)
- ***Fernandez, Paul G.***
  Midland, MI 48640 (US)
- **Gral, Timothy J.**
  Saginaw, MI 48603 (US)
- **Gaudreau, Dean W.**
  Midland, MI 48640 (US)
- **Hozeska, Robert J.**
  Saginaw, MI 48603 (US)
- **Grinwis, Donald J.**
  Midland MI 48642 (US)
- ***Gavit, Gregory J.***
  Freeland, MI 48623 (US)
- **Sheehan, Joseph Jr.**
  Lake Jackson TX 77566 (US)
- **Thomas, Lowell V.**
  Midland, MI 48642 (US)

(74) Representative: **Jordan, Volker, Dr. et al**
Weickmann & Weickmann
Patentanwälte
Postfach 860 820
81635 München (DE)

Remarks:
This application was filed on 29 - 10 - 2001 as a divisional application to the application mentioned under INID code 62.

(54) Process control interface system having triply redundant remote field units

(57) A process control interface system (10) having a network of distributed triply redundant input/output field computer units (12). The system includes a plurality of selfcontained remotely located triply redundant field computer units (12) connected to decision making redundant process control computers (14) through a bi-directional communication network having at least two concurrently active communication channels (46, 48). Each of the field computer units include a set of at least three redundant field computers (92, 94, 96) for arbitrating both input and output signals. The field computer units also include individual abort circuits (510, 606) for *each output signal to be transmitted to a device* (84, 86) which affects the operation of the physical process. These abort circuits effectively enforce the output value signals arbitrated independently through each of the three redundant field computers using a voting procedure.

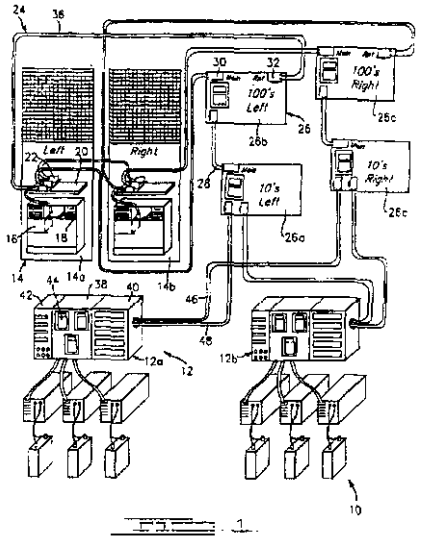


FIG. 1

EP 1 193 576 A2

Description

BACKGROUND OF THE INVENTION

[0001] The present invention generally relates to the interface between a process control computer and its remotely located field instrumentation. More specifically, the present invention relates to a process control interface system which is comprised of a distributed network of triply redundant remote field units that communicate with redundant process control computers over redundant fiber optic paths.

[0002] One of the most difficult and elusive goals to achieve in the design of any automated process control system is to provide an accurate, fast and yet highly reliable control system which is capable of withstanding the rugged demands of controlling a physical process non-stop for years at a time, if possible. This is particularly true for the process control applications in a chemical plant where the cost of shutting down a complex large-scale process for computer system repairs may be enormous due to the time, effort and waste incurred in attempting to bring such a process back on line.

[0003] In order to achieve maximum economic efficiency and optimum product quality, the demands for more comprehensive process control automation have continued to increase in both quantity and sophistication. As the reliance on computer-based control for the operation of a chemical process increases, it is clear that a number of computers are required to work together in order to accomplish all of the desired control tasks. This, of course, adds further complexity to a control system for which maximum fault tolerance is desired.

[0004] In order to increase the reliability of a process control computer system, many attempts have been made to provide a backup computer for one or more of the computers being used to actively control the process. However, a rapid hand-off of control from an active computer to a backup computer is difficult to achieve if the goal is to provide a seamless or transparent transfer to the devices which affect the operation of the physical process. Additionally, the conditions under which a transfer of control should be made may be complex and consume needed processor time during normal operations.

[0005] Another approach to this problem is to provide triple redundancy with three actively operating computers. While the provision of three computer processors certainly increases the overall cost of the control system, it does permit the use of 'majority voting" for decision making. The benefit of majority voting not only adds to the ability of the computer system to withstand a fault in one of the computers, it also helps to ensure that the decisions being made are accurate. In other words, the agreement of two out of three computers on any particular decision increases the likelihood that the decision is ultimately correct.

[0006] Nevertheless, even when triply redundant control is found to be desirable, a myriad of design problems must first be confronted in order to achieve a truly effective triply redundant control system, including the handling of internal failures within different areas of the triply redundant control system. While there have been a number of attempts to appropriately manage the interrelationships between a set of three or more computers, there is still considerable room for advancement in this art, particularly as it relates to large scale chemical process control applications.

[0007] Accordingly, it is a principal objective of the present invention to provide a distributed network of triply redundant field computer units which communicate with redundant process control computers to maximize both accuracy and the overall system's tolerance to faults in the process control system that could affect the physical process being controlled.

[0008] It is another objective of the present invention to provide a distributed network of triply redundant field computer units which enables broadcast downloading of updated software to each of these units without affecting the process being continuously controlled.

[0009] It is a further objective of the present invention to provide a triply redundant field computer unit which permits circuit boards in one of the computers contained in the unit to be replaced without affecting the process being controlled or requiring control to be forced to one or the other of the remaining computers.

[0010] It is an additional objective of the present invention to provide a triply redundant field control unit which enables a unique arbitration process of field inputs and outputs to be achieved.

[0011] It is also an objective of the present invention to provide a triply redundant field computer unit which is capable of automatically aborting potentially erroneous output signals.

[0012] It is yet another objective of the present invention to provide a triply redundant field computer unit which enables any two computers contained in the unit to temporarily reset, and if necessary, more permanently reset the remaining computer.

[0013] It is still an additional objective of the present invention to provide a triply redundant field computer unit which includes one or more "smart" multi-function input circuits for interpreting raw sensor information and one or more "smart" output circuits for independently determining the manner in which a desired output value is achieved.

[0014] It is still a further objective of the present invention to provide a method of testing both digital and analog output circuits which is non-intrusive to the process being continuously controlled.

[0015] It is yet another objective of the present invention to provide a triply redundant field computer unit which includes a high current output power supply circuit and a battery backup that may be periodically tested under load conditions.

SUMMARY OF THE INVENTION

[0016] To achieve the foregoing objectives, the present invention provides a plurality of self-contained remotely located triply redundant field computer units which are connected to decision making redundant process control computers through a bi-directional communication network having at least two concurrently active communication channels. Each of the field computer units include a set of at least three redundant field computers for converting raw analog and digital input signals into arbitrated input value signals at predetermined times. The input arbitration method provided by the redundant field computers enables a plurality of selectable default input conditions for each input signal, such as select HIGH and select LCW, in the event that a majority agreement cannot be reached among valid input signals.

[0017] Messages containing these arbitrated input value signals are transmitted to the redundant process control computers from each of the field computer units over a multilevel fiber optic network. The fiber optic network is designed to permit substantial communication testing, and enable the direction of signal transmission on the primary level of signal distribution to be reversed in the event of a communication fault. Once the appropriate process control decisions are made, the field computer units receive output value signals from the redundant process control computers over the fiber optic network.

[0018] The field computer units also include a set of individual abort circuits for each output signal to be transmitted to a device which affects the operation of the physical process. These abort circuits effectively enforce the output value signals arbitrated independently through each of the three redundant field computers. The software arbitration process involves using a tiered voting procedure which includes a plurality of selectable default output conditions, such as fail SAFE and fail LAST. Each of the default input and output conditions are determined through software implementation, such as at the redundant process control computers. With the software implementation according to the present invention, each of the default input and output conditions may be rapidly changed in response to changing process conditions.

[0019] Additional features and advantages of the present invention will become more fully apparent from a reading of the detailed description of the preferred embodiment and the accompanying drawings in which:

BRIEF DESCRIPTION OF THE DRAWINGS

[0020]

Figure 1 is a diagrammatic view of a process control interface system according to the present invention.

Figure 2 is a diagrammatic representation of a portion of the fiber optic communication network shown in Figure 1 which particularly illustrates the multi-function breakout circuits of the network.

Figure 3 is a block diagram of the process control interface system shown in Figure 1.

Figure 4 is a block diagram which illustrates the flow of data communication in the process control interface system of Figure 1.

Figure 5 is a perspective view of the processor chassis for the triply redundant field computer unit shown in Figure 1.

Figures 6A-6U comprise a schematic diagram for one of the triply redundant field computers shown in Figure 5.

Figures 7A-7C comprise a schematic diagram for a smart serial input circuit according to the present invention.

Figures 7D-7M comprise a series of flow charts associated with the operation of the smart serial input circuit of Figures 7A-7C.

Figures 8A-8E comprise a schematic diagram for a multiple-mode pulse input circuit according to the present invention.

Figures 8F-8Q comprise a series of flow charts associated with the operation of the multiple-mode pulse input circuit of Figures 8A-8E.

Figures 9A-9D comprise a schematic diagram for resistance measurement circuit according to the present invention.

Figure 10A is a block diagram of a portion of the triply redundant field computer which particularly illustrates the abort circuits for the digital output signals. Figure 10B is a similar block diagram which particularly illustrates the abort circuits for the analog output signals.

Figures 11A-11C comprise a schematic diagram for a digital output circuit capable of non-intrusive testing.

Figures 12A-12F comprise a schematic diagram for a smart analog output circuit according to the present invention.

Figures 13A-13D comprise a schematic diagram for a network controller circuit according to the present invention.

Figures 14A-14E comprise a schematic diagram of a breakout serial communication circuit shown in Figure 4.

Figure 15A comprises a schematic diagram of a fiber optic receiver circuit employed in the network shown in Figure 1. Figure 15B comprises a schematic diagram of a fiber optic transmitter circuit employed in the network shown in Figure 1.

Figures 16A-16G comprise a schematic diagram of a power supply circuit for the triply redundant field computer unit.

Figures 17A-17I comprise a set of flow charts which illustrate the arbitration methods according to the present invention for digital input and output values.

Figures 18A-18T comprise a set of flow charts which illustrate the arbitration methods according to the present invention for analog input and output values.

Figures 19A-19M comprise a set of flow charts which illustrate the method of non-intrusively testing the digital output circuits shown in Figure 10A.

Figures 20A-20V comprise a set of flow charts which illustrate the method of setting the analog abort switches and conducting non-intrusive testing of the analog output circuits shown in Figure 10B by a field I/O computer controller.

Figures 21A-21S comprise a set of flow charts for the software which controls the operations of each of the smart analog output circuits shown in Figure 10B.

Figures 22A-22R comprise a set of flow charts which illustrate the output control routine shown in Figure 21B.

Figures 23A-23I comprise a set of flow charts which illustrate the non-intrusive testing method performed by the analog output circuits.

Figures 24A-24G, 25A-25Z, 26A-26Z and 27A-27M comprise a set of flow charts which illustrate the method of downloading software in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0021] Referring to Figure 1, a process control interface system 10 having a network of distributed triply redundant input/output field computer units 12 is shown. In this regard, it should be appreciated that Figure 1 includes only two field computer units 12 for purposes of illustration, and that the interface system 10 has the capability of handling a significant number of field computer units. For example, in one embodiment according to the present invention, the interface system 10 is capable of utilizing a maximum of sixty four field computer units 12.

[0022] The field computer units 12 serve as the primary interface between the field instrumentation and a centralized process control computer system. In the embodiment discussed herein, the centralized process control computer system is generally comprised of a pair of redundant process control computers, which are generically referred to by reference number 14. While the redundancy of two concurrently operating process control computers has certain fault tolerance advantages over a single decision making process control computer, it should be understood that the principles of the present invention are not limited to any particular process control computer design or configuration. Thus, for example, it may be desirable to employ only one or even three process control computers in the place of the two process control computers 14 shown in Figure 1 under the appropriate circumstances.

[0023] In the present embodiment, the redundant process control computers 14 preferably operate concurrently on all of the signals transmitted from the field computer units 12. In other words, each of the process control computers 14 are capable of making independent decisions based upon the data received by these redundant computers from the field computer units 12. The decisions made by the process control computers 14 determine the output signal values which are ultimately directed to specific output devices (for example, valves, pump motors and reactor heaters) by the appropriate field computer units 12. While the output signal values are preferably reconciled at least to some extent between the two process control computers 14 before the transmission of these signals to the proper field computer units 12, it should be understood that two independent sets of output signal values could be communicated to the field computer units. In this regard, the input values received from a field computer unit 12 could be arbitrated at the process control computers 14, which should make it unnecessary to reconcile or arbitrate output values. This is because both of the process control computers would then be working from the same set of arbitrated input values.

[0024] As an example of a preferred form of possible value reconciliation, corresponding output value tables in each of the process control computers 14 could be compared during a preset time period, and one of the values could be chosen for each output value signal to be transmitted to the field computer units 12. This selection of output control values could be made on a suitable criteria to the process being controlled, such as the use of the value determined by the Left process control computer 14a when the value determined by the Right process control computer 14b is within a certain predetermined percentage limit (for example, 2.5%). Otherwise, the distinct output control values of both the Left and Right process control computers 14 could each be sent to the proper field computer units 12 when these values are found to be outside the predetermined percentage limit. Alternatively, the selection of different output control values from the Left and Right process control computers could be made on the basis of a software implemented preference. Thus, for example, under certain process conditions, it may be considered more appropriate to select either the high or low value for transmission to the field computer unit 12, regardless of whether the value was determined

by the Left or Right process control computer.

[0025] Each of the process control computers 14 preferably include a network controller 16, a debug panel 18 for the network controller, and a tray 20 upon which to support the fiber mount boards 22 to which various fiber optic conduits 24 are connected. As will be more fully discussed in connection with Figures 13A-13D, the network controller 16 is used to direct communication traffic both to and from the process control computers 14 via the fiber optic conduits 24. The debug panel 18 includes both a display and a set of numeric/function keys in order to provide a window into specific operations of the network controller 16.

[0026] As will be discussed more fully in connection with Figures 15A-15B, each of the fiber mount boards 22 contain the transmission circuit required to convert electrical signals to optical signals, as well as the receiver circuit required to convert optical signals to electrical signals. As for the fiber optic conduits themselves, these conventional light conductors may be made of either glass or plastic. However, it should be appreciated that the use of glass fibers permit significantly greater transmission distances to be achieved. While it is preferred that fiber optic conduits be employed to convey messages between the field computer units 12 and the process control computers 14 for their high speed throughput and substantial security, it should be understood that other suitable communication mediums could be used in the appropriate applications.

[0027] As illustrated in Figure 1, the fiber optic network which connects each of the process control computers 14 with each of the field computer units 12 includes a set of breakout circuits 26 for each of the redundant process control computers. As will be more fully discussed in connection with Figures 14A-14E, each of the breakout circuits are designed to facilitate multiplexed serial communication between a plurality of field computer units 12 and one of the redundant process control computers 14.

[0028] Thus, for example, the breakout circuit 26a is configured to provide multiplexed serial communication between the Left process control computer 14a and up to ten field computer units 12. The breakout circuit 26a is in turn connected via fiber optic conduits 28 to the breakout circuit 26b which is configured to provide multiplexed serial communication between the Left process control computer 14a and several groups of field computer units 12. In this regard, the breakout circuit 26a represents one group of field computer units 12 to the breakout circuit 26b.

[0029] It should be noted that the breakout circuit 26b is connected to the Left process control computer 14a through both a main port 30 and a repeat port 32. Specifically, the fiber optic conduits 34 provide a connection between the main port 30 of the breakout circuit 26b and the Left process control computer 14a, while the fiber optic conduits 36 provide a connection between the repeat port 32 of the breakout circuit and the Left process control computer. The fiber optic conduits 34-36 thereby form a ring around the Left process control computer 14a and the breakout circuit 26b. As will be discussed in more detail below, the breakout circuits are designed to be multi-functional in that they have the capability of not only multiplexing communication, but also conveying messages that are received at the main port 30 out to the repeat port 32. This ability to repeat messages also enables the network to extend for great distances, as will be described in connection with Figure 4.

[0030] Additionally, the network controller 16 also has the ability to direct that messages be transmitted from the process control computer 14a to the repeat port 32 of the breakout circuit 26b. This important feature permits communication to continue without significant interruption in the event that communication cannot proceed through the fiber optic conduits 34. In other words, the direction of signal communication on the ring between the process control computer 14b and the breakout circuit 26b may be reversed in the event of a communication fault.

[0031] Additionally, it should be appreciated through Figure 1 that a substantially identical communication network between the Right process control computer 14b and each of the field computer units 12 is provided by the breakout circuits 26c-26d and their associated fiber optic conduits. Thus, it should be appreciated that the capability to change the direction of signal flow at the primary (or first) level of signal distribution is provided for each of the network communication rings connected to the Left and Right process control computers through their respective network controllers 16.

[0032] In accordance with the present invention, the integrity of each of these network communication rings is tested before any signals are transmitted to the field computer units 12. Indeed, it may be possible with the present invention for the integrity of the entire network to be periodically tested as a preliminary part of the signal communication process. Thus, for example, with an overall process and communication cycle of one second, the integrity of at least the primary network communication rings is preferably tested each second, as this integrity check will help to avoid wasted or incomplete communication efforts.

[0033] Specifically with reference to Figure 1, a synchronization pulse (for example, a 1 byte message) is transmitted from the network controller 16 to, and around, the ring formed by fiber optic conduits 34, breakout circuit 26b and fiber optic conduits 36. The purpose of this synchronization pulse is to permit the Left process control computer to determine whether or not signals may be successfully transmitted in this counter-clockwise direction. In this regard, a reception of the synchronization pulse from the repeat port 32 of the breakout circuit 26b via fiber optic conduits 36 within a predetermined amount of time (for example, a timeout of 300 micro seconds) will indicate that there are no breaks in the communication path or circuit faults which would interfere with the proper transmission of signals on this portion

of the network. A similar synchronization pulse will then be transmitted from the network controller 16 in the opposite direction, namely around the ring formed by fiber optic conduits 36, breakout circuit 26b, and fiber optic conduits 34, to determine whether or not signals may be successfully transmitted in this clockwise direction.

[0034] As will be more fully appreciated from Figure 2, it will be seen that a plurality of breakout circuits 26 may be connected in series to provide the primary level of signal distribution for the network. In this regard, the successful circulation of the first synchronization pulse around the ring shown will establish that each of the breakout circuits 26e-26k were able to receive and repeat this pulse. More specifically, each of the breakout circuits 26 preferably respond to the synchronization pulse by transmitting a signal which identifies itself to the network controller 16. However, if for example, breakout circuit 26k did not repeat this synchronization pulse back to the network controller 16, then the subsequent transmission of a synchronization pulse in the opposite direction will help to establish not only where the signal interruption occurred, but will enable the process control computer 14 and its network controller 16 to determine the path required to transmit signals to or receive signals from each breakout circuit 26 on the primary level of signal distribution. As a result of the integrity testing process, the network controller 16 will store the path information necessary to transmit or receive signals from each of the field computer units 12 in random access memory ("RAM"). In other words, signals directed to some of the field computer units 12 may be transmitted via fiber optic conduits 34, while signals directed to other field computer units 12 may be transmitted via fiber optic conduits 36 in the same overall timing cycle (for example, one second) period.

[0035] Figure 2 also serves to point out that the breakout circuits 26 may serve to function as signal repeaters, such as breakout circuits 26e-26f and 26h-26j. Thus, where the field computer units 12 are located at significant distances from the process control computer (for example, 1.6 kilometers), then one or more of the breakout circuits 26 may be used to provide the signal re-transmission necessary to permit an accurate signal reception at such remote field computer units.

[0036] Referring again to Figure 1, each of the field computer units 12 are shown to include a processor chassis 38, a DC chassis 40 and an expanded DC chassis 42. The processor chassis 38 includes three redundant computer circuits, which may also be referred to as field I/O controllers, and their associated analog input ("AI"), analog output ("AO") and digital output ("DO") processing circuits. In one form of the present invention, the digital input ("DI") circuits may be contained on the field I/O controller circuit boards. As illustrated in Figure 1, the processor chassis provides a debug panel 44 for each of the redundant computer circuits in the field computer unit 12 to enable a technician to view selective internal operations of these circuits. The DC chassis 40 generally provides three functions. The primary function of the DC chassis 40 is to provide a connection point for DC field instrumentation. Additionally, the DC chassis 40 provides a mounting location for the fiber mount board utilized for terminating the fiber optic conduits 46 and 48 of the communication network. The DC chassis also provides a mounting location for a passive element board, which is used to provide protection to circuit elements of the field computer unit 12 from high energy surges that may be encountered in the field (for example, lightening). The passive element board includes a passive element circuit for each analog and digital input signal. These passive element circuits include positive temperature coefficient (PTC) resistors and zener diodes in conventional circuit protection configuration. The expanded DC chassis 42 provides a mounting location for additional DI and AI circuits and passive element circuits in the event that the not all of the DIs and AIs may be accommodated by the DC chassis 40.

[0037] Figure 1 also shows that each of the redundant computer circuits in the processor chassis 38 is preferably connected to a separate power supply 50. The circuit for these power supplies 50 will be discussed in connection with Figures 16A-16G. Each of these power supplies 50 is preferably provided with its own backup battery 52. The batteries 52 facilitate uninterrupted operation by the field computer unit 12 in the event that the source of alternating current normally provided for the power supplies becomes temporarily unavailable. Thus, it should be appreciated that a fault at any one of the power supplies 50 or even an interruption in the supply of alternating current power to the field computer unit 12 will not affect the underlying physical process being controlled by the field computer unit 12. Alternatively, it should be appreciated that a conventional uninterruptable power supply could be used as an option to avoid a potential loss of electrical power.

[0038] Referring to Figure 3, a block diagram of the distributed interface system 10 is shown. In this regard, Figure 3 serves to point out the bi-directional nature of the flow of signal communication through the use of the arrows 54 which are pointed in opposite directions. Additionally, Figure 3 illustrates that each of the breakout circuits 26 is preferably provided with a debug panel 56. Each of the debug panels discussed herein, namely debug panels 18, 44 and 56, are simply provided to assist a field technician during the maintenance or repair of the various circuits to which these debug panels are attached. Furthermore, Figure 3 illustrates generic devices for the DI's, DO's, AI's and AO's which are connected to the field computer unit 12. However, as will be appreciated from the discussions below, each of the field computer units 12 is capable of handling a substantial number of such field instrumentation inputs and outputs.

[0039] Referring to Figure 4, a block diagram of the flow of data/command/program signal communication for the interface system 10 is shown. In this regard, three circles 58-62 are used to illustrate exemplary signal inputs to the

field computer unit 12. Thus, an exemplary AI signal 58 may be comprised of a 4-20 ma current signal input, while an exemplary DI signal 60 may be comprised of a signal which is indicative of the closure or non-closure of a switch. When these signals are received by the field computer unit 12, they are referred to as "raw data" (block 64), and it should be understood that all of the raw data signals are read by each of the redundant computer circuits in the field computer unit 12. While each of the redundant computer circuits in the field computer unit 12 could be provided with its own set of corresponding input sensors, it is preferred that each of the redundant computer circuits receive the same input signals. In the event that it is desirable to provide two or more sensors to detect a particular process condition, it is still preferred that each of the redundant computer circuits receive the input signals from each of these corresponding sensors. In such a case, the redundant computer circuits would process each of these corresponding signals as a separate input signal. In other words, if three flow meters were used to detect the flow rate of a fluid at the same location in a fluid stream, then each of the three redundant computer circuits would process each of these three input signals and share these three input signals with each other through neighbor to neighbor communications. In this way, the full power of these redundant computer circuits may be utilized to enable the best opportunity for accurate decisions to ultimately be made. It should also be noted that block 64 indicates that the raw data signals includes DOT and AOT values. These values are feedback or track signals which are used to permit the appropriate circuits and software in the field computer unit 12 to determine if the output values sent to the field instrumentation are in accordance with commanded values received from the process control computers 14a-14b. These feedback or track signals are also transmitted to the process control computers 14a-14b for possible use as an assurance that the output is in the desired state.

[0040] Once the raw data signals have been received, each of the redundant computer circuits will independently determine whether or not the data is valid (block 66). This initial validity check helps to prevent the transmission of inaccurate input data, such as could occur if an input board was not property plugged in or it was inoperative. Each of the redundant computer circuits will also exchange the data that they have read from the field. In the case of analog input signals, each of the redundant computer circuits compares the difference between its input data signal and the input data signal from its neighbors, on a channel by channel basis. against a predetermined tolerance boundary to determine if the signal is within both a relatively broad range and a relatively narrow range of acceptable levels.

[0041] The validated signals for each input are independently arbitrated by the redundant computer circuits (block 68), as will be more fully discussed in connection with the flow charts of Figures 17A-17E and 18A-18N. Once the validated data signals have been arbitrated in software, the redundant computer circuits have effectively selected the specific input value to be transmitted to the process control computers 14a-14b via the fiber optic conduits 46-48 (block 70). In this regard, it should be understood that three redundant computer circuits are included in the field computer unit 12, while only two sets of fiber optic conduits 46-48 are employed in this embodiment to convey signals. Accordingly, it should be appreciated that the arbitrated data signals will be concomitantly transmitted from two of the three redundant computer circuits to the process control computers 14a-14b via the breakout circuits 26 (blocks 72-74) and the network controller 16 (block 76).

[0042] Once the process control computers 14a-14b make their process control decisions, then the (independent or reconciled) output value signals will be transmitted concomitantly to the appropriate field computer units 12 via both the Left and Right network rings. In accordance with the present invention, it is not necessary for the output value signals to be simultaneously transmitted to the appropriate field computer units 12 through both the Left and Right network branches. Specifically, it should be noted at this point that the network controllers 16 for the Left and Right process control computers 14a-14b operate under their own clocks, even though the timing of these clocks are preferably adjusted in software once per second to a clock signal in their respective process control computers. In a similar way, one of the process control computers (for example, computer 14b) preferably adjusts its clock signal to the clock signal of the other process control computer (for example, computer 14a). Likewise, the clocks for each of the redundant computer circuits in the field computer unit 12 preferably adjust themselves to one of their clocks (for example, the Left computer circuit) with each process control cycle. Accordingly, it should be appreciated that the clocks in each of the process control computers 14a-14b, the network controllers 16 and the field computer units may undergo a periodic adjustment in order to maintain the clock signals within a desired tolerance (for example, 4 milliseconds).

[0043] In any event, when the output value signals are received at a field computer unit 12, they are communicated to each of the redundant computer circuits, and are referred to as Unarbitrated Data in block 78. Then, in accordance with the present invention, each of the redundant computer circuits independently arbitrate these output value signals in software (block 80). Finally, each of the redundant computer circuits transmit each of the arbitrated output value signals to the field DO devices 84 and the field AO devices 86 (block 82) through a set of abort circuits which will be discussed below in connection Figures 10A and 10B. However, at this juncture it should be noted that the abort circuits enforce the decisions made via software arbitration by each of the redundant computer circuits.

[0044] Referring to Figure 5, a perspective view of the processor chassis 40 is shown. The processor chassis 40 generally includes a metal housing 88 and a mother board 90. The mother board 90 may be referred to as a backplane board, as it is vertically supported against the back wall of the housing 88. The backplane board 90 includes the

necessary connectors and conductors for interconnecting the various circuit boards which are mounted to the backplane board. In this regard, Figure 5 shows that an individual circuit board is provided for each of the three redundant computer circuits 92-96 contained in the field computer unit 12. In this way, it should be appreciated that any of these individual computer circuit boards 92-96 may be quickly removed and replaced without affecting the operation of the remaining computer circuit boards. Indeed, one of these computer circuit boards 92-96 may simply be pullec om the processor chassis 40 for repair or replacement. However, it is preferred that electrical power for this computer circuit board be temporarily shut down while it is being removed or re-installed into the processor chassis 40. Nevertheless, no other command or software changes need to be made during replacement, even though the physical process is continuing to be controlled by the output signals from the field computer unit being serviced.

[0045] Figure 5 also illustrates that individual AI, DO and AO circuit boards are also mounted to the backplane board 90. Each of these input and output circuit boards is capable of handling a plurality of different signal inputs or outputs as the case may be. It should also be noted that a high speed analog input circuit board could also be contained in one of chassis locations within the field computer unit 12 for measuring electrical parameters in an alternating waveform power system. A description of this high speed power analyzer may be found in the commonly assigned Glazer et. al. patent application Serial No. 502,050, entitled "High Speed Power Analyzer", filed on March 30, 1990, now U.S. Patent No. 5,151,866. This U.S. patent is hereby incorporated by reference.

[0046] Referring to Figures 6A-6U, a schematic diagram for one of the redundant computer circuits will now be discussed. For sake of simplicity, this redundant computer circuit or field I/O controller will be generically referred to herein as controller 100. It should also be understood that in this embodiment, the controller 100 will be replicated for each of the redundant computer circuits 92-96. However, it should be appreciated that other suitable redundant computer circuits may be employed in the appropriate application, and that one or more of these circuits could be replaced with an updated circuit without necessarily requiring the replacement of all of the redundant computer circuits.

[0047] Figure 6A shows that the controller 100 includes a microprocessor circuit chip U40. While in one form of the present invention, the microprocessor U40 is comprised of a 80C31BH-1 microprocessor chip manufactured by Intel, it should be understood that other suitable chips may be used for this or any of the other circuit chips identified herein as the application or technological advance may warrant. The microcomputer kernel for the controller 100 also includes a 128K x 8 EPROM memory (58255P-551) U41, a 128K x 8 battery-backed RAM memory (58255P-551) U42, and a memory address latch (74HC573). The microcomputer kernel for the controller 100 also includes a memory controller (EP1810) U44, which is shown if Figure 6B. In this embodiment, the program for the controller could be stored in either the EPROM circuit or the battery-backed RAM circuit. The use of a battery-backed RAM is particularly advantageous in at least one respect. Namely, the battery-backed RAM U42 helps to permit an updated program to be downloaded to the controller 100 from the process control computers 14a-14b through the fiber optic network at any available communication time slot without having to electrically configure the memory device for a change in the information stored therein.

[0048] Importantly, it should be noted that the process of downloading an updated program to one or more of the field computer units 12 does not interfere with the ongoing operation of the physical process being controlled. More specifically, the program for only one controller 100 is updated at a time, so that the other two remaining controllers may continue under their existing programs to process field inputs and outputs. In one form of the present invention, the RAM U42 has a storage capacity of 128K bytes, even though the actual program storage requirement does not exceed 64K. This is to permit both data and program memory to be stored on the same chip. The doubling of memory capacity allows an updated program to be loaded and verified, while the controller is not doing process control, without disturbing the current contents of the program memory. After this validity check is completed, then the updated program is moved to the lower 64K memory locations of the RAM U42 for use on the next program cycle.

[0049] Once the updated program has been property downloaded into the RAM U42 for one of the controllers 100 in a field computer unit 12, it is successively loaded into the RAM U42 for each of the other controllers 100 in turn. As will be discussed below, each of the controllers 100 include neighbor to neighbor serial communication links which will permit, among other things, an updated program sent to one of the controllers to be copied to the RAM memory U42 of another controller in the field computer unit 12. Such neighbor to neighbor links also enable one of the controllers to completely restore the program memory in another controller should such an action be required. Thus, each of the field computer units 12 in the distributed interface system 10 may be provided with updated application programs without any manual steps needed to be taken at the field computer units or any interruption required in the physical process itself. Indeed, it is also possible for a broadcast downloading operation to be employed with the fiber optic network in which some or all of the field computer units 12 concomitantly receive an updated program through a generally addressed network message. In other words, the process control computers 14a-14b could transmit an updated program to as many field computer units 12 as appropriate in the distributed interface system 10 by setting the addresses to each of the corresponding breakout circuits 26 in the broadcast message to direct the message to the selected field computer units.

[0050] The RAM memory U42 and the ROM (and bootstrap) memory U41 share a multiplexed address/data bus

"P0" (pins P0-1..P0-7), as well as a common address bus "P2" (pins P2-0..P2-7). In this regard, it should be appreciated that the memory address latch U43 creates an address bus "AD" (pins AD-0..AD-7) from the multiplexed address/data bus for use by various components in the controller 100. In other words, the memory address latch U43 will capture an address or partial address on pins P0-1..P0-7 for subsequent use by components such as the EPROM memory U41. For example, pins AD-0..AD-3 and AD-7 are directed to the memory controller U44, which is a programmable logic device. Depending upon the digital state of these address pins and other needed input pins (such as "/WR"), the memory controller will generate an output signal in accordance with the internal software configuration for the chip. As an example of one such output, the memory controller will generate a "/RAM" signal which is directed to the "/CE" port of the RAM memory U42. This particular signal from the memory controller U44 will enable the RAM memory chip U42 to read or write data in combination with other associated signals, such as the "/RAM-WR" signal generated by the memory controller.

[0051] Figure 6A also shows a manually actuated reset switch "SW4", which may be conveniently located on the front panel of the field computer unit 12 in order to permit a technician to reset microprocessor U40 of the controller 100. However, in accordance with the present invention, a neighbor controlled reset circuit 102 is also provided which will enable any two controllers in the field computer unit 12 to reset the remaining controller without operator intervention. The reset circuit 102 has two input signals, namely "N1RST" and "N2RST". Each of these signals represents a reset request to the controller from one of the other neighbor controllers. The N1 RST signal is directed to the opto-coupler (MOC8021) U36, while the N2RST signal is directed to the opto-coupler U35. The output of opto-coupler U36 is connected to the other input to opto-coupler U35, so that the reset circuit 102 requires the combination of both the N1RST and N2RST signals to produce a high output "RESET" signal for transmission to the RST port of the microcomputer U40 through comparator (LM339) U24 and micro manager (DS1236-5) U28. The comparator U24 is employed to produce a Low "EXTRNRST" signal when the microprocessor U40 is to be reset. The micro manager circuit U28 will respond to the Low EXTRNRST signal by producing the High RESET signal.

[0052] Thus, for example, where two of the controllers in the field computer unit do not receive communication from the remaining controller within a predetermined period of time, then each of the other controllers may independently arrive at a decision that the non-responsive or otherwise errant controller should be temporarily reset or permanently shut down. Nevertheless, the reset circuit 102 requires the concurrence of both of the other neighboring controllers to temporarily reset or shut down the remaining controller by causing a reset condition (and holding this controller in the reset condition when it is to be permanently shut down). A permanent reset condition at the microprocessor level will disable the operation of the controller until at least one of its neighboring controllers changes the digital state of its reset request signal. In accordance with the method of operation under the present invention, the non-responsive controller is temporarily reset before a decision is made to permanently reset the controller. The initial decision to temporarily reset the non-responsive controller is preferably made after valid input and output communication messages have not been received for two consecutive process control cycles (for example, 2 seconds). Accordingly, it should be appreciated that this method allows for a fault tolerance for communications between neighboring controllers of at least one process control cycle. If the non-responsive controller does not begin communicating with its neighbors within a predetermined period of time after being temporarily reset (for example, 20 seconds), then its neighboring controllers will independently request a permanent reset of the non-responsive controller. Once the non-responsive controller has been replaced or repaired, then the permanent reset condition may be terminated through a software value change in the appropriate data table location of a neighboring controller to re-activate the previously non-responsive controller. Additionally, each of the controllers 100 preferably maintains a count of the number of times that they have requested a reset condition of a neighboring controller, so that a record may be available for health and welfare analysis as needed.

[0053] It should be noted that each of the controllers preferably communicates three times it a process control cycle (for example, one second) with its neighboring controllers. Specifically, each of the controllers will communicate the following signals to neighboring controllers: the input signals received from the field, the output signals received from one of the process controller computers, and various diagnostic signals to be discussed more fully below. In one form of the present invention, each of these communications may take place during predetermined time windows (for example, 8 milliseconds each).

[0054] The micro manager circuit U28 also monitors the voltage level of the normally +5 volt VCC power line. This monitoring function enables a temporary reset condition to be applied in the event that the VCC power line drops momentarily below a predetermined level (for example, +3 volts). Additionally, the micro manager circuit U28 is adapted to switch the supply of electrical power for the RAM memory U42 to the lithium backup battery B1 in the event that the VCC power line drops to zero. The micro manager circuit U28 controls the PROT-CERAM signal. This signal usually follows the CERAM signal, but is latched high during battery backed conditions. Importantly, this procedure will disable these memory circuits from writing any new data into their respective memory locations. This procedure is employed to prevent potential corruption of the data contained in RAM memory due to an interruption in electrical power.

[0055] It should also be pointed out that the opto-couplers U35-U36 electrically isolate the controller 100 from both

of its neighbors. In this particular embodiment, opto-couplers are used on the reception end to isolate all of the communication paths between the redundant controllers 100, in order to prevent an electrical fault in one of the controllers from affecting the operation of its neighboring controllers.

[0056] Neighbor to neighbor signal transmissions from the microprocessor U40 of Figure 6A are facilitated through the serial communications driver (74H138) U38 of Figure 6H. As illustrated in Figure 6H, the "TXDATA" signal from the serial output port of the microprocessor U40 is coupled to the "/G2B" input port of the serial communication driver U38. Accordingly, it should be appreciated that the serial communication driver U38 is used to direct the TXDATA signal from the microprocessor U40 to one or more of a plurality of different communication paths. These communication paths include the "NF1TXD" and "NF2TXD" signals, which each represent a serial communication signal to a different neighboring controller 100. Four additional serial communication output signal streams are also provided, namely "TXDATAAO", "TXDATA1", "TXDATA6" and "TXDATA11". The TXDATAAO signal is directed to the analog output circuits in the field computer unit 12 to convey analog output values and direct the non-intrusive testing to be described below. In this regard, it should be appreciated that the analog output value signals which are transmitted from the process control computers 14a-14b to the field computer unit 12 are subsequently processed (for example, software arbitration) by the microprocessor U40 of the controller 100 and directed to the appropriate analog output circuit boards of the field computer unit through the serial communication driver U38. Additionally, it should be noted that the arbitrated analog output value signals are not transmitted to any neighboring controllers, as there is no need to do so in accordance with the present invention. Thus, it should be appreciated at this juncture that none of the other controllers are aware of specific analog output value signals transmitted to their respective analog output circuits. The other three serial communication signals (TXDATA1, TXDATA6 and TXDATA11) are directed to specific analog input circuits for requesting value and configuration data.

[0057] The last two remaining output signals of the serial communication driver U38 of Figure 6H are the "MAIN_XMIT" and "RPT_XMIT" signals. The MAIN_XMIT signal is directed to a transmitter circuit, such as that shown in Figure 15B, for communication with one of the process control computers 14a-14b through the fiber optic network. In this regard, the MAIN_XMIT signal is directed to the appropriate port of breakout circuit 26 connected to the field computer unit 12. The RPT_XMIT signal simply provides additional communication capacity if desired. With respect to the controller 100 which is mounted in the Middle slot of the field computer unit 12 between the Left and Right controllers, there is no connection provided for the MAIN_XMIT and RPT_XMIT signals in this particular embodiment. However, it should be appreciated that the fiber optic network could be modified to provide a set of fiber optic conduits for each of the controllers 100 contained in the field computer unit 12, particularly when three redundant process control computers 14 are provided.

[0058] Figure 6C illustrates a signal distribution circuit 104 which is coupled to the multiplexed data/address bus P0 of the microprocessor U40. The signals directed to the distribution circuit 104 from the microprocessor U40 are buffered by a pair of octal D type latch circuits (74HC573) U37 and U32. Latch circuit U32 is used to transmit signals to the debug panel 44 for the controller 100, while latch circuit U37 creates a distribution bus "RP" (pins RP-0..RP-7) for use by several other circuit chips. Each of the circuit chips connected to the RP bus in Figure 6C are comprised of an 8-bit addressable latch circuit (75HC259).

[0059] The latch circuit U30 and a portion of the latch circuit U39 are used to transmit individual "set" digital output signals (pins SDO-1..SDO-10) to specific digital output circuits which are connected to the controller 100 through the backplane board 90. Accordingly, it should be appreciated that the digital output value signals which are transmitted from the process control computers 14a-14b to the field computer unit 12 are subsequently processed (for example, software arbitration) by the microprocessor U40 of the controller 100 and directed to the appropriate digital output circuit boards of the field computer unit through the latch circuits U30 and U39.

[0060] The latch circuits U22, U26, U34 and U39 are used to transmit abort analog output signals "AAO" and abort digital output signals "ADO" to the analog output circuits and digital output circuits, respectively, of neighboring controllers. For example, latch circuit U22 generates abort digital output signals ADO2-3..ADO2-10, while latch circuit U26 generates abort digital output signals ADO1-2..ADO1-9. This notation means that all of the abort digital output signals from latch circuit U22 are directed to the digital output circuits for the controller 100 designated as 'neighbor 2" relative to this particular controller circuit. Similarly, all of the abort digital output signals from latch circuit U26 are directed to digital output circuits for the controller designated as "neighbor 1". Additionally, the specific signals with corresponding final digits, such as ADO1-9 and ADO2-9, refer to the same digital output channel. Thus, it should be appreciated that a series of corresponding abort digital output signals are sent to the digital output circuits for the neighboring controllers within the field computer unit 12.

[0061] With respect to the abort analog output signals, it should be understood that these signals are not analog in nature. Rather, as in the case of the abort digital output signals, the abort analog output signals are either in a High digital state (logical "1") or a Low digital state (logical "0"). Additionally, a corresponding notation is employed for both the abort digital and abort analog output signals. Accordingly, it should be appreciated that a series of individual abort analog output signals are sent to the analog output circuits for each of the neighboring controllers within the field

computer unit 12. As will become more clear from the discussion of the analog and digital output circuits below (for example, Figures 10A-10B), these "abort" output signals are used to enforce the software arbitration decisions made by each of the controllers 100. These arbitration decisions are represented by the "set" digital output signals and the analog output signals already discussed above.

[0062] The signal distribution circuit 104 of Figure 6C also includes a latch circuit U33 which is used for various functions of the controller 100. For example, several temperature control signals are shown, such as "FANON", "COOLON" and "HEATON", for maintaining the field computer unit interior within an acceptable temperature range. As the signal names imply, the field computer unit 12 may be provided with one or more fans, a heater and/or an air cooling device in the event that the field computer unit is located in an environment where such measures would be desirable. The "BAT" signal is used to turn off a charger for the batteries 52 in order to begin a load test to be described in connection with the power supply circuit 50. The "BATTOFF" signal is used to shut down a +5 volt power supply line to the field computer unit when the batteries 52 are drained of power. Similarly, the "/CONSERVE" signal is used to turn off a +26 volt power line to the field computer unit in order to conserve battery power. The "XGFLT" signal is used to control the circuitry that tests for a difference between the ground potential of the field computer unit and the true ground.

[0063] The "DEADSET" signal is directed to a retriggerable monostable multivibrator circuit (74LS122) U21 which is used as deadman timer and abort opening circuit. In this regard, the capacitor C49 and the resistor R102 determine a basic pulse time, and the DEADSET signal is used to prevent the "/ABRES" and 'DEAD' output signals from switching to their shutdown states. As illustrated in Figure 6C, the /ABRES signal is directed to the /CLR port of the latch circuits U22, U26, U30, U34 and U39. Accordingly, the /ABRES signal serves to simultaneously reset all of these identified latch circuits when the DEADSET strobe is not received from the microprocessor U40 to a retrigger a timer in multivibrator circuit U21. The DEADSET signal is transmitted once each process control cycle when the microprocessor U40 is functioning property. The DEAD signal is directed to the analog output circuits in order to prevent them from sending power to the field.

[0064] Figure 6B also illustrates that the PLD circuit U44 generates demultiplexed output signals (OUT0..OUT7) which are directed to the enable port for several of the circuit chips that have been discussed above. For example, the OUT5 signal is transmitted to latch circuit U22 to enable this latch circuit to capture the HIGH/LOW data signal on line RP-0 and direct it to the output port addressed by lines RP-1..RP-3. Additionally, the OUT6 and OUT7 signals are directed to a digital to analog converter circuit U1 which will discussed in connection with Figure 6K.

[0065] The PLD circuit U44 also generates demultiplexed output signals (IN0-IN6), which are directed to the various "read" circuits shown in Figures 6F and 6G. Thus, for example, the IN3 signal from PLD circuit U44 is directed to the enable ports (/1G and /2G) of the tri-state buffer circuit (74HC244) U16 of the "read" remote address circuit 106 shown in Figure 6F. In this regard, switches SW1 and SW2 (230034G) determine the field address of the controller 100, which may be read by the microprocessor U40 from bus P0 when it is desired to receive a message from or form a message to one of the process control computers 14a-14B. Figure 6F also includes a read function circuit 107 similar to the read remote address circuit 106. The read function circuits 107 includes a switch SW3 which is set to inform the microprocessor U40 of the power supply configuration for the controller and/or other hardware specific settings. Additionally, the read function circuit 107 includes a set of KEY0..KEY3 signals which respond to the keys depressed on the debug panel 44. These keys include a function key, a key to read an element of memory and a key to put a value into a memory location.

[0066] Figure 6D shows another read circuit 108. This read circuit includes a set of jumpers "J7-J10", which may be used to permit the microprocessor U40 to know which hardware version or revision is being utilized for the controller 100. Additionally, a switch "SW6" is employed in order to provide space for future enhancements. The signals provided by the jumpers J7-J10 and the switch SW6 are captured by the tri-state buffer circuit (74HC244) U61 and transmitted to the P0 bus of the microprocessor U40.

[0067] Figure 6E shows a display circuit 109, which is comprised of an octal flip-flop circuit U62 and an LED bank (LEDBAR10) "LED1". This display circuit is employed on the controller circuit board to permit a technician to readily see various health and welfare indicia for the controller during maintenance.

[0068] Turning to Figure 6G, a set of three read circuits 110-114 are shown. These read circuits are used to inform the microprocessor U40 as to how to interpret the data being read from a plurality of analog signal input circuits, such as those shown in Figures 7A-7C and 8A-8F. For example, the "TYPEAC" and "TYPEDC" signals inform the microprocessor U40 whether the input signals from the left expansion chassis 42 represent alternating current "A.C." or direct current "D.C." signals. Additionally, signals such as "FAM1-5A" and "FAM1-5B" transmitted to buffer circuits U23-U27, respectively, provide digital indications of broad linearization routines that should be employed by the microprocessor U40. For example, these signals indicate whether a particular signal received by the microprocessor U40 has been transmitted from a smart input circuit board or a standard input circuit board. The "AITYPE1-A" and "AITYPE3-B" signals indicate specific linearization routines that should be employed by the microprocessor U40 (for example, type-J v. type-S thermocouples).

[0069] The buffer circuit U31 receives signals, such as "AISENSE1-5", which inform the microprocessor U40 as to which input and output circuit boards are installed in the field computer unit 12. The switch SW4 is used to configure signals, such as "USE-DOAC1", which inform the microprocessor U40 whether the controller 100 is being used as a Left, Middle or Right controller.

[0070] Figure 6I illustrates a sixteen channel multiplexor circuit (506A) U9 which is configured to direct a plurality of digital input signals to the main multiplexor circuit (506) U11 shown in Figure 6J. Specifically, the digital input signals are labeled "MDI-1..MDI-10". These signals are derived from the pull down circuits shown in Figures 6T and 6U. Address lines "HDEV0..HDEV3" are used to select one of these digital input signals for output to the main multiplexor circuit U11. The output port of the multiplexor U9 is connected to an operational amplifier (3140A), which is configured as a voltage follower, in order to generate the "DI-LOCAL' signal for transmission to the main multiplexor U11.

[0071] The main multiplexor U11 of Figure 6J is used to individually select one of a plurality of different input signals for transmission in a successive pattern to the microprocessor U40 through a successive approximation circuit 116. These input signals include the analog level or analog serial input signals (for example, "MAI6-10L"), analog output status and track signals (for example, "AOT"), serial communication signals from neighboring controllers (for example, "NP2RXD"), and serial communication signals from the fiber optic network (for example, "MAIN_RCV"). Additionally, the main multiplexor circuit U11 receives a "DI_DISTANT" signal which represents a plurality of multiplexed analog voltage level signals from digital inputs circuits in the left expansion chassis 42, and a "DO_DISTANT" signal which represents a plurality of multiplexed analog voltage level signals from analog input circuits in the left chassis. The "DACCAL" signal is a signal which could be used to provide external calibration of the DAC circuit U1. The "BOARD_FUNC" signal represents a plurality of multiplexed signals from the multiplexor circuit U10 of Figure 6K. The "DO_LOCAL" signal represents a plurality of multiplexed informational signals from one or more digital output circuit boards, such as track values and return values from non-intrusive testing.

[0072] The successive approximation circuit 116 receives the multiplexed output from the main multiplexor U11 through the resistor R41. The successive approximation circuit 116 enables the microprocessor U40 to determine the voltage level of a signal output from the multiplexor U11. In this regard, the output from the main multiplexor U11 provides one input to a comparator (LM339) U3. The other input to the comparator U3 is provided by a digital to analog converter "DAC" circuit (DAC708KH) U1, shown in Figure 6K as a continuation of the successive approximation circuit 116. Specifically, the successive approximation circuit permits the microprocessor U40 to receive a plurality of both digital and analog input signals through a single input line "RXDATA". This is achieved through the toggling of the comparator U3 output in response to a changing "VOUT" signal level from the DAC circuit U1. The microprocessor U40 transmits a series of different digital voltage levels to the DAC circuit U1 via the RP bus until such time as the comparator U3 changes output states. In this regard, the microprocessor U40 preferably performs a binary search by starting with a digital voltage level in the middle of the acceptable range, determining if this value is high or low, and then stepping up or down from that point. The microprocessor U40 then determines the voltage level output from the main multiplexor U11 through its knowledge of the last digital voltage level transmitted to the DAC circuit U1. Accordingly, it should be appreciated that the combination of this successive approximation procedure and the use of multiplexors substantially reduces the number of input pins that would otherwise be required to read all of the digital and analog inputs signals being gathered by the field computer unit 12.

[0073] Figure 6K also shows that the DAC circuit U1 is addressed through an octal D flip-flop circuit (74HC374) U17, which creates the address lines "DAC0..DAC2" from the RP bus. Additionally, this flip-flop circuit also creates address lines "LDEV0..LDEV3", which are directed to level shifting buffer circuits (NC14504B) U18 and U19. The LDEV address lines are shifted from a 0/5 volt signal to a 0/15 volt signal, as required by the configuration desired for the multiplexor circuits U9, U10 and U11. Similarly, the address lines P1-4..P1-7 are shifted by the buffer circuit (MC14504B) U13 to generate address lines HP1-4..HP1-7 for the multiplexor U11. In this regard, it should be noted that the ground "GND" potential of these multiplexor circuits is set to 10 volts rather than 0 volts. This is because the particular multiplexor chip chosen (506) limits the potential difference between V+ and GND to 22 volts. However, with the GND potential set to 10 volts, the V+ potential may be set to 25.2 volts and the V- potential set to -5 volts, thereby allowing the multiplexor circuits to operate from a +/-15 volt supply. In such a configuration, it is necessary to shift the level of the LDEV address signals in order to permit the multiplexor chip to operate properly.

[0074] As indicated in Figure 6K, the multiplexor circuit U10 receives several diverse input signals for selection and transmission to the main multiplexor U11 via the "BOARD_FUNC" signal. These input signals include the present status of reference voltage levels (for example, "+10VREF"), and various temperature levels (for example, "BDTEMP").

[0075] Figure 6L illustrates a simple temperature sensor circuit 118 which is used to provide an indication of the temperature at or near the controller circuit board. This temperature is sensed by the transducer circuit formed by (AD502) Q13 and resistor R52, and filtered by capacitor C13.

[0076] Figure 6M illustrates two temperature control output circuits 120-122. The output circuit 120 is responsive to a "HEATON" signal from the latch circuit U33 of Figure 6C, while the output circuit 122 is responsive to a "COOLON" signal from this latch circuit. Opto-couplers U14-U15 are used to galvanically isolate the controller 100 from the external

heating and cooling devices through the transmission of optical signals "PHEAT" and "PCOOL" respectively. These opto-couplers are driven by current sources (T1317C) Q17-Q18 and the concurrence of either of the HEATON or COOLON signals.

[0077] Figure 6N illustrates a filter circuit 124 for the identified humidity and temperature signals. For example, the "EXTEMP_1" external temperature signal input is labeled "MEXTEMP" at the output, which is then transmitted to the multiplexor circuit U10 of Figure 6K. This external temperature signal may be used as a redundant cold reference junction temperature signal. The humidity signal "HUMITY_1" may be derived from a sensor within the field computer unit housing 88. One or more of these temperature signals may be used by the microprocessor U40 to determine whether the PHEAT or PCOOL signals should be generated. In one form of the present invention, it is preferred that the interior environment of the field computer unit 12 be maintained within a temperature range between 10 and 50 degrees celsius.

[0078] Figure 6O is a very simple impedance circuit 126 which operates in conjunction with the serial communication driver circuit U38 of Figure 6H for communicating with neighboring controllers. Specifically, the circuit 126 receives the "NF1TXD" and "NF2TXD" signals, which each represent a serial communication signal to one of the neighboring controllers. This impedance protects driver circuit U38 from damage in the event that a short occurs on a signal line outside of the controller 100. It should also be noted that Figure 6S provides a serial communication receiver circuit 128 for accepting communication from neighboring controllers. These neighbor signals are passed through to the opto-coupler circuit U12 for optical isolation. These signals are then transmitted to the main multiplexor circuit U11 as the signals "NP2RXD" and "NP1RXD".

[0079] As mentioned earlier, the neighboring communication paths may be used to convey input and output value signals, as well as updated or revised program data. Accordingly, it should be appreciated that the combination of serial communication transmitter and receiver circuits between the three controllers 100 in the field computer unit 12 provide the field computer unit with the ability to arbitrate both incoming and outgoing data through the mutual exchange of such data by the controllers. Thus, when the Left controller board 92 receives output value signals for the field instrumentation via fiber optic conduits 48, these signals are also transmitted by the Left controller board to the Middle controller board 94 and the Right controller board 96. Similarly, when the Right controller board 96 receives output value signals for the field instrumentation via fiber optic conduits 46, these signals are also transmitted by the Right controller board to the Middle controller board 94 and the Left controller board 92. In this way, each of the three controller boards 92-96 are provided with three sets of output value signals which may be used for independent arbitration in software. In one form of the present invention, the Middle controller 94 receives output value signals from both the Left controller board 92 or the Right controller board 96. A further discussion of the arbitration procedure for output values will be provided in connection with Figures 17F-17I and 18O-18T.

[0080] Figure 6P illustrates a ground fault circuit 130, which is used to inform the microprocessor U40 that a ground fault condition has occurred through the signal "GNDFLT" and multiplexor U9. In this regard, the "XGFLT" signal is derived from the latch circuit U33 of Figure 6C, while the "GND_FAULT" signal is derived from the field through the backplane board 90. A ground fault condition occurs when there is a very low potential difference between the chassis ground and the FLTGND terminal. The microprocessor U40 may respond to this condition by setting an error bit that is available to the process control computer 14.

[0081] Figures 6Q and 6R are shown simply to illustrate two representative power conditioning circuits which are contained on the controller 100. The "MM15" output signal shown in Figure 6R is used to permit monitoring of the -15 volt power line. Similar power conditioning circuits are also contained on other circuit boards in the field computer unit 12. As should be appreciated from the above discussions, the controller 100 requires several different voltage levels to drive the circuit chips forming part of the controller, and these power conditioning circuits are adapted to produce the desired voltage levels.

[0082] Figures 6T and 6U illustrate digital input pull down circuits 132 and 134 respectively. In this regard, each of these circuits include a current source circuit (TL317), such as Q12, which is set to drive 2.5 milliamps through a current loop associated with each of the indicated digital input signal lines (e.g, DI-1..DI-5). These digital input lines may be used, for example, to sense the opening or closing of a set of switch contacts. When one of these switches is open, the current source will unsuccessfully attempt to push 2.5 milliamps into an essentially infinite load, so the voltage level measured from the sensing line (for example, MDI-1) will be in excess of 20 volts. When one these switches closes, the associated digital input line will be pulled to ground through a low impedance path, and its connected sensing line (for example, MDI-1) will transmit a signal level to the multiplexor U9 on the order between 2.5-7.5 volts. This voltage level will depend upon how many controller boards are connected to the particular signal input to the field computer unit 12. In this regard, it should be noted that if the voltage level sensed is below 1.5 volts, then the microprocessor U40 will assume that a field short condition has occurred, as the resistance in the sensing circuit is below that which would otherwise be available if the digital input circuit was operating properly.

[0083] Referring now to Figures 7A-7C, a schematic diagram for a smart serial input circuit 200 for processing analog signal information is shown. The input circuit 200 is capable of asynchronously processing the signals received on 5

separate serial input channels. Each of these channels are adapted to receive a digital signal stream which is representative of analog input signal information. In one form of the present invention, the field computer unit 12 may employ three such "analog" input circuits for each of the three redundant computer circuits 92-96. In this regard, the input circuit 200 will be mounted in one of the card slots shown in the processor chassis 40 of Figure 5 (for example, AI1-5 and AI6-10). While not shown in Figure 5, a slot is also provided for an "AI11-20" analog input circuit. Thus, it should be appreciated that the field computer unit 12 is capable of handling up to twenty distinct analog input signals.

[0084] The input circuit 200 is designed to operate in conjunction with a suitable transmitter device which will generate the appropriate digital stream. Preferably, a Honeywell transmitter is employed to read the analog signal and generate a digital stream or message therefrom, such as a (Series 100, 200 or 300) Smart Pressure Transmitter, a Smart Temperature Transmitter or a Smart MAGNEW Flow Transmitter. These Honeywell transmitters generate a three part digital message approximately three times each second. Specifically, the digital message includes the transmitter status, the primary analog value sensed, and configuration/status data. The digital message may also include a secondary variable value, such as head temperature.

[0085] The input circuit 200 is referred to as being a "smart" circuit in that it is capable of doing considerably more than merely sending on to the controller 100 the raw data that it receives from the transmitters. In this regard, input circuit 200 decodes the serial data stream from the transmitters and converts these streams to a format which is compatible with the controller 100 (that will ultimately be transmitted to the process control computer 14 as a 16-bit signed integer percent of full value). The input circuit 200 also provides for various error bits that the controller may utilize to interpret the data or otherwise transmit informed error messages. For example, these error bits include a "No Xmitter" bit, a 'Parity Error bit, and a 'Comm Error" bit. The No Xmitter bit is set when the transmitter has failed to send a serial data stream to the input circuit 200 within a predetermined time period (for example, 382 msec.). The Parity Error bit is set when: (a) an input signal is detected less than 48.9 msec. after the completion of the previous message, (b) the current byte being assembled from the serial transmission fails the parity test, or (c) the binary value of the start/stop bits are wrong. The input circuit 200 also formulates a message to the controller 100 which permits the controller to perform a "checksum" verification of the message it receives from the input circuit. The debug panel 44 for the controller 100 may also be utilized to examine the status bytes which contain the above identified error bits at the field computer unit 12. For example, the technician may use the debug panel to enter the memory address for the particular status byte in question, and the contents of this byte will be presented for visual inspection on the display device of the debug panel.

[0086] Figure 7A shows a receiver circuit 202 for the input circuit 200. While only one receiver circuit 202 is shown, it should be appreciated that the input circuit 200 should include an individual receiver circuit for each transmitter. The connector pin 'C3' is used as the entry point of the circuit to convey the digital signal stream from a transmitter to the receiver circuit 202. The receiver circuit 202 then employs a comparator (LM339) AU5 to produce an appropriate digital signal level input "HON1" (for example, High +5 volts, Low 0 volts) for further processing. The comparator AU5 is preferably set in an inverting mode to trigger at 0.9 volts with a hysterisis band of 0.42 volts, so that a logic "0" is detected when the voltage input to the circuit exceeds 1.25 volts, and a logic "1" is detected when the voltage input to the board is below 0.83 volts.

[0087] The HON1 signal is directed to the "P1" port of a 16MHz microprocessor (80C31) AU2, which is shown in Figure 7B. An 8Kx8 EPROM (27HC64) chip AU1 is used to store the program employed by the microprocessor AU2. The EPROM chip AU1 is directly connected to the "P2" port of the microprocessor AU2 and indirectly connected to the "P0" port of the microprocessor through memory address latch (HC573) AU3. The multiplexed data output from the microprocessor AU2 is transmitted to the controller 100 through the "TXDATA" signal. The TXDATA signal corresponds to one of the "MAI" prefix signals connected to the main multiplexor U11 of the controller 100. The microprocessor AU2 also receives signals from the controller 100 through the "RXDATA" signal line stemming from connector pin "C12".

[0088] Figure 7C shows a configuration circuit 204 for the input circuit 200. The configuration circuit 204 includes a switch "ASW1" which has four output lines (TYPE1..TYPE4). A pull up resistor is connected to each of these lines through resistor bank chip "ARP1". Additionally, an inverter from hex inverter circuit (HC04) AU4 is connected to each of the output lines from the switch ASW1 to provide an isolated set of configuration lines to the microprocessor AU2. The switch position for each of these lines is used to inform the microprocessor AU2 (through the "P3" bus) of the type of transmitter device connected to each of the receiver circuits by employing a suitable four bit code. The switch output lines are also directed to the controller 100. These output lines correspond to the "AITYPE" prefix signals shown on Figure 6G.

[0089] Referring to Figures 7D-7M, a series of flow charts associated with the operation of the smart serial input circuit 200 are shown. In this regard, Figure 7D provides an overall flow chart 206 entitled "AISER MAIN". The flow chart 206 includes an initialization block 208 which ends with the enablement of one or more interrupts. Program flow control is then passed to diamond 210, which determines whether or not a request for data has been sent by the controller 100, referred to here as "FIO". If data has been requested, then the UPLOAD routine is called (block 212).

The UPLOAD routine is shown in Figure 7F. If an upload request is not present, then the microprocessor AU2 determines if all of the data read through the flow chart of Figure 7E has been analyzed (diamond 214). If the data received in response to a series of data interrupts has not been analyzed, then the ANALYZE routine of Figure 7G is called (block 216).

[0090] Figure 7F indicates that the UPLOAD routine 212 includes the transmission of seven debug bytes to the controller 100 (block 218). These bytes are preferably stored in the internal RAM memory of the microprocessor AU2, and they may be accessed through the debug panel 44 for the controller.

[0091] Figure 7G indicates that the ANALYZE routine calls the SERVICE routine 220 shown in Figure 7H for each of the analog input signals received. The ANALYZE routine performs a variety of validity checks on the digital signal stream from a transmitter. For example, the flow chart 220 includes a diamond 222 which determines whether the channel is clear (CHNCLR), and a diamond 224 which determines whether the channel is in the process of assembling a byte of information from the serial data stream. If a byte is being assembled, then diamond 225 determines if the information being processed is from the proper interrupt. A bit count is then used to determine if valid start, parity and stop bits have been received. If the answer is negative for any of these questions, then the BADPARITY bit is set (block 226). Assuming that the data passes these checks, then the contents of the bit buffer "BITBUFF" are copied into the memory buffer 'MBUFF' (block 228) for subsequent transfer to the upload buffer "UPBUFF" (block 229). The contents of the upload buffer are then transmitted to the controller 100 in response to an upload request.

[0092] Figures 7I-7M illustrate flow charts for programs associated with the interpretation of signals received by the controller 100 from the input circuit 200. In this regard, the "AI31" flow chart includes a set up block 230 which calls a Smart AI Interface routine. The Smart AI_Interface routine provides a timeout of 5 msec. within which a upload response must be received and checked for communication errors. If a communication error was detected, then the status check routine "STCHK" is called. The STCHK routine sets one or more specific error bits depending upon the detected error (for example, a bad parity bit or a bad checksum bit). If no communication errors were detected, then a jump is made to the "OKAIS" routine of Figures 7J-7L is made (block 232).

[0093] As indicated by block 234 of Figure 7J, the OKAIS routine determines if a primary variable value was contained in the message sent from the input circuit 200. If the primary value is determined to be good, then a flag will be set which will cause a Fail-Last value to be sent to the process control computer 14 on the next failure (block 236 of Figure 7K). Then, the "IETOPS" routine of Figure 7M will be called (block 238) to convert the primary value to a fixed point value and store it as a percent of the maximum scale value of an acceptable input. In the event that a bad primary value was received, diamond 240 will determine whether a Fail-Last condition was set for this process control cycle. If it was, then the last known good primary value will be sent to the process control computer 14 and a flag will be set to not Fail-Last in the next process control cycle. However, if a Fail-Last condition was not requested, then the primary value will be loaded with a number corresponding to-100% of the maximum acceptable value (block 242).

[0094] *Blocks 244-246 and diamond 248 indicate that if a secondary value is present* (for example, temperature), then it will be converted to a percent of full scale. Diamond 250 then shows that this part of the OKAIS procedure will be implemented for all five analog inputs being sensed. Diamond 252 indicates that the controller 100 will then load the primary variables for channels 11-15, that were stored by block 258, into the proper IRAM locations. Block 254, diamond 256 and block 258 combine to temporarily store the primary variables for channels 11-15 and re-execute the routine to collect the data for channels 16-20. This allows one call of the routine to process 10 channels of data. The conclusion of the OKAIS routine is an indication that the analog input signals are now available for subsequent software arbitration by the field computer unit controllers.

[0095] Referring to Figures 8A-8E, a schematic diagram for a multiple-mode pulse input circuit 300 according to the present invention is shown. *The input circuit 300 is also referred to herein as the pulse train board "PTB" circuit.* The PTB circuit 300 is a five channel analog input "daughter" circuit board that may be used to measure frequency (1 Hz to 65kHz) with a high degree of accuracy (for example, .075% of the measurement) and/or count pulses (1 to 32767 pulses per second). Since the PTB circuit 300 has three different modes of operation, the controller 100 has two different methods of processing data (that is, pulse or frequency), and three methods of outputting this analog data (that is, only pulses, only frequency or both), even though the controller uses the same data to calculate both frequencies and count pulses. In the frequency mode, the frequency value stored in the AI table of the controller 100 is in a pseudo-floating point format, as will be discussed further below. This form is preferred in order to ensure that the floating point conversion would introduce no more than .025% of error into the final value to be transmitted to the process control computer 14. In the pulse counting mode, a true integer number is stored in the AI table. The number of pulses received since the last reported value is reported to the process control computer 14 as an integer stored in the AI table. In the event that *the values received by the PTB circuit 300 are over their respective ranges, then the controller 100 preferably* reports a full range value.

[0096] Since the field computer unit 12 preferably reports all of its input data to the process control computers 14a-14b each second, it should be appreciated that measured frequency values lower than IHz present a special problem, as the field computer unit will not be able to update the measurement once per second. Accordingly, the PTB 300 is

adapted to report a frequency of 1Hz in the time intervals that a pulse was detected. If no pulse was detected within the reported second, then a zero value will be transmitted to the process control computer 14. In the case where a pulse train starts after a period of zero input, and the PTB circuit 300 is in the frequency mode, the first second will not be used to report a frequency value. Rather, this first second will be used to report the total number of pulses received in that second. Only in the next second will the data be a true frequency value. This procedure is utilized to permit a summation of the total pulses over a known time interval. If no pulses are received over a second, the PTB circuit 300 will be unable to measure the time interval.

[0097] Figure 8A illustrates a receiver circuit 302 for the PTB circuit 300. In this regard, it should be understood that a receiver circuit 302 should be provided for each of the input pulse signal channels connected to the PTB circuit 300. The receiver circuit 302 includes a connector "BC3" which is used to couple the circuit to a pulse emitting transducer, such as a Hall Effect device, through the protection provided on the passive element board. The receiver circuit 302 also includes a signal line labeled "AI-1C" which provides a path to ground through a PTC resistor, such as resistor "VR3" shown in Figure 8D. The receiver circuit also includes a low pass filter, which is comprised of resistor "RLP" and capacitor "CLP". This low pass filter effectively removes any high frequency noise that may otherwise be induced in the field wiring. It should also be noted that the resistor RLP and the capacitor CLP are derived from a function module chip "BU13" which contains several of the other passive components in the receiver circuit 302. The capacitor CLP is connected in parallel with a diode (1N5819) "CR4" which clamps the negative going portions of the pulse signal to GND in order to prevent the comparator (LM339) "BU12" from being saturated.

[0098] The comparator BU12 receives the filtered pulse signal input and a reference voltage potential derived from a ten volt source. The reference voltage potential is set by voltage divider network comprised of the threshold resistor "RTH", a 10K resistor "R14" and the hysterisis resistor "RH". When the pulse signal is above the reference voltage potential, the output of comparator BU12 is pulled to GND. The low output from the comparator BU12. in effect, puts the resistor R14 in parallel with the resistor RH. This effect lowers the threshold resistance and allows the comparator output to stay low longer. This substantially eliminates unwanted oscillation that could be caused by low level noise on the input pulse signal.

[0099] The values for the passive components RTH, RH, RLP and CLP are preferably determined in accordance with the following approximation equations for large signal applications:

$$RTH = (100{,}000/Vth) - 10{,}000$$

where Vth = (.30) x Amax
where Amax is maximum amplitude of the signal

$$RH = ((1/Vh) - 1/5) \times 50{,}000$$

where Vh = 2 x (peak to peak noise level)

$$RLP \times CLP = T$$

where T = (1/Fmax) x (duty cycle of input)/3.14,
where T = (1/Fmax) x (1 - duty cycle)/3.14, depending upon whichever is smaller, and Fmax = the maximum frequency of the signal

In this regard, it should be noted that Vth is the threshold voltage where the comparator BU12 will decide that an input has a great enough voltage to be considered a high input. The value of the hysterisis resistor RH should be selected to allow the proper amount of hysterisis to be placed in the receiver or detector circuit 302. In this context, hysterisis is the difference between the threshold point and the point at which the comparator BU12 determines that the signal has dropped enough to be considered low. The provision of hysterisis in the receiver circuit 302 is useful in preventing mid-frequency, low-amplitude noise from affecting the output of the comparator BU12. The value of "T" is the period of the fastest component in Fmax. This calculation is useful as most signals are not on and off for equal periods of time (for example, a 50% duty cycle). Thus, to allow a pulse of 20% duty cycle to pass, the low pass filter must be capable of handling a frequency 1/(2 x .2) or 2.5 times greater than the true Fmax. Conversely, if the duty cycle is greater than 50%, the low pass filter must be capable of handling the zero part of the signal that is at a higher frequency than expected by a 50% duty cycle Fmax. Thus, for example, a pulse signal with a duty cycle of 75% should have a filter designed for 1/(2 x (1 -.75)) or 2 times Fmax. In this regard, it is preferred that the value for CLP be chosen to enable the value of RLP to stay in the range between 1 ohm and 10K ohms. Where the frequency of the input signal is relatively

low (for example, 50 Hz), the following values may be provided through the function module BU13: RTH = 10k, RH = 100K, RLP = 2.7k and CLP = 100 pf.

[0100] In large signal applications, the error induced in the approximations by RH is small, and thereby making the calculation for Vth a standard voltage divider. However, for small pulse signals, the error may be significant. Accordingly, for hysterisis levels greater than 1% of Vth, the following equations should be employed:

$$10000 \text{ x RH RTH} = \frac{10000 \times RH}{\frac{(100000+(10 \times RH)-(10000 \times Vhc))-RH-10000}{(10-Vth)}}$$

where Vhc is the high value output from the comparator BU12 (for example, 5 volts)

To use this formula, the value of RH must be known. In this regard, the value of RH may be approximated according to the following formula:

$$RH = \frac{(5-Vh) \times 50000}{5 \times Vh}$$

[0101] Once the pulse signal passes through the comparator BU12, it is an inverted 0/5 volt signal with a relatively slow rise time due to the capacitor (.001 micron) "C8". To speed up signal transitions and shape the signal into a more precise digital form, an inverter gate with hysterisis (74LS14) "BU6" is used. The inverter gate BU6 improves the rise time of the signal and inverts the output pulse signal "PTB1" to the original orientation of the pulse train received by the circuit.

[0102] As indicated in Figure 8B, the pulse signal output from each of the receiver circuits 302 (PTB1..PTB5) are coupled to a programmable logic device (Altera 1810) "BU7". The programmable logic device BU7 is set to provide five internal counters (one for each input pulse channel), and the associated internal addressing is set to permit it to be addressed as a memory mapped I/O device. In this regard, the internal configuration for the programmable logic device BU7 looks like five individual eight bit counters with their output control lines being set by logic driven by the address lines. The necessary multiplexing function for the programmable logic device outputs is accomplished by using tri-state buffers internal to the device. The internal counters permit pulses with a frequency greater than one-half the sample rate (that is, the Nyquist limit) to be measured.

[0103] Figure 8B also indicates that the PTB circuit 300 includes a microprocessor (80C31) "BU2", a memory address latch (HC573) "BU3" and an 8Kx8 EPROM chip "BU1". The jumper "J1" is set between pins 1-2 for EPROMs up to 256K, and the jumper J1 is set between pins 2-3 for EPROMs that are 256K or larger. The 16 MHz crystal oscillator "BY1" used to create the microprocessor clock signal is preferably accurate to +/- .005% in order to minimize the measurement error of the PTB circuit. When the microprocessor BU2 accesses a counter in the programmable logic device BU7, it reads the counter value and determines the number of pulses that have elapsed by subtracting the previous count from the current count. This procedure allows up to 255 pulses to occur between sample periods. It should also be noted that the PTB circuit 300 includes a light emitting diode "LED1", which will be on when the circuit is functioning property, as an aid to troubleshooting in the field. A flashing green light will indicate that the controller 100 is attempting to reset the PTB circuit 300. The debug panel 44 may be used to view the contents of an error byte for the PTB circuit 300. For example, individual bits of this error byte will indicate whether there has been a communication failure between the controller 100 and the PTB circuit 300, or whether a read error has occurred on a particular input pulse channel.

[0104] In terms of communication with the controller 100, the "RXDATA" signal line connected to the microprocessor BU2 is used to receive signals from the controller 100, such as a request to send data to the controller. Conversely, the "TXDATA" signal line is used to transmit the processed pulse data to the controller 100.

[0105] Figure 8C illustrates a current driver circuit 404, which is used for those pulse transducers which need to receive their electrical power from the PTB circuit 300. The current driver circuit is designed to provide a 25 milliamp current source to the field device at approximately 17 volts. A similar current driver circuit may also be employed in other input circuit boards, such as the input circuit 200 discussed above. As illustrated in Figure 8C, each of the pulse transducers may receive their electrical power through an individual current driver, such as current driver (LM317) "BU15".

[0106] Figure 8E illustrates a switch circuit 306, which is used to set the operating mode of the PTB circuit 300. In this regard, the switch "BSW1" sets the function for all five channels on the PTB circuit 300. For example, a selection of "0" may be used for the frequency mode, while a selection of '3' may be used for the pulse counting mode. Additionally, a selection of "4" may be used to enable both the frequency and pulse counting modes to be employed. In this regard, the controller 100 will transmit a set of both frequency and pulse counting data to the process control computer 14 for

each of the channels contained on the PTB circuit 300. The output lines of the switch BSW1 are coupled to the "P1" port of the microprocessor BU2 shown in Figure 8B. Thus, it should be appreciated that the switch circuit permits the PTB circuit 300 to be configured in the field, while also providing a way for the controller 100 to know how the data should ultimately be processed.

[0107] Referring to Figures 8F-8Q, a series of flow charts associated with the operation of the PTB circuit 300 are shown. Figures 8F-8J relate to software resident on the PTB circuit 300 itself, while Figures 8K-8Q relate to software resident on the controller 100. More specifically, the software represented by Figures 8F-8J is responsible for sampling the 1-5 pulse signal inputs, totaling the number of pulses received, measuring the elapsed time, and communicating this data back to the controller 100. In contrast, the software represented by Figures 8K-8Q is responsible for taking the data delivered from the PTB circuit 300, converting it into a frequency value and a total pulse count, and then sending these values to the process controller computer 14 upon request.

[0108] Figure 8F shows an overall flow chart 308 for the PTB circuit 300. The flow chart 300 includes a system initialization routine (block 310), which is illustrated in Figure 8G. After initialization has been completed, the program for the microprocessor BU2 of the PTB circuit 300 checks to see if data communication has been requested by the controller 100. If the answer is no, then the program checks to see if there is data to process. If pulse data has been received, then program control is directed to the process data routine (block 312), which is shown in Figure 8H. Once all of the data has been processed, then the program control returns to check for a communication request. If the controller 100 has made a request for data, then the send data routine is called (block 314). The send data routine is shown in Figure 8I.

[0109] Figure 8F also shows an interrupt or sampling routine (block 316), which is shown in Figure 8J. The interrupt routine is not shown to be connected to any other program control block, as it is clock controlled to ensure the accuracy of the sampling rate. Specifically, the interrupt routine is controlled by the "T1" clock signal of the microprocessor BU2 (see block 318 of Figure 8G). This interrupt preferably has priority over all of the other programmed functions of the PTB circuit 300 in order to ensure that sampling occurs at precise time intervals. In one form of the present invention, the sampling rate has an interval of 1/1999 sec. This particular sampling rate is considered advantageous due to the ability to evenly divide this rate into the maximum number of instructions/second (1,333,333) of the microprocessor BU2 and its ability to maintain a maximum error of .05%. As will be discussed further below, this sampling rate is preferably compensated for the length of time required to execute different instructions.

[0110] The function of the interrupt routine 316 shown in Figure 8J is to sample the counters in the programmable logic device BU7 and store the data in a buffer for later analysis. This is accomplished by reading each of the five internal counters four successive times (that is, read counter for channel 1 four times, then read counter for channel 2 four times, etc.), and then storing the data in a temporary buffer of the microprocessor BU2. This procedure is illustrated by blocks 320-322 in Figure 8J. The interrupt routine 316 then sorts through the readings to find the first two consecutive readings that were equal for each channel in order to prove the validity of the data read (for example, diamonds 324-328). The routine then starts filling up a buffer of data (for example, blocks 330-332) to be used by the process data routine 312, which runs in the spare time between interrupts.

[0111] The responsibility of the process data routine 312 shown in Figure 8H is to look at the data in the buffer, decide if a pulse has arrived, and then act on this decision. In order to accomplish this, five registers (blocks 338-339) are kept in the microprocessor BU2 for each channel of the PTB circuit 300. These registers are referred to as: Total Pulses, Total Interrupts, Number of Interrupts, Interrupts Since Last Pulse, and Previous Counter Reading. The Total Pulses register contains the number of pulses counted since the last transmission to the controller 100 (during the one second interval). This is the actual value transmitted to the process control computer 14 when the PTB circuit 300 is in the pulse counting mode. The Total Interrupts register contains the number of interrupts that have elapsed between the first and last pulses in the Total Pulses register. In other words, the Total Interrupts register provides an interval timer which is started by the last pulse received (leading edge) before the previous transmission to the controller 100 and *ended by the last pulse received before this transmission to the controller. The Number of Interrupts Since Last Pulse* register is used for pulse trains that are slower than 2kHz (that is, pulse trains under the sample rate). This register stores the number of interrupts that have occurred since the last pulse was detected and allows the Total Interrupts register to truly reflect the number of interrupts that have elapsed while the microprocessor BU2 was reading the Total Pulses register. The Previous Counter Reading register stores the last counter reading taken from the programmable logic device BU7, and it is used to determine how many pulses were received between samples.

[0112] Before proceeding to discuss the process data routine 312, it should be noted that the interrupt routine 316 includes a block 336 for controlling the timer controlled by the T1 clock. As the instruction set for the microprocessor BU2 includes instructions which may take one or two bus cycles to execute, a problem is presented when writing software that must be interrupted after a precise time interval. This is because this particular microprocessor will not service an interrupt until it is finished with the current instruction. The preferred solution to this problem is to load the T1 "count up" counter register of the microprocessor BU2 with the value of "FFFF" minus the number of bus cycles to elapse before an interrupt is to occur. The T1 counter will then count up until it hits "0000", and then the interrupt would

occur. Thus, for example, with a one bus cycle instruction, the interrupt routine would begin with a T1 value of 6 (to allow for the time needed to process the interrupt call), while an interrupt at the beginning of a two bus cycle instruction would enter the interrupt routine with the T1 counter having a value of 7. By adding the value of T1 to the appropriate constant and loading this value into the T1 counter register, it is possible to allow the average time between interrupts to be constant. This constant is determined by the number of bus cycles needed between interrupts and the number of bus cycles between the value of the timer and loading the timer. Thus, for example, where an interrupt is desired every 1/1999 sec. (or every 667 bus cycles), and it takes 5 bus cycles between the reading and loading operations, the value loaded into the T1 register would be: FD69 = FFFF-666dec + 5 dec.

[0113] As illustrated in Figure 8H, the process data routine 312 works by first incrementing all of the Number of Interrupts Since Last Pulse registers (block 338). Next, the current count "CC" from the buffer created by the interrupt routine 316 is compared with the Previous Counter Reading value "PC" to determine if a pulse has been received (diamond 340). If a pulse has not been received, the routine will move on to process the data from the next channel (block 342). If a pulse was received, then the number of pulses would be added to the corresponding Total Pulses register (blocks 344-348). The Number of Interrupts Since Last Pulse would also be added to the Total Interrupts register (block 350), the Number of Interrupts Since Last Pulse would be zeroed (block 352), and the processing would move on to the next channel (block 354).

[0114] Figure 8I shows the send data routine 314 which is called in response to a data request from the controller 100. In this regard, the PTB circuit 300 first sends the controller 100 the contents of seven bytes of debug data (block 356). Then, the error byte and constants, such as the sampling rate, are sent (block 358). Subsequently, the Total Pulses read in the last second and the Number of Interrupts that elapsed while reading the Total Pulses are sent for each of the input channels in turn (block 360). Finally, an Exclusive OR sum of all the transmitted bytes "XSUM", excluding the XSUM byte, is sent (block 362).

[0115] Figure 8K shows an overall flow diagram 364 for the software used in the controller 100 for processing the data received from the PTB circuit 300. The flow chart 364 begins with a get data routine (block 366), which is shown in Figure 8L. If the controller 100 is unable to obtain data from the PTB circuit 300, the controller will place the PTB circuit into a reset mode for three seconds (block 368), increase the error count by one (block 370), and send the previous second's data to the process control computer 14 with a flag to indicate that this group of analog inputs has bad data (block 372).

[0116] Assuming that the data has been received without error (diamond 374), the program will then convert the raw data into both total pulses (block 376) and a pseudo-floating point form (block 378). For the total pulse counting mode, the program takes the number of pulses received and places this value into the analog input table "AI XRAM" (block 376). This conversion routine is shown in Figure 8M. For the frequency mode (block 378), mathematical manipulations are performed to convert the Total Pulses and Total Interrupts data into a pseudo-floating point value. This is a two part process which begins by forming a 24 bit intermediate result, and then is completed by converting this result to a 16 bit pseudo-floating point form used to encode frequency. The pseudo-floating point number is a 16 bit value comprised of a power of four exponent and a fractional mantissa. The exponent represents the smallest power of four that can be divided into the original frequency (while maintaining a fraction) less one. This prevents the representation of numbers less than one, since fractions of one are not allowed. However, this procedure allows numbers up to 65535 to be represented. For example, given a frequency of 7692 Hz, the smallest power of four that can be divided into this frequency value and still retain a fraction is $4^7 = 16384$. Since the exponent of the power of four is stored in a "less one" format, the value of the exponent stored in the upper 3 bits of the floating point number is six. The mantissa value is the frequency as a fraction of the power-of-four value stored in the exponent. It is a 13 bit integer that is a fraction of 8191 (1FFFh, where "h" stands for hexadecimal). In other words, dividing the value in the mantissa by 8191 and multiplying the answer by four raised to the exponent plus one power will result in the original frequency. Thus, for the example shown above, the fractional mantissa would be:

$$\frac{7692}{16384} = .4694824$$

This fractional mantissa would be stored in the 13 available bits as 3845 decimal or 0F05h. Therefore, the final pseudo-floating point value produced for a frequency of 7692 Hz would be:

1100111100000101 = CF05

[0117] An overview of this pseudo-floating point conversion process is shown in Figure 8N. In this regard, Figure 8O provides a detailed flow chart of the block 380 for converting number of pulses data to a 24 bit mantissa. Similarly, Figure 8P provides a detailed flow chart of the block 382 for converting the 24 bit mantissa to the 16 bit pseudo-floating

point form. Finally, Figure 8Q illustrates a flow chart of the block 384 for making an adjustment when the frequency value is less than 1 Hz.

[0118] With respect to Figure 8O, the following should be noted. If there were not any pulses (block 386), then the 24 bit frequency mantissa value is stored as zero (block 388). If the number of pulses (that is, Total Pulses) is less than 255, than the exponent value "EXP" is set to zero, and the constant "K" is set to 800h (block 390). The variable "RPS" stands for Reads Per Second, and this is the number of interrupts that occur every second (that is, 1999 dec). The constants 800h and 08h are necessary to slide the 24 bit answer to the proper position so that no resolution is lost when doing the conversion to the 16 bit pseudo-floating point value. These constants will slide the value of 1 out of the 12th bit position where it belongs in the 16 bit pseudo-floating point value. The use of these constants also has the added advantage of allowing greater precision since more bits are calculated before they exceed the limits of the divide routine.

[0119] As illustrated in the flow chart 382 of Figure 8P, the conversion to the pseudo-floating point value is accomplished by polling the 14th and higher bits of the 24 bit result. If any of them are not zero, the result is shifted to the right by two places (that is, divided by four), and the exponent is increased by one (block 392). This shifting process is continued until bits 14, 15 and 16 are zero. Once the result is reduced to 13 bits (block 394), the final bit shifted off is rounded back into the 13 bits. When the bit is one, a one is added to the 13 bit mantissa (block 396). This reduces the error of the pseudo-floating point number to .025%. Finally, the exponent is ORed into the upper 3 bits (16, 15, 14) of the 16 bit frequency value (block 398). If the final result is greater than 65535, the output is forced to positive full value, 65535. If the final result is less than one, the output is forced to the representation of one, as indicated by the flow chart 384 of Figure 8Q. The process control computer 14 may then average the pulses over many seconds in order to obtain a true frequency value.

[0120] Referring to Figures 9A-9D, a schematic diagram for a multi-functional bridge circuit 400 according to the present invention is shown. The bridge circuit 400 may be used to measure 5 individual temperature or weight values. Specifically, the bridge circuit 400 is designed to accept standard platinum resistance temperature devices "RTDs" or heavy duty RTD's when the circuit is placed in the temperature measuring configuration using the switch "CSW1" of Figure 9D. Additionally, when the bridge circuit 400 is placed in the weight measuring configuration, the circuit will accept the wire terminations of a weight cell (for example, A-D excitation, and B-C mv input with B positive). As indicated in Figure 9D, the switch setting also permits the bridge circuit 400 to inform the controller 100 that the temperature should be recorded in a Celsius or Fahrenheit format.

[0121] Figure 9C shows a voltage source circuit 402 for providing electrical power to the temperature/weight transducers. Figure 9B shows the multiple-wire input signal filtering provided to the bridge circuit for each of these transducers. As shown in Figure 9A, these input signals are directed to multiplexors (506A) "CU1-CU2". The output signal from multiplexor CU2 is coupled to an operational amplifier (3140A) "CU5", which is shown to be in a voltage follower configuration. The output signal "MAI-L" from the operational amplifier CU5 is transmitted to the main multiplexor U11 of controller 100.

[0122] The output signal from the multiplexor CU2 also provides one input to the differential amplifier circuit (AD521) "CU3". The other input to the differential amplifier circuit CU3 is received from the multiplexor CU1. The output from the differential amplifier CU3 is amplified via operational amplifier (3140A) CU4 and directed to the main multiplexor U11 of controller 100 as signal "MAI-H". Figure 8A also shows a precision resistor assembly (S2CH) "CU6", which is comprised of a set of resistors used for calibration and gain purposes.

[0123] Referring to Figure 10A, a simplified block diagram of a portion of the triply redundant field computer is shown to particularly illustrate the abort circuits for the digital output signals. In this regard, a set of abort circuits are located on each of the digital output circuits 500-504. As should be appreciated from Figure 10A, each of the controllers 92-96 is provided with its own digital output circuit. Accordingly, it should be understood that a field computer unit 12 contains a set of three redundant digital output circuits 500-504 whenever digital output signals are to be sent to the field. While each of these redundant digital output circuits preferably has a plurality of output signal channels (for example, 1-10 individual output signal channels), only one such channel is shown in Figure 10A for illustration purposes.

[0124] Each of the controllers 92-96 transmits a "SET DODC" signal to their respective digital output circuits 500-504 for each digital output signal to be sent to the field. Each of these SET DODC signals represents the result of an arbitration process which is individually performed at each of the controllers 92-96. As indicated above, the digital output value signals received by the field computer unit 12 from the process control computers 14a-14b are shared with each of the redundant controllers 92-96. Assuming that the transmission of any particular digital output signal value (that is, a High or Low value) has been completely successful and all of the controllers 92-96 have correctly processed this value, then the "SET DODC-L", 'SET DODC-M" and "SET DODC-R" signals will be identical. The "L", "M" and "R" suffix is simply used herein to indicate that the signal originated from the Left, Middle or Right controller. However, there may be instances when these SET DODC signals are not the same. Additionally, there may be instances when it is desirable for the digital output signal from a particular digital output circuit to be prevented from being transmitted to the field.

[0125] As indicated by Figure 10A, the output conductors from each of the digital output circuits 500-504 are tied together at a common node 506, which is connected to a digitally controlled device 508 (for example, a solenoid). This means that if the output signal from any one of the digital output circuits 500-504 is High, then the device 508 could receive a High input signal, even though the other two digital output circuits are generating Low output signals. However, such a situation is prevented from occurring in accordance with the present invention through the combined use of redundant abort circuits 510-514.

[0126] As shown in Figure 10A, each of the abort circuits 510-514 includes a set of three electronically controlled switches 516-520 (for example, MOSFET devices). The switch 516 is controlled by the SET DODC signal. However, even though the switch 516 may be closed, a High output signal (for example, 26 volts) cannot be transmitted to the device 508 unless at least one of the switches 518-520 is also closed. The switches 518-520 are controlled from the "ABORT" signals generated by the other two neighboring controllers. For example, in the case of the abort circuit 510, the switch 518 is controlled by the "ABORT R-L" signal from controller 96, and the switch 520 is controlled by the "ABORT M-L" signal from the controller 94. As illustrated in Figure 6C, these ABORT signals are determined individually by the microprocessor U40 of each controller.

[0127] Thus, it should be appreciated that in order for the controller 92 to transmit a High SET DODC-L signal to the field, it needs the concurrence or agreement of either the controller 94 (through a High ABORT M-L signal) or the controller 96 (through a High ABORT R-L signal). In this way, the software arbitration decisions by the controllers 92-96 are enforced in the digital output circuits 500-504 through the abort circuits 510-514. If the controllers 94-96 determine that a particular digital output signal from controller 92 should be prevented from being transmitted to the field, then each of the controllers 94-96 will generate a Low ABORT signal for that particular digital output signal, which will open the abort switches 518-520.

[0128] Each of the digital output circuits 500-504 includes a "TEST" line, such as the TEST line 522 for digital output circuit 500. A diode, such as diode 524, is also included to isolate the digital output circuit (and hence the TEST line) from the common voltage seen by the device 508. A TRACK feedback line 526 is also provided in order to permit each of the controllers 92-96 to see the actual digital state presented as an input to the device 508. As will be more fully described in connection with Figures 11 A-11C, the digital output circuits 500-504 are designed to facilitate non-intrusive testing. The method of non-intrusively testing the digital output circuits 500-504 will be discussed in connection with Figures 19A-19M.

[0129] Referring to Figure 10B, a block diagram is shown of the redundant analog output circuits 600-604 according to the present invention. In this regard, a detailed block diagram is presented for the analog output circuit 600, while a single block is used to illustrate the identical analog output circuits 602-604 for neighboring controllers. Due to the detail presented in the block diagram for analog output circuit 600, the discussion of the schematic diagram for this circuit, as shown in Figures 12A-12G may be somewhat abbreviated. In any event, Figure 10B illustrates that the analog output circuit 600 includes an abort circuit 606 for each analog output signal channel contained in the analog output circuit (for example, 5 independent channels). The abort circuit 606 is similar to the abort circuit 510 discussed above, insofar as the abort switches DN1-DN2 correspond generally in placement to the switches 520-518. However, an amplifier is used in the place of the switch 516, as an analog signal rather than a digital signal is to be transmitted to the field. Additionally, opto-isolators are used as the abort switches instead of MOSFETs. Accordingly, it should be appreciated that each of the redundant analog output circuits 600-604 are provided with an abort circuit for the same reason that an abort circuit is provided in the digital output circuits 500-504.

[0130] The analog output circuit 600 receives instructions from its controller, which is generically indicated in Figure 10B as controller 100. In this regard, the analog output circuit 600 receives a desired output value for each channel from its controller, and the analog output circuit is left by the controller to determine how this output value is to be achieved. For this reason and for the analog output circuit's ability to conduct non-intrusive testing on its own, the analog output circuit is considered to be a "smart" circuit that frees the controller 100 to perform other needed functions in the meantime. In order to achieve these goals, the analog output circuit 600 is provided with a microprocessor and the necessary support circuitry to operate with relative independence from the controller 100, as indicated by block 610.

[0131] The capacity for intelligent independence in accordance with the present invention is also important from the standpoint of determining how a common field device should be driven from three concurrently operating analog output circuits to a common output value. This is a particularly difficult problem where, as here, a rapid response to changing conditions is desired. In this regard, each of the analog output circuits 600-604 will be commanded by their respective controllers 92-96 to achieve a desired output value on each channel. Accordingly, each of the analog output circuits 600-604 will want to drive the field device in response to a goal output value independently given to them by their own controller once each process control cycle (for example, one second). Thus, an unstable output could result, since it is also desired that the analog output circuits operate with relative independence from each other during the process control cycle for fault tolerance purposes. However, in accordance with the present invention, intelligent, yet independent methods of controlling the output are provided for each of the analog output circuits through the microcomputer control circuit 610. In accordance with these methods, not only is output level sharing optimally achieved, but each of

the analog output circuits is able to respond at high speed to changing conditions.

[0132] The microprocessor for the analog output circuit digitally transmits multiplexed output signal voltage values for each of the actively operating output channels to a digital to analog converter circuit 612. The analog output values from the digital to analog converter circuit 612 are then sequentially processed through an amplifier circuit 614, and forwarded to a multiplexor circuit 616. The multiplexor circuit 616 then directs the amplified analog output signals to the appropriate abort circuits, such as the abort circuit 606 for the "AO-1" signal.

[0133] As in the case of the abort circuits 510-514 for the digital output circuits 500-504, each of the abort circuits for the analog output circuits include a provision for creating a feedback signal. With respect to the abort circuit 606, this feedback provision is shown to be comprised of a resistor 618 and a pair of signal lines 620-622. The signal line 620 provides a high feedback signal "MEH-1" on the upstream side of the resistor 618, and the signal line 622 provides a low feedback signal "MEL-1" on the downstream side of the resistor 618. Additionally, a Track resistor 624 and a pair of signal lines 626-628 are provided by the field computer unit 12 in order to permit each of the analog output circuits 600-604 to see the actual analog output signal value being received at an analog controlled output device 630. The signal line 626 provides a high track signal "AOT-H-1" on the upstream side of the Track resistor 624, and the signal line 628 provides a low track signal 'AOT-L-1' on the downstream side of the resistor 624. Additionally, the abort circuit 606 is also shown to include a signal line 632 which provides a feedback signal "OAT-1" immediately following the amplifier 608. In this way, the operability of the analog output circuit 600 up to this point may be tested with both of the abort switches DN1-DN2 in an open condition in accordance with the non-intrusive testing method to be described below.

[0134] As illustrated in Figure 10B, the analog output circuit 600 includes a pair of multiplexor circuits 634-636 which feed a differential amplifier 638. The multiplexor circuits 634-636 operate under the address instructions from the microprocessor of the analog output circuit to successively pair corresponding High/Low signals as an input to the differential amplifier 638 to produce a signal indicative of the voltage drop across the feedback and track resistors, which is directly proportional to the output being sent to the field. Thus, for example, the MEH-1 signal would be presented at the output of the multiplexor 634 at the same time that the MEL-1 signal is presented at the output of the multiplexor 636. After an amplification step, a final multiplexor 640 is then employed to successively transmit these differential voltage signals, the "OAT-1..OAT-5" signals, or the mulitplexor outputs referenced to ground to an analog to digital convener circuit 642. The analog to digital converter circuit 642 is in turn connected to the microprocessor block 610 for analysis.

[0135] The analog output circuit 600 is preferably a 5 channel (0-22 ma) circuit device which is capable of testing it's outputs in such a way that the testing is non-intrusive to the field. The analog output circuit 600 is also designed to be a high speed device, so that if one of the three redundant analog output circuits 600-604 fails, then the other analog output circuits will pick up the additional load within a relatively short period of time (for example, 80 msec.). The operation of the analog output circuit 600 may best be described as providing a proportional integral "PI" control loop, as the circuit responds to an output value (for example, a setpoint) received from the controller 100. This output value is preferably a fraction or percentage of the maximum output capability (for example, 22 ma). As mentioned above, the actual field output is measured by each of the redundant analog output circuits 600-604 across the Track resistor 624, which is located on the passive element board of the field computer unit 12. In order to filter out any noise that might appear on the Track signal, one fourth of the difference between the last Track value and this measurement is added to the last Track value. If the difference is greater than 8%, the old Track value is completely replaced in order to speed the system's response to large errors.

[0136] The software control loop of the analog output circuit 600 involves a comparison between the voltage across the Track resistor 624 and the desired output value. A fraction of the error between the desired output and measured Track values (up to one fourth) is then added to the desired digital to analog output value (that is, the integral value), which is stored in the memory of the microprocessor for the analog output circuit 600. This enhanced value is then transmitted to the digital to analog converter circuit 612 and processed through the multiplexor 616 to the designated abort circuit (for example, abort circuit 606). The analog output circuit 600 then determines its contribution to the total output provided to the field device 630 by measuring the voltage drop across the "ME" feedback resistor 618. This is done to assure that the analog output circuit 600 is contributing 100% of the output to the field device 630 during the non-intrusive testing method described below. The analog output circuit 600 also compares the OAT signal to the output of the digital to analog converter circuit 616 (via its "DAC-OUT" signal shown in Figure 10B), to determine whether or not the operational amplifier 608 is operating property. For example, if too much power is being transmitted to the field device 630, and this channel's output should be zero, but the OAT measurement says that it is not zero, the analog output circuit 600 disables this channel and flags an "OAT<>DAC" signal to the controller 100.

[0137] The analog output circuit 600 also provides for the automated application of abort switches (for example, abort switches DN1-DN2) in the event of a failure which sends too much power to the field. The primary path for opening an abort switch is a zero output ensurance mechanism which forces the abort switches open for a channel when that channel is commanded to have a zero output. The secondary path for opening the abort switches is derived from a

request of one or more of the analog output circuits 600-604. For example, in the event that a particular output channel for an analog output circuit is 2% too high, according to the analog output circuit's own analysis, then this analog output circuit will request its controller to have the offending output channel be aborted by opening either of the abort switches DN1-DN2. However, as these abort switches are responsive to the neighboring controllers, an exchange of abort request information is required at the controller level. In accordance with one form of the present invention, the exchange of abort requests between each of the controllers 92-96 takes place during the next output communication cycle (for example, in the next process control cycle). If any two controllers 92-96 agree that a particular channel for one of the analog output circuits 600-604 should be disabled, then these controllers will generate the necessary signals to open both of the abort switches DN1-DN2 on the offending analog output circuit. If an analog output circuit requests an abort on a particular output channel, and neither of the neighboring controllers have requested an abort on the same channel, then an abort disagreement has occurred. These disagreements are preferably handled by counting the number of sequential disagreements on a particular channel and flagging an error to the process control computers 14a-14b when the count exceeds a predetermined value (for example, 32 decimal, 20 hex). When there is no abort disagreement on a particular channel, the counter for that channel is zeroed. It should be appreciated that the secondary path for opening the abort switches enforces the arbitration decisions made by each of the controllers 100. Accordingly, it is not necessary for any of the three analog output circuits 600-604 to know the arbitrated output values that were sent to the other analog output circuits by neighboring controllers.

[0138] Additionally, if an analog output circuit is determined to be dead, the neighboring controllers will open the abort switches for all of the channels on the dead analog output circuit to isolate this circuit from the field. In this regard, an analog output circuit will be considered dead if the smart analog output board is not communicating, if a memory test of the circuit has failed, if a test of the digital to analog converter circuit 612 has failed, or if a test of the analog to digital converter circuit 642 has failed. The controller 100 responsible for the "dead" analog output circuit will not open the abort switches of the neighboring analog output circuits due to a loss of its own analog output circuit. Rather, this controller will examine the controller to controller communications to determine if the opening of these other abort switches is warranted. This will permit a 3-2-1 failure scenario, rather than a 3-2-0 failure procedure. Accordingly, in the event that only one working analog output circuit remains, then no aborts on the operating channels for that analog output circuit will be opened, unless an output is commanded to zero.

[0139] In the event of a controller to controller communication failure, the abort switches for the analog output circuit corresponding to the controller 100 that did not communicate will not be opened. This procedure permits the fail SAFE/LAST mechanism described below to work property. The two remaining controllers that are able to communicate will then act as a dual redundant field computer unit, where only one abort request is needed to open an abort circuit. If both neighboring controllers fail to communicate, then an abort request will not be serviced, and the fail SAFE/LAST selections in software arbitration will control the outputs from the field computer unit for all of the analog outputs.

[0140] Once a pair of abort switches have been opened due to an excessively high output, it is preferred that these abort switches be closed only after a replacement of the analog output circuit is sensed or the controller 100 for that analog output circuit is restarted. The exception to this procedure occurs in the case where there is a triple abort request for a particular output channel. In such an occurrence, all of the abort switches for this channel are reclosed to prevent a total loss of power to the field.

[0141] From the above discussion, it should be appreciated that a failure associated with one or more output channels may take two process control cycles to open the appropriate abort switches DN1-DN2. Thus, for example, where an overall process cycle of one second is provided, then a one second period will be used to communicate an abort request to the controllers from the analog output circuits, and then another one second period will be used to permit controller to controller communication. Nevertheless, an abort on zero output to the field will take place in the same cycle that the controllers 92-96 receive a zero output value from the process control computers 14a-14b.

[0142] Referring to Figures 11A-11C, a schematic diagram for the digital output circuits 500-504 is shown. Figure 11A provides a schematic diagram of the abort circuit 510, which was diagrammatically illustrated in Figure 10A. Again, it should be noted that such an abort circuit is provided for each digital output channel of the field computer unit 12. In other words, in a field computer unit having ten digital output channels, a set of ten abort circuits would be provided for each of the three controllers 92-96, thereby providing a total of thirty abort circuits.

[0143] Figure 11A shows that the switches 516-520 are each comprised of a MOSFET (IRFD120) transistor. Each of these transistors receive their gate signals from an opto-isolator, such as opto-isolator (PS2603) DU1 for transistor 516. The "SET_DODC-1" input signal for the opto-isolator DU1 generally corresponds to the "SET DODC-L" signal of Figure 10A. Similarly, the "ABORT1-1" input signal corresponds to the "ABORT R-L" of Figure 10A, and the "ABORT2-1" input signal corresponds to the 'ABORT M-L" signal of Figure 10A. The parallel connection of transistors 518-520 in Figure 10A is demonstrated in Figure 11A by the fact that the drain and source terminals of these two transistors are tied together. The source terminal of transistor 516 is also connected to the drain terminals of the transistors 518-520, and the drain terminal of transistor 516 is connected to the +26 volt power supply "DPS1" (shown in Figure 11 C) through fuse "DF1". In other words, the transistor 516 is connected in series with both transistors 518 and 520. Pull

down resistor (100K) RP7 and diode (1N459A) 524 are connected to the source terminals of transistors 518-520 to provide the output line labeled "DODC-1" on the downstream side of diode 524. Thus, it should be appreciated that when transistor 516 is turned on by a High SET_DODC-1 signal and at least one of the transistors 518-520 are turned on by their respective gate signals, then the conductive states of these transistors will permit current to flow from the +26 volt power supply to the DODC-1 output line. Since the conduction of the transistor 516 is required to transmit electrical power to the field device 508, this transistor may be referred to as a power switch. In contrast, the transistors 518-520 may be referred to as abort switches, as these transistors operate in combination to inhibit or prevent electrical power from being transmitted to the field device when the power switch is closed (that is, the transistor 516 is in a conductive or On state).

[0144] As indicated above, the digital output circuits 500-504 are designed to enable non-intrusive testing to be performed. In this regard, it should be noted that the abort circuit 510 includes a resistor (10K) RP1 connected in parallel across the drain and source terminals of the transistor 516, and a resistor (10K) RP3 connected in parallel across the drain and source terminals of the transistor 520. Additionally, Figure 11 A shows that the TEST-1 line 522 is connected to the node or junction which is provided between the source terminals of the transistors 518-520, the pull down resistor RP7 and the anode of diode 524. Accordingly, it should be appreciated that the resistors RP1, RP3 and RP7 provide a voltage divider network which will enable the transistors 516-520 to be selectively actuated and the change in voltage detected via the TEST-1 line. For example, when the transistor 516 is turned on, the voltage on the TEST-1 line will rise, as the resistor RP1 is effectively short-circuited by this transistor. Similarly, when either of the transistors 518-520 are turned on, the voltage on the TEST-1 line will rise, as the resistor RP3 is effectively short-circuited by the conducting transistor. Nevertheless, substantial current is not permitted to flow through the DODC-1 line unless the transistor 516 and one of the transistors 518-520 are switched to a conductive state.

[0145] Figure 11 B shows a feedback circuit 526 for the digital output circuit 500. The feedback circuit 526 includes a pair of multiplexor circuits DU33 and DU35 which are addressed by the controller 100 through the address lines HDEV-0..HDEV-3 and the enable line HP3-5. The TEST lines for each of the digital output channels are connected as input signals to the multiplexor DU33, while the DODC signals for each of these channels are connected as input signals to the multiplexor DU35. The output lines 528-530 from the multiplexors DU33 and DU35, respectively, are coupled together, and the multiplexed feedback signals on these output lines are then processed through a pair of operational amplifiers (3140A) DU32 and DU31 which are connected in series. Accordingly, it should be appreciated that each of the digital output circuits 500-504 provide a serially multiplexed stream of feedback signals to their respective controllers 92-96.

[0146] Referring to Figures 12A-12F, a schematic diagram for the analog output circuits 600-604 is shown. Figure 12A provides a schematic diagram of the microcomputer circuit shown as block 610 in Figure 10B. The microcomputer circuit 610 includes a 16MHz microprocessor (80C31) EU3, a memory address latch circuit (HC573) EU2, an 8Kx8 CMOS EPROM (57C64) EU1, and a programmable logic device (EP910) EU4. The microprocessor EU3 receives the output value for each of the analog output channels on the serial RXDATA line from the controller 100, and the microprocessor transmits status data to the controller on the serial TXDATA line. The EPROM EU1 is used to store the operating program for the analog output circuit 600. The PLD EU4 is used to generate various signals which control the functions of specific portions of the analog output circuit 600. For example, the "DACWR" and "DACA" signals from the PLD EU4 are transmitted to the digital to analog converter circuit 612 of Figure 128 in order to cause the D/A converter to capture a digitally coded analog value on the data bus (DATA <7 0>) of the microprocessor EU3 and convert this coded value to a corresponding analog level.

[0147] The microcomputer circuit 610 also includes Green and Red LEDs to provide a visual indication of the health status of the analog output circuit 600 (sometimes referred to as the SAO board for "Smart Analog Output'). If the board is functioning property, the Red LED will be OFF and the Green LED will be ON. However, the microprocessor of the controller will cause the Green LED to flash under certain conditions, such as when the communications between the analog output circuit 600 and its controller 100 have failed. Similarly, the Red LED may be caused to flash when the microprocessor circuit 610 is not functioning property or it is trying to communicate with its controller 100. The Red LED will be turned ON under several possible conditions, such as if a non-intrusive test has failed, a channel on the SAO board has been aborted, or a track problem has been detected. Conversely, the Green LED will be turned OFF if a hardware component of the SAO board has failed or a failure of the controller 100 has occurred. Accordingly, it should be appreciated that these status LEDs are preferably put to multiple uses, so that a variety of different problems may be visually discerned during a field inspection from just two LEDs.

[0148] Figure 12B shows the digital to analog converter circuit 612, amplifier circuit 614 and multiplexor circuit 616 discussed in connection with Figure 10B. In this regard, it should be noted that the D/A converter 612 (AD7248) has a resolution of 12 bits, but it need not be designed for absolute accuracy. Rather, in accordance with the control methods of the present invention, the accuracy of the D/A converter 612 is not nearly as important as the ability to make small changes.

[0149] The amplifier circuit 614 is comprised of an operational amplifier EU34 (3140A). This single stage amplifier

provides a "2.21" multiplier that boosts the 10 volt maximum output to a maximum of 22.1 volts. In this regard, it is preferred that a 1.21k ohm resistor be employed in the feedback leg between the output and the inverting input of the operational amplifier. This provision prevents a differential input greater than 10 volts by limiting the amount of current that can be drawn through the non-inverting input, and thus preventing the device from being put into a positive feedback mode that could take several seconds to recover from. This provision also allows the amplifier circuit, in conjunction with the 1.21k ohm resistor, to amplify its input by 2.21.

[0150] Figure 12C illustrates the abort circuit 606 which was discussed in connection with Figure 10B. In this regard, the operational amplifier (3140A) EU15 or 608 is responsive to the "SET-AO1" signal from the multiplexor 616. However, the abort circuit includes provisions to prevent electrical power from being transmitted to the field if either the microcomputer circuit 610 or the controller 100 fail to operate properly. Specifically, the operational amplifier EU15 may be disabled by the conduction of the transistor EQ3 via a Low signal on the appropriate pin of the "P1" bus of the microprocessor EU3. In other words, the analog output circuit 600 may pull its own analog output to zero. Additionally, the presence of a Low "DEADMAN" signal from the deadman timer circuit 649 of Figure 12D will also cause the analog output from the operational amplifier EU15 to be pulled to zero. The timer (LS122) EU9 of the deadman timer circuit 649 is responsive to periodic "DEADSET" signal pulses from the controller 100 to maintain the DEADMAN signal in a High state. Thus, if a DEADSET pulse is not received within a predetermined period of time (for example, 64 msec), then the analog output circuit 600 will automatically pull down all of its analog output lines to zero.

[0151] As in the case of the digital abort circuits 510, the analog abort circuit 606 includes opto-isolators (EU32-EU33) to electrically insulate the analog output circuit 600 from its neighboring analog output circuits 602-604. However, these opto-isolators (ILD31) are also capable of passing current to drive the field control device to which the analog output circuit is connected. Accordingly, the output line 646 from the operational amplifier EU15 is connected to the collector terminal of the transistor in each of the opto-isolators EU32-EU33. Additionally, it should be noted that the abort circuit 606 includes a diode 648 which separates the ME resistor 618 from the track resistor 624.

[0152] Figure 12E indicates that the multiplexor circuit 634 of Figure 10B is actually comprised of multiplexors EU24 and EU26. Similarly, the multiplexor circuit 636 of Figure 10B is shown to be comprised of multiplexors EU23 and EU25. Accordingly, the differential amplifier circuit 638 is also comprised of a set of five operational amplifiers (OPA2107) EU11, (OPA2107) EU21 and (OPA602) EU12. The operational amplifiers EU11 provide the multiplexed "OUT-L" and "OUT-H" signals from the ME and track resistors that allow the measurement of these signals with respect to ground. The operational amplifiers EU21 buffer the output of the multiplexors as the first stage of the differential amplifier 638 formed by operational amplifiers EU21 and EU12. The "A/D_IN" signal produced by the differential amplifier 638 represents an amplified voltage difference between the outputs of the multiplexors (for example, amplified by 4.545).

[0153] The differential amplifier circuit 638 provides a gain of 4.545 in order to convert the 2.2 volt maximum track differential to 10 volts. This amplification permits the entire range of the analog converter 642 to be utilized. Additionally, it should be noted that the operational amplifiers have negative and positive rails of -5 volts and +26 volts respectively. In this regard, the operational amplifiers operate within 5 volts of the negative rail and 3.0 volts of the positive rail. The operational amplifiers should also have a slew rate greater than 1 volt/msec, and as low a voltage offset as possible. In this way, the differential amplifier circuit 638 has the ability to operate relatively fast, perform well near the supply rails and reject common mode voltages across a wide range.

[0154] Figure 12F completes the analog output circuit 600 by receiving the OUT-L, OUT-H and A/D_IN signals and further multiplexing these signals with the OAT-1..OAT-5 signals. The analog output of the multiplexor 640 is processed through operational amplifier (OPA602) EU5, and then converted into a digital signal stream by A/D converter (ADS574) 642. The A/D converter 642 is in turn connected to the DATA <7 0> bus of the microprocessor EU3 of the analog output circuit 600.

[0155] Referring Figures 13A-13D, a schematic diagram for the network controller 16 is shown. As indicated above, the network controller 16 serves as the communication director for the entire fiber optic network, and it preferably has the capability to communicate at a rate of at least 500K baud. The network controller 16 is equipped with its own microcomputer circuit 800, as illustrated in Figure 13A. The microcomputer circuit 800 includes a microprocessor (80C31BH-1) FU10, a 32K program memory FU11, a 32K data memory chip FU6, a PLD memory controller chip FU5 and latch chips FU2-FU3. In this respect, the microcomputer circuit 800 is similar in design to that shown for the controller 100 in Figure 6A, and the same or similar components may be used in both circuit designs. A 16MHz oscillator circuit 802 is also shown to be connected to the microprocessor FU10, which serves to point out that the network controller 16 operates under its own clock, even though the microprocessor FU10 receives a "MODSYNCIN" synchronization signal from the process control computer 14.

[0156] The network controller 16 is connected to its process control computer 14 via a 16-bit wide "B" bus, which is shown in Figure 13B. The network controller 16 also receives a set of encoded control signals ("MOD-D0..MOD-D3", "MOD-CP" and "MOD-ST") from the process control computer 14 which facilitate communication between these two computer systems. In this regard, these encoded control signals are connected to a decoder circuit (22V10) FU13,

which deciphers these control signals and directs these control signals to the circuits indicated in Figures 13A-13B. Thus, for example, the "/MODSETDATA" signal is sent to a pair of three-state flip flop circuits (74HT574) FU14-FU15 in order to capture data presented on the "B" bus. Similarly, the "/MODREADATA" signal is sent to a pair of latch circuits FU16-FU17 in order to enable these latch circuits to pass data captured from the "P0" bus of the microprocessor FU10 to the "B" bus of the process control computer. The flip flop circuits FU14-FU17 also receive enable/clock signals from a 3 to 8 decoder circuit (74HC138) FU4, which is connected to the "AD" bus of the microprocessor FU10.

[0157] Figure 13B also shows that a flip flop circuit FU18 provides a further input interface between the "B" bus from the process control computer 14 and the "P0" bus of the microprocessor FU10 of the network controller 16. In this regard, the process control computer transmits a SETCODE signal to the network controller 16 which is used to indicate to the network controller 16 what data elements were loaded into the flip flop circuits FU14-FU15 by the process control computer 14. Additionally, the process control computer 14 sends a predetermined set code value (for example, 10hex) to flip-flop circuit FU18, which is used to indicate the start of a new process control cycle (for example, a new second). During the anticipated time that this code should be transmitted, the network controller 16 repeatedly polls the flip-flop circuit FU18 in a tight loop in order to detect the start of a new process control cycle. When the new process control cycle set code is detected, then the microprocessor FU10 will read and store its own corresponding clock signal. Then, the microprocessor FU10 will change the appropriate register which stores the clock data by an amount which will enable the clock signal of the network controller 16 to be adjusted to that of the process control computer 14. Finally, Figure 13B shows a decoder circuit (74HC541) FU1 which is connected to the keyboard of the debug panel 18 for the network controller 16 via signal lines "KEY0..KEY3". Communication to the debug panel 18 is provided by the RPDBUG signals shown in Figure 13A. Thus, it should be appreciated that the circuits illustrated in Figure 13B provide a way to effectively make multiplexed use of the "P0" bus of the microprocessor FU10 for purposes of bi-directional communication with the process control computer 14 and bi-directional communication with the debug panel 18.

[0158] Figure 13C shows a receiver circuit 804 for the network controller 16. The receiver circuit 804 generally comprises a multiplexor circuit FU8, a digital to analog converter circuit FU12 and a comparator circuit FU7. The multiplexor circuit FU8 is connected to an "RXD" bus, which is essentially a set of individual signal lines that extend from an edge connector on the network controller circuit board. These signal lines include the "MAIN_RXD" and the "REPEAT_RXD" signal lines which illustrate the network controller's ability to communicate in opposite directions. In this regard, the MAIN_RXD line is ultimately connected to both of the two fiber optic cables 34 shown in Figure 1 through an interface circuit to be described below. Similarly, the REPEAT_RXD line is ultimately connected to both of the two fiber optic cables 36. In this way, both of the cables in each network ring are utilized to form one communication link. Additionally, the multiplexor FU8 also receives the signal lines labeled "NEIGH1_RXD" and "NEIGH2_RXD". One of these NEIGHbor lines could be used to receive high speed optical communication between the process control computers 14a-14b. The other of these NEIGHbor lines is also available to facilitate such communication when the process control computer 14 is comprised of three redundant process control computers. Alternatively, these NEIGHbor signal lines could be used to provide additional redundant communication links between the process control computers.

[0159] As in the case of many of the input signals for the controller 100, the digital to analog converter circuit FU12 and the comparator circuit FU7 operate in combination to produce an "RXDATA" signal which is connected to the microprocessor FU10. This arrangement permits a plurality of both analog and digital signals to be processed through the same circuitry, which ultimately generates a single input line to the microprocessor FU10.

[0160] Figure 13D shows a transmitter circuit 806 for the network controller 16. Specifically, the transmitter circuit 806 is shown to be comprised of a decoder/demultiplexor circuit (74HC138) FU9. The decoder circuit FU9 is connected to the address bus "P1" of the microprocessor FU10, and the decoder circuit also receives the "TXDATA" signal from the microprocessor for transmitting signals to the fiber optic network. The decoder circuit FU9 produces signals which are complimentary to the "RXD" signals discussed in connection with Figure 13C. Specifically, the "MAIN_TXD" signal is ultimately connected to one of the fiber optic cables 34, and the "REPEAT_RXD" signal is ultimately connected to one of the fiber optic cables 36. Similarly, one of the "NEIGH1_TXD"/"NEIGH2_TXD" signals could be used to provide a transmission link between the process control computers 14a-14b.

[0161] Referring to Figures 14A-14E, a schematic diagram of the breakout serial communication circuit 26 is shown. In this regard, the breakout circuit 26 has several circuit similarities to the network controller 16. Specifically, the microcomputer circuit 808 of the breakout circuit 26 (shown in Figure 14A) is similar to the microcomputer circuit 800 for the network controller 16. The microcomputer circuit 808 includes a microprocessor (80C31BH-1) GU10, a 32K program memory GU13, a 32K data memory chip GU11, a PLD memory controller chip GU14 and latch chips GU3 and GU8. Additionally, the transmitter circuit 810 of the breakout circuit 26 (Figure 14D) is similar to the transmitter circuit 806 of the network controller 16, and the receiver circuit 812 of the breakout circuit (Figure 14D) is similar to the transmitter circuit 804 of the network controller.

[0162] Figure 14B shows a power supply circuit 814, which serves to illustrate that the breakout circuit 26 may receive its electrical power from the process control computer 14 (labeled "MOD") or from an external source. Figure 14C shows the connectors "S1..S15" for each of the communication signal lines available on the breakout circuit 26. These

connectors are in turn coupled to fiber optic receiver/transmitter circuits, such as those shown in Figures 15A-15B respectively. Thus, for example the MAIN_RXD and MAIN_TXD signals are coupled through connector S1, and the REPEAT_RXD and REPEAT_TXD signals are coupled through the connector S3. Additionally, as the name "breakout" implies, a set of connectors S6-S15 are provided to direct signals received by the breakout circuit 26 to specific communication channels that are associated with individual field computer units 12.

[0163] Accordingly, it should be appreciated that the breakout circuit 26 has the capability to multiplex or demultiplex communication signals for up to ten individual field computer units 12. Additionally, it should also be appreciated that the breakout circuit 26 may be configured to provide a "repeater" function, such as that shown for the breakout circuit 26e in Figure 2. In this regard, the signals received on the MAIN_RXD line may be processed through the microprocessor GU10 and re-transmitted on the REPEAT_TXD line to the next breakout circuit, such as the breakout circuit 26f of Figure 2. In this way, the breakout circuit 26e may be used as a signal re-transmitter.

[0164] Figure 14E shows a configuration circuit 816, which is used to control the signal directioning function of the breakout circuit 26. Specifically, a pair of switches "GSW1-GSW2" are provided to facilitate the multiplexing/demultiplexing of signals between the main/repeat ports 30-32 of the breakout circuit 26 and the communication channels "CH1..CH10". In one form of the present invention, the switch GSW1 is used to determine a start channel and the switch GSW2 is used to determine a stop channel. Thus, the combination of these two range switches will enable the microprocessor GU10 to know which set of adjacent channels are actively connected to field computer units 12. In contrast, the setting of switch GSW3 informs the microprocessor GU10 whether the breakout circuit is connected on the primary level of signal distribution (for example, breakout circuits 26b and 26d of Figure 1) or whether the breakout circuit is connected on the secondary level of signal distribution (for example, breakout circuits 26a and 26c of Figure 1). The setting of switch GSW3 also informs the microprocessor GU10 as to whether the breakout circuit is being used as a repeater. Additionally, Figure 14E also shows a connector "GS5" which is used to couple the debug panel 56 for the breakout circuit 26 to the microprocessor GU10 via the "RPDBUG" bus.

[0165] Referring to Figures 15A-15B, a schematic diagram of two fiber optic interface circuits are shown. Specifically, Figure 15A shows a receiver circuit 900, and Figure 15B shows a transmitter circuit 902. The receiver circuit 900 includes an optical to electrical converter circuit "HU2" which feeds a high speed comparator circuit (LT1016) "HU4". The high speed comparator HU4 produces a "RX OUT" signal which has an electrically variable component that corresponds to the optically variable component of the optic input signal. When plastic optical fibers are employed to conduct communication signals, it is preferred that an HP-2522 converter be utilized for the converter HU2. However, when glass optical fibers are employed, it is preferred that an HP-2402 converter be employed for the converter HU2.

[0166] The transmitter circuit 902 of Figure 15B includes a NAND gate (75451) HU3 which feeds an electrical to optical signal converter circuit HU1. When plastic optical fibers are employed to conduct communication signals, it is preferred that an HP-1522 converter be utilized for the converter HU1. However, when glass optical fibers are employed, it is preferred that an HP-1404 converter be employed for the converter HU1.

[0167] Referring to Figures 16A-16G, a schematic diagram of the power supply circuit 50 is shown. The power supply circuit 50 is a 500 watt power supply that is capable of powering up to five field computer unit sides. In this regard, it is preferred that one power supply circuit be used to power only corresponding controllers 92-96 in each field computer unit 12. In other words, one of the power supply circuits 50 may be used to provide electrical power to the Left controller 92 in 1-5 field computer units. The power supply circuit 50 may also be used to provide power to one or more of the breakout circuits 26 as well. Additionally, the power supply circuit 50 is also used to charge the batteries 52 from which it may ultimately derive power in the event of an interruption in its A.C. input power. The batteries 52 are preferably a set of two 12 volt sealed batteries which are connected in series.

[0168] The power supply circuit is also preferably contained in its own enclosure, as shown in Figure 1. An enclosure may also be provided to house a field computer unit 12, a set of power supply circuits 50 and a set of batteries 52. The enclosure for the power supply circuit 50 is preferably equipped with a set of LEDs which will indicate the status of various functional aspects of the power supply circuit 50. For example, one LED may be used to indicate that the power supply circuit 50 is receiving A.C. electrical power, while another LED may be used to indicate the battery 52 has sufficient power available. As will be discussed below, the power supply circuit 50 has the ability to test the battery 52 by conducting a load test.

[0169] Figure 16A shows a fan controller circuit 904 which is responsive to the "FANON" signal from the controller 100. The FANON signal will cause the transistor in the opto-isolator circuit IU8 to conduct, and thereby transmit electrical power to a fan in the enclosure for the power supply circuit 50. Power to the fan may also be provided from the signal generated by a pair of temperature sensing devices (AD592), which are connected to pins 1-4 of the connector "S3". If the temperature being sensed in the power supply enclosure is sufficiently high, the temperature sensing devices (not shown) will turn on the fan (also not shown). The POWER-TEMP signal is transmitted back to the controller 100 to allow the controller 100 to monitor the temperature of the power supply and turn on the fan if necessary.

[0170] Figure 16B shows a power converter circuit 906 which may receive either 120 VAC or 240 VAC electrical power. Figure 16B also shows an opto-isolator circuit (H11G2) IU1, which is used to sense that A.C. power is available

to the power supply circuit 50. While not shown in this schematic diagram, a suitable A.C. converter (for example, a Vicor VI-FKE6-CMX circuit) is preferably employed to produce modulated D.C. power on the lines labeled "+HV" and "-HV". A set of three 200 watt power supply circuits (VI-200) "PS3-PS5" are connected in parallel to convert this high voltage input power to a regulated 28 volt D.C. output. A voltage divider circuit "R3-R5" is used to adjust the output voltage to precisely +28 volts. This voltage level is necessary to charge the batteries 52. The batteries 52 are charged through the bank of positive temperature coefficient (PTC) resistors "VR2..VR7", which are used to limit current flow to the batteries. As the batteries 52 draw more current, the PTCs heat up and restrict the flow of current to the batteries.

[0171] The charging voltage is transmitted on conductor line 908 to a relay "K2" on Figure 16C, which is used to connect the batteries 52 to the charger circuit of Figure 16B. In this regard, the positive terminal of one or more sets of batteries 52 is connected to conductor line 910 on the downstream side of the relay K2. The relay K2 is controlled by the "LOAD_TEST-B" signal, which is derived from the controller 100. The LOAD_TEST-B signal is used to cause the batteries 52 to be disconnected from the charging circuit in order to test the state of charge on the batteries. As will be seen below, this test is conducted under load conditions which will reflect the amount of current draw that could occur if the batteries were called upon to provide the primary power source for one or more field computer units 12.

[0172] In order to conduct this "load" test, the batteries 52 are alternately switched between a low current drawing load (for example, 125 ohms) and a high current drawing load (for example, 0.75 ohm). The low current load is provided by (5 watt) resistors R28-R29, while the high current load is provided across pins 3-6 of connector "S4". The high current load may be any resistive device capable of pulling the maximum allowable current from the batteries 52, such as a pair of Dale HLZ-165 1.5 ohm power resistors in parallel. A switch "K1" is used to alternately connect the batteries 52 to the high/low current loads during the testing procedure in response to a "LOAD_TEST-A" signal which is received indirectly from the controller 100. The LOAD_TEST signal resets a (555) timer circuit IU9, which is configured to generate a High signal for approximately 180 seconds. With the polarity shown for the opto-isolator circuits IU7 and IU10, the LOAD_TEST-A and LOAD_TEST-B signals may actually be the same signal from the controller 100. In other words, the batteries 52 will be charged while the LOAD_TEST-B signal is High, and the timer circuit IU9 will be held in a reset condition. However, when the LOAD_TEST-B signal is brought Low, the switch K2 will energize and connect the positive terminal of the batteries 52 to the switch K1. The timer circuit IU9 will then start counting and cause the batteries 52 to be switched to the high current load for approximately 60 seconds. Then, the batteries 52 may be switched to the low current load.

[0173] During the load test, the battery voltage "BATTERY V" will be measured by the controller 100 through isolation circuit (AD202) IU3. In this regard, the discharge voltage of a battery is both a function of the load and the amount of energy stored. Accordingly, the controller 100 will be able to determine the approximate amount of energy stored from the BATTERY V signal and the known resistance value of the high current load. In other words, the controller 100 will direct a load test where the power supply circuit 50 provides the controller with a high current load battery value during a time span of approximately 60 seconds. The low current load may also be used to fully discharge the batteries 52 if needed. The isolation circuit IU3, as well as the isolation circuit IU4, are used to permit the power supply circuit 50 to have two separate GND potentials. The GND potential which is isolated from the battery GND is referred to herein as ISOGND.

[0174] The power supply circuit 50 also generates several other signals which are related to the state of the circuit or the state of the batteries 52. For example, Figure 16C shows that the power supply circuit 50 includes a comparator circuit (LM339) IU6, which generates a "BATT LOW" signal. As the name implies, the BATT LOW signal is indicative of whether the battery voltage is too low (for example, <10 volts). Similarly, a "BATTERY >26V" signal is used to indicate that the battery voltage is too high (for example, over 26.1 volts), via one of the comparator circuits IU6. The "CHARGER V" signal is used to provide the controller 100 with an indication of the voltage being applied to charge the batteries 52. Assuming that this charging voltage is above 25 volts, one of the comparator circuits IU6 will generate a High "CHARGER OK" signal. Since the toggle point of this comparator is set to 4.17 volts by the regulator (AD587) IU5 and the resistors R20 and R23, the CHARGER V signal is divided down across resistors R32-R31.

[0175] Turning to Figure 16D, a control interface circuit 912 for a group of five power supply circuits 50 is shown. The control interface circuit 912 includes a pair of decoder circuits (22V10) JU1-JU2 for interpreting command signals from the controller 100, such as the replicated "FANON" and "ICONSERVE" signals. As will be seen from the discussion below, the ICONSERVE signal is used to turn off the supply of 26 volt power to the field computer units. The "BATOFF" signal is used to turn off the supply of 5 volt power to the field computer units. In this regard, it should be appreciated that the controller 100 may first direct the power supply circuit 50 to conserve battery power by turning off the 26 volt power source, and subsequently shut down the 5 volt power source after a suitable time has elapsed (as determined by the controller 100). The "BAT TEST" signal is used to generate a "LOAD_TEST_ON" signal which corresponds to the LOAD_TEST-A/LOAD_TEST-B signals.

[0176] Figures 16E-16F show a set of connector circuits 914-916 which are replicated for each of the field computer units 12 that are powered by the power supply circuit 50. The connector circuit 914 simply shows the various command signals that are transmitted to each of the field computer units 12. Similarly, the connector circuit 916 shows the trans-

mission of the 26 volt power source and a "VCC" power source to each of the field computer units 12 via fuses "CB1-CB2".

[0177] Figure 16G shows an output power circuit 918 for the power supply circuit 50. The output power circuit 918 includes a power line labeled "VSOURCE" which corresponds to the +28 volt power source output from converters PS3-PS5 of Figure 16B. The VSOURCE line feeds three 150 watt converter circuits (VI-200) "KPS2-KPS4" and a 100 watt converter circuit (VI-200) "KPS1". The converter circuits KPS2-KPS4 combine to produce a +26 volt power source across lines 920-922, while the converter circuit KPS1 produces a +5 volt power source across lines 922-924. It should be noted that jumpers KJ3-KJ4 are provided to connect the output of the +5v power source to the sense circuit of the power source.

[0178] A set of opto-couplers (MOC8021) "KU1-KU4" are used to control the on/off operation of the converter circuits KPS1-KPS4 in response to the "SHUTDOWN" and "5V OFF" command signals. Specifically, a High SHUTDOWN signal (which was derived from the ICONSERVE signal) will cause the opto-isolator circuit KU1 to become non-conductive, and thereby turn on transistor KQ1. This will cause the gate signal input to the converters KPS2-KPS4 to be driven low, and thereby shut these converters off. This will in turn remove the +26 volt power source from the field computer unit. A similar control procedure is also utilized to shut off the +5 volt power source through opto-isolator KU4 and transistor KQ2. Additionally, the opto-isolators KU2-KU3 are responsive to the +28 volt line 926 to simultaneously turn on the converters KPS1-KPS4 when the converter circuits PS3-PS5 of Figure 16B are receiving power from the AC line.

[0179] Referring generally to Figures 17A-17I and Figures 18A-18T, a set of flow charts is shown to illustrate the arbitration methods performed at the field computer unit 12 according to the present invention. Figures 17A-17E relate to the arbitration of digital inputs, and Figures 17F-17I relate to the arbitration of digital outputs. Similarly, Figures 18A-18N relate to the arbitration of analog inputs, and Figures 18O-18T relate to the arbitration of analog outputs.

[0180] In order to put the field computer unit 12 software arbitration methods in perspective, the following observations may be made. These methods represent the procedures according to the present invention for how input and output values are selected in response to both agreements and disagreements between the values provided to each of the three controllers 92-96 contained in the field computer unit 12. In this regard, it is important to understand that these arbitration methods are performed by each of the controllers 92-96. It should also be understood that each of these arbitration methods are performed within each process control cycle (for example, each second).

[0181] In general, the value data used in these arbitration methods must first be validated as an initial step. Then, if the value data (that is, a AO, AI, DI or DO value) from at least two controllers agree, then the Leftmost value is selected. In other words, the AI or DI value determined at the Left controller 92 will be transmitted to the process control computer 14 if the Left controller 92 and the Middle controller 94 agree. Similarly, the AO or DO value determined at the Middle controller 94 will be transmitted to the field if the Middle controller 94 and the Right controller 96 agree. However, as each of the controllers 92-96 perform this arbitration process, it should be appreciated that it is possible that the controllers may transmit arbitrated values from different agreement combinations on a channel by channel basis for both input and output values. Such a situation could occur, for example, as a result of a communication failure to or from one of the controllers 92-96, so that the data values for that controller may not be shared with the other two controllers.

[0182] In the event that three valid data values exist, but none of the three controllers 92-96 are in agreement, then in accordance with the present invention a software selectable default condition is used for that value. In the case of input values, a choice may be made between a Select-High or Select-Low value to be sent to the process control computer 14. In the case of output values, a choice may be made between a Fail-Safe or a Fail-Last value to be sent to the field. One of the advantages of the present invention is that these software selectable default conditions may be rapidly changed in order to provide the most effective process control decisions possible in response to changing conditions in the field. In one form of the present invention, these default value conditions can be changed and are transmitted to the field computers units 12 with each process cycle signal communication for each input and output channel being processed by the field computer unit.

[0183] While these default value conditions are stored in each of the controllers 92-96 so that a communication interruption will not prevent the most current default value conditions from being applied, a procedure is nonetheless provided to ensure that the most appropriate default value conditions will be applied. For example, when a process is first started, the most appropriate output default value condition may be a Fail Safe value (for example, a zero output). Whereas, after the process has been operating properly for some period of time, the most appropriate output default value condition may be the Fail-Last condition. In this regard, the Fail-Last condition applies the last arbitrated data value for the channel in question in the event of a loss of communication from the process control computer 14. When the Fail-Last condition is invoked for an analog output in response to a complete disagreement between valid data, then the value which is numerically nearest the last arbitrated data value will be selected. In the event that no valid data is available for either an input or an output value, then the last arbitrated data value should be used.

[0184] Turning to Figures 17A-17E, the flow charts for the arbitration of digital input data will now be described. Before proceeding to discuss these flow charts, it should be noted that each of the three controllers 92-96 independently

perform this arbitration process. However, the Middle controller 94 will not send its arbitration results to the process control computer 14 unless an additional fiber-optic communication link is provided for this controller. Such a fiber-optic communication link should be utilized, for example, in the event that three process control computers 14 are provided.

[0185] Figure 17A shows an overall flow chart 1000 for the arbitration of digital input data. Block 1002 indicates that the data values for the first 10 digital input channels are loaded into memory. These data values were obtained from the multiplexor U9 of the controller 100 shown in Figure 61. Then, various constants, pointers and counters are initialized to set up the arbitration process (block 1004). Assuming that the digital input circuits are contained on the controller circuit board or the microprocessor U40 detects that a chassis mounted digital input circuit is plugged in, then a 'good bit' is set to indicate that valid data is available (block 1006).

[0186] Diamonds 1008-1010 test whether valid neighbor to neighbor communication messages have been received at the controller (for example, using a checksum calculation). In other words, the controller 92 will test to see if valid data passing messages have been received from the controllers 94-96, while the controller 94 will test to see if valid data passing messages have been received from the controllers 92 and 96. Next, the controller will "get" the valid digital input values for the first channel (block 1011). Then, the valid digital input values for this channel will be converted from "N1" (for example, controller 94), "N2" (for example, controller 96) and "ME" (for example, controller 92) values, to Left, Middle and Right values for arbitration software purposes (block 1012).

[0187] At this point, the flow chart 1000 shows a series of three broken-line boxes 1014-1018 which each represent a separate flow chart. Specifically, the "Determine Send-Low" block 1014 is shown in Figure 17B, the "Determine Which Input to Send" block 1016 is shown in Figures 17C-17D, and the "Set/Clear DIC Bit" block 1018 is shown in Figure 17E. Once the process steps shown in these flow charts are completed, then the arbitrated digital input value for the first channel is stored in a message buffer for transmission to the process control computer 14 (block 1020). The program then repeatedly loops back to get and arbitrate the next digital input channel until all of the digital input values have been arbitrated (block 1022). Again, it should be noted that this process is performed by each of the controllers 92-96, particularly where three process control computers 14 are provided. However, in the embodiment illustrated in Figure 1, only the Left and Right controllers 92 and 96 transmit their arbitration result to their respective process control computers 14a-14b.

[0188] The flow chart 1014 of Figure 17B is directed to determining whether a Low default value should be sent to the process control computer 14. In this regard, the flow chart 1014 checks to see if a valid Send Low bit is available for at least one of the Left, Middle and Right controllers 92-96 (for example, diamonds 1024-1028). Then, the program checks to see if there is an agreement between the valid Send Low bit of two controllers (for example, diamonds 1030-1032). If there is an agreement, then the Leftmost Send Low bit is used (for example, block 1034). However, if there is a disagreement between valid Send Low bits when only two valid Send Low bits exist, then the state of the last valid Send-Low bit will be used (for example, blocks 1036-1038).

[0189] The flow chart 1016 of Figures 17C-17D represents the primary arbitration routine for each of the digital input channels. While the process starts out testing the validity of the Left digital input (block 1040), it should be appreciated that the apparent bias toward the values of the Left controller 92 is not necessary, even though this selection promotes overall system and software uniformity. Assuming that the Left digital input value is valid, the Middle digital input value is checked for validity (block 1042). Then, assuming both values are good, and they match (block 1044), then the Left digital input value will be selected for transmission to the process control computer 14 (block 1046). In other words, if both the Left and Middle controllers 92-94 provide a High digital value, then the digital value stored in memory that represents the Left value will be sent to the data table of values which will ultimately be transmitted to the process control computer 14. Nevertheless, the process does not end at this point, as a Left-Right match determination is made (block 1048) if a valid digital input value is available from the Right controller 96. In the event that there is a disagreement (for example, Left = High, Right = Low), then the Left-Right compare bit "DICLR" will be "set"; that is, the DICLR bit will be provided with a High/one value (block 1050). These specific compare bits may be counted and/or sent to the process control computer 14 with each process control cycle, so that an indication is available of continued disagreements. In this regard, the accumulated compare bits may be used to decide that a service call to the field should be made or that a particular digital input circuit board or controller 100 should be shut down in the appropriate circumstances.

[0190] The remaining portion of the flow chart 1016 generally follows the analysis discussed above. However, it should be noted that block 1052 indicates that an Arbitration Failure bit is set when there is a Left-Middle disagreement and the Right digital input value is not valid. At this point, diamond 1054 indicates that the program checks to see if the process control computer 14 has requested that a Low value be sent as the default value. If the answer is no, then the Left value will be selected if it is High (block 1056), and the Middle value will be selected if the Left value is Low (block 1058). This is because the Middle value must be High, as there was a disagreement with the "Low" Left value. if the Send-Low default value was requested, then the Left value will be checked first to see if it is High (block 1060). As blocks 1058 and 1062 indicate by implication, the Low value will ultimately be sent to the process control computer

14.

[0191] The flow chart 1018 of Figure 17E is directed to determining the state of a general digital input compare bit "DIC". If a disagreement between any two valid digital input values has been detected from the state of the specific compare bits, then the DIC bit will be set (block 1064). Otherwise the DIC bit will be cleared (block 1066).

[0192] Referring to Figure 17F-17I, the arbitration method for the digital output values will now be described. In this regard, it will be seen that the flow charts of Figures 17F-17I generally follow the analysis discussed above for the arbitration of digital input values. Thus, for example, the flow chart 1068 of Figure 17F corresponds to the flow chart 1000 of Figure 17A, and the flow chart 1070 of Figure 17G corresponds to the flow chart 1014 of Figure 17B. However, in the case of flow chart 1070, the determination is made as to whether a "Fail-Last" request has been sent to the field computer unit 12 from the process control computer 14.

[0193] The flow chart 1072 of Figure 17H provides the primary arbitration routine for each of the digital output channels. As the selection of digital outputs generally follows the analysis described in connection with the selection of digital inputs, only a few comments need to be made. Specifically, block 1074 indicates that a specific "Nomatch" bit (that is, the Compare bit) and a 'Negotiation Failure' bit (that is, the "DOAF" bit) will both be set when the only two valid digital output values are not the same. Additionally, block 1076 indicates that the "DOAF" bit will be set in the event that none of the Left, Middle and Right digital output values are valid.

[0194] Block 1076 also indicates that the present invention provides a mechanism in response to a failure of communications. Specifically, a programmable "timeout counter" will be decremented from an initial value, which would otherwise prevent any change in output status to be made until communications have been re-established. In this regard, a desired timeout value may be transmitted from the process control computer 14, which would then be arbitrated by the controllers 92-96 for use as a fail safe timeout counter for all digital and analog outputs. For example, this timeout value may represent the number of seconds before moving from a fail-last status to a fail-safe status. Diamond 1078 is used to test whether a timeout has occurred (for example, a zero counter value). If the timeout has not yet occurred, then diamond 1080 tests whether a Fail-Last default value has been requested. If the Fail-Last default value has been requested, then block 1082 indicates that the last arbitrated digital output value will be sent to the field (for example, digital output circuit 500). If the Fail-Last default value has not been requested, then a Fail-Safe value (for example, a Low, zero or deenergized state) will be sent to the field (block 1084). If a timeout condition has occurred, then diamond 1078 and block 1084 indicate that a Fail-Safe value is sent to the field.

[0195] The flow chart 1086 of Figure 17I generally corresponds to the flow chart 1018 of Figure 17E. However, block 1088 indicates that a general digital output compare bit "DOC" will be set if a disagreement was found between any two controller values for the particular digital output channel being processed. Finally, block 1090 of Figure 17F indicates that the selected digital output value will be stored in a memory table location for subsequent transmission to the appropriate digital output circuit channel.

[0196] Turning to Figures 18A-18N, the flow charts for the arbitration of analog input data will now be described. In this regard, Figures 18A-18B combine to show an overall flow chart 1100 for the arbitration of analog input data. As an initial procedure, block 1102 indicates that the program checks the Family-Type codes from each of the three analog output circuits 600-604. The detailed process steps represented by block 1102 are shown in Figures 18C-18D. Specifically, the program routine starts by checking to see if valid Family-Type codes were received from each of the two sets of analog input circuits (for example, diamonds 1104-1108). Then, the program determines whether or not there is a match between the Family-Type codes for the controller conducting the arbitration and the Family-Type codes for the other two controllers (for example, diamonds 1110-1112). If a match is found, then a specific "OK" bit is set in each instance (for example, blocks 1114-1116). However, if a particular match was not found, such as for the "ME" and "Neighbor1" codes, then a "Nomatch" bit may be set (block 1118 in Figure 18D).

[0197] Now that the controller conducting the arbitration method knows how to process the analog input data, the program flow jumps back to block 1122 of Figure 18A in order to obtain the data values from the three analog input circuits for the first channel. Diamond 1124 indicates that the program then conducts several tests relative to the Neighbor1 analog input circuit. Specifically, the controller conducting the arbitration checks to see if the Neighbor1 circuit board is inserted and if a complete communication message has been received from the controller for the Neighbor1 analog input circuit. In this regard, it should be noted that this may be achieved by looking to see if the "OK" bit has been set for the Family-Type codes of the ME and N1 boards.

[0198] Next, the difference between the analog value received by the controller conducting the arbitration and the analog value received from the Neighbor1 analog input circuit (through a Neighbor to Neighbor communication message) is determined (block 1126). This difference in analog values is then compared against a Narrow Tolerance threshold value (block 1130). The Narrow Tolerance value is dependent upon the particular type of analog input sensing hardware being used. For example, for a sensor providing a 4-20 ma current loop input value, the Narrow Tolerance value may be set to 0.6%. In other words, if the "ME" value was 10.0 ma and the Neighbor1 value was between 9.88-10.12 ma, then these values would be determined to be within Narrow Tolerance agreement. Substantially tighter Narrow Tolerance values may be employed with other analog input values which are quite stable, such as those derived

from thermocouples.

[0199] Block 1132 indicates that the Neighbor1 Narrow Tolerance bit will be set in the event that there is Narrow Tolerance agreement. However, if the Neighbor1 value was outside of the Narrow Tolerance range, then a test will be made to determine if this value is at least within a Wide Tolerance value (block 1134). The Wide Tolerance value is a suitably less strict value, such as a value which is double that of the Narrow Tolerance value. As will be seen below, the Narrow Tolerance value test is used to initially qualify an input channel for arbitration, referred to herein as being "in service". In contrast, the Wide Tolerance test is used to permit a previously qualified input channel to remain in service. Assuming that the "ME" value and the Neighbor1 value are sufficiently in agreement, then the Wide Tolerance bit will be set (block 1136). Regardless of outcome of this decision, the program will then proceed to test the Neighbor2 value in the same way that the Neighbor1 value was tested (for example, diamonds 1138-1142), assuming that the Neighbor2 analog input circuit board was inserted. Then, assuming that both the Neighbor1 and Neighbor2 analog input circuit boards were inserted and the necessary Neighbor to Neighbor communication messages were received, then the analog input values from these two circuits will be subjected to the Narrow Tolerance and Wide Tolerance value tests (for example, diamonds 1144-1148). The ME, Neighbor1 and Neighbor2 values will then be converted to Left, Middle and Right values for software arbitration purposes (block 1150).

[0200] Next, a set of "in service" test routines is provided for each of the Left, Middle and Right analog input values, as indicated by blocks 1152-1156. Each of these routines are used to determine whether these values should remain in service. The significance of the 'in service' designation is that a value must first be judged to be in service before it may be used in the primary arbitration routine. Figure 18E provides a flow chart for the block 1152, Figure 18F provides a flow chart for the block 1154 and Figure 18G provides a flow chart for the block 1156. Due to the similarity between these three flow charts, only the flow chart 1152 for the Left analog input value will be discussed.

[0201] As will be seen from the flow chart 1152 of Figure 18E, the program starts off with an assumption that the "In-Service" bit for the Left input value is already set. However, if the Family-Type code for the Left input value is wrong (diamond 1158), then the In-Service bit will be cleared (block 1160). Assuming that the Family-Type code is correct, then the program will check to see if the In-Service bit for the Left input value is presently set (diamond 1162). Assuming that the In-Service bit is set, then the In-Service bit for the Middle input value will be checked (diamond 1164). Assuming that the In-Service bit for the Middle input value is set, then the program will check to see if the L-M Wide Tolerance bit was set (diamond 1166). If the Wide Tolerance test was satisfied, then the Left In-Service bit will remain set. Otherwise, the Right input value will be tested in the same way, as indicated by diamonds 1168-1170. If the L-R Wide Tolerance bit was not set, then the M-R Wide Tolerance bit will be examined (diamond 1172). If the series of tests represented by diamonds 1166-1172 all fail, then the Left In-Service bit will be cleared (block 1160).

[0202] After the "in service" designation has been tested for each of the Left, Middle and Right values, then the flow chart 1100 of Figure 18B proceeds to block 1014. In this regard, it should be noted that block 1014 references the same flow chart as that shown in Figure 17B for digital inputs. Accordingly, it should be appreciated that the process of determining whether the process control computer 14 has requested a Low input value in the event of a default condition is the same for both digital inputs and analog inputs.

[0203] The analog input arbitration process then proceeds to the primary selection routine, which is indicated by block 1174 in Figure 18B. The flow chart represented by block 1174 is collectively shown in Figures 18H-18J. The program will first check to see if any of the Left, Middle or Right values are in service (for example, diamonds 1176-1180 in Figure 18H and diamonds 1182-1184 in Figure 18I). If none of these values are in service for the analog input channel being processed, then the controller performing the arbitration will select its own value (block 1186) and the Arbitration Failure bit will be set (block 1188). However, if both the Left and Middle values were found to be in service (from their respective In Service bit settings), then these two values would be subjected to the Wide Tolerance value test (diamond 1190). Assuming that the Left and Middle values were in sufficient agreement, then the Left value would be selected (block 1192).

[0204] Importantly, block 1192 also indicates that a value labeled "Difference" is added to or subtracted from the Left value selected. The summation of the value selected with the Difference value is used to avoid a process bump in the event of a failure, as explained below. If the Left analog input value was selected during the last process cycle, then the Difference value will be zero and the Left value from the present process cycle will be sent to the process control computer 14 without modification. However, if the Left value was found to be out of service during the present cycle, and the Middle value was selected for transmission to the process control computer 14 (for example, block 1194 in Figure 18I), the Difference value provides an "offset" that may be added to or subtracted from the Middle value before transmission of the resulting value to the process control computer 14.

[0205] Thus, assuming for example that the Left in service value for the last process control cycle was 10.00 ma and the Middle in service value was 10.05 in the same process cycle, then a valid of 10.00 ma would still be transmitted to the process control computer 14. However, if the Left value in the next process control cycle was unavailable and the Middle in service value was selected for this cycle, then the 0.05 Difference value from the last process control cycle would be subtracted from the present Middle in service value by the controller performing the arbitration. In other

words, if the present Middle in service value was 10.12, then 0.05 from this amount and the analog input value for this channel would be transmitted to the process control computer 14 as 10.07 ma. As each of the controllers 92-96 perform the arbitration process shown in Figures 18H-18J, it should be understood that these controllers will know the specific Difference value that should be added or subtracted from the present Middle in service value selected prior to transmission of this analog input value to the process control computer 14. Alternatively, it should be appreciated that the Difference value could be transmitted to the process control computer 14 to permit interpretation of the analog input values to be made by the process control computer.

[0206] Even though the Left value has been selected, the arbitration process does not end at this point. As illustrated by diamond 1196, the program proceeds to determine if the Right value is currently in service. Assuming that the Right value is in service, then the Wide Tolerance test is checked for both the Left-Right and Right-Middle value combinations (diamonds 1198-1200). If either of these tests fail, then the appropriate compare bit could be set, such as the specific R-M compare bit (block 1202). In this way, the process control computer 14 could ultimately be apprised of disagreements between in service analog input values. The number of these disagreements may be counted to enable a suitable response to be taken in the event of a continued disagreement, such as alerting an operator or even shutting down an affected controller 100 in the appropriate circumstances.

[0207] In the event that one of the three analog input values are not in service, such as the Middle value, then the program will proceed to a comparison between the two remaining in service values (for example, block 1204). If these two in service values are in Wide Tolerance disagreement, then the Arbitration Failure bit will be set (block 1206). Additionally, block 1206 indicates that the specific compare bit affected could also be set. If this disagreement represents a new failure (block 1208), then the arbitration analog input value for the Last process control cycle will be sent to the process control computer 14 (block 1210). However, if this failure was present in the immediately preceding process control cycle, then the program will check to see if the process control computer 14 has requested a Low default value (diamond 1212). In either event, the program will test to see which one of the two in service values is greater than the other (diamonds 1214-1216). If the Low value was requested, then blocks 1218-1220 indicate that the lower value of the two in service values will be sent. Similarly, blocks 1220-1222 indicate that the higher of the two in service values will be sent when the Select-Low bit for this analog input has not been set. In any event, it should be appreciated from blocks 1218-1222 that the Difference value may also be factored in during the arbitration process or it could be sent to the process control computer 14 along with the analog input value selected. As the remaining portions of Figures 18I-18J carry out a similar decision tree analysis as that described above for those times in which the Left and/or Middle values are not in service, no further discussion of these flow charts is necessary.

[0208] Referring again to Figure 18B, a block 1224 indicates that a set of Difference values is calculated for use during the next process control cycle. Specifically, the difference between the actual value selected and each of the Left, Middle and Right values is calculated and stored. In the event that the Left value was selected, then the Difference value would be zero. However, in the example set forth above, the Difference value for the Left-Middle combination would be 0.05 ma. A similar Difference value is also calculated for the Left-Right and Middle-Right combinations, assuming that these values were also in service at the time.

[0209] Next, a set of "in service" test routines is provided for each of the Left, Middle and Right analog input values, as indicated by blocks 1226-1230. Each of these routines are used to determine whether these values should be put in service for the next process control cycle. Figure 18K provides a flow chart for the block 1226, Figure 18L provides a flow chart for the block 1226 and Figure 18M provides a flow chart for the block 1230. Due to the similarity between these three flow charts, only the flow chart 1226 for the Left analog input value will be discussed.

[0210] Diamond 1232 indicates that the Left value will simply remain in service if it is already in service. However, in the event that the Left value was found to be out of service, then diamonds 1234-1238 indicate that the Middle and Right values will be checked for their respective in service availability. If both the Middle and Right values are in service, each of these values is compared against the Left value to determine if there is Narrow Tolerance agreement (diamonds 1240-1242). If both Narrow Tolerance tests are successful, then the In-Service bit for the Left value will be set for use in the next process control cycle (block 1244). However, if the Left-Middle Narrow Tolerance test fails and the Left-Right Narrow Tolerance test passes (diamond 1246), then the difference between the Left value and the input sent to the process control computer will be calculated (block 1248). Then, diamond 1250 will test whether the Left-Sent value is less than the Narrow Tolerance threshold. If the Left-Sent value was less than the Narrow Tolerance threshold, then the Left In-Service bit will be set. Otherwise, the Left value will remain out of service.

[0211] In the event that the Left and Right values were found to be in service, and the Middle value was out of service, then the Left-Right Narrow Tolerance test need only be passed in order for the Left In-Service bit to be set (diamond 1252). In the event that none of the Left, Middle or Right values were found to be in service, then the program will check to see if one of the Middle and Right values were at least 'good' (diamonds 1254-1256). In this regard, a good value is one where the analog input board was plugged in and a complete neighbor to neighbor message was received. If either the Left-Middle or the Left-Right combinations pass the Narrow Tolerance test (diamonds 1258-1260), then the Left In-Service bit will be set (e.g, block 1262).

[0212] Once this procedure is completed for each of the Left, Middle and Right analog input values, then the flow chart of block 1264 is executed, as shown in Figure 18N. In this regard, the general analog input compare bit "AIC" will be set if any of the specific analog input compare bits have been set (block 1266). Thus, for example, if the comparison between the Left and Middle values failed the Wide Tolerance test (diamond 1268), then the AIC bit would be set.

[0213] Finally, as indicated by block 1270 in Figure 18B, the arbitrated analog input value is stored in a data table which will be transmitted to the process control computer 14. Then, the program will proceed to arbitrate the next analog input channel in a loop which is indicated by ellipse 1272. This AI loop will be repeated until all of the analog input channels are arbitrated for the first set of redundant analog input circuit boards. Then, the entire arbitration process will be repeated until all of the analog input channels have been arbitrated (for example, 4 sets of 5 analog input channels being arbitrated at a time).

[0214] Referring now to Figures 18O-18T, the process of arbitrating analog outputs will now be described. Figure 18O shows an overall flow chart 1274 for the analog output process. As flow chart 1274 follows the analysis employed by the flow chart 1068 of Figure 17E for digital outputs, the flow chart 1274 needs only to be briefly discussed. For example, it should be noted that the "Determine Fail-Safe/Fail-Last" block 1070 is the same for both digital and analog outputs. The substantive difference between the analog and digital overall flow charts is ultimately contained in the "Determine which Output to Use" block 1276 and the "Set/Clear AOC bit" block 1278. Figures 18P-18S illustrate the flow chart for block 1276, while Figure 18T illustrates the flow chart for block 1278.

[0215] Referring first to Figures 18P-18S, the flow chart 1276 is shown to generally follow the analysis discussed above for selecting digital outputs (flow chart 1072 of Figure 17G). However, instead of matching digital output values, valid pairs of analog outputs are compared relative to an Output Tolerance value. Specifically, the difference between two analog output values is calculated (for example, block 1280), and then a determination is made as to whether this difference is beyond the Output Tolerance value (for example, block 1282). The Output Tolerance value is preferably selected to be 0.1% of full scale.

[0216] If the Output Tolerance test is successful, then the Leftmost value is selected (for example, block 1284). However, if the Output Tolerance test fails, then the specific Disagreement bit will be set and the general Negotiation failure bit "AOAF" will be set (block 1286). The program will then proceed to determine if a Fail-Last request has been made by the process control computer 14 (diamond 1288). If the Fail-Last request has not been made, then the lowest of the two valid analog output values will be sent to the field (diamond 1290). This lowest of the two valid analog output values provides a Fail-Safe selection for the analog output channel.

[0217] In the event that a Fail-Last value was requested by the process control computer 14, then the program will proceed to find out which of the two valid analog output values was closest to the last arbitrated value. For example, as block 1292 indicates, the difference between the Right analog output value and the Last arbitrated output value will be calculated. Similarly, block 1294 indicates that the difference between the Left analog output value and the Last arbitrated output value will be calculated. Then, diamond 1296 will compare these two value differences and the lowest difference will be used to pick the Left or Right value as the case may be.

[0218] Finally, the flow chart 1278 of Figure 18T is used to set or clear the general analog output compare bit "AOC". In this regard, the diamonds 1298-1302 and block 1304 indicate that the AOC bit will be set if any specific comparison bits were found to be set. Otherwise, the AOC bit will be cleared if no disagreements have been found (block 1306).

[0219] It should also be noted that the analog output track "AOT" values and the digital output track "DOT" values may be arbitrated in a similar manner to that described in connection with the arbitration of analog output and digital output values described herein. Indeed, even the clock signal received by the controllers 92 and 96 may be arbitrated as well in a similar manner. In this regard, the clock signal arbitration preferably follows the analysis set forth in Figure 17C to determine which clock signal should be selected.

[0220] Referring to Figures 19A-19M, a set of flow charts is shown to illustrate the method non-intrusively testing the digital output circuits 500-504 according to the present invention. This testing method includes both passive and active testing procedures. Figures 19A-19C combine to provide an overall flow chart 1400 for the non-intrusive testing process. As indicated by blocks 1402-1406 and diamonds 1408-1416, a series of health checks are made before any testing of the digital output circuits is permitted. In this regard, no errors must be found from the immediately preceding process cycle for the digital output circuit to be tested, and the controller 100 conducting the test must be able to communicate with its neighboring controllers. In the event that any of the conditions represented by diamonds 1408-1416 are not met, then the continuation of flow chart 1400 in Figure 19B indicates that the appropriate error codes are set.

[0221] Assuming that the digital output circuit is permitted to be tested, then diamond 1418 indicates that the digital output circuits for the neighboring controllers will be checked for errors. If any errors are found, then the passive testing procedure of block 1420 will be bypassed. Figures 19D-19E combine to provide the flow chart for the passive testing procedure. While the passive testing procedure could be conducted on the digital output circuit of only one of the controllers 92-96 at a given time, it should be appreciated that each of the controllers 92-96 could conduct the passive testing procedure simultaneously. This is because active cooperation between neighboring controllers is not required

during the passive testing procedure.

[0222] As indicated by block 1422, polygons 1424-1426 and diamond 1428, the passive test will begin with Channel 1, and then loop through all ten channels if no errors are encountered. Diamond 1430 indicates that the program will detect whether or not the channel being tested has changed states. If the channel has changed states, then the program will proceed to test the next channel. However, during the initial pass through the loop, the answer will be no, and the test and track voltages will be read (blocks 1432-1434).

[0223] Diamond 1436 indicates that the controller 100 will determine whether the channel being tested is On or Off from the arbitrated command value. If the channel is commanded On, the controller will check to see that the test voltage (for example, TEST-1) was greater than a predetermined threshold level (for example, 19 volts). If the test voltage was greater than this level, then this portion of the test will have been successfully passed, and program will loop back to test the next channel through the OK polygon 1440. If the test voltage was too low, then the appropriate errors codes will be set, as a number of different errors could have occurred (for example, a blown fuse or a set switch open). Once an error is detected, the passive test is ended in this embodiment. However, it should be appreciated that the other channels could be subjected to passive testing in the appropriate application.

[0224] If the channel is commanded to be in an Off condition, then the controller 100 will check to see if the test voltage is greater than a predetermined Low test level (for example, 350 milli-volts) through diamond 1446. If the test voltage is below this level, then an open fuse condition will be detected for the fuse in the abort circuit under examination (for example, fuse DF1 of Figure 11A), and the appropriate error code will be set. Assuming that the test voltage exceeds the predetermined Low test level, then the controller 100 will check to see if the track voltage is below a Low track level (for example, 4.4 volts) through diamond 1448. If the track voltage is above this Low level, then the controller 100 checks to see if the track voltage is less than a predetermined high track voltage (for example 14.4 volts) through diamond 1450. If the track voltage is above this High level, then an error is present. However, the exact source of the error cannot be determined, so the test is continued with another channel. In this regard, the active testing procedure to be described below will need to be employed to help identify the source of the error.

[0225] In the event that the track voltage is below the Low voltage level, then further checks are performed in order to determine if there, nevertheless, is still an error that could be detected. In other words, the track voltage should be below the predetermined Low level when the channel is off, but there still may be a hidden problem that could be uncovered. In this regard, the test voltage will be examined to see if there is an error related to the diode 524 of the abort circuit (diamond 1452). If the test voltage is greater than a predetermined High test voltage (for example, 15.8 volts), then an open diode condition will be determined by the controller, and the appropriate error codes will be set (block 1454). In this regard, it should be noted that these error codes may be used by the controller 100 to request an abort of the channel by its neighboring controllers. Additionally, the controller which is conducting the test may also signal the presence of an error in its digital output circuit to the process control computer 14 in the next message sent to the process control computer. The process control computer 14 could also request that the field computer unit 12 transmit specific error code or status bits for analysis through a health and welfare process. In this regard, it should be noted that the process control computer 14 could be connected to another computer which would perform the health and welfare analysis.

[0226] If the test voltage was found to be less than its predetermined high voltage level, then the controller 100 will test for the presence of a voltage drop across the diode 524 by comparing the test and track voltages (diamond 1456). If a voltage drop was not found, then the controller 100 will determine the presence of a shorted diode condition, and set the appropriate error code (block 1458). If a voltage drop was found, then the controller 100 will check to see if the track voltage is below a predetermined Minimum level (for example, 240 milli-volts) through diamond 1459. If the track voltage is below this Minimum level, then the controller 100 will determine that the passive test was successful for this channel. If the track voltage is above the minimum level, then the controller 100 will determine that an error in the field has occurred, and the appropriate error code will be sent (block 1460). It should be understood that each of the High, Low and Minimum threshold values are determined by the +26 volt power supply level and the resistance values set for the resistors RP1, RP3 and RP7 in the abort circuit 510 shown in Figure 11A.

[0227] From the above discussion, it should be appreciated that the controller 100 is able to passively test each of the channels of its digital output board, in that none of the digital output channels have to be intentionally set on or off as part of the test procedure. In this regard, block 1462 of Figure 19A points out that the controller 100 must reserve a certain period of time in which to passively detect and analyze the functioning of its digital output circuit through the test and track signals. Additionally, it should be appreciated that the passive test according to the present invention also has the capability to determine the type of errors that may be encountered, including an error associated with the output control device in the field.

[0228] Referring specifically to Figure 19B, the controller 100 will wait until the time has expired for the passive testing procedure (for example, 10 milli-seconds) before proceeding to the active test procedure (block 1464). A decision is then made as to which one of the controllers 92-96 will conduct the active test procedure. In one form of the present invention, it is preferred that a different controller 92-96 undergo active testing each process control cycle. This is

accomplished by dividing the "second" clock value of the process control computer 14 by the number of controllers contained in the field computer unit 12 (that is, 3), as shown in block 1466. The remainder is used to determine which controller will undergo active testing. For example, at a reading of 12 seconds, the remainder value is 0. Therefore, as indicated by diamond 1468, the Left controller 92 will conduct the active test procedure (polygon 1470) during this process control cycle. Additionally, the result of diamond 1468 indicates that the other two controllers 94-96 will enter a listening mode (polygons 1472-1473).

[0229] Figures 19F-19G combine to provide an overall flow chart 1470 for the active test procedure. In this regard, the first channel of the Left controller 92 will be used to illustrate the operation of the active test procedure. Assuming that the digital output circuit board 500 for the controller 92 is in place and no errors are found on any of the digital output circuits 500-504 (diamonds 1474-1482), then the block 1482 indicates that one of the digital output channels will be selected for the active test procedure. In this particular embodiment, only one of the digital output channels will be tested during a single process control cycle. Accordingly, it should be appreciated that it will take 30 seconds to actively test all 10 of the digital output channels in the digital output circuits 500-504, where the process control cycle is set for a period of 1 second. In the event that the state of the channel in line for testing has not changed (diamond 1484) and a field error has not been found from passive testing of this channel (diamond 1486), then a determination will be made as to whether this channel is On or Off (diamond 1488). If the channel is Off, then the active-Off test will be performed (polygon 1490). Conversely, if the channel is On, then the active-On test will be performed (polygon 1492).

[0230] The flow chart 1490 for the active-Off test is shown in Figure 19H. As illustrated by flow chart 1490, the active-Off test is comprised of a series of three separate tests (blocks 1494-1498), which will all be completed assuming that no errors are found. In the first test (block 1494), the SET_DODC-1 signal will be set High by the controller 92 in order to turn on the transistor 516 of Figure 11A. While not specifically stated in block 1494, the transistors 518-520 will both be off, as the abort switches are programmed to open automatically when the channel is Off. Accordingly, the conduction of transistor 516 will not cause the abort circuit 510 to drive the field device 508. As the resistor is shorted across the conducting transistor 516, the TEST-1 voltage signal should rise by an amount determined by the resistance divider network in the abort circuit 510. Accordingly, as indicated by block 1494, the controller 92 will check to see that a sufficient voltage increase (delta-test) was achieved, and that the TEST-1 voltage stays below its maximum allowable value. If this test was unsuccessful, then an Active Test Error bit will be set Regardless of the outcome, the SET_DODC-1 signal will be toggled back to its off state. Diamond 1500 indicates that the controller 100 will check to see if the Active Error bit was set, and if it was, then program flow will be turned over to the active error procedure 1502 of Figure 19F.

[0231] Assuming that no errors were encountered, then the second active-Off test will be performed (block 1496). Under this test, the controller 100 will request that its neighbor1 controller (for example, controller 94) set the ABORT1-1 signal High in order to turn on transistor 518. However, as the SET_DODC-1 signal will remain Low, the abort circuit 510 will not be able to drive the field device 508. Nevertheless, the TEST-1 signal voltage should rise, as resistor RP3 is effectively shorted by the conducting transistor 518. The controller 100 will check to see if the appropriate voltage level increase was achieved, and set the Active Test Error bit if this increase was not achieved. The controller 92 will then request its neighboring controller to toggle the ABORT1-1 signal back to a Low state. Diamond 1504 indicates that the controller 92 will then check to see if this message was received via the Communication Error bit.

[0232] Assuming that no errors were encountered, then the third active-Off test will be performed (1498). This test is the mirror image of the second active-Off test, except that the ABORT2-1 signal will be toggled by the remaining neighboring controller (for example, controller 96). If no errors were encountered, then program control will loop back to the flow chart of Figure 19G in order to test the next digital output channel in the next process control cycle (polygon 1506).

[0233] Turning to Figures 19I-19J, the flow chart for the active-On test 1492 is shown. The active-On test is comprised of a series of five test procedures (blocks 1508-1516). In test block 1508, the SET_DODC-1 signal is set Low, while the ABORT1-1 and ABORT2-1 signal remain High. Accordingly, the controller 92 checks to see that the TEST-1 voltage level drops by the delta-voltage amount. The SET_DODC-1 signal is then toggled back to its High state. In test block 1510, the ABORT1-1 signal is toggled Low (through a request to the neighbort controller), while both of the ABORT2-1 and SET_DODC-1 signals are High. Accordingly, the controller 92 checks to see that the TEST-1 signal has not experienced a voltage drop. If a voltage drop is found, then a failure has occurred relative to the transistor 520, the opto-isolator DU3 or the ABORT2-1 signal, as a properly conducting transistor 520 would cause the TEST-1 signal to maintain its voltage level. The third active-On test (block 1512) repeats the second active-On test, except that the ABORT2-1 signal will be toggled Low.

[0234] In the fourth active-On test (block 1514), the controller 92 requests both of its neighboring controllers 94-96 to set the ABORT1-1 and ABORT2-1 signals low. Then, the controller 92 will check to see that TEST-1 signal voltage drops by the predetermined delta-voltage value. During this time, the other two controllers 94-96 will continue to drive the field device. Finally, in the fifth active-On test, the controller 92 will request its neighboring controllers 94-96 to switch their SET_DODC-1 signals Low for the channel being tested. When this happens, it should be understood that

the abort circuit 510 alone will be driving the field device 508. Accordingly, the controller 92 will check to see that the TEST-1 voltage level does not drop, in order to make sure that the abort circuit 510 is capable of driving the field device 510 by itself if necessary. Additionally, the presence of a voltage drop across the diode 524 will also be checked for, in order to be certain that the diode is functioning properly. Assuming no errors were found, then program control will be passed to the no error procedure 1506, which will set up the next channel to test (block 1518).

[0235] During the active-Off and active-On tests, it should be understood that the neighboring controllers 94-96 need to cooperate with the controller 92 by acting on the requests to change their ABORT1-1, ABORT2-1 and SET_DODC-1 signals. This cooperation is achieved through the listening mode procedure 1472 shown in Figures 19K-19M. As these neighbor to neighbor communications are outside of the input and output data exchanges which are performed at specific times once each process control cycle, the successive approximation digital to analog converter circuit shown in Figures 6J-6K must be set up at each of the controllers 94-96 to receive signal change requests from the controller 92 (block 1520). An internal timer will then be set up by each of the controllers 94-96 within which signal change requests or commands must be received (block 1522). If the appropriate commands are not received in this time (diamond 1524), then the get out procedure 1526 of Figure 19B will be performed.

[0236] Diamonds 1528-1530 indicate that the controller 92 may signal the controllers 94-96 to end the active test process. If the command received was not an end test command, the neighboring controllers 94-96 will check to see if any errors were encountered on their respective digital output circuits 502-504 during passive testing (block 1532). If any error was encountered, then the neighboring controller detecting its own error will signal back to the controller 92 that it cannot execute the requested command (1534), and set the amount of time that it expects a further message from the controller 92 (block 1536). As the existence of any board error will terminate active testing, the controller 92 will preferably respond with the end test command. In such a case, the Error code representing the type of error will be stored, as will an identification as to which channel the error was detected during passive testing (block 1534).

[0237] Assuming that no errors were found, then the neighboring controllers 94-96 will determine whether the controller 92 has requested a specific change in the ABORT signal (diamonds 1540-1542) or a change in the SET signal (diamond 1544). For example, in the case of the 'Abort On" command, then the neighboring controllers 94-96 will extract the channel to be affected from the command message (block 1546), and check to see if there is a field error (diamond 1548). Assuming that an error has not been detected for the field device 508 of the channel being tested, then each of the controllers 94-96 will check to see if the channel is On (diamond 1550). If the channel is On, then the abort transistor (for example, transistor 518) will already be on. Accordingly, the controller receiving an Abort On command at this juncture will determine that a bad message has been received (polygon 1552), and send a reply message to the controller 92 that this command cannot be executed (block 1536). However, assuming that the channel was Off, then the controllers 94-96 will determine which abort switch has been commanded to be changed to an Off state (block 1554). Then, the Reset Wait routine 1556 of Figure 19M will be performed.

[0238] The Reset Wait routine 1556 of Figure 19M begins with the neighboring controllers 94-96 sending a reply message to the controller 92 which echoes back the command received (block 1558). This echoing procedure enables the controller 92 to know that its message was property received. Then, the controllers 94-96 will turn On or Off the specific switch commanded by the controller 92 (block 1560), and set a timer to permit an automatic toggling back of this switch to its previous state (block 1562). If a toggle-back message from the controller 92 is not received before the timer reaches zero (or the predetermined time out value), then the affected neighboring controller will automatically toggle this switch back to its previous state (block 1564). Otherwise, the controllers 94-96 will reset their respective switches (block 1566), and reply with an echo message to the controller 92 (block 1568). Ultimately, as shown in Figure 19G, the controller 92 will send a message to its neighboring controllers to end the active testing procedure (blocks 1570-1572).

[0239] As indicated above, each of the analog output circuits 600-604 enable tests to be conducted of their abort and drive capabilities. These tests are considered to be non-intrusive, because they will not disturb the analog output values being supplied to the field. The non-intrusive testing will be conducted on all 5 channels of one analog output circuit 600-604 at a time, and such testing preferably takes place only when all of the controllers 92-96 and their respective analog output circuits are fully functioning. While one analog output circuit is undergoing this non-intrusive testing, at least one of the other two neighboring analog output circuits will generate the electrical current necessary to maintain the desired output power to the field.

[0240] Figures 20A-20V provide a set of flow charts for the software on the controllers 92-96 which makes abort determinations and directs the non-intrusive testing of the analog output circuits 600-604 according to the present invention. In this regard, Figure 20A shows an overall or main flow chart 1600 for this controller software. For ease of description, operations represented by this software will be discussed using controller 92 as the example. However, it should be appreciated that these operations are performed concurrently by each of the controllers 92-96. Block 1602 indicates the necessary data for abort decisions and non-intrusive testing is copied from the external RAM memory (U42 of Figure 6A) to the internal RAM of the controller's microprocessor (U40 of Figure 6A). Then, the controller 92 will sequentially perform a set of routines, as indicated by the broken-line blocks 1604-1612. The Calculate Needed

Aborts routine 1604 is shown in Figures 20B-20L. The Choose and Set Up the Non-Intrusive ("NI") Test routine 1606 is shown in Figures 20M-20P. The Communicate to the Smart Analog Output ("SAO") Board routine 1608 is shown in Figures 20Q-20S. The Handle Errors routine 1610 is shown in Figures 20T-20U. The Send Abort Positions to the Hardware routine 1612 is shown in Figure 20V. Once all of these routines are completed, then the necessary status bytes needed by the Process Information ("PI") system are created (block 1614). Finally, the IRAM data is copied back to the XRAM (block 1616).

[0241] Referring to Figures 20B-20L, a set of flow charts for the Calculated Needed Aborts routine 1604 is shown. In this regard, Figure 20B provides an overall flow chart for this routine. Block 1618 indicates that the data transferred from the N1 and N2 output communications will first be examined to see if there are any hardware abort disagreements. A hardware abort disagreement arises when the ME controller 92 has aborted a particular channel and neither of the neighboring controllers N1-N2 have done the same. If this condition exists, the disagreeing abort switch will be closed. In any event, any abort request from a neighboring controller is honored by opening the abort switch for the channel of the SAO identified by the request data (block 1620). The controller 92 will then clear out the abort requests and start to process its own independent abort determinations for the next process control cycle (block 1622).

[0242] Diamonds 1624-1626 are used to determine if either of the neighboring SAO boards were replaced, and if so, then blocks 1628-1630 indicate that the abort switches for a replaced SAO will be closed in order to permit it to operate. Next, the controller 92 will check to see if its SAO board sent a communication during the last process control cycle (diamond 1632). If a communication was not sent or a problem was reported, then a flag will be set to indicate that this SAO board is considered "dead" (block 1634). A similar procedure is then performed for both of the neighboring SAO boards through the messages provided from the controllers 94-96 (diamonds 1636-1638). Then, in the event that both of the neighboring controllers 94-96 failed to communicate with the controller 92, then no abort switches will be opened by the controller 92 at this point (diamond 1640). This is to permit the outputs determined by the Fail Safe/Last mechanism to reach the field even though none of the controllers 92-96 are able to communicate with each other.

[0243] Assuming that the controller 92 is able to communicate with at least one of its neighbors, then the Open Needed Aborts routine 1642 will be performed. The Open Needed Aborts routine 1642 is shown in Figures 20C-20J. The controller 92 will then perform the Handle Abort Disagreements routine 1644 of Figure 20K. Finally, the controller 92 will perform the Clean Up from the Aborts routine 1646 of Figure 20L

[0244] Referring to Figures 20C-20J, the Open Needed Aborts 1642 routine will now be discussed. Diamond 1648 indicates that an initial check is made as to whether the SAO board for the controller 92 was flagged as being dead. If this SAO board is considered alive or operational, then program control will jump to point "A" on Figure 20E. However, even if this SAO board is considered dead, the controller 92 will still set up to process abort decisions for all five analog output channels, and point to the first of these channels (block 1650). Diamonds 1652-1656 indicate that a check will be made to see if either of the neighboring SAO boards were flagged as dead.

[0245] Assuming that both of the neighboring SAO boards are operational, then diamonds 1658-1660 are used to detect for the presence of an "OOCH·ME = 0" flag from each of the neighboring SAO boards, as relayed by the controllers 94-96. The "OOCH" term of this status signal stands for "Out Of Control High". As indicated above, if any of the SAO boards detect more power going to the field than there should be (for example, more than 2% of the maximum allowable value), then any SAO board detecting such an occurrence will attempt to ramp itself down to zero. If it is able to ramp itself out of the contribution of power being transmitted to the field (that is, ME = 0) and the OOCH condition still exists, then it will set the Out Of Control High ME = 0 flag for communication to neighboring controllers through a message from its own controller. Thus, for example, if the "N1 OOCH ME = 0" signal is received by the controller 92, and the "N2 OOCH ME = 0" flag is not set, then block 1662 indicates that the controller 92 will open the abort switch for the first channel on the controller designated as N2 (for example, controller 96). This action is taken because it is clear at this point that the SAO board for the controller designated as N1 (for example, controller 94) is not the source of the problem. However, if both the "N1 OOCH ME = 0" and "N2 OOCH ME = 0" signals were received by the controller 92, then a flag will be set to indicate to the process control computer 14 that uncontrolled power is being transmitted to the field for this analog output channel (block 1664).

[0246] In the event that the answer to diamond 1652 is YES and the answer to diamond 1654 is NO, then the controller 92 will look for the "N2 OOCH ME = 0" flag (diamond 1666). If this signal is present, then the controller 92 will set the uncontrolled power to the field flag (block 1668). Additionally, as extra measure, the controller 92 will re-open the abort switch for this channel of the SAO board for the controller designated as N1. This is because (although the abort switch should have been opened) it could nevertheless be possible that the N1 SAO board could erroneously be sending too much power to the field, even though the N1 controller could not communicate with the controller 92, and the N2 SAO appears to be able to drive the load. Diamond 1670 and block 1672 indicate that this procedure is followed in the event that the N1 SAO is functioning properly and the N2 SAO board is considered dead (or its controller did not communicate with controller 92 in this process control cycle). In the event that diamonds 1652 and 1654 are both answered YES, then this channel's Triple Abort flag will be cleared (block 1674). This flag is used to enable all abort switches to be closed in order to prevent a total loss of power to the field.

[0247] Figure 200 shows that this process is continued and repeated until all of the five analog channels have been processed. Additionally, Figures 20E-J combine to demonstrate that this process is performed in a similar manner when the SAO board for the controller 92 is functional and the neighboring SAO boards may or may not be functional. Thus, for example, diamond 1676 indicates that the controller 92 will test for the presence of its own "OOCH ME = 0" flag when its neighbors have failed to communicate or their SAO boards are considered dead. In this example, block 1678 indicates that the proper amount of power is being transmitted to the field for the SAO board for controller 92 has not ramped itself down to a non-contribution level (for example, a zero output). In contrast, if this SAO has ramped itself out, then the uncontrolled power to the field flag will be set and the N1 and N2 abort switches for this channel will be opened by the controller 92 to assure that they are outputting no power (block 1680).

[0248] Additionally, it should be noted that a YES answer to diamond 1682 in Figure 20F indicates that the neighboring controllers 94-96 will independently handle the necessary abort decisions (for example, open the abort switches for SAO board of controller 92), if such action is warranted by the process described above. Furthermore, a NO answer to diamond 1684 of Figure 20J shows that the Safe Disagreement flag will be set (block 1686). This is a situation where all of the SAO boards are functioning, communication has been received from both the N1 and N2 controllers, the SAO board for controller 92 has set the "OOCH ME = 0", and the other two SAO boards have not set their respective "OOCH ME = 0" flags. In this situation, the Safe Disagreement flag is set because the three SAO boards are functioning, so it is possible to employ majority decision making to determine whether an abort should be opened. The Safe Disagreement flag is used to indicate to the Abort Disagreement routine of Figure 20K that a problem has occurred. However, if the answer to diamond 1684 is YES, then the controller 92 will open the N1 abort switch for this channel (block 1688). This is because two SAO boards (ME and N2) have independently noticed the output to the field was too high and independently pulled their outputs down to a non-contribution level, but the SAO board for the N1 controller has not

[0249] Referring to Figure 20K, a flow chart for the Handle Abort Disagreements routine 1644 of Figure 20B is shown. This routine examines a counter which is set up for each analog output channel to record the number of Safe Disagreements between this controller's SAO board "OOCH ME = 0" flag and the other two functioning SAO boards. If this count gets too high (for example, 32 decimal) on any one of the five analog output channels, then an abort disagreement error flag will be set (block 1690). This error flag will cause the controller 92 to shut down its own SAO board, because the disagreement with the neighboring boards indicates that this board would not be capable of driving the output if it had to (that is, the output would be too low). Diamond 1692 and blocks 1694-1696 indicate that only continuous disagreements will be accumulated to eliminate undue transient conditions.

[0250] Referring to Figure 20L, a flow chart for the Clean Up from the Aborts routine 1646 of Figure 20B is shown. This routine is used to respond to a situation where the controller 92 is informed that both of its neighboring controllers 94-96 have opened the abort switches on one of the channels for the SAO board of the controller 92. If the controller 92 had also opened the abort switches on this channel, both of the abort switches for this channel will be closed by the controller 92, so that at least one of neighboring SAO boards will be able to transmit power to the field (block 1698). If the controller 92 had not opened the aborts on the channel, the SAO board would be told to shut down since one of its channels was aborted and the board would have to be removed for repair.

[0251] Referring to Figures 20M-20P, a preferred form of the non-intrusive testing method according to the present invention is shown. In this regard, these flow charts represent the Choose and Set Up the NI Test routine 1606 of Figure 20A. Diamond 1700 shows that this testing will only be initiated if the controller conducting the test is able to communicate with both of its neighbors, and at least one of the controllers was able to communicate with the process control computer 14 within the last process cycle. Similarly, diamond 1702 indicates that if any errors were encountered, then the non-intrusive test procedure will be by-passed until such errors are corrected.

[0252] As indicated by diamond 1704, the non-intrusive testing is timed to begin at exact multiples of 5 minutes, according to a clock signal of the process control computer 14. In this regard, each of the field computer units 12 will receive a synchronization pulse from both the Left and Right process control computers 14a-14b each second. The controllers 92-96 then adjust their clocks accordingly. The non-intrusive testing then uses that clock to follow a specifically timed schedule. As it takes approximately 1.5 minutes for one of the analog output circuits to complete the testing routine, the 5 minute interval allows sufficient time to complete non-intrusive testing for all of the analog output circuits 600-604. In this regard, the Table below identifies the preferred timed operations for the non-intrusive testing. The 'Displayed Time' listed on the Table is the time which is visually presented on the debug panel 18 of the process control computers 14a-14b. Each of the test numbers identified in this Table correspond to specific test procedure identified in Figures 20J-20M.

Time	Displayed Time	Action
00:00-00:35	00:00-00:23	Left Test #1
00:36	00:24	Test #2

(continued)

Time	Displayed Time	Action
00:37	00:25	Test #3
00:38	00:26	Test #4
00:39	00:27	Test #5
00:40	00:28	Test #6
00:41 - 01:21	00:29 - 01:15	Test #7
01:22 - 01:57	01:16 - 01:39	Middle Test #1
01:58	01:3A	Test #2
01:59	01:3B	Test #3
02:00	02:00	Test #4
02:01	02:01	Test #5
02:02	02:02	Test #6
02:03 - 02:43	02:03 - 02:2B	Test #7
02:44 - 03:19	02:2C - 03:13	Right Test #1
03:20	03:14	Test #2
03:21	03:15	Test #3
03:22	03:16	Test #4
03:23	03:17	Test #5
03:24	03:18	Test #6
03:25 - 04:05	03:19 - 04:05	Test #7

[0253] While each of these seven tests will discussed below, these tests may be identified as follows. Test #1 may be referred to as the "Rampdown" test, as the controller conducting the test (controller 92 in this example) will slowly reduce its contribution to the analog output current to 0% of the commanded output value. The SAO boards for the N1 and N2 controllers will react by increasing their output current to maintain the proper output upon each reduction. The SAO board for the N1 controller is preferably instructed to contribute the majority of the output. This operation generally takes several seconds. If a failure is reported during this step, the probable cause of the failure will be due to a shorting of the blocking diode 648 (shown on Figure 12C).

[0254] Test #2 may be referred to as the 'Generate Test Voltage' test, as the SAO board for controller 92 will be instructed to output a voltage which is not large enough to affect the current being transmitted to the field. In other words, the test voltage level should be set lower than the threshold of the blocking diode 648 (for example, 400 mV). If a failure is reported during this step, then the probable cause of the failure will be due to the inability of the operational amplifier 608 to output the desired test voltage level.

[0255] Test #3 may be referred to as the "ME Aborted Test", as the DN1 and DN2 abort switches will be commanded to be opened. The SAO board for the controller 92 will measure its output on the high side of the ME resistor 618 with respect to ground to determined if in fact the output is zero volts. In this regard, it should be noted that in all of these tests, it is preferred that each of the five channels are tested simultaneously. Accordingly, under Test #3, all of the analog output channels on the SAO board for the controller 92 will be aborted.

[0256] Test #4 may be referred to as the "N2 Abort Switch" test, as the DN2 abort switch will be closed while the DN1 abort switch is opened. The SAO board for the controller 92 will then measure its output on the high side of the ME resistor 618 with respect to ground to determine if the abort test voltage (for example, 400 mV) is present at the output for each of its channels.

[0257] Test #5 is a test of the deadman circuitry. It begins by repeating Test #3 to assure the aborts DN1 and DN2 have been opened. Then, the deadman circuitry is activated, a voltage is output to detect the activation of the deadman, and then a determination is made whether the deadman was activated. Test #6 is a repeat of Test #4, except that the DN2 abort switch is open while the DN1 abort switch is closed.

[0258] Test #7 may be referred to as the "ME 100% Load" test, as the SAO board for the controller 92 will ultimately

be commanded to drive 100% of the commanded output value to the field. Accordingly, the DN1 and DN2 abort switches will be closed and the SAO boards for the N1 and N2 controllers will slowly ramp down to 0%. The SAO board for the controller 92 will then measure the output for each channel across the ME resistors 624 to make sure that the SAO board has the ability to drive the required output value without any help from either of its neighbors.

[0259] Returning to Figure 20M, the block 1706 indicates that the test time will be incremented by one second each instance that this procedure is repeated. This time count or value will then be evaluated through a series of diamonds 1708-1724 in view of the fact that the seven tests for each of the three controllers 92-96 follow the time chart set forth in the Table above. Additionally, it should be noted that a two digit nomenclature is used in the flow charts of Figures 20M-20P to identify the non-intrusive ("NI") tests of the present invention. The first digit refers to the identity of the controller conducting the NI test, whereas the second digit refers to the specific test number. In this regard, the first digit is either "0", "1", "2" or "x". The "0" digit refers to the ME controller, which is controller 92 in this example. The "1" and "2" digits refer to the N1 and N2 controllers respectively. The "x" digit is essentially a wild card that could refer to any of the controllers 92-96. Additionally, the "x" designation may also be used as wild card for the test number digit as well.

[0260] Thus, if the test time is between 0-35 seconds, the controller will allow the extraction of the first test (that is, Test #1), as indicated by the "x1" nomenclature (block 1726). Next the controller conducting the NI test procedure will then check to see if it is the Left or the Middle controller (diamonds 1728-1730). In this example, the answer to diamond 1728 will be YES, and the program will go to block 1732 of Figure 20P. Block 1732 permits the SAO board for controller 92 to extract a "0x" test number, which at this point in the procedure "x" was previously identified as Test #1. As this NI test procedure will also be conducted independently, but concurrently in the other controllers 94-96, the program will jump to points "C" or "D" of Figure 20P, respectively, for each of these controllers. In this regard, it should be appreciated that the points "C" and "D" provide entry points for other parts of the NI test program. Thus, for example, the diamonds 1734-1738 are used to direct program flow to different procedures depending upon which test is currently being extracted. In the case of Tests #3-#5, the Right controller 96 must open the abort switches for its neighbor N2 (that is, controller 92), provided that the channels of the controller 92 were able to ramp down as required under Test #1 (block 1740). In the case of Test #6, the Right controller will close the abort switch for each of the channels on the SAO board for the controller 92 (block 1742).

[0261] Finally, block 1744 indicates that the last second's NI test number and this second's test number will be stored. Then, during the next process control cycle, which in this example is a one second period, the NI test procedure of Figures 20M-20P will be repeated. In this way, each of the controllers 92-96 will direct the NI tests performed on the SAO boards. Additionally, it should be appreciated that these controllers will also cooperate with each other by toggling abort switches and ramping down/up as required by the specific test number being conducted. This cooperation is provided through the time chart set forth above, as each of the controllers independently performs the same test procedure program. In other words, it is not necessary for one controller to request or command another controller to take the necessary action. Rather, each of the controllers 92-96 will look at the time and take the appropriate action, unless one of the problem conditions set forth in diamonds 1700-1702 is detected.

[0262] Referring to Figures 20Q-20S, the flow chart for the Communicate to the SAO Board routine 1608 of Figure 20A is shown. This routine is used to facilitate bi-directional communication between a controller and its SAO board. In this regard, a first data exchange between the controller and its SAO board is referred to as "Primary" communication (for example, NI test directions and output values). Conversely, any subsequent data exchange between the SAO board and its controller is referred to as a "Secondary" communication (for example, track values). Accordingly, Figure 20Q shows two entry points, one for Primary communication (oval 1746) and one for Secondary communication (oval 1748).

[0263] Figure 20Q shows several communication set up blocks which are consecutively performed. In this regard, it should be noted that set up block 1750 provides an initial wake up message to the SAO board, to which the SAO board must respond within a specific timeout period. If the SAO board responds property, then data will be exchanged with the SAO board (block 1752). If the validity check failed (for example, an incorrect checksum), then the appropriate bad communication flags will be set (blocks 1754-1756). Additionally, all of the analog output track ("AOT") values will be zeroed to prevent old data from remaining in the data tables, and thereby prevent a technician from misinterpreting the old data.

[0264] Figure 20R indicates the appropriate status information and values will be stored depending on whether the message was a Primary or Secondary communication (blocks 1758-1760). Additionally, diamond 1762 is used to check for any failures in the Non-Intrusive testing. The controller, such as controller 92, will respond by setting a flag which will be transmitted to its neighboring controllers to either stop or continue the NI test procedure (blocks 1764-1766). Regardless of this outcome, a flag will be set for the SAO board of the controller 92 to indicate that any test failure is a false alarm (block 1768). As will be seen below, this flag may be cleared during a later part of this procedure.

[0265] The controller 92 will then begin to examine the NI test error counters for each of the five analog output channels (block 1770). If the NI test conducted in the last second was not Test #1 or Test #7, then the NI test counter

will be incremented or decremented depending upon whether a test failure was reported by the SAO board (blocks 1772-1774). If a test failure was reported and the test error counter exceeds a predetermined limit (for example, 30 hex), then a flag will be sent to the neighboring controllers to stop testing and the false alarm flag will be cleared (blocks 1776-1778). In this regard, it should be appreciated that the NI test procedure will permit a transient error to be reported before deciding to halt the NI test procedure.

[0266] Figure 20S shows that the controller 92 will again check for a failure of an NI test (diamond 1780). This is done because the "test failure' flag will be cleared if an NI test failure occurred, but the test error counter did not exceed the predetermined limit. If such an failure is detected, then the false alarm flag will be cleared and the SAO board for controller 92 will be instructed to shut down (block 1782). This procedure will then be repeated for each of the analog output channels (block 1784 and diamond 1786). An NI testing report is also generated when a new error is detected (block 1788).

[0267] Referring to Figures 20T-20U, the flow chart for the Handle Errors routine 1610 of Figure 20A is shown. This routine begins with a check to see if an SAO board was replaced during the last process control cycle, and then it proceeds to check for other health indicia (diamonds 1790-1798). If the status report indicates a failure or the controller was not able to receive a communication from its SAO board, then the SAO board will be flagged as dead, and the controller will request its neighboring controllers to open the abort switches for this SAO board (block 1800 on Figure 20U). However, if the SAO board was flagged as being alive for the last process control cycle, then the error handling routine will look at the data from each of the analog input channels (block 1810).

[0268] Diamond 1812 examines the value of a "Recovery" counter, which is used to give the system time to re-synchronize when the controller 92 is unable to communicate with either of its neighboring controllers (see diamond 1814 and block 1816 of Figure 20U). If the Recovery count is not zero, then the "OOCH ME = 0" status byte will be cleared in order to prevent an abort from being opened as the system is synchronized (block 1818). If a test failure is detected on any of the analog output channels, then the NI testing will be stopped (block 1820). If an "OAT <> DAC" error has been flagged, then an abort request will be transmitted to the neighboring controllers 94-96 for the particular channel under inspection (block 1822). This is because the "OAT <> DAC" error means that the SAO board's operational amplifier 608 on this channel is not functioning properly.

[0269] Figure 20U also shows that a "Retry" counter is employed to handle a situation where the communication from the controller to its SAO board is imperfect (diamond 1824). If the Retry counter is greater than a predetermined value (for example, 5), then the controller will cause a hardware reset of the SAO board in an attempt to correct the problem (block 1826). In other words, the bad communication flag from block 1754 of Figure 20Q will be used to permit the controller to track the existence of a communication problem with its SAO board, and after a sufficient number of tries, then the controller will reset the microprocessor EU3 of the SAO board in an attempt to restore valid communication.

[0270] Referring to Figure 20V, the flow chart for the Send Abort Positions to the Hardware routine 1612 of Figure 20A is shown. This routine examines the abort decision information for each of the analog output channels and responds by opening or closing each of the abort switches for its neighboring SAO boards (blocks 1828-1830). It should also be noted that the controller will look at the arbitrated analog output value to be sent to the field (diamond 1832). If the output value is zero for any of the analog output channels, then the controller 92 will send a flag to its neighboring controllers to open the abort switches on its SAO board for those channels (1834).

[0271] Referring now to Figures 21A-21S, a set of flow charts is shown for the software resident on the SAO boards. Additional flow charts for the SAO board software will also be discussed in connection with Figures 22A-22S and 23A-23I. Figures 21A-21B provide an overall or main flow chart 1900 for the SAO board software. As should be appreciated from the discussions above this software is contained in the program memory circuit EU1 of each of the SAO circuit boards 600-604.

[0272] The flow chart 1900 begins with a call to a startup routine 1902, which is shown in Figures 21C-21D. The microprocessor EU3 of the SAO board will preferably read the software version level from memory (block 1904), and proceed to test the hardware components for the SAO board (block 1906). This hardware test routine is shown in Figures 21E-21K. The SAO microprocessor will then check to see if a Deadman condition exists (diamond 1908). A Deadman condition could exist if the controller 100 shuts down, the microprocessor on the SAO board shuts down, or if the SAO board puts itself into a Deadman condition for diagnostic testing purposes. If a Deadman condition exists, then all of the analog output channels will be zeroed (block 1910) and the program will jump to the warm start point in the startup routine 1902 of Figure 21C, unless the SAO board is currently testing its ability to disable the operational amplifier 608. While not shown in Figure 21A for simplicity, a check may be made at this point to determine if the SAO board is currently testing this Deadman capability. This Deadman test will be described below in connection with Figures 23E-23G. If the SAO board is testing the Deadman capability, then the Deadman test will be repeatedly conducted (for example, 30 times) before returning to an appropriate location in flow chart 1900, such as block 1904.

[0273] Figure 21A also shows that the SAO board may be re-started if too many interrupts are received from an internal timer of the SAO microprocessor (diamond 1912). These timed interrupts provide a way to permit the SAO

microprocessor to determine whether a communication from the controller for this SAO board has been received within a reasonable period of time.

[0274] Assuming that the SAO board is "alive", the SAO microprocessor will strobe the "DEADSET" signal (block 1914), and call the communications routine (block 1916). The communications routine is shown in Figure 21 M. After this communications routine, then a Testing routine will be called (block 1918). The Testing routine 1918 is shown in Figure 21 L A routine will then be performed to gather feedback data from the field (block 1920). This Read Data routine is shown in Figures 21N-21Q. Next, a Handle Error Conditions routine 1922 of Figure 21R will be performed. The program will then proceed to a Calculate the Output routine 1924, which is shown collectively in Figures 22A-22S. Thereafter, the non-intrusive testing routine 1926 will be performed. This NI testing routine is shown collectively in Figures 23A-23I.

[0275] Once all of these steps are performed, then the SAO microprocessor will point to the next channel to be serviced (block 1928) and repeat the procedure until all five analog output channels are done (diamond 1930). The SAO microprocessor will then update its record of "five channel cycles" since the last communication from its controller, such as controller 92 (block 1932), and then determine if it is the appropriate time to check the field loops (diamond 1934). The routine for checking the field loops (for example, measuring the field loop resistance values) is shown in Figure 21S (block 1936). In either case, the main program for the SAO board will ultimately loop back to the beginning in order for the program to be continuously repeated. Thus, it should be appreciated that the field loops will be measured and the hardware tested each process control cycle (for example, one second).

[0276] Referring to Figures 21C-21D, the flow chart for the startup routine 1902 is shown. The Red LED will be turned on to indicate that the SAO board hardware is not ready to send power to the field, as a series of tests will be conducted (block 1938). In this regard, the first test relates to the data memory for the SAO board (block 1940). This test is similar to the memory test described below for the controller's data memory. Then, due to the fact that the SAO board is entering a cold start, a counter which keeps track of the number of process cycles executed by the SAO board will be set to "01" to allow the hardware test routine to function property (block 1942).

[0277] Then, as shown in Figures 21C-21D, a hardware test routine will be performed at four different points during the startup routine (blocks 1944-1950). This hardware test routine is shown collectively in Figures 21E-21K. In this regard, it should be noted that the repeated testing of the hardware components for the SAO board is not necessary. Rather, this testing routine is performed during spare times as an extra measure to increase the confidence level in the ultimate operation of the SAO board. Thus, for the example, the hardware test routine will be performed in between times that the controller is trying to communicate with the SAO board (diamonds 1952-1954). As indicated above, the controller will communicate twice with the SAO board (blocks 1956-1958) in order to send timing information, output values, and assure the controller/SAO communication link is functioning property. Ultimately, the Red LED will be turned off (block 1960) and the Deadman timer will be reset (1962).

[0278] Turning to Figures 21 E-21 G, an overall flow chart 1906 for the hardware test routine is shown. Assuming that this is the first cycle for the SAO board, then the SAO microprocessor will read the "0 volt" input to the differential amplifier 638 via the multiplexors EU23-EU26 shown in Figure 12F (block 1968). Then, a check will be made to determine whether or not the voltage being read is within specifications (diamond 1970). If this voltage is outside of the proper specification level, then a routine will be performed to flag an analog to digital problem (block 1972). The flag ADC problem routine is shown in Figure 21 H. The SAO microprocessor will then read the "1/8 reference" signal shown on Figure 12B as an input to multiplexor EU24 (block 1974). This voltage signal level (for example, .275 volts) will be stored for use during the Calculate Slope routine of Figures 21J-21K (block 1976). Then a check will be made to determine whether or not this voltage signal is within specifications (diamond 1978). In this regard, the value which is produced by the differential amplifier 638 for the "1/8 reference" signal will be tested against a predetermined range (for example, 1.25 volts +/- .078 volts). A similar procedure is also implemented for the "1/2 reference' signal (for example, 1.10 volts).

[0279] Then, as shown in Figure 21 F, the SAO microprocessor will cause the digital to analog converter ("DAC") 612 to output a series of different voltage levels (blocks 1980-1986), and then it will check the actual output from the DAC through the analog to digital converter ("ADC") 642 (diamonds 1988-1994). If any of these voltage levels were determined to be outside of specifications, then the "Flag DAC Problem" routine 1996 will be performed. As shown in Figures 21 H and 211, both the Flag ADC Problem routine 1972 and the Flag DAC Problem routine 1996 increment or decrement a problem counter (blocks 1998-2000) as needed. Additionally, either or both of these Flag routines may cause the Red LED to turn ON if the problem count exceeds a predefined limit (diamond 2002 and block 2004). Then, as shown in Figure 21 G, this problem counter will be evaluated (diamonds 2006-2010), and the problem counter will be decremented if a problem was not detected during this pass through the hardware test routine (block 2012). Once this problem counter is greater than a decimal 2, then the Red LED will be turned On, and the SAO board shut down. A similar procedure could also be implemented to test the operational amplifier 608, as was performed for the DAC test. Thus, for example, the DAC 612 could be instructed to output a predetermined voltage (for example, 2.2v), and then the OUT-H and OUT-L signals could be read to see if these signals were within specifications.

[0280] Referring to Figures 21J-21K, a self-explanatory flow chart for the Calculate Slope routine 1976 is shown. As will be seen from the flow chart, this routine evaluates the slope of an artificial line created between the 1/2 and 1/8 reference signal levels, and operates to adjust stored slope and intercept values by one (each pass through the routine) until there is equality with the measured values. The values created by this routine are used to correct the field measurements for offset and gain errors introduced by the analog circuitry.

[0281] Referring to Figure 21L, a flow chart of the Testing routine 1918 of Figure 21A is shown. This routine detects whether the SAO board is plugged into a test jig rather than the field computer unit 12 itself (diamond 2014). If the SAO board is plugged into the test jig, then a set of predefined output values will be used to test the operation of the SAO board (block 2016).

[0282] Referring to Figure 21 M, a flow chart for the Communications routine 1916 of Figure 21 A is shown. While this flow chart is also self-explanatory, is should be noted that the watchdog interrupts referred in diamond 1912 of Figure 21A will be turned off (block 2018) and subsequently reset during this routine (2020).

[0283] Referring to Figure 21 N, a flow chart of the Read Data routine 1920 of Figure 21A is shown. The ADC converter control block 2022 of this routine is shown as its own flow chart in Figure 21O. In this regard, it should be appreciated that the SAO microprocessor needs to command a specific input signal selection for the differential input multiplexors EU25-EU26 and the convertor input multiplexors EU23-EU24. The Read Data routine will then proceed to the Linearize routine 2024 of Figure 21 P. As shown in Figure 21 P, the slope value determined from the Calculate Slope routine will be evaluated (diamond 2026). If the slope value is greater than one, then this slope value will be compared with the commanded output value (diamond 2028). If the output value is greater than twice the slope, then the Linearize routine will be ended because linearization of the data will result in an overflow in the mathematics. Otherwise, a calculation will be made, as shown in block 2030. The purpose of this calculation is to correct the measured voltages for offset and gain errors introduced by the analog circuitry.

[0284] Once the Linearize routine 2024 is completed, the Read Data routine 1920 will proceed to the Filter the Track routine 2032 of Figure 21Q. This routine begins with comparing the newly measured track value and the track value stored from the calculation performed on this channel in the last 5 channel cycle (block 2034 and diamond 2036). If the absolute value of the difference between the new and old track values exceeds a first predetermined amount, then the old track value will be completely replaced with the new track value to speed the response of the SAO board in its effort to achieve the commanded output value (block 2038). If the absolute value of this difference in track value is less than the first predetermined amount, then a check will be made to see if this difference is less than a second, smaller predetermined amount (diamond 2040). The result of this decision will determine whether the Unstable Track flag will be set. In any event, the difference value will be divided by four (block 2042), and a portion of this divided difference value will be added to or subtracted from the old track value depending upon whether the difference value was positive or negative (diamond 2044 and blocks 2046-2048). This proportionate change in the stored track value filters out most noise found on the track signal.

[0285] The Read Data routine 21 N will then point the multiplexors EU25-EU26 at the ME resistor High/Low values, and read and store these values (block 2050). A similar operation will then be performed for the OAT values via multiplexor 640 (block 2052).

[0286] Referring to Figure 21R, a flow chart for the Handle Error Conditions routine 1922 of Figure 21 B is shown. This self-explanatory flow chart demonstrates how the Red LED flag will be set and used to cause the DAC to rampdown (block 2054). In this regard, the Rampdown DAC -outine 2054 will be discussed in connection with Figure 22I. Similarly, the Send the DAC to the Field routine 2056 will be discussed in connection with Figure 22O.

[0287] Referring to Figure 21S, a flow chart for the Check the Field Loops routine 1936 of Figure 21B is shown. As will be seen from this flow chart, the SAO microprocessor will measure the actual output signal for each of the analog output channels and perform the checks identified on the magnitude of this signal (diamonds 2058-2062). If the signal being sent to the field is outside of any of these test bounds, then the appropriate flag will be set or preserved for further processing (blocks 2064-2068). These tests assume that the field load is modeled by a resistor in series with an inductor, and that the load being driven is between 50 and 470 ohms (+/- 30 ohms). Thus, for example, diamonds 2060 and 2062 compare the measurement from the low side of the track resistor with respect to ground with the maximum and minimum acceptable voltages for this output value. However, it should be noted that the loop resistance check will not be performed if the output value (block 2058) for the channel is below 2 mA, because the present hardware prevents the signal from being read reliably when the output value is below this magnitude.

[0288] It should also be noted that a 100 ohm PTC resistor is preferably connected in series between the low side of the track resistor 624 and the field loop. Thus, the maximum and minimum acceptable measurements at full scale (for example, 22 mA) can be calculated from the following formulas:

$$Vmax = (Rmax + Rptc) * (22mA),$$

where Rmax = 470 ohm

$$Vmin = (Rmim + Rptc) * (22mA),$$

where Rmim = 50 ohm

Using these formulas, it should be appreciated that the maximum and minimum voltage levels employed by diamonds 2060-2062 may be calculated for any desired output value (in mA). Thus, the test employed by the field loop routine 1936 are specifically tailored to the output value commanded by the controller for the SAO board.

[0289] Referring to Figure 22A, an overall flow chart for the Calculate the Output routine 1924 of Figure 21B is shown. This Output control routine provides an intelligent PI control loop as will be seen from the description below. The Output control routine includes a setup routine 2100, which is shown in Figure 22B. In this regard, Figure 22B shows that an initial evaluation of the commanded output value will be made (diamond 2102). If the output value is nearly 100% of the maximum allowable value, then the output for the channel being processed will be forced to a level just below this maximum value (block 2104). This is done so that an output above the 99.75% level can be seen and no more than 22ma of current will be transmitted to the field.

[0290] The Output control routine also includes a Calculation routine 2106, which is shown in more detail in Figure 22C. Once output error is calculated (block 2108), which is the difference between the output value and the measured track value, then it will be determined whether an increase or decrease in the analog output must occur (diamond 2110), and the appropriate status indicators will be set.

[0291] Referring again to Figure 22A, an evaluation will then be made as to whether the remainder of the Output control routine should be skipped (diamond 2112). In this regard, the Output control routine may be skipped when a problem has been detected on the board by the Handle Error Conditions routine. Assuming that the Output control routine is to be performed, then a check will be made to see if the red LED is ON (diamond 2114). If the red LED is ON, then a determination will be made as to whether the calculated output error is too large (diamond 2116). If the error is too large (for example, 3.5%), then a flag will be set to indicate that this SAO board is controlling the field (block 2118), and the Out of Control routine 2120 will be performed. Otherwise, the opposite indication will be flagged, the SAO board will back off its output to zero (block 2122), and the In Control routine 2124 will be performed.

[0292] As should be appreciated from the procedure described thus far, the three SAO boards 600-604 will effectively compete with one another to drive the load in accordance with the present invention. However, when any of the SAO boards detect that one of the other SAO boards is controlling the output, it will start backing off to a non-contribution level. In this way, only one of the three SAO boards 600-604 operates to drive the load at any one time, unless one of the other SAO boards determines that its contribution is necessary to achieve the commanded output value.

[0293] If the red LED is OFF, then a Back Calculation routine 2126 will be performed. This Back Calculation routine is shown in Figures 22D-22E. As will be seen from Figures 22D-22E, the Back Calculation routine is used to set a "Back.Calc" constant, and subtract or add this constant to the output error (block 2128). The Back.Calc constant is used in the PI control loop to account for any differences in the track measurements (due to any hardware differences between the SAO boards), and thereby allow the smoothest exchange of output contribution. The Back.Calc constant is the difference between the output value and the track value (block 2130). In this regard, it will be appreciated that the Back.Calc calculation will depend upon factors such as which NI test is being performed (for example, diamonds 2132-2134),because these are the cycles where the SAO boards must exchange responsibilities. In other words, the driving board must lower its output to zero and another board must drive the output.

[0294] Once the Back Calculation routine 2126 is performed, an "Output In Control ?" routine 2136 will be executed by the SAO microprocessor. The Output In Control routine 2136 is in the form of a question, because it will exit into either the In Control routine 2124 or the Out of Control routine 2120 depending upon the conditions being evaluated during its execution. The Output In Control routine 2136 is shown collectively in Figures 22F-22H. In this regard, Figure 22F shows that a series of evaluations will be made to determine if an NI test is being conducted (diamond 2140), and if so, then identify which test is currently being conducted (diamonds 2140-2148). The answers to these questions and answers to their depending questions (that is, diamonds 2152-2164) will determine which mode the SAO board is in. Specifically, Figure 22F identifies three modes of operation, namely "Tight Control", "Monitoring" and "Stay Clear". The use of the operating modes will become apparent from a review of Figures 22F-22H and the description below.

[0295] If the NI test is "00", it should be understood that no NI test is actually being conducted. As indicated by diamonds 2152-2154, the Tight Control mode is assumed when the SAO board's contribution to the field output is other than 0% of the commanded output value. Diamond 2166 of Figure 22G shows that an evaluation will be made in the Tight Control mode to determine if the output error (the output value - track difference) is within a tight deviation range (for example, 0.05% of 22ma). If the output is outside of this tight deviation range, then the Out of Control routine 2120 will be performed, as shown in Figure 22H. Otherwise, the In Control routine 2124 will be performed.

[0296] If the SAO board was not contributing anything to the output (diamond 2154), then the Stay Clear mode will

be assumed. In the Stay Clear mode, a check will be made to see if the output error is outside of a wide deviation range, such as 1.6% (diamond 2168). If the output error is within the wide deviation range, then the In Control routine 2124 will be performed. Otherwise, the Out of Control routine 2120 will be performed.

[0297] If the NI test is Test #1 (diamond 2142), then a determination will be made to see if the SAO board is driving more than 50% of the commanded output value (diamond 2156). If the answer is YES, then the Tight Control evaluation of diamond 2166 will be performed. Otherwise, the SAO board will assume the Monitoring mode. In the Monitoring mode, a determination will first be made to see if the SAO board is driving any of the output (diamond 2170). If the answer is YES, then a check will be made to see if the output error is within a monitor deviation, such as 0.10% (diamond 2172). If the answer to this question is NO, then the Out of Control routine 2120 will be performed. However, if the answer to this question is YES, then a determination will be made as to whether the output value was greater than the track value measured (diamond 2174). The determination of diamond 2174 will also be made if the NI test is "01-07" and the output is within the wide deviation (diamonds 2176-2178).

[0298] If the output value was greater than the track value (diamond 2174), then the In Control routine 2124 will be performed. Otherwise, a series of questions will be posed (diamonds 2180-2190) before entering the In Control routine 2124. Thus, for example, if the NI test is Test #07 and the output has not achieved more than 93.75% of the maximum possible output, then the Rampdown DAC routine 2192 will be performed. This action prevents more than 22ma from being sent to the field, as it should be noted that block 2174 established that the track is already greater than the output value.

[0299] The Rampdown DAC routine 2192 is shown in the flow chart of Figure 22I. In this regard, the flow chart indicates that the output will be ramped down in relatively small or large increments, depending upon whether the SAO board is driving more than 25% of the output value (diamond 2194). For example, when the small decrement constant is employed (block 2196), the output may be ramped down on the order of 0.1%/call to this routine. While the controllers 92-96 operate on a specific process control timing cycle, this is not strictly the case for the SAO circuit boards 600-604, as the SAO microprocessors will repeatedly execute their programs (as shown in Figures 21A-21B) as quickly as possible. In other words, each SAO board may execute all of its programs on the order of 50-100 times per process control cycle (for example, one second) of the controllers 92-96.

[0300] Figure 22H also shows that a Power Rampdown routine 2198 may be employed if the series of questions is resolved to the point where it is determined that the output error is greater than the monitor deviation (diamond 2190). The Power Rampdown DAC routine 2198 is shown in the brief flow chart of Figure 22J. In this regard, it will be appreciated that a very rapid decrement rate will be employed due to the fact that the output has been detected to be beyond the acceptable monitor deviation limit.

[0301] Figure 22F also shows that the Tight Control mode will be assumed whenever it is determined that the ME SAO board is driving 100% of the desired output value (diamonds 2158, 2162-2164). Otherwise, if the answer to any of the diamonds 2158, 2162-2164 is NO, then the Monitoring mode will be assumed. Similarly, if it is determined that ME SAO board is not driving any of the output (diamonds 2150 and 2160), then the Stay Clear mode will be assumed.

[0302] Referring to Figure 22K, a flow chart for the In Control routine 2124 is shown. This routine begins by clearing the "Almost Out of Control High ME = 0" flag (block 2200). The clearing of this flag is used to signify that an OOCH ME = 0 condition will not be signalled the next time the Out of Control counter reached a preset limit. Then an evaluation will be made as to whether the output value is greater than 99.7% of the maximum allowable output value (diamond 2202). If the output value is essentially less than this maximum value, then the program flow will skip down to the end of this routine, where an Out of Control counter will be decremented (block 2204). However, if the output value is at its maximum value, then three additional evaluations may be made (diamonds 2206-2210). If the NI test is one of the test numbers Test #01 through Test # 06, then the Out of Control counter will be decremented. However, if the NI test is one of those listed in diamond 2206, then the Rampdown DAC routine 2192 will be performed. Similarly, if the NI test is Test # 07 and the SAO board is outputting more than 93.7% of the maximum output value (diamonds 2208-2210), then the Rampdown DAC routine 2192 will be performed.

[0303] In the event that the Rampdown DAC routine 2192 routine is implemented at this point, then the Send the Output to the Field routine 2212 will be immediately executed. The Send the Output to the Field routine 2212 will be discussed in connection with Figure 22N. The Send the Output to the Field routine 2212 is also shown on Figure 22A as the next routine to be executed in any event once the In Control routine 2124 is completed. Nevertheless, if it is determined that the output of this SAO board should be decreased, then it is preferred that it should be permitted to begin backing off at the earliest opportunity.

[0304] Referring to Figures 22L-22M, a flow chart for the Out of Control routine 2120 is shown. This routine is used to change the DAC output value in response to a number of factors, such as the magnitude of the error detected. In the first place, block 2214 indicates that this routine will cause subsequent routines are to be skipped. This is because NI testing should not be performed if the output is not correct Then, a sibling wait counter will be evaluated (diamonds 2216-2218). The sibling wait counter is used to delay reaction to an output error and enable one of the neighboring SAO boards to react instead. Then, the Out of Control counter will be incremented (block 2220). Next, the magnitude

of the output error will be evaluated in order to determine the rate at which the DAC output value should be changed (diamonds 2222-2224).

[0305] As shown in Figure 22L, the program will branch depending upon whether the output error was negative (diamond 2226). If this difference was negative, then the DAC value will be decreased accordingly (Figure 22M, block 2228). Otherwise, the DAC value will be increased to the appropriate value (block 2230). Thus, for example, the DAC value will be set to a 10v output amount in block 2232 to prevent a futile attempt to send 20v to the field if the device will not allow the track to reach the output value at maximum voltage out. This action lowers the bump if a disconnected field wire is attached.

[0306] Referring to Figure 22N, a brief flow chart of the Send the Output to the Field routine 2212 is shown. After a setup step (block 2234), this routine simply calls the DAC Control routine 2236 to write the two byte value into the digital to analog converter circuit. The DAC Control routine is shown in the self explanatory flow chart of Figure 22O.

[0307] Referring to again to Figure 22A, the next routine to be executed is shown to be the Check for a Test #07 Error routine 2238. This routine is shown in the flow chart of Figure 22P. As shown in Figure 22P, a series of evaluations are made to determine if the NI Test Fail counter should be incremented (block 2240), and ultimately flag an NI Test Failure (block 2242) if too many tests have failed (diamond 2244). In this regard, it will be recalled that during Test #07, the ME SAO board must be driving the entire output by itself for each of its channels. Thus, if the SAO board is not driving the entire output by itself, its output voltage is at the maximum, and current is going to the field, then the NI Test Failure counter will provide a period of time to reach the required goal. However, if the goal of driving the output by itself cannot be reached within a reasonable period of time (for example, the NI Test Failure counter has exceeded 30), then an error condition will be flagged.

[0308] Figure 22A shows that the final routine to be executed is the Handle Output Problems routine 2246. The Handle Output Problems routine 2246 is shown in Figures 220-22R. As will be seen from these figures, this routine is used to set or clear a number or different flags depending upon the conditions specified. Thus, for example, if the Out of Control count for the channel being evaluated has not exceeded a predetermined amount (for example, 53), then three different flags will be cleared (diamond 2248). If the Out of Control count exceeded a predetermined amount, then an evaluation will be made as to whether the track measurement was lower than the output value (diamond 2252). If the answer is YES, then the "Almost Out of Control High", the "Out of Control High" and the "Out of Control High ME = 0" flags will be cleared (block 2254). Additionally, the "Out of Control Low" flag will be set, as the output to the field is lower than it should be.

[0309] In contrast, if the error is on the high side (block 2250 generates a NO), then the "Out of Control High" flag will be set and the "Out of Control Low" flag will be cleared (block 2258). Then, the operational amplifier track signal OAT will be evaluated to see if it is near zero (diamond 2260). If it is not near zero, then the OAT signal will be re-measured, as the DAC was commanded previously to reduce its output (block 2262). If this additional measurement does not show the desired response, then the "OAT <> DAC" flag will be set (block 2264).

[0310] Figure 22R shows that the "Almost Out of Control High Me = 0" flag will first be set (block 2266) and the Out of Control count will be zeroed (block 2265) if the difference between the output value and the track measurement is greater than an abort deviation value, such as 2% (diamonds 2268-2270). Then, during the next pass through this routine that the error count has exceeded a predetermined amount, the "Out of Control High ME = 0" flag will be set (block 2272) if the "Almost Out of Control High ME = 0" flag has not been cleared (block 2274). Forcing this delay in the setting of the OOCHME = 0 bit prevents false errors from being reported.

[0311] Referring now to Figures 23A-23I, a set of flow charts is shown for the NI Testing routine 1926 of Figure 21 B. Diamond 2300 indicates that this NI Testing routine may be skipped, such as when an error has been detected by the Handle Error Conditions routine. Diamond 2302 indicates that the NI Testing routine will not be performed during those one-second periods when the Test #00 insignia is utilized. Additionally, diamonds 2304-2306 indicate that the NI Testing routine will not be performed when an error is encountered on the channel to be tested or when the controller for this SAO board commands an output value which is less than a minimum value (for example, 4 mA). While NI testing could be performed when the commanded output value is near zero, it is preferred that NI testing be deferred, as the abort switches for any zero output channel will be opened and it will not be possible to conduct a complete test (e.g, Test #7).

[0312] In the event that this SAO board or one of the other SAO boards is being tested (diamond 2308), then this SAO board will look to see which test is being conducted. In this regard, it should be appreciated that this SAO board (for example, SAO circuit board 600) does need to take any action for Tests #12-16 or #21-26, as any necessary action will be taken by its controller (e.g, controller 92). In the event that Test #11 is being conducted (diamond 2310), then the NI Testing routine will cause this SAO board to assume the necessary output being shed by its neighboring SAO board designated as N1 (for example, SAO board 602). However, it should be noted at this point that the NI Testing routine 1926 does not specifically test for Test #21. This is due to the fact that the NI Testing routine being performed by the N1 SAO board will have the N2 SAO board designated as its neighbor N1. In other words, the NI Testing routine 1926 builds in a preference for which SAO board should begin to assume the output being shed by another SAO board.

Specifically, in this instance, the preference is made for the SAO board which has most recently completed Test #7, as this particular test evaluates the SAO board's ability to assume the entire output

[0313] In the event that the NI test being conducted is not Test #11, then the sibling wait counter will be cleared to permit immediate action if necessary (block 2312). Then, it will be determined if the NI test being conducted is Test #17 or Test #27 (diamond 2314 of Figure 23B). If the answer is NO, then the NI Test routine 1926 will be ended for this call. However, if one of these two NI tests are being conducted, then the sibling wait counter will be loaded with a value which will permit the SAO board under test time to ramp up its output (block 2316). Then, a determination will be made as to whether this SAO board is driving any of the output (diamond 2318), the appropriate rampdown rate will be chosen (blocks 2192 and 2198) as the result, and the output value will sent to the field (block 2212). If the DAC output is not zero, then a flag will be set to indicate that this SAO board has not finished ramping down (block 2320).

[0314] If this SAO board is currently being tested (diamond 2308), then a flag will be set to indicate that all lower NI routines in this SAO cycle should be skipped (block 2322). Then, if Test #07 is being conducted (diamond 2324) or if Test #11 is being conducted, a determination will be made as to whether this SAO board is driving any power (diamond 2326) by examining the voltage across the ME resistor and the appropriate flag will be set (block 2328). Then, the contribution to the field will be evaluated (diamonds 2330-2336) by examining the voltage drop across the ME resistor. If this SAO board is driving 100% of the output, the DAC output is at its maximum and the track output is at the proper value, then the NI test will be successfully completed (diamond 2338). Otherwise, additional determinations will need to be made and the appropriate action taken during this pass through the NI Testing routine 1926. For example, if this SAO board is driving more than 25% of the output value (diamond 2334), but less than 100% of the output value (diamond 2336), then 0.05% will be added to the value supplied to the DAC (block 2340). Then, the NI Testing routine 1926 will exit at this point until it is called upon again to evaluate the contribution that this SAO board is making to the output. If the DAC output is at its maximum, and this board is not driving 100% of the output, the test failure counter is increased.

[0315] If the answer to diamond 2324 on Figure 23A was NO, then the NI Testing routine 1926 will jump to point "A" on Figure 23C to begin checking to see which of other the NI tests are being conducted (diamonds 2342-2344 on Figure 23C, diamonds 2346-2348 on Figure 23D, diamond 2350 on Figure 23E, and diamond 2352 on Figure 23H). As will be appreciated from a review of Figures 23C-23I, the NI Testing routine follows a specific regimen for each of the NI tests. Thus, for example, in the case of Test #01, the SAO board will attempt to ramp itself down until a zero output is achieved (diamonds 2354-2356). Once a zero output is achieved, the NI Testing routine 1926 will jump to point "D" on Figure 23I. If the rampdown is unsuccessful, the controller is flagged not to test this channel and subsequent tests in the cycle will locate the problems on the neighboring boards.

[0316] As indicated in Figure 23I, a check will be made to see if the voltage measured on the low side of the ME resistor with respect to ground is too high for a DAC output of zero (diamond 2358). If the voltage is too high (for example, 0.037), the diode has been shorted and the NI Test Failure counter will be incremented (block 2360). Then, the NI Test Failure counter itself will be checked to see if the present count has exceeded its predetermined limit, such as 40 failures (diamond 2362). This failure count is set relatively high in comparison to the failure count maintained by the controllers (e.g, only 1 failure is permitted at the controller level), in light of the fact that the SAO boards are repeatedly executing their programs many times relative to the process control cycle timing employed by the controllers 92-96. If the count limit has been exceeded, then a flag will be set to indicate that an NI test failure has occurred (block 2364). However, as indicated by diamond 2366 and the additional entry points "B" and "E", the NI test failure flag will only be set if this SAO board was conducting the NI test, as opposed to one of its neighboring SAO boards.

[0317] In the case of Test #02, Figure 23D shows that the DAC output will be evaluated to determine if the SAO board was able to ramp down this channel (diamond 2368). Assuming that this channel was able to ramp down to zero, then the SAO microprocessor will set the channel output to the abort test voltage (block 2370), allow time for the output to settle (block 2372), and measure the operational amplifier track ("OAT") voltage signal (block 2374). Then, a determination will be made as to whether the OAT voltage level for this channel is in the expected or acceptable band, such as 150-700 mV (diamond 2376). If the answer is YES, then Test #02 will be successfully completed for this particular channel. However, if the answer is NO, then the NI Testing routine 1926 will jump to point "E" on Figure 231, where the NI Test Failure counter will be incremented. In any event, it should be appreciated that each of the analog output channels will be serviced in turn each time the NI Testing routine 1926 is called from the main SAO program 1900.

[0318] In the case of Test #03, Figure 23E shows that the NI Testing routine 1926 will ultimately measure the voltage on the high side of the ME resistor with respect to ground (block 2378), provided that this channel was able to ramp down to zero (diamond 2380) and determine if it is low enough (e.g,. 150mv) (diamond 2382). If the voltage is not sufficiently low, then one or both of the abort switches have not opened. In this regard, it should be noted that the opening of the DN1 and DN2 abort switches will be performed independently by the N1 and N2 controllers, respectively, according to the time chart discussed above.

[0319] In the case of Test #5, Figures 23E-23G show that two tests are actually conducted. First, the operation of the abort switches DN1 and DN2 are again tested through a ME resistor measurement while the ATV signal is being

produced (block 2384). Then, assuming that this test was successful, the ability to disable the operational amplifier will be tested. This test is accomplished by first checking to see if the SAO board Deadman is "open" (diamond 2386). This check is made by causing the microprocessor 610 to read the "NOT DEAD" signal from the Deadman Timer 649 of Figure 12D. If the answer is Yes, then the operational amplifier 608 should be disabled. If the answer is NO, then all of the operational amplifier's 608 on the SAO conducting the test will be disabled (block 2388). The DAC will then be commanded to output the Deadman Test Voltage, such as 3v (block 2390). The NOT DEAD signal will be checked again (diamond 2392), and then OAT signal will be read for the channel being tested if the Deadman is not open (block 2394). In this case, all of the SAO board operational amplifiers 608 will be re-enabled (block 2396), and then the OAT voltage will be checked to see if it is high enough (diamond 2398). Assuming that the OAT was high enough (for example, the Deadman Test Voltage level), or if the Deadman was not already opened, then the operational amplifiers will be disabled (block 2400). Next, the OAT voltage will be measured (block 2402). Thereafter, the DAC will be re-set to the ATV level (block 2404), and the operational amplifiers will be re-enabled (block 2406). After this step, then the voltage from the Deadman voltage input will be evaluated to see if was possible to disable the operational amplifier (diamond 2408).

[0320] In the case of Tests #04 and #06 (diamond 2352), Figure 23H shows that a voltage measurement will be made on the high side of the ME resistor (block 2410). Again, it should be appreciated that the necessary steps of opening and closing the abort switches DN1 and DN2 are handled by the neighboring controllers in accordance with the time chart set forth above.

[0321] From the above description of the preferred embodiment, it should be appreciated that the field computer units 12 operate in accordance with a predetermined process control cycle. In other words, all of the signal communication and input/output processing functions of the field computer units are performed within a single process control cycle, such as a one second interval. While the clock signals for each of the network controllers 16 and the field computer unit controllers 92-96 are all adjusted during this process control cycle to maintain the clock signals within a given tolerance, an adjustable timeline is generally provided to facilitate cooperation between these interface system components. For example, in one form of the present invention, the synchronization message is sent by the network controllers 16 to each of the field computer units 12 at the beginning of a new process control cycle. The field computer units 12 will in turn be looking for this two byte message within a given period of time (for example, 1.5 milli-seconds). After the network controllers 16 determine the necessary communication paths, they will send the appropriate digital and analog output values to each of the field computer units. Then, the controllers 92-96 will exchange this information in order to perform the independent arbitration methods described above. However, in the event that communication from the network controllers 16 is not received by a field computer unit or communication is not received by one of the controllers 92-96 from its neighboring controllers, these components will nevertheless proceed to perform their tasks after a suitable period of time. Thus, for example, the previously supplied Fail-Last and Fail-Safe instructions may be implemented according to the output arbitration methods discussed above.

[0322] Additionally, the action timeline should also permitted the non-intrusive testing of digital and analog outputs to be performed periodically as set forth above. The timeline may also be constructed to permit further testing of system components. For example, it may be advantageous to test the RAM memory U42 in each of the controllers 92-96 within an available time slot. This test may be accomplished by first writing a specific value (for example, 55hex) into each storage location of an unused section of the RAM memory, and then reading each location to verify the integrity of this section of memory. Then, a portion of the input or output data table may be moved to this verified section of RAM memory, and the memory section from which this data was taken could be verified in the same manner. However, it is preferred that a different value is written into this used section of memory (for example, AAhex). The data could then be replaced once it was determined that there were no memory errors. In this way, the entire RAM memory U42 may be periodically tested. If a memory error was found, then this memory section could be tested again and/or a general "problem" status bit could be set to inform the process control computer 14 of the presence of a error. As with the other errors discussed above, the process control computer may request the status of a specific error bit which would identify an error in the RAM memory U42.

[0323] Referring generally to Figures 24A-27M, a set of flow charts are shown to illustrate the methods of downloading updated software according to the present invention. In this regard, the present invention advantageously provides the ability to download updated software throughout the process control interface system 10 without having to interrupt the physical process being controlled. More specifically, the present invention permits updated or new software to be selectively transmitted from one of the network controllers 16 to each of the breakout circuits 26 in the interface system 10, and to each of the field computer units 12 in the interface system.

[0324] Thus, the software contained in each of the major components of the distributed process control interface system 10 according to the present invention may be individually updated or collectively updated in groups. In other words, it may be beneficial to update the software for each of the field computer units 12 at one time and update the software for each of the breakout circuits 26 at another time. Conversely, it may be appropriate to update the software throughout the interface system 10, starting with the breakout circuits 26 and ending with the field computer units 12.

[0325] Importantly, each of these updating operations may be carried out while process control operations are continuing. For example, while one of the process control computers 14a-14b is being used for process control, the other process control computer may switch over to perform one or more downloading operations. Another advantage of the method and system according to the present invention is the ability to download updated software into a plurality of breakout circuits 26 or field computer units 12 during the same downloading operation. Thus, for example, when a successful downloading procedure has been verified for each of the field computer units, then the redundant controller 92-96 in each of the field computer units 12 which received the new software may startup using this software in the same process control cycle.

[0326] In one form of the present invention, it is preferred that a successful download operation be verified for all interface system components to which the new software was addressed before any of these system components is permitted to startup on the new software. In other words, if the Left controllers 92 in all of the field computer units 12 verified a completely accurate reception of the new software, then they will all be permitted to startup on the new software. Otherwise, they will all be commanded to start back up using the old software which was previously contained in these controllers 92. At this point, the downloading procedure may be tried again, or the hardware for the controller (s) that were unable to verify the correctness of the new software could be checked.

[0327] Once the updated software has been verified for all of the Left controllers 92, then these controllers may be commanded to transmit a copy of this software to the Middle controllers 94 in each of the field computer units 12. In this regard, it should be appreciated that the serial communication links between the controllers 92-96 in the field computer units 12 enable one of the controllers 92-96 to transfer a copy of updated software into one or both of the other controllers. Alternatively, it should be appreciated that once the Left and Middle controllers 92-94 are operating with updated software, then the Right controller 96 could receive a copy of this updated software from its process control computer (for example, process control computer 14b). In other words, the process control computer 14a could return to its process control operations, and the process control computer 14b switched over to a downloading operation.

[0328] Of course, both of the process control computers 14a-14b could be shut down from a process control standpoint, so that both the Left controller 92 and the Right controller 96 in each of the field computer units could receive the identical updated software. However, this could require an interruption in the physical process being controlled. In any event, it should be appreciated that the only downloading function that could be implemented with both of the process control computers 14a-14b running process control operations, is the transfer of updated software from either the Left controller 92 or the Right controller 96 to the Middle controller 94, as the process control computers 14a-14b do not need to be involved in this procedure in accordance with the present invention.

[0329] Referring specifically to Figure 24A, an abbreviated flow chart of the field computer unit main "Femmai" 2420 is shown. Flow chart 2420 indicates that each of the field computer units 12 will generally be conducting the process control activities discussed above (block 2422), unless a "DOWNLDF" bit has been set in response to a download command (diamond 2424). The setting of the DOWNLDF bit is actually accomplished in the serial port interrupt routine 2426 shown in Figure 24B. In this particular application, the field download command is simply identified as command "113" (diamond 2428). Prior to the clearing of the DOWNLDF bit (block 2430), the value of this bit will be placed in a neighbor communication message in order to inform the neighboring controllers that this controller is receiving new software. This action will prevent the neighboring controllers from attempting to reset the controller receiving updated software. It should also be noted that the field communication routine "Fcomm" of Figure 24C is used to receive download commands from the process control computer 14.

[0330] Assuming that the process control computer 14 has issued the download command, then the field computer unit 12 will jump to the "FIO_DOWN_LD" routine shown in Figures 24E-24G (block 2432). The FIO_Down_LD routine 2432 is sometimes referred to herein as the FIO Download routine. As will be apparent from this flow chart, the FIO_DOWN_LD routine provides a main routine for a series of subroutines, which are shown in Figures 26G-26P and 27I-27M. These sub-routines enable the field computer unit to receive and verify the downloaded software, assuming that this software is intended for the field computer unit. However, before discussing the these flow charts further, the transmission of downloading commands will first be examined.

[0331] Referring to Figure 25A, an abbreviated flow chart of the Netmain program or routine 2500 is shown. In this regard, the Netmain program 2500 represents a main program for the network controller 16. This Netmain program follows a normal process control timeline, such as indicated by the "Do Process Control" block 2502. Nevertheless, at an early point in the main loop of the Netmain program, it is determined whether a downloading operation has been requested (diamond 2504). This request is determined by checking for the presence of a "DOWNLD" bit, which is set in the flow chart shown in Figure 25B. If the DOWNLD bit is set, then the Netmain program will jump to the GET_CODE routine 2506 shown generally in Figures 25C-25E. Otherwise, normal process control functions, such as transferring data received from the field computer units 12 to the process control computer 14, will be performed, assuming that the process control computer has not been taken off its process control regimen.

[0332] Thereafter, the NCOMM routine 2508 will be performed. This routine is shown through the flow chart of Figure

25P. As indicated by this flow chart, the NCOMM routine relates to the loading of updated software into the Middle controller 94 of the field computer units 12. More specifically, the NCOMM routine will check to see if a command has been entered to load the Middle controllers 94 with updated software (diamond 2510). As will be appreciated from the description below of the CBTDEC routine of Figure 25B, the request for a Middle download may be entered by an operator through the debug panel 18. If a Middle download request has been made, then a specific command will be sent downstream by the network controller (block 2512) to all of the field computer units 12 through a Send Command routine 2514 shown in Figure 25Q. While it is preferred that all of the Middle controllers 94 be updated together, it should be appreciated that in the appropriate application it may be permit a selection of some but not all Middle controllers 94.

[0333] The Middle download command will be received and acted upon by the BCOMM routine 2516 of Figure 24D, which is contained in each of the breakout circuits 26 connected to one of the process control computer 14a-14b. The BCOMM routine 2516 will pass the Middle download command to all of its output ports to eventually be acted upon by the FCOMM routine 2518 in each of the field computer units 12. The FCOMM routine is shown in Figure 24C. The FCOMM routine 2518 writes the Middle download command into XRAM, where it is read by the SIDE_LOAD routine 2520 of Figures 26Q-26R. The SIDE_LOAD routine 2520 in the Left controller 92 or the Right controller 96 determines the port address of the Middle controller 94, sends the Middle download command to the Middle controller, and listens for an answer. The NEIGHBOR subroutine 2522 of Figure 26S in the Middle controller 94 receives this command, sets its serial port to receive from the neighboring controller that sent the command, and then jumps out of its process control time line to the FIO_DOWN_LD routine 2524 of Figures 24E-24G to receive the new software.

[0334] In the meantime, the NCOMM routine 2508 will enable the CHECK_MID routine 2526 of Figures 26R-26S (block 2528 in Figure 26P) and initialize a waiting period for the CHECK_MID routine to be executed (block 2530). The CHECK_MID routine 2526 is also shown as a block in the Netmain loop of Figure 26A. The CHECK_MID routine 2526 is used to verify that a copy of the updated software from either the Left controller 92 or the Right controller 96 has been successfully transferred to the Middle controller 94. In this regard, the Middle controller 94 will perform checksum calculations and comparisons, and upon successful completion, it will respond to the sending controller with its checksums. These checksums may be comprised of "exclusive or", "rotated exclusive or" and "sum of code" checksums. These checksums will be compared with the checksums which are embedded in the software code sent to the Middle controller 94. The sending controller will compare the checksum from the Middle controller 94, and if they agree with its own checksums, then a bit will be set in a byte which will be transmitted to the network controller 16 during normal input communication.

[0335] Once all of the Good Checksum messages have been received by the network controller 16, then a similar confirmation message will be displayed on the debug panel 18 of the network controller 16. The display of the Good Checksum message on the debug panel 18 will enable the operator to know that the Middle controller 94 may be started up on the new software. In this regard, the operator may then depress the buttons on the debug panel 18 which will cause a "Transplant" command to be sent to each of the field computer units 12 via the NCOMM routine 2508 of Figure 26P. However, if a checksum error has been detected, then a "Cold Feet" command will automatically be sent to all of the field computer units via the NCOMM routine 2508. The Cold Feet command will cause the Middle controllers 94 to start up (that is, be reset) using the old or prior software. A suitable message to this effect will also be displayed on the debug panel 18.

[0336] The relevant portion of the common button decoder "CBTDEC" routine 2528 is shown in Figure 25B. The CBTDEC routine 2528 is referred to as being common in that it is preferably contained in each of the interface system components that contain a debug panel (that is, the network controllers 16, the breakout circuits 26 and the field computer units 12). This is why the CBTDEC routine 2528 contains a determination as to whether this component is a network controller 16 for each of the functions listed (for example, diamond 2530). Each of the functions identified in the CBTDEC routine refer to a specific downloading operation. Thus, for example, the Function 1E is used to initiate the downloading of updated software into the Middle controllers 94. As indicated by diamond 2532, the interface system 10 will only permit the Middle download command to be transmitted after at least one of the Left or Right controllers has successfully received updated software. Once the operator has depressed theappropriate debug panel buttons, then the MID_LOAD bit will be set (block 2534). One or more messages may then be displayed on the debug panel, such as "Loading Middle Field I/O" (display block 2536).

[0337] Function 1D is used to automatically cause the Cold Feet command to be sent to all of the components to whom new software code was addressed. In this regard, the downloaded software code will be ignored, and the components will startup on the old software code.

[0338] Similarly, Function 1C is used to enable the operator to cause the Transplant command to be sent to all of the devices to whom new software code was addressed. The Transplant command can also be sent via the NCOMM routine 2508 to start the Middle controller 94 on the new software code if the network controller 16 is executing its process control time line. Once this command is received, then the REPROG routine 2538 of Figure 26D will be executed. The REPROG routine 2538 will cause the newly downloaded software to be copied from data memory (for

example, XRAM) into program memory. It should be noted that the CBTDEC routine 2528 will not permit the Transplant command to be sent if the checksum verifications have indicated the presence of an error (diamond 2540).

[0339] Function 1B is used to move new software from one of the process control computers 14a-14b to the XRAM circuit contained in its network controller 16. The selection of Function 1B will cause the command code "113" to be transmitted from the network controller 16. In this regard, diamond 2542 indicates that this function will not be performed if this process control computer is currently being used for process control. The downstream devices or interface system components which receive the new software code is determined from the 'start and stop' switches on the breakout circuits 26. Since the breakout circuits 26 do not know what type of device or devices they are connected to downstream, it is preferred that all of these devices will receive new code intended for the breakout circuits when that option is selected. In this regard, the preferred procedure is for the new "overheads" software code to have an embedded program ID that may be used downstream to determine whether the receiving device should use the new software code. While the network controller 16 will initially know which devices are connected to it downstream from a call to the process control computer, it should be appreciated that the network controller 16 could poll the fiber optic network prior to the downloading operation to determine which devices are currently connected to it.

[0340] Verification of downloaded breakout circuit software code and field computer unit software code is accomplished at the network controller 16 by polling the known field computer units 12 on the fiber optic network. In this regard, it should be noted that each of the breakout circuits will preferably verify new breakout circuit software received before transmitting this software to any devices to which they are connected. Thus, for example if the breakout circuit 26f of Figure 2 detects that it has not received a complete or accurate transmission, it will not send this software to the breakout circuit 26g. In one form of the present invention, the breakout circuits will not attempt to verify the accuracy of new field computer unit software, as the breakout circuits 26 are not provided with sufficient free memory to check this software. More specifically, new field computer unit software is transmitted in two packets (for example, 32K each), whereas new breakout circuit software only requires a single transmission (for example, 32K). However, it should be understood that the memory capacity of the breakout circuits 26 could be increased in the appropriate application.

[0341] When the network controller 16 receives the checksums that agree with the checksums of the transmitted program, from all of the known field computer units 12, it will present the operator with a choice of starting on the new software code or on the old software code via a message prompt on the debug panel 18. However, if the network controller 16 receives a bad checksum or times out while requesting a checksum message from any of the known field computer units 12, then all of these devices will be sent the Cold Feet command code to automatically cause a start up on the old software. Indeed, even if all of the known field computer units 12 sent good checksum messages, it is preferred that the interface system automatically cause a start up on the old software, if the operator does not respond to the prompted choice within a predetermined timeout period. In any event, if the time-out timer expires during the verification process, then the downloading operation will automatically terminate with a "Time-Out" message being displayed on the debug panel 18.

[0342] Once the DOWNLD bit has been set via Function 1B of the CBTDEC routine 2528 (block 2544), this bit will be detected by the Netmain routine 2500 of Figure 25A. This will in turn cause the network controller 16 to jump to the GET_CODE routine 2506 of Figures 25C-25E. The GET_CODE routine 2506 detects what devices have been selected for software updating and reacts accordingly. As indicated by diamond 2546, the Middle controller 94 in the field computer units 12 may be downloaded through the GET_CODE routine 2506. However, this procedure is only implemented when both of the process control computers 14a-14b are "down" with respect to process control operations. In this case, the GET_CODE routine calls the JUMPOUT routine 2548 shown in Figure 25H. The JUMPOUT routine 2548 will cause a one second burst of back to back download commands to be transmitted out the main port of the network controller 16. These consecutive download commands will cause the breakout circuits 26 and/or one side of the field computer units 12 to jump out of their process control time line, and sit in a tight receive loop (with a time-out timer running) looking for further instructions upstream. From this point, the downloading and verification process will be automatically performed.

[0343] Assuming that the Middle controller 94 is not involved with the downloading process at this point, then the network controller will then receive new software from the process control computer 14. In one form of the present invention, this software is preferably sent in the following four blocks or packets: (1) network controller software (for example, 32K), (2) breakout circuit software (for example, 32K), and (3) field computer unit software (for example, two passes of 32K each). In this regard, the read "Which One" block 2550 refers to the numbers (1), (2) or (3) for these software transfers. As the field computer unit software requires two transmissions or passes, the diamond 2552 indicates that the network controller 16 will check whether or not it is receiving the second pass of the number (3) software transfer. If any other number is detected, then the transfer request will be interpreted as a bad selection (diamond 2554), and the network controller 16 will revert to the Netmain routine (block 2556).

[0344] Assuming that the software transfer request is acceptable, then the network controller will determine if the software being transferred is network controller software (diamond 2558). If the software is not network controller software, then the FIO table will be check to see if it is empty "MT" (diamond 2560 on Figure 25D). In this regard, it

should be noted that the term FIO stands for Field Input/Output, and it is simply another way of referring to the field computer units. Assuming that the FIO table is not empty, or the software is network software, then the network controller 16 will request the next 32K packet of software (block 2562). The network controller 16 will then look for the next command code from the process control computer 14 (block 2564). The command code is received in two bytes, as indicated in Figure 25G.

[0345] Assuming that this is not the second pass for FIO software (diamond 2566), then the checksums will be stored in XRAM (block 2568). At this point, the network controller 16 will check if this software is FIO software (diamond 2570), and verify the accuracy of the transmission if the software is not FIO software (block 2572). In this regard, Figure 25O shows the flow chart of the Verify routine 2572. If the checksums did not match those embedded in the software (diamond 2574), then a "Bad Checksum" message will be displayed on the debug panel 18, and the network controller 16 will revert to the exiting "Old" program (block 2576).

[0346] If the checksums matched those embedded in the transferred software, then the network controller 16 will check if this packet is network controller software (diamond 2578). If the software is not network controller software, then the network controller 16 will call the Jumpout routine 2548 of Figure 25H, and then put downstream devices in a receive loop (block 2580). The network controller 16 will then request the next software transfer (block 2582). The network controller 16 will then check if the received software is FIO software (diamond 2584). If the software is FIO software, then a check will be made to see if this is the first or second pass (diamond 2586). If it is the first pass, then the network controller 16 will bump the "Which One" number to (4) to set up the second pass (block 2588). If this was the second pass, then the network controller will call the Verify Downloaded Program routine 2590 shown in Figures 25K-25N. Assuming that the Verify Downloaded Program routine 2590 did not terminate with a revert to Old Program block, then the a message will be displayed on the debug panel 18 (block 2592), which will permit the operator a choice of implementing the New Program (block 2594) or reverting to the Old Program (block 2596).

[0347] Figure 25I shows the New Program routine 2594, while Figure 25J shows the Old Program routine 2596. In this regard, it should be noted that the New Program routine 2594 calls the Reprog routine 2538 shown in Figures 26D-26F. As shown in Figure 25E, the New Program routine 2594 will be executed in response to the selection of Function 1 C on Figure 25B. In this regard, the selection of Function 1C will cause the transmission of command code "114" from the network controller 16. Figure 25E also shows that the Old Program routine 2596 will be executed in response to the selection of Function 1D on Figure 25B. The selection of Function 1D will cause the transmission of command code "115" from the network controller 16.

[0348] It should be noted that the Verify Downloaded Program routine 2590 calls the Get One routine 2598, which is shown in Figure 25F. The Get One routine 2598 is simply a way of providing relatively large delays, such as for a one second timeout. As shown in Figure 25F, the Get One routine controls the decrementing of several counters (for example, block 2600).

[0349] Turning now to the downloading process at the breakout circuits 26, the BCOMM routine 2516 of Figure 24D will call the Breakout Download routine 2602 of Figures 25T-25U. As shown in Figures 25T-25U, the Breakout Download routine 2602 will call various subroutines, such as the Jumpout routine 2604 of Figure 25Z, the Rcv_Init routine 2606 of Figure 27D, and the Get_One routine 2608 of Figure 25Y. The Breakout Download routine 2602 is also responsive to various commands received from process control computer 14 through the network controller 16. For example, in response to command "118", the Breakout Download routine 2602 will call the Check_Sums subroutine 2610 of Figures 25V-25W. Command code "118" is a request from the sending device which will cause the receiving device to send back the checksums received with the transmitted software. This will permit the sending device to compare these checksums with the embedded checksums in its program memory. Similarly, in response to command "122", the Breakout Download routine 2502 will call the Receive subroutine 2612 of Figures 26A-26B. The Receive routine 2612 will in turn call the Download subroutine 2614 of Figure 25X. The Breakout Download routine 2604 will also call the Tellall subroutine 2616 of Figure 26C, which will pass the command code to downstream devices.

[0350] If the software is determined to be Breakout circuit software (diamond 2618), then the Breakout Download routine 2602 will call the verify routine VXRAM 2572 of Figure 25O. If the checksums are correct (diamond 2620), then the Reprog routine 2538 of Figure 26D will be executed. Otherwise, a Bad Checksum message will be displayed (block 2622), and the breakout circuits will ultimately revert to the existing software through a timeout implementation. Figure 25U also shows that the Breakout Download routine 2602 will also respond to the command "115", which is used to cause a start up on the exiting software code (block 2624). In this regard, the Tellall subroutine 2616 will be called to pass this command downstream, and then a jump will be made back to the main program for the breakout circuits 26 (block 2626).

[0351] Referring again to Figures 24E-24G, it will be appreciated that the FIO Download routine 2524 has a number of similarities to the Breakout Download routine 2602 of Figures 25T-25U. Thus, for example, the FIO Download routine 2524 will call the Receive routine 2524 of Figure 26G in response to command "122". Additionally, the receipt of command "114" will cause the field computer units 12 to determine if the downloaded code is FIO software (diamond 2628). If the software received is not FIO software, then the Old Program subroutine 2630 of Figure 27I will be called. Other-

wise, the verify routine 2572 of Figure 25O will be called.

[0352] It should also be noted that the FIO Download routine 2524 will call the Neighbor subroutine 2632 in response to command "123". The Neighbor subroutine 2632 is shown in Figures 26J-26K. The Neighbor subroutine 2632 is used to transfer new software from one controller 100 to both of the neighboring controllers in the same field computer unit 12. In this regard, the Neighbor subroutine 2632 causes the serial port to be pointed at the Neighbor1 controller (block 2634), and a burst of command code "113" signals is sent to get the neighboring controller out of its process control time line (block 2636). The serial port is then pointed at the Neighbor2 controller (block 2638), and the command code "113" signals are sent to this controller (block 2640). An enable data mode command code "122" is also sent to these controllers. Upon receiving the command code "122", the neighboring controllers will branch to the Receive sub-routine 2524, and then perform the checksum test with the Verify routine 2572.

[0353] Figure 24G also shows that the FIO Download routine 2524 will check for command code "124" (diamond 2642). This command code is a request for the neighboring controller which received new software to send the checksums back to the sending controller. In this regard, it should be noted that the sending controller will wait a sufficient period of time for the neighboring controller to receive and verify the software before transmitting command code "124". If the checksums match the embedded checksums in the sending controller's program, then the process will be repeated for the other neighboring controller. If the checksums do not match, then the downloading process is terminated by the sending controller jumping to the start of its main program.

[0354] Referring to Figure 26V, a flow chart of the My Side Receive routine 2644 is shown. Due to the fact that the programs for the field computer units 12 are stored in RAM, the My Side Receive routine 2644 is used for loading the overheads software into a controller 100 which has just been installed in a field computer unit. The My Side Receive routine 2644 begins with a search for a program source. In this regard, the new controller will point to its Neighbor1 controller (block 2646), and then call the Neighbor subroutine 2648 of Figure 26W. The Neighbor subroutine 2648 will send a command code "120" signal to this neighboring controller (block 2650), and then it listens for a command code "121" signal reply (block 2652). If the new controller does not receive the expected reply within the timeout period set, then it will repeat the process with the Neighbor2 controller (block 2654). Again, if the expected reply is not received, then the new controller will point to the main serial port (block 2656) in order to receive its program software from the interface network. If the new controller detects a command code "113" while it is pointing at its main port, then it will jump to the FIO Download routine 2524 to receive its software as explained above.

[0355] If the new controller does receive the expected command code "121", then the Command subroutine 2658 of Figures 26X-26Y will be called to receive the overheads software. If the neighboring controller in module Side_Load of Figures 26Q-26R receives the command code "120", it will remember which port address this request came from, answer with the command code "121", and write this command code into the "DOWN" byte in XRAM. On the next invocation of the Side_Load routine, the sending neighbor will send the command code "122" to the new controller in order to put this controller into a data receiving mode, and send a block of program memory from its own program memory "PRAM". In this method of program transfer, the new software in written directly into the program memory of the receiving controller, and verification is not attempted until the program begins to run. If the transfer is unsuccessful, then the entire My Side Receive routine will be repeated again.

[0356] The present invention has been described in an illustrative manner. In this regard, it is evident that those skilled in the art once given the benefit of the foregoing disclosure, may now make modifications to the specific embodiments described herein without departing from the spirit of the present invention. Such modifications are to be considered within the scope of the present invention which is limited solely by the scope and spirit of the appended claims.

Claims

1. A method of non-intrusively testing an analog output circuit (600) in a field computer unit (12) having at least three redundant analog output circuits (600, 602, 604) which are each controlled by a correspondingly redundant computer (92, 94, 96), where each of said analog output circuits has a plurality of output channels and each of said output channels has an analog signal driver (608) which is responsive to one of said redundant computers and a set of abort switches (DN1, DN2) which are each controlled by a neighboring redundant computer, comprising the steps of:

   (a) selecting one of said analog output circuits to undergo said non-intrusive testing;
   (b) determining if the analog signal level provided by an analog signal driver for at least one of said output channels of said selected analog output circuit is below a predetermined threshold level;
   (c) causing said analog signal driver to increase its analog signal level to a predetermined test level which will not change the analog signal level provided to a field device for this output channel;

(d) determining if the analog signal level output from said analog signal driver was able to achieve said predetermined test level.

2. The method according to Claim 1, wherein said predetermined test level is a voltage level which is below the forward cut-in potential of a blocking diode in the output channel of said selected analog output circuit.

3. The method according to claim 1 or 2, further comprising the steps of:

(k) selecting one of said analog output circuits to undergo said non-intrusive testing;
(l) causing an analog signal driver for at least one of said output channels of said selected analog output circuit to reduce its analog signal level to a predetermined level over a period of time which will permit at least one of said neighboring redundant computers to increase its analog signal level output in order to maintain the analog signal level provided to a field device for this output channel before said non-intrusive testing was begun;
(m) determining if the analog signal level provided to said field device was reduced by a predetermined amount during the time that that analog signal level from said analog signal driver was being reduced; and
(n) restoring the analog signal level output of said analog signal driver to the analog signal level provided before it was reduced.

4. The method according to Claim 3, wherein the analog signal level output from a plurality of analog signal drivers for said selected analog output circuit are concomitantly reduced, so that a plurality of said output channels are tested during the same period of time.

5. The method according to Claim 4, wherein the analog signal level output from all of said analog signal drivers for said selected analog output circuit are concomitantly reduced, so that all of said output channels are tested during the same period of time.

6. The method according to one of Claims 3 to 5, wherein said predetermined level is a level where the analog signal driver no longer provides a substantial contribution to the analog signal level transmitted to said field device.

7. The method according to Claim 6, wherein said predetermined level is a substantially zero level contribution.

8. The method according to one of Claims 3 to 7, wherein the analog signal level output of said analog signal driver is restored to the analog signal level provided before it was reduced in the event that it was determined that the analog signal level provided to said field device was reduced by said predetermined amount during the time that that analog signal level from said analog signal driver was being reduced.

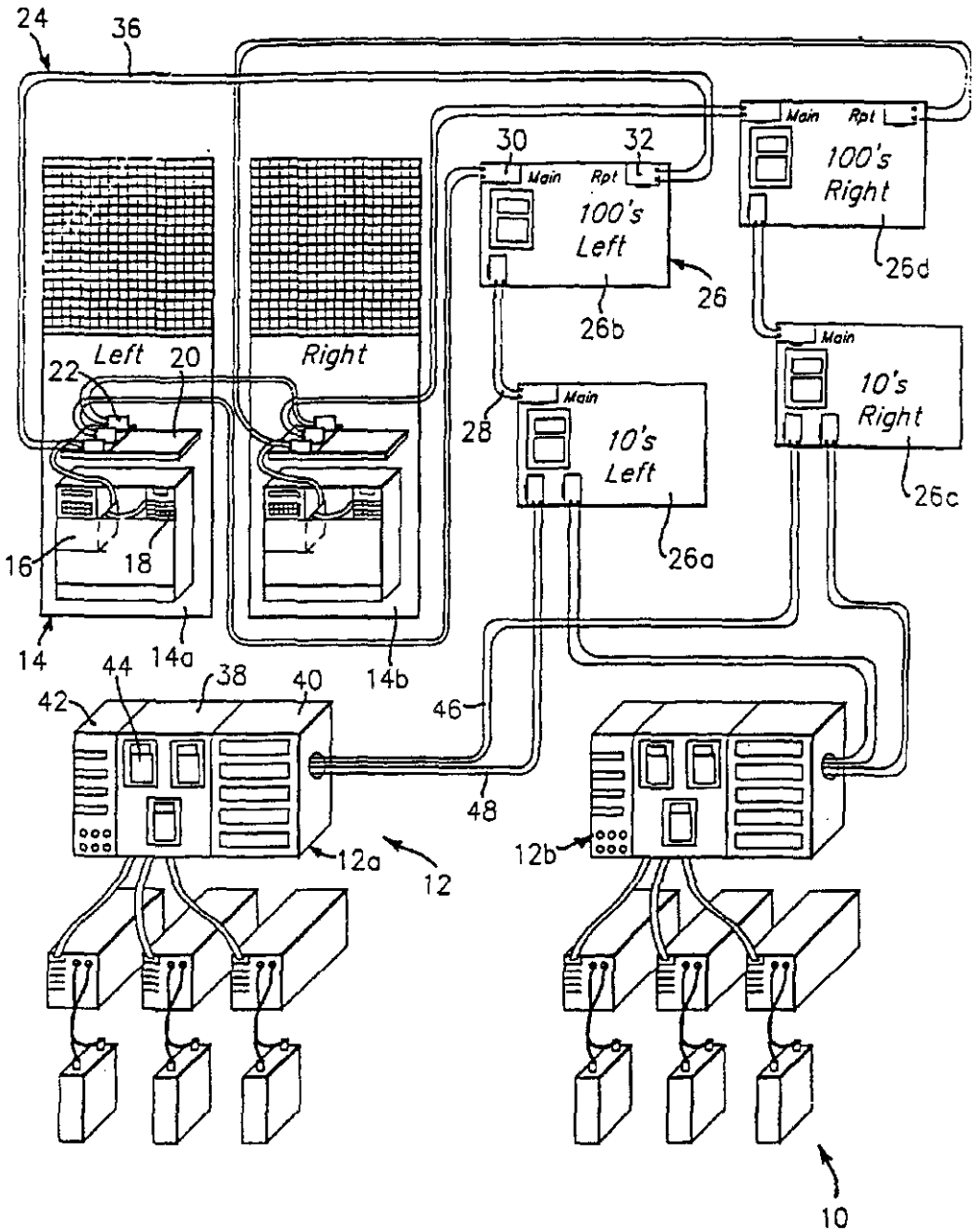
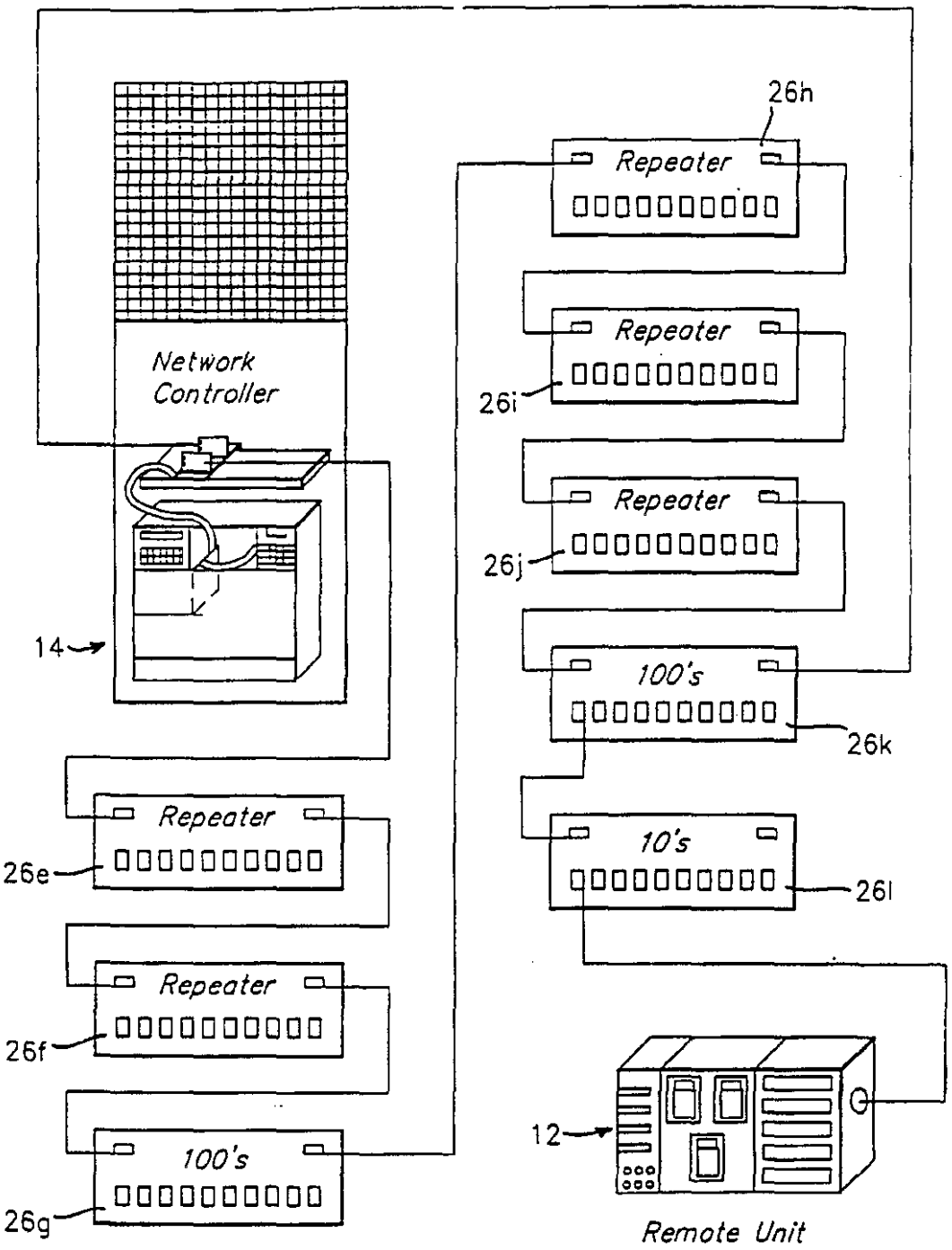

24
36
30
32
Main
Rpt
100's Left
26
26b
Main
Rpt
100's Right
26d
Main
10's Left
26a
Main
10's Right
26c
28
Left
20
22
Right
16
18
14
14a
14b
44
38
40
42
46
48
12a
12
12b
10

FIG. 1.


Network Controller
14
Repeater
26e
Repeater
26f
100's
26g
26h
Repeater
Repeater
26i
Repeater
26j
100's
26k
10's
26l
12
Remote Unit

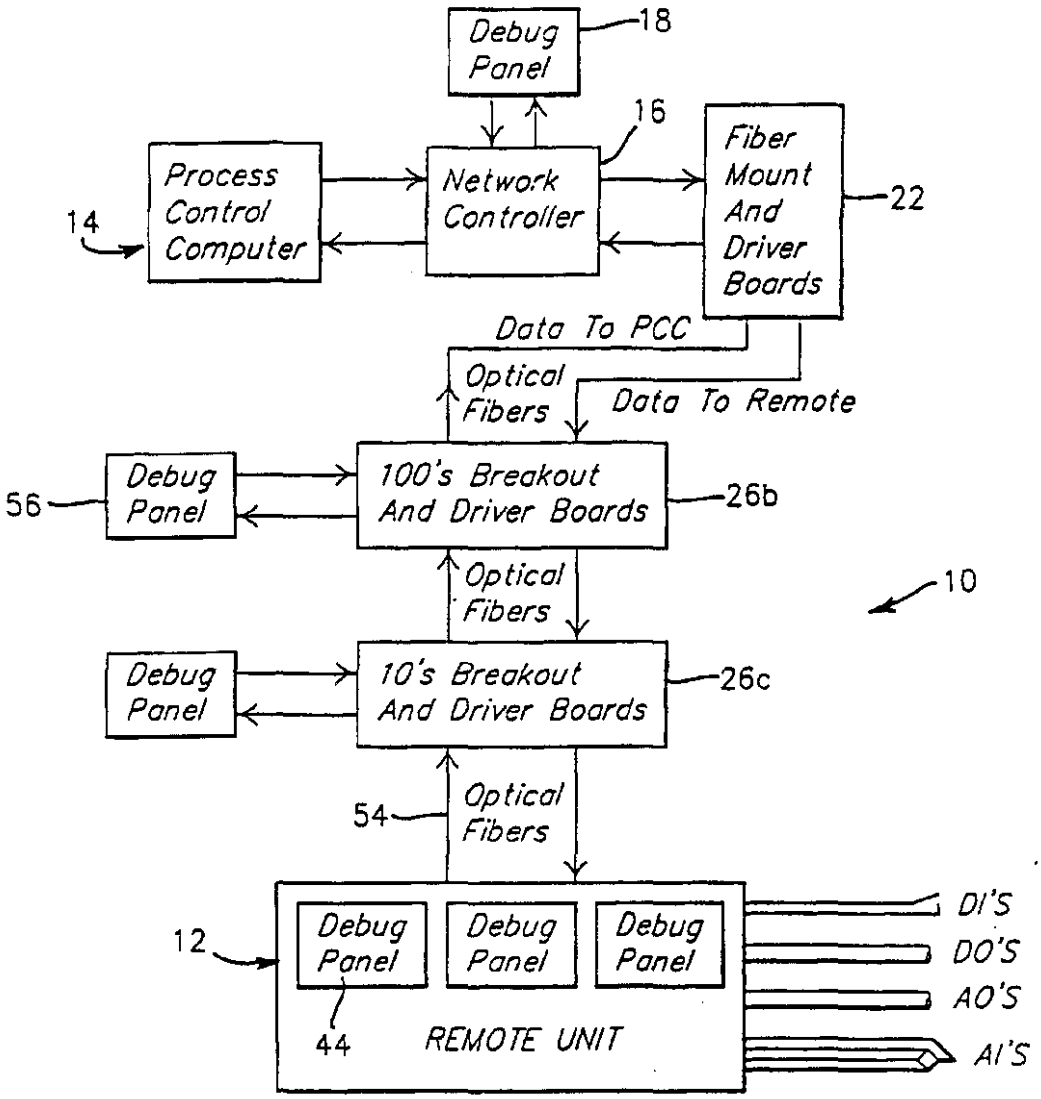
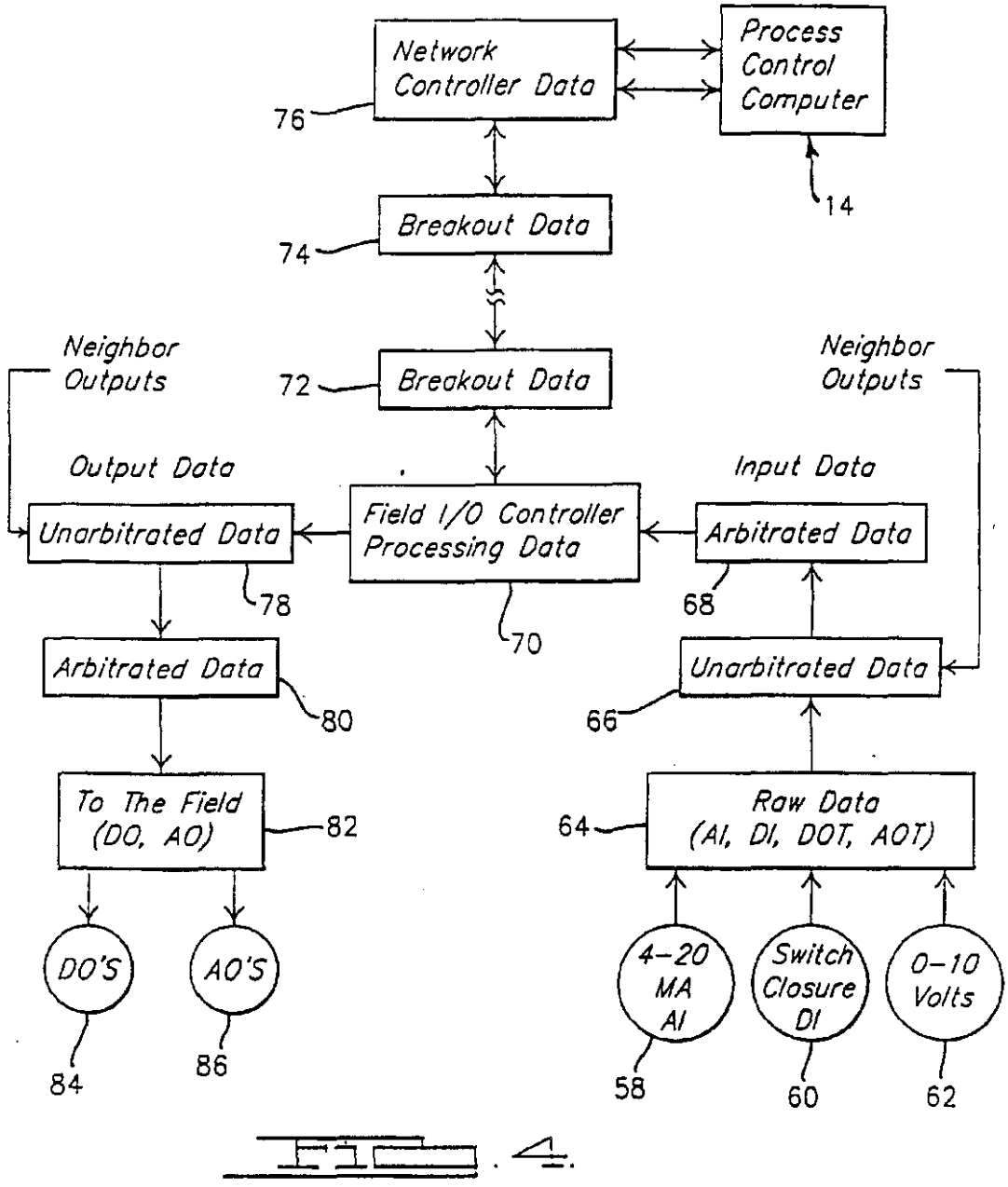

Debug Panel
18
16
Process Control Computer
14
Network Controller
Fiber Mount And Driver Boards
22
Data To PCC
Optical Fibers
Data To Remote
Debug Panel
56
100's Breakout And Driver Boards
26b
Optical Fibers
10
Debug Panel
10's Breakout And Driver Boards
26c
54
Optical Fibers
12
Debug Panel
Debug Panel
Debug Panel
44
REMOTE UNIT
DI'S
DO'S
AO'S
AI'S

FIG. 2.


Network Controller Data
Process Control Computer
76
14
Breakout Data
74
Breakout Data
72
Neighbor Outputs
Neighbor Outputs
Output Data
Input Data
Unarbitrated Data
Field I/O Controller Processing Data
Arbitrated Data
78
70
68
Arbitrated Data
Unarbitrated Data
80
66
To The Field (DO, AO)
82
Raw Data (AI, DI, DOT, AOT)
64
DO'S
AO'S
4-20 MA AI
Switch Closure DI
0-10 Volts
84
86
58
60
62

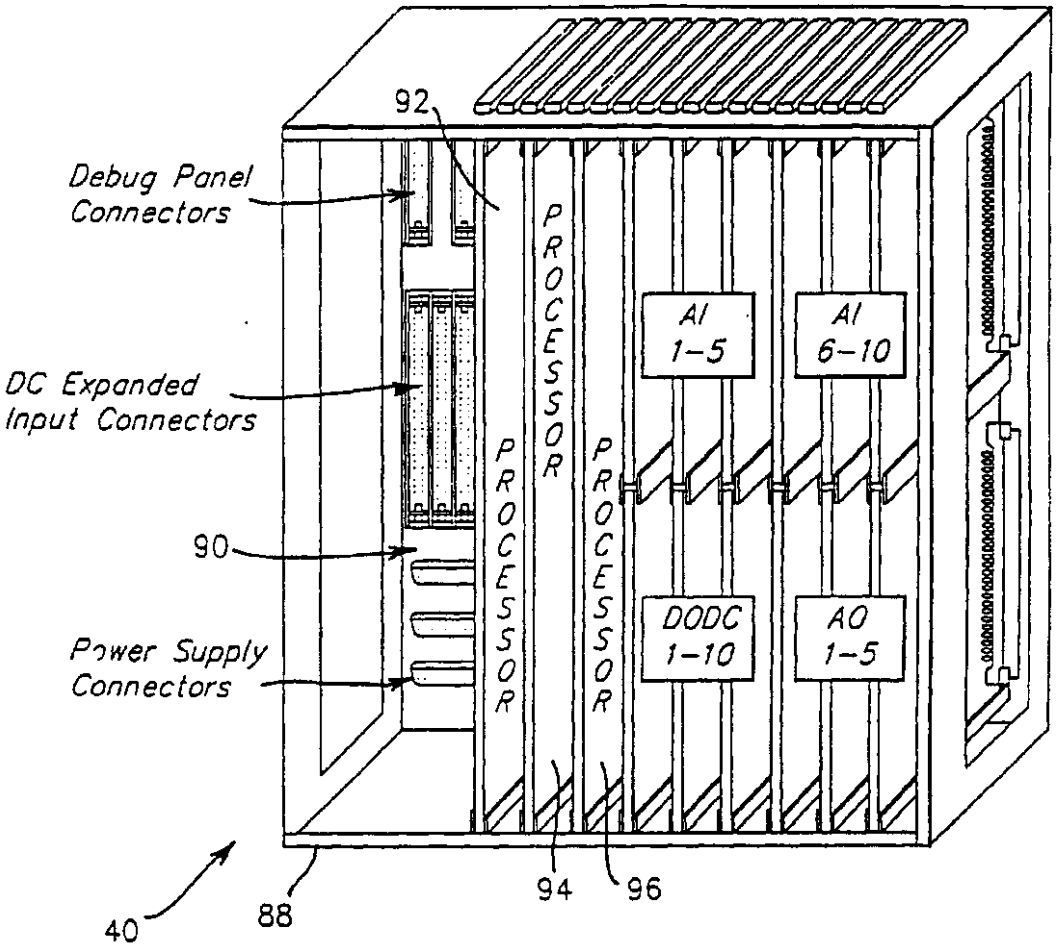

92
Debug Panel Connectors
DC Expanded Input Connectors
90
Power Supply Connectors
PROCESSOR
PROCESSOR
PROCESSOR
AI 1-5
AI 6-10
DODC 1-10
AO 1-5
40
88
94
96

FIG. 5.

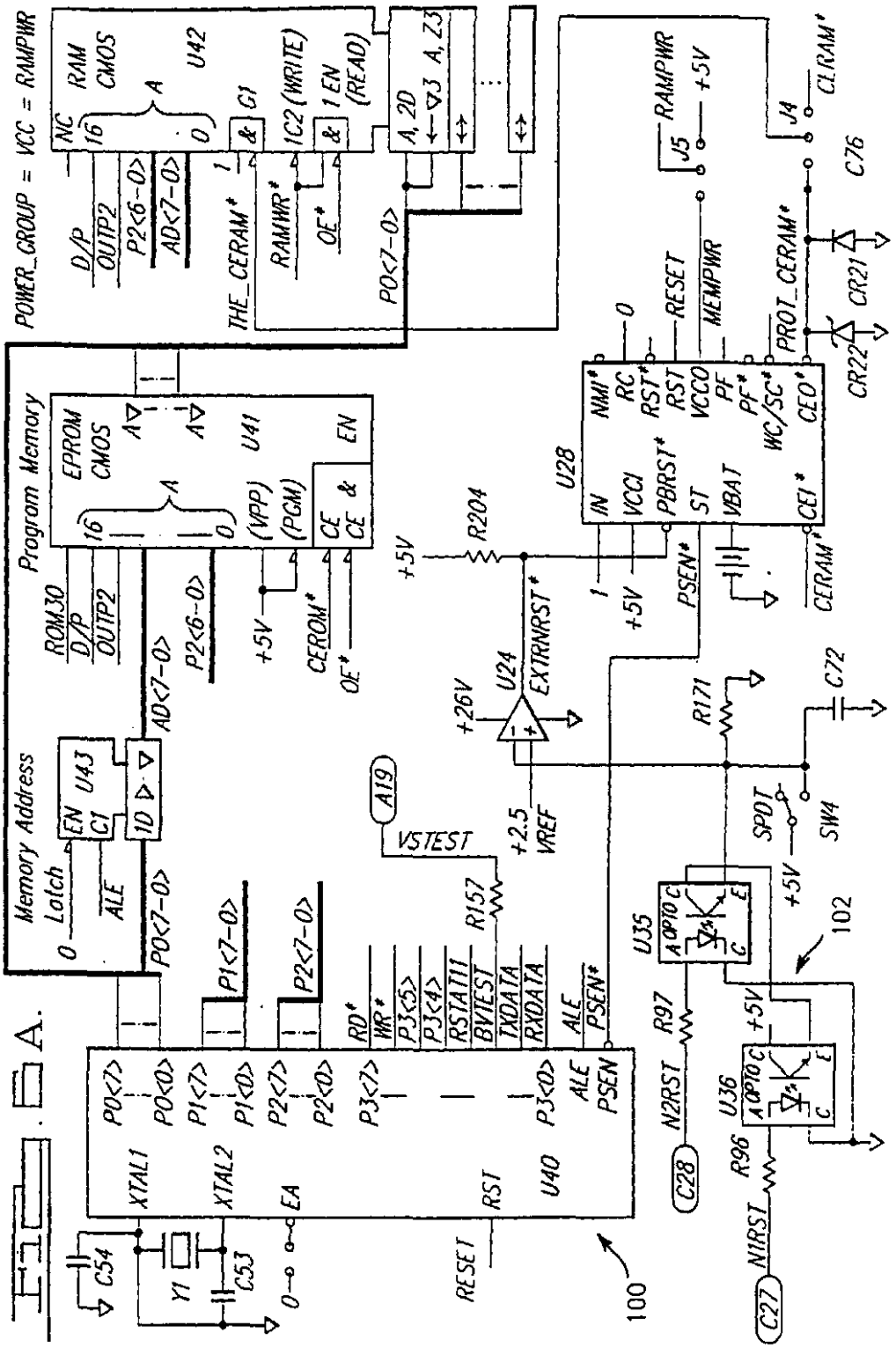

POWER_GROUP = VCC = RAMPWR
Program Memory
Memory Address
U42
U41
U43
U40
U28
U24
U35
U36
U24
EXTRNRST*
VSTEST
A19
R157
R204
R171
R97
R96
C72
C76
C54
C53
Y1
C28
C27
N2RST
N1RST
SPDT
SW4
CR21
CR22
J4
J5
RAMPWR
CLRAM*
RESET
MEMPWR
PROT_CERAM*
CERAM*
PSEN*
+5V
+26V
+2.5
VREF
100
102

FIG. 6A.

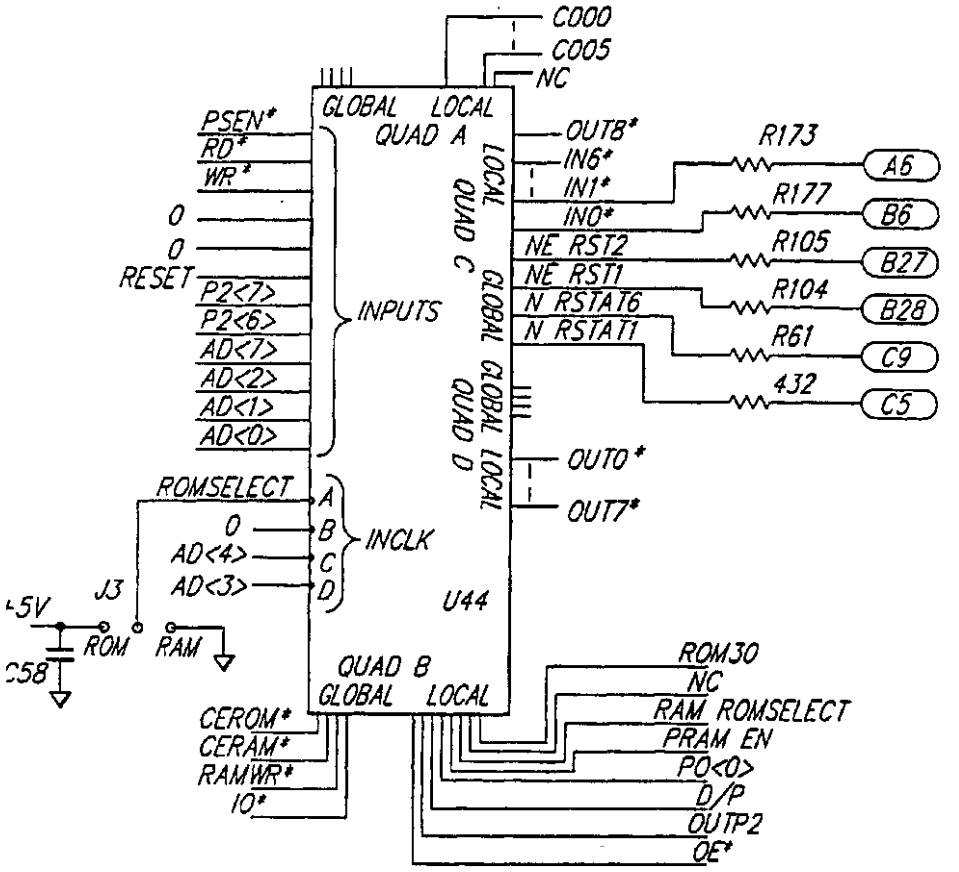

C000
C005
NC
GLOBAL LOCAL
QUAD A
PSEN*
RD*
WR*
0
0
RESET
P2<7>
P2<6>
AD<7>
AD<2>
AD<1>
AD<0>
INPUTS
LOCAL QUAD C
GLOBAL GLOBAL
QUAD D LOCAL
OUT8*
IN6*
IN1*
IN0*
NE_RST2
NE_RST1
N_RSTAT6
N_RSTAT1
OUT0*
OUT7*
R173
R177
R105
R104
R61
432
A6
B6
B27
B28
C9
C5
ROMSELECT
0
AD<4>
AD<3>
A
B
C
D
INCLK
U44
J3
+5V
ROM
RAM
C58
QUAD B
GLOBAL LOCAL
CEROM*
CERAM*
RAMWR*
IO*
ROM30
NC
RAM ROMSELECT
PRAM EN
PO<0>
D/P
OUTP2
OE*

FIG. 8B.

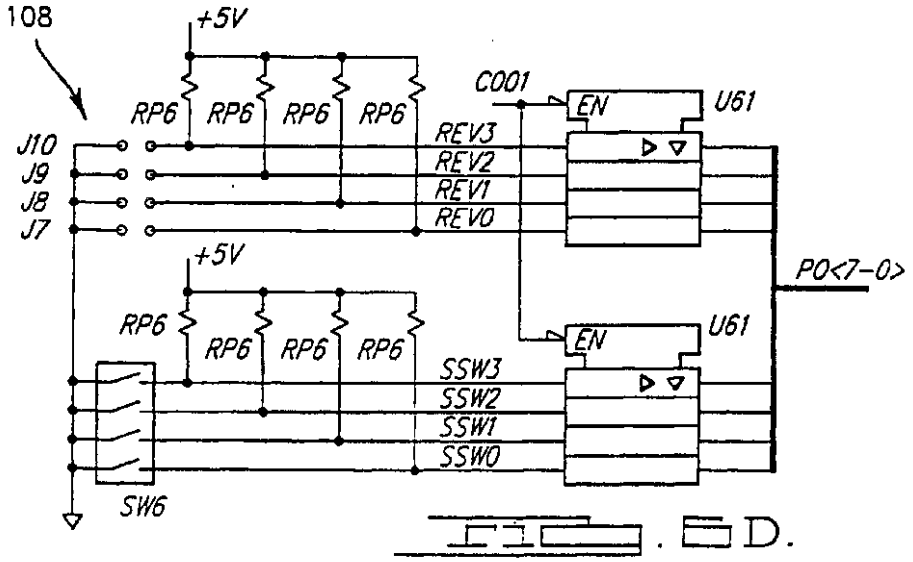

108
+5V
RP6
RP6
RP6
RP6
C001
EN
U61
J10
J9
J8
J7
REV3
REV2
REV1
REV0
+5V
RP6
RP6
RP6
RP6
RP6
EN
U61
SSW3
SSW2
SSW1
SSW0
PO<7-0>
SW6

FIG. 8D.

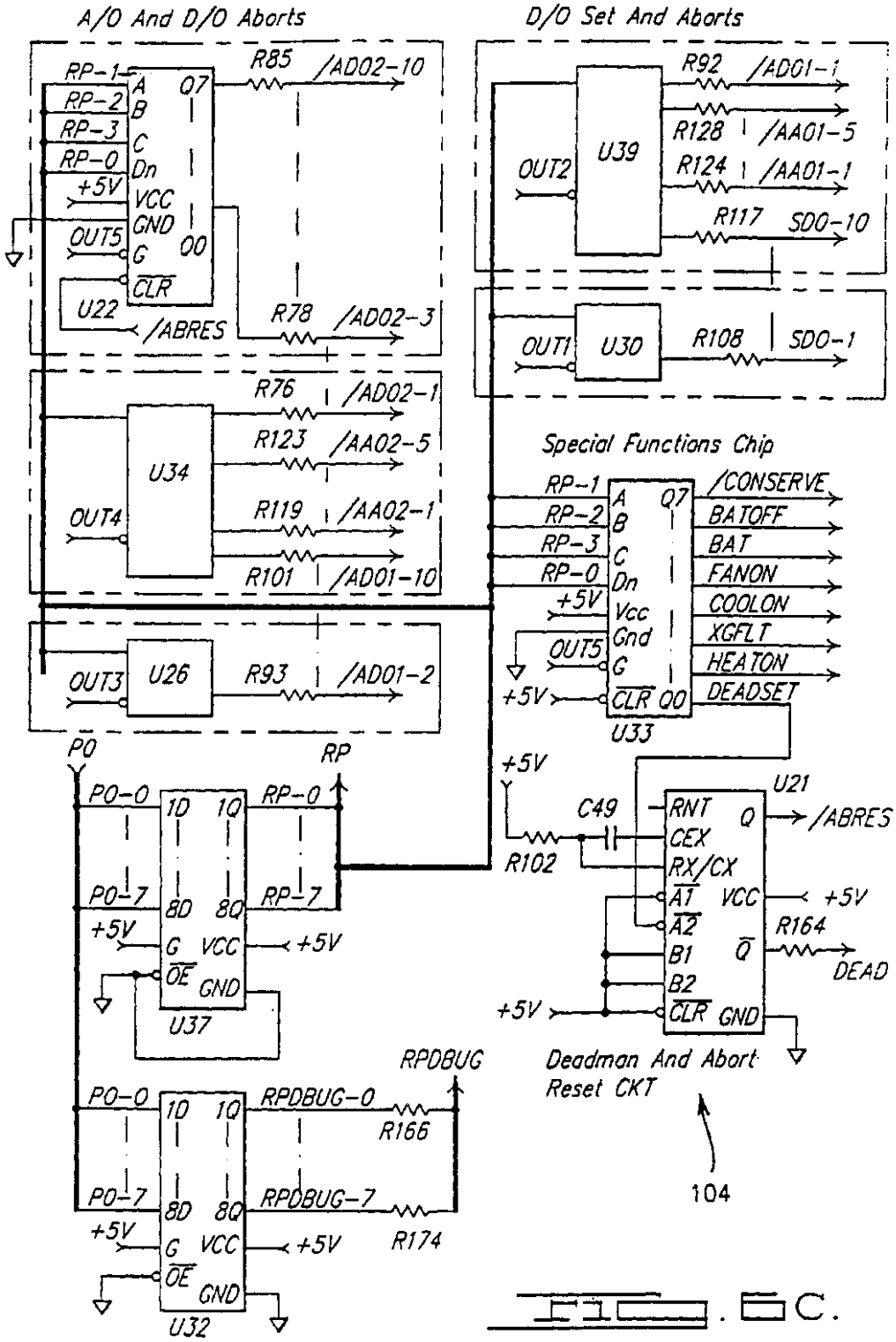

A/O And D/O Aborts
D/O Set And Aborts
RP-1
RP-2
RP-3
RP-0
+5V
OUT5
U22
/ABRES
R85
/AD02-10
R78
/AD02-3
U34
OUT4
R76
/AD02-1
R123
/AA02-5
R119
/AA02-1
R101
/AD01-10
U26
OUT3
R93
/AD01-2
U39
OUT2
R92
/AD01-1
R128
/AA01-5
R124
/AA01-1
R117
SDO-10
U30
OUT1
R108
SDO-1
Special Functions Chip
U33
/CONSERVE
BATOFF
BAT
FANON
COOLON
XGFLT
HEATON
DEADSET
P0
RP
P0-0
P0-7
RP-0
RP-7
U37
U32
RPDBUG
RPDBUG-0
RPDBUG-7
R166
R174
C49
R102
U21
/ABRES
R164
DEAD
Deadman And Abort
Reset CKT
104

FIG. 6C.

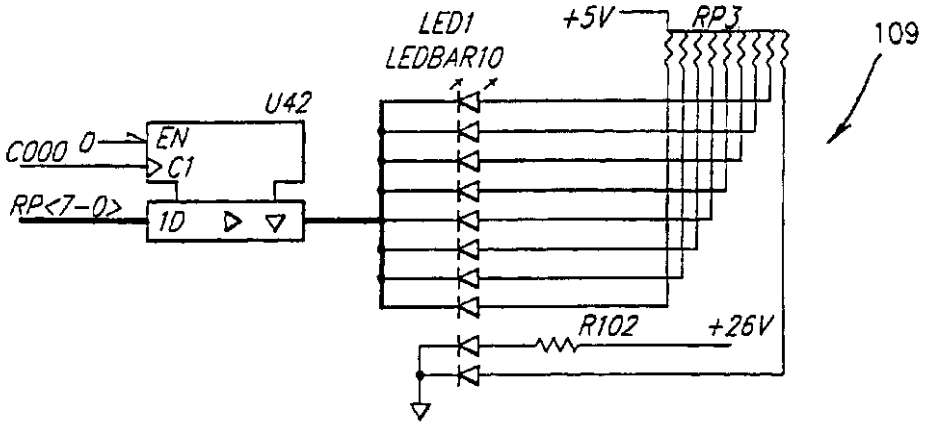

LED1
LEDBAR10
+5V
RP3
109
U42
C000 0
EN
C1
RP<7-0>
1D
R102
+26V

FIG. E.

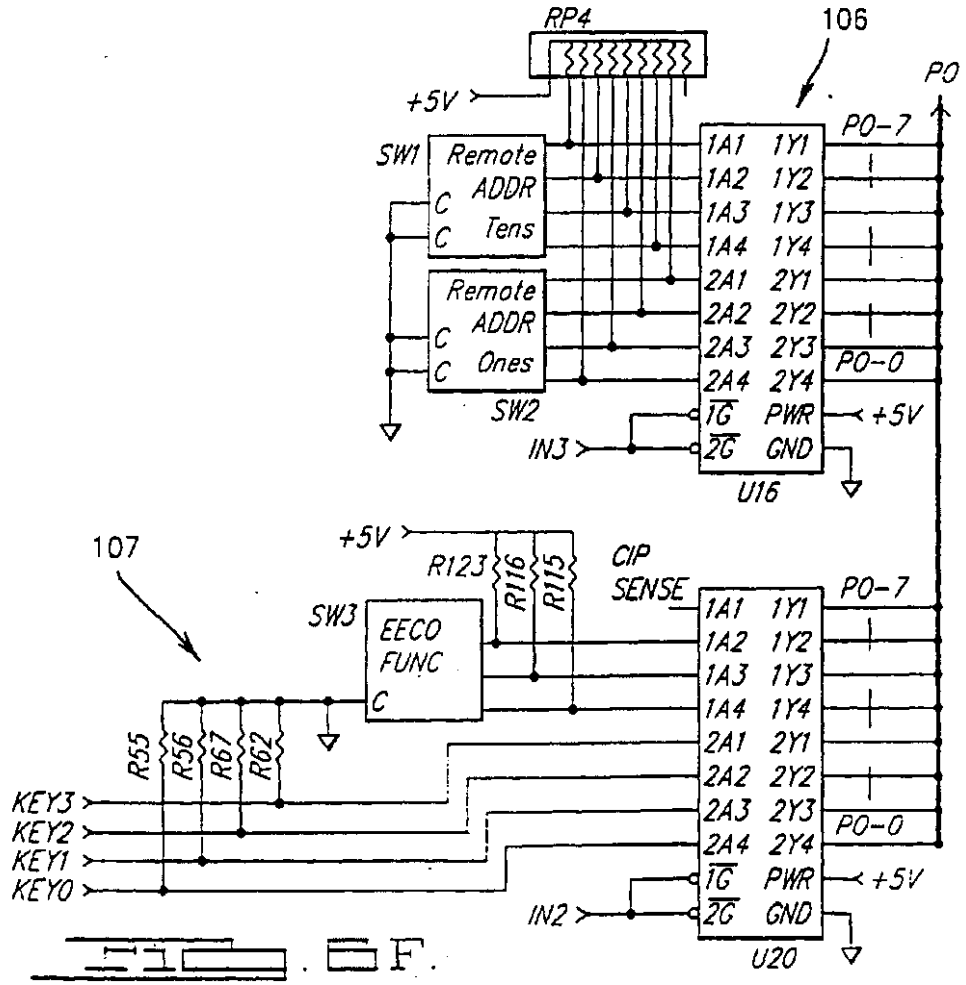

RP4
106
+5V
P0
P0-7
SW1
Remote
ADDR
Tens
Remote
ADDR
Ones
SW2
1A1 1Y1
1A2 1Y2
1A3 1Y3
1A4 1Y4
2A1 2Y1
2A2 2Y2
2A3 2Y3
2A4 2Y4
P0-0
1G PWR
2G GND
+5V
IN3
U16
107
+5V
R123
R116
R115
CIP
SENSE
SW3
EECO
FUNC
C
R55
R56
R67
R62
KEY3
KEY2
KEY1
KEY0
P0-7
P0-0
+5V
IN2
U20

FIG. F.

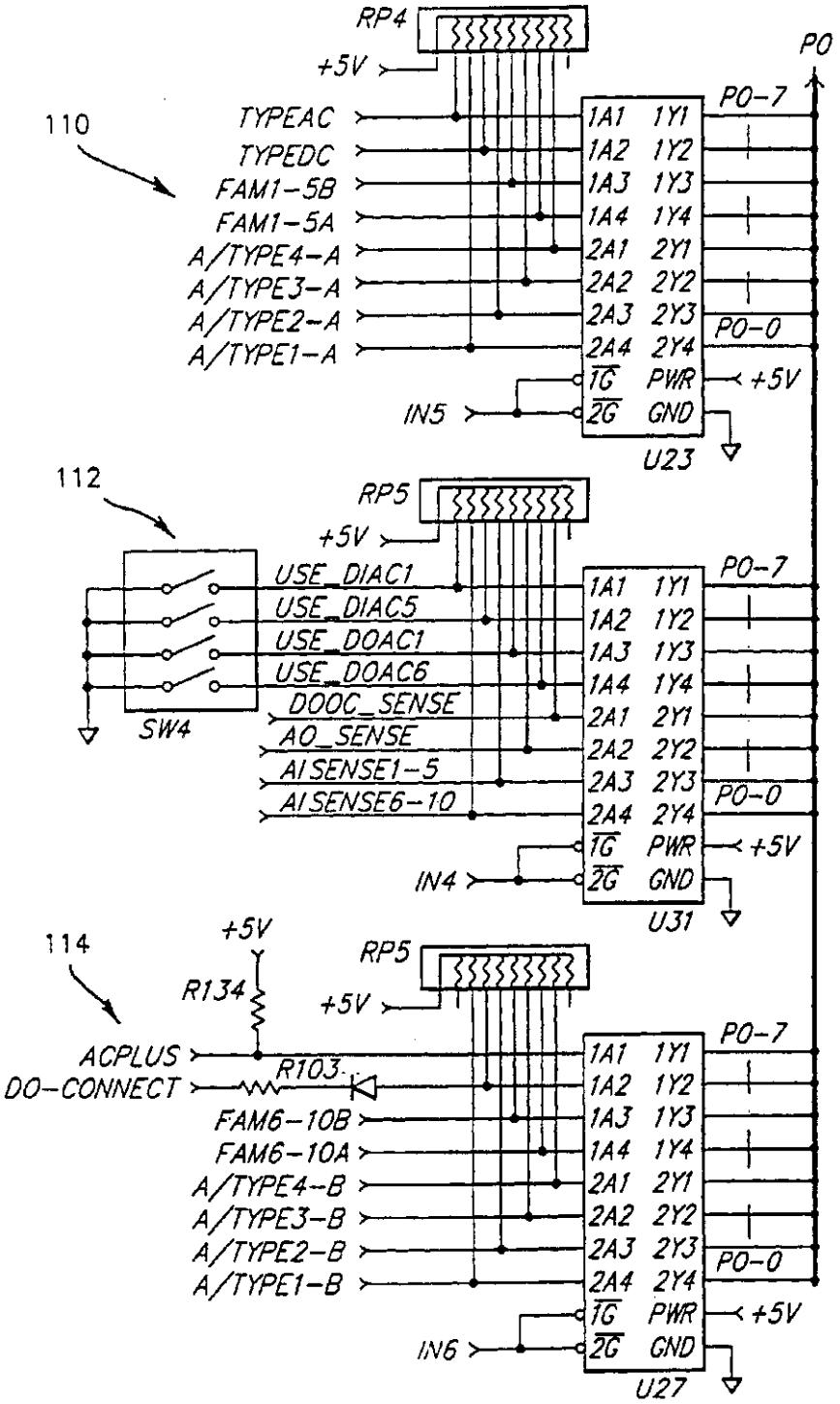

RP4
+5V
P0
P0-7
110
TYPEAC
TYPEDC
FAM1-5B
FAM1-5A
A/TYPE4-A
A/TYPE3-A
A/TYPE2-A
A/TYPE1-A
1A1 1Y1
1A2 1Y2
1A3 1Y3
1A4 1Y4
2A1 2Y1
2A2 2Y2
2A3 2Y3
2A4 2Y4
P0-0
1G PWR +5V
2G GND
IN5
U23
112
RP5
+5V
USE_DIAC1
USE_DIAC5
USE_DOAC1
USE_DOAC6
DOOC_SENSE
AO_SENSE
AISENSE1-5
AISENSE6-10
SW4
P0-7
P0-0
IN4
U31
114
+5V
R134
RP5
+5V
ACPLUS
DO-CONNECT
R103
FAM6-10B
FAM6-10A
A/TYPE4-B
A/TYPE3-B
A/TYPE2-B
A/TYPE1-B
P0-7
P0-0
IN6
U27

FIG. 6G.

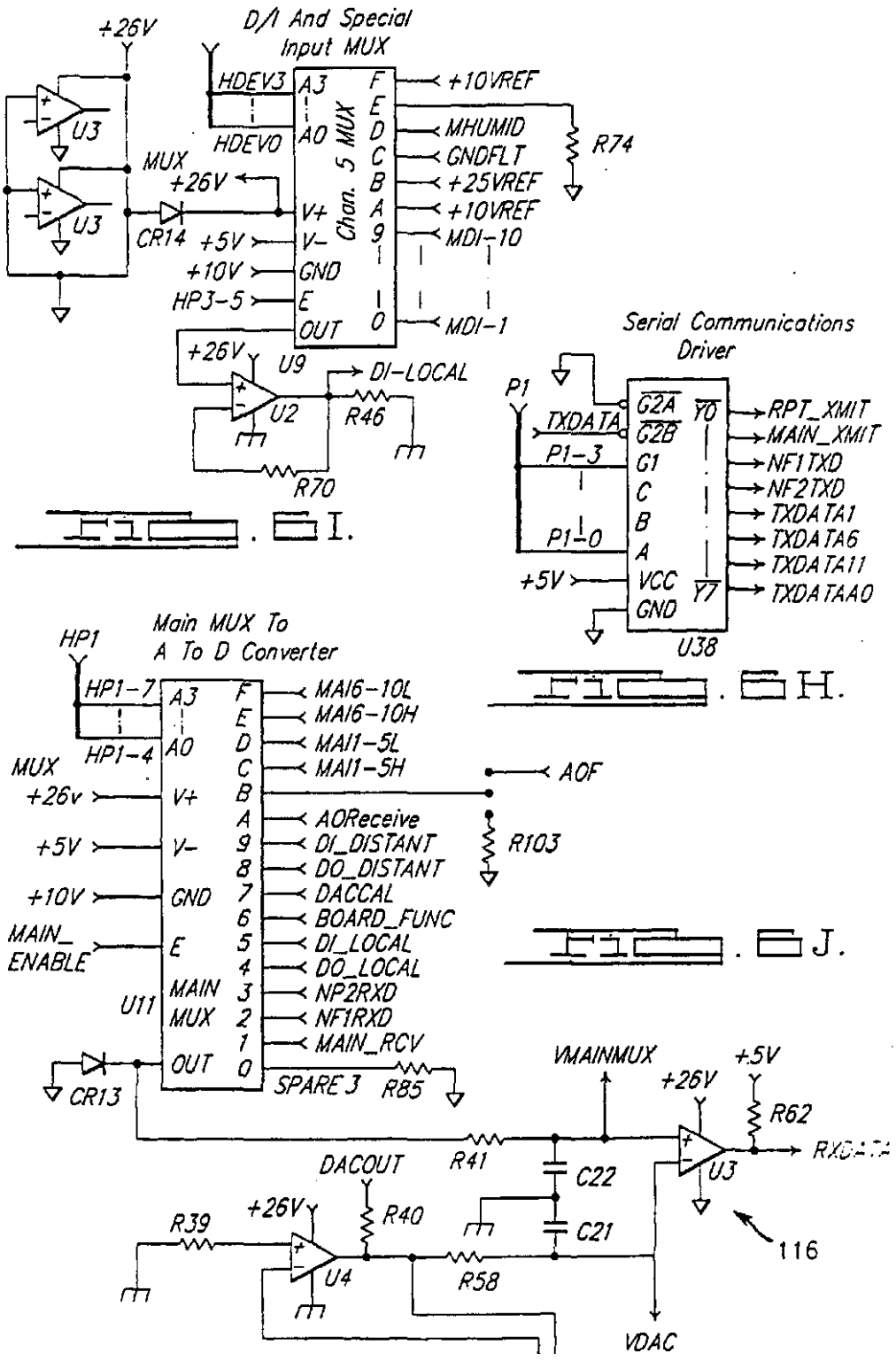


FIG. I.

FIG. H.

FIG. J.

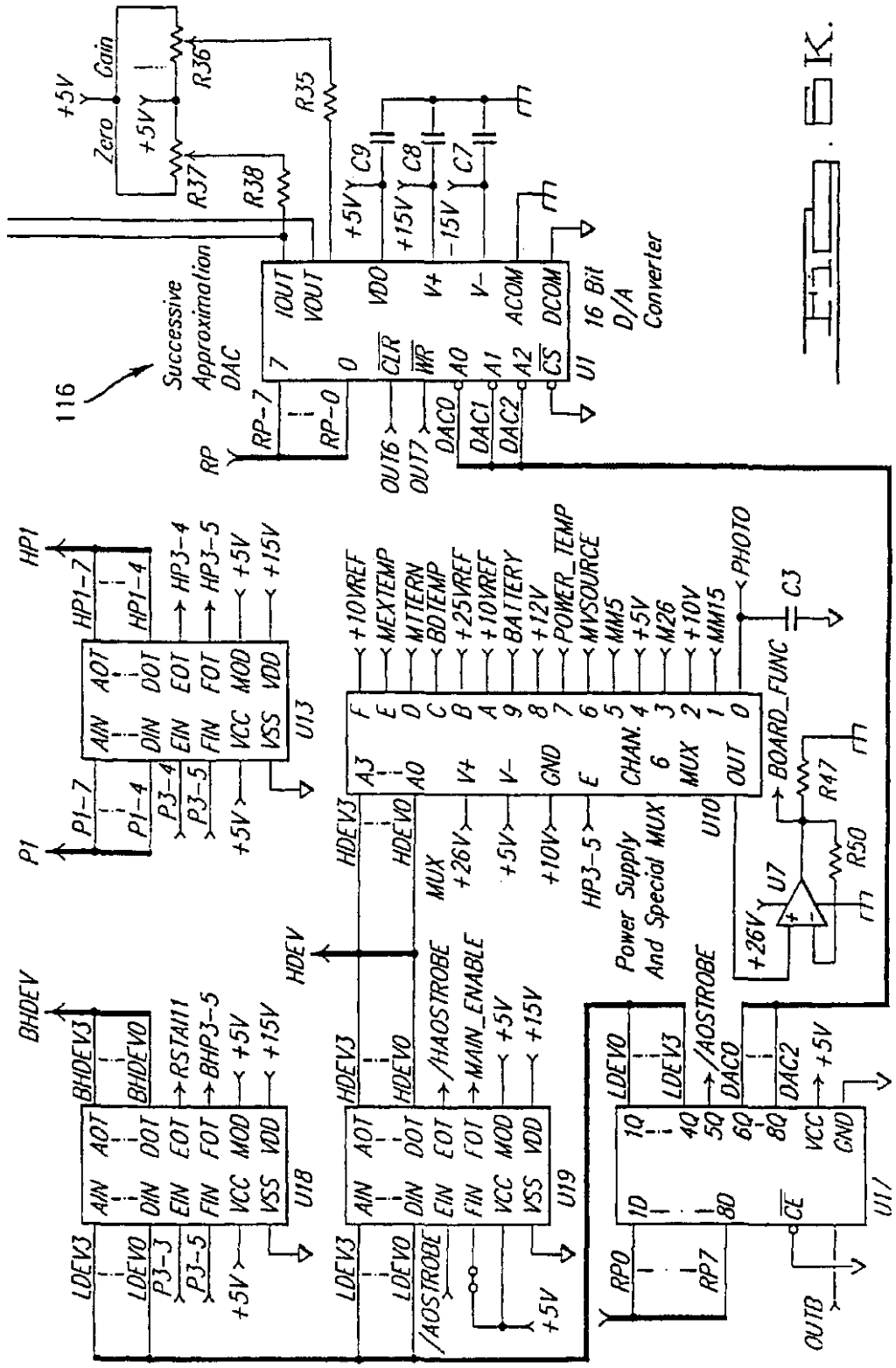

116
Successive Approximation DAC
16 Bit D/A Converter
Power Supply And Special MUX
Zero
Gain
+5V
R36
R37
R38
R35
C9
C8
C7
+15V
-15V
IOUT
VOUT
VDO
V+
V-
ACOM
DCOM
CLR
WR
A0
A1
A2
CS
U1
RP
RP-7
RP-0
OUT6
OUT7
DAC0
DAC1
DAC2
HP1
HP1-7
HP1-4
HP3-4
HP3-5
P1
P1-7
P1-4
P3-4
P3-5
AIN
AOT
DIN
DOT
EIN
EOT
FIN
FOT
VCC
MOD
VSS
VDD
U13
+10VREF
MEXTEMP
MTTERN
BDTEMP
+25VREF
+10VREF
BATTERY
+12V
POWER_TEMP
MVSOURCE
MM5
M26
MM15
PHOTO
C3
BOARD_FUNC
R47
R50
U7
+26V
HDEV3
HDEV0
MUX
HDEV
BHDEV
BHDEV3
BHDEV0
RSTAI11
BHP3-5
U18
LDEV3
LDEV0
P3-3
/AOSTROBE
/HAOSTROBE
MAIN_ENABLE
U19
LDEV0
LDEV3
/AOSTROBE
DAC0
DAC2
U17
RP0
RP7
OUTB

FIG. 6K.

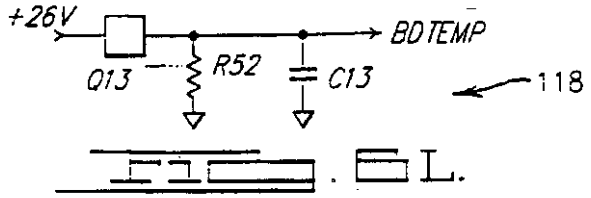


FIG. 6L.

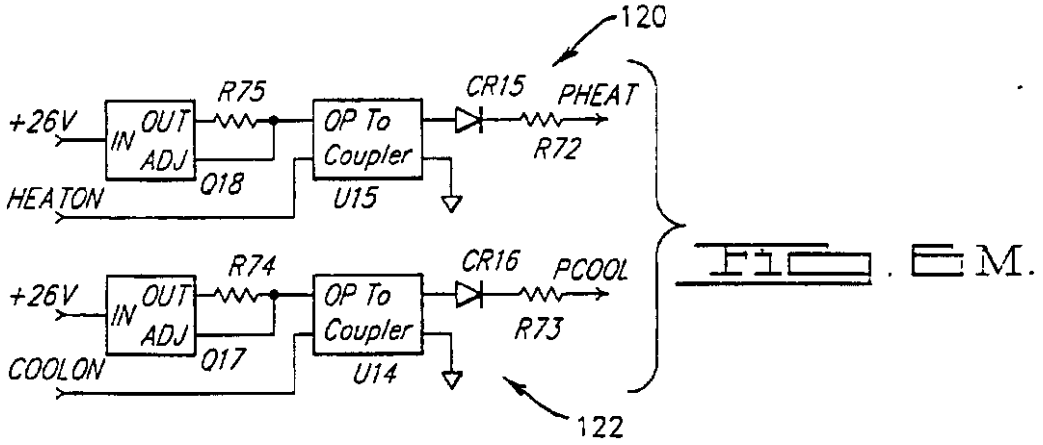


FIG. 6M.

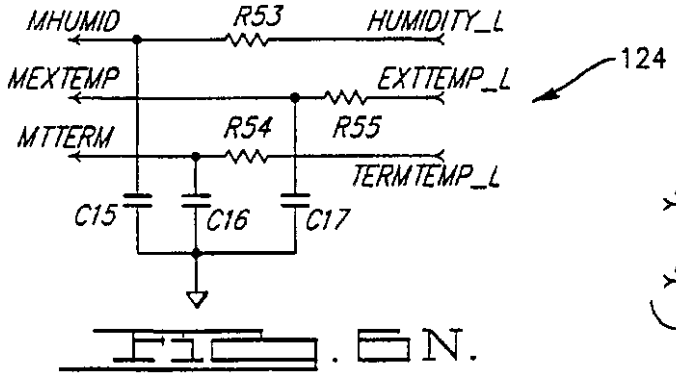


FIG. 6N.

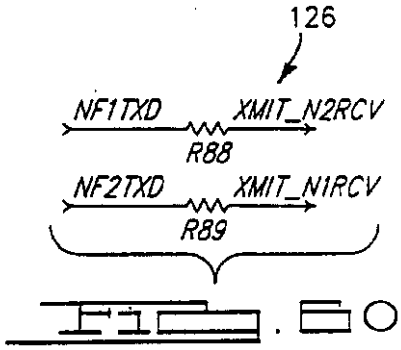


FIG. 6O.

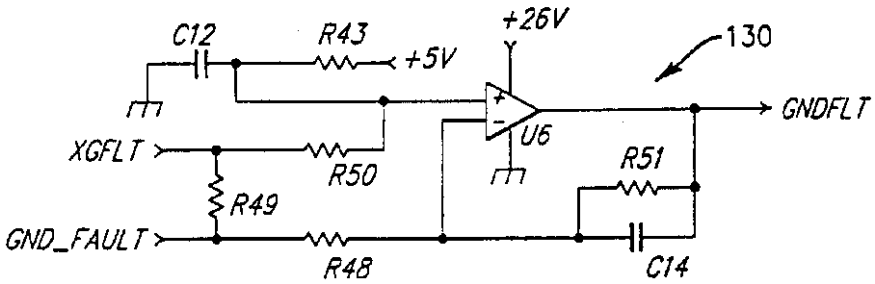


FIG. 6P.

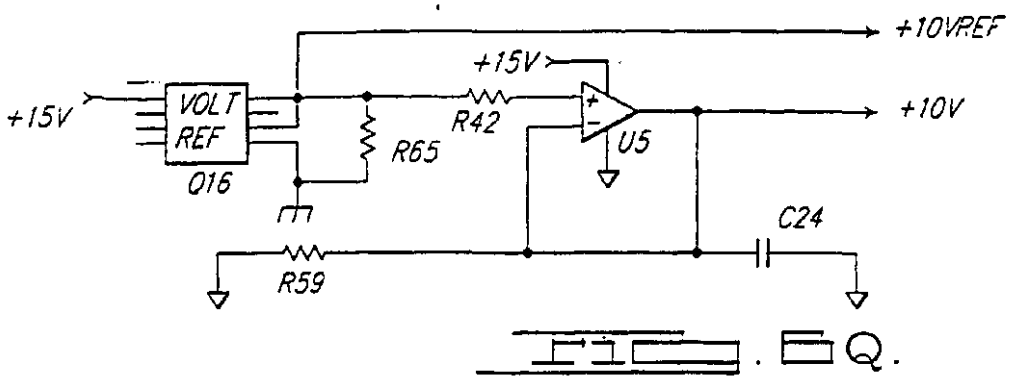


FIG. 5Q.

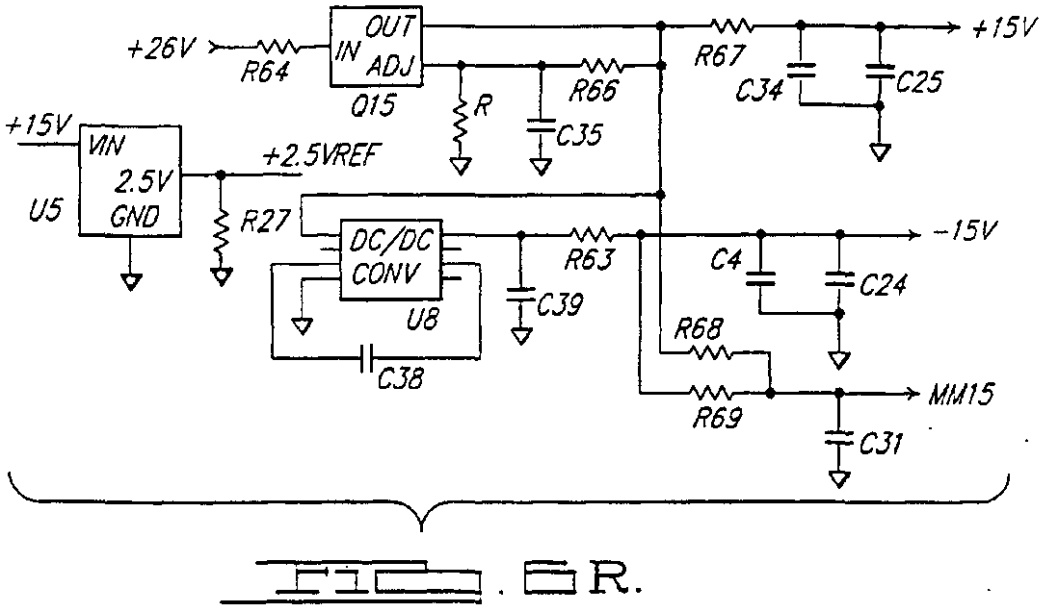


FIG. 5R.

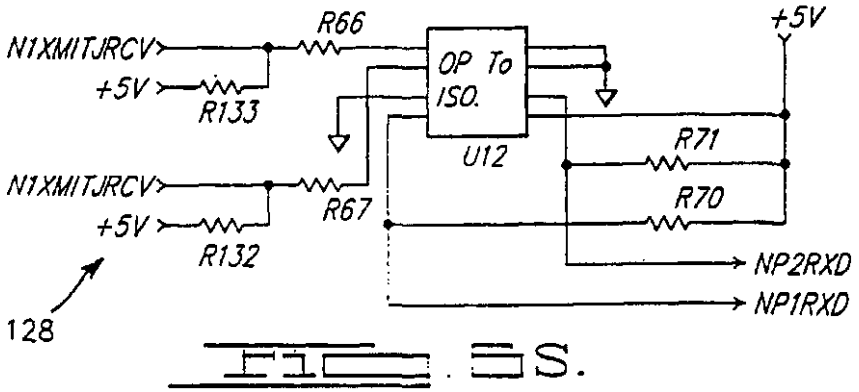


FIG. 5S.

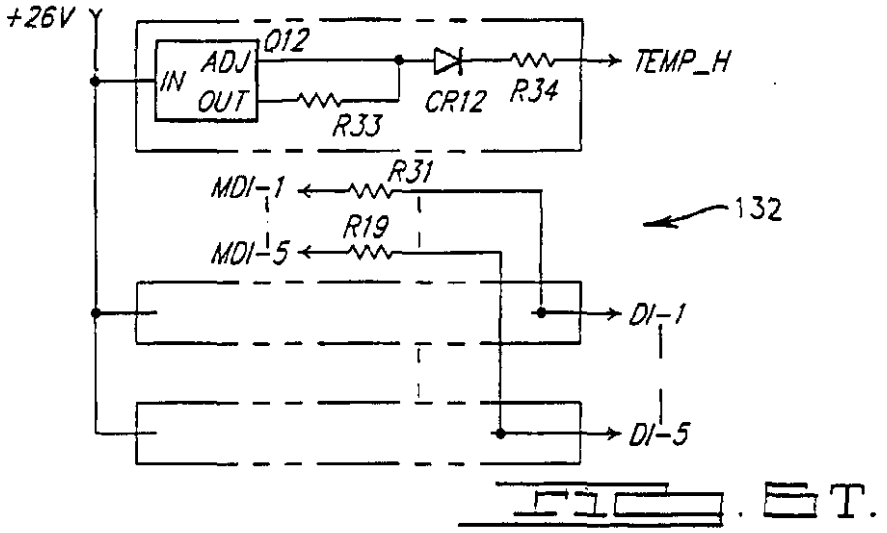


FIG. 5T.

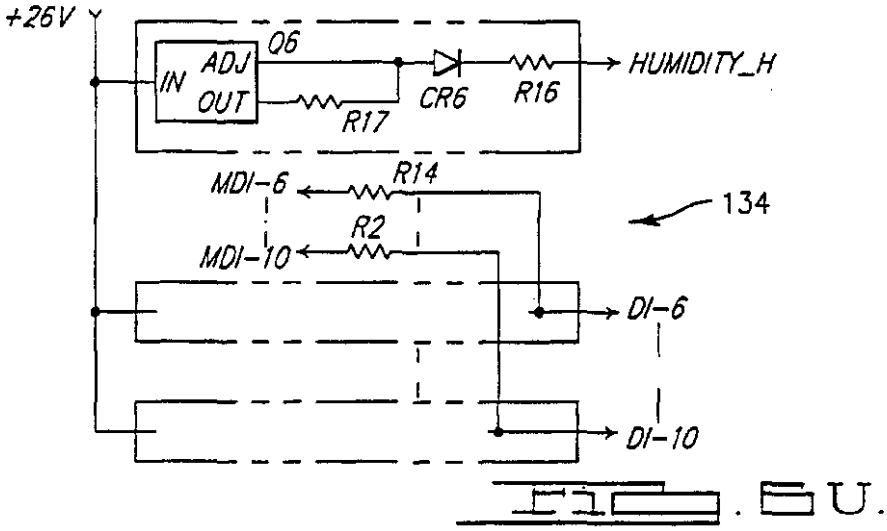


FIG. 5U.

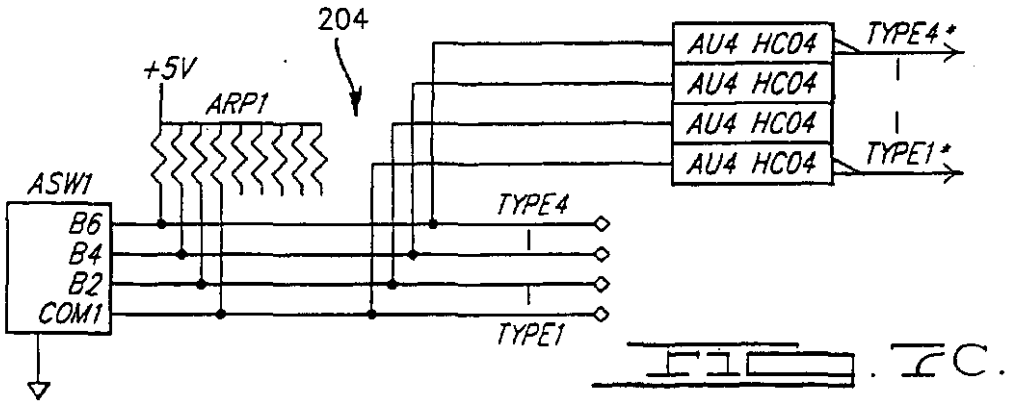


FIG. 5C.

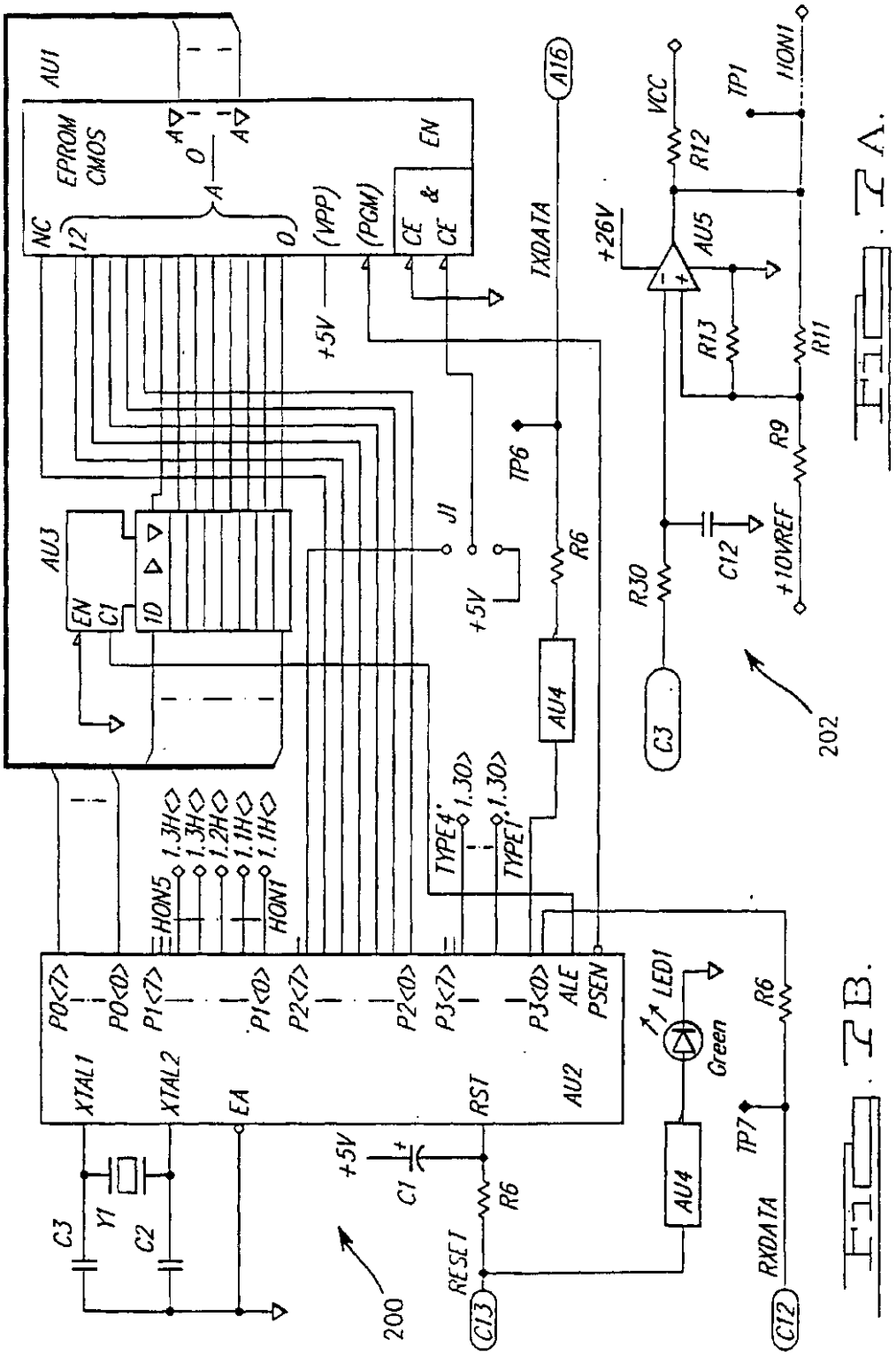

AU1
EPROM
CMOS
NC
12
0
A
O
A
(VPP)
(PGM)
CE
CE
&
EN
+5V
AU3
EN
C1
1D
J1
+5V
TXDATA
A16
TP6
R6
AU4
TYPE4'
1.3O>
TYPE1'
1.3O>
HON5
1.3H<>
1.3H<>
1.2H<>
1.1H<>
1.1H<>
HON1
P0<7>
P0<0>
P1<7>
P1<0>
P2<7>
P2<0>
P3<7>
P3<0>
ALE
PSEN
XTAL1
XTAL2
EA
RST
AU2
C3
Y1
C2
+5V
C1
R6
RESET
C13
200
LED1
Green
R6
TP7
AU4
RXDATA
C12
VCC
R12
TP1
HON1
+26V
AU5
R13
R11
R9
R30
C12
+10VREF
C3
202

FIG. 7A.

FIG. 7B.

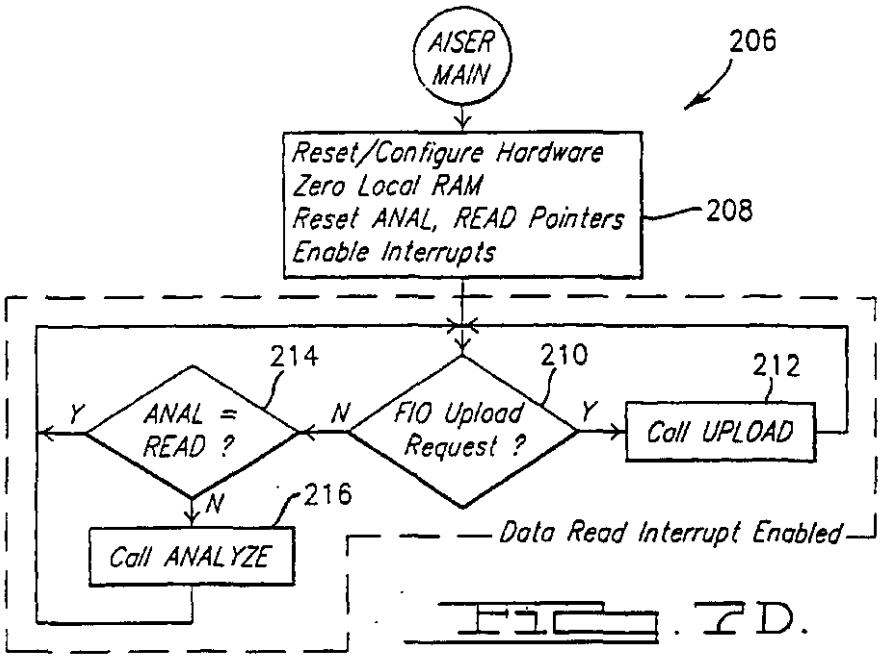


FIG. 7D.

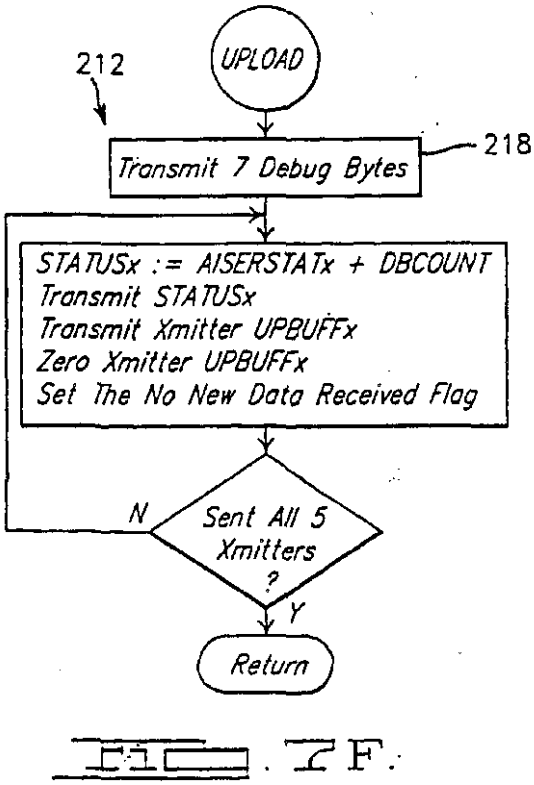


FIG. 7F.

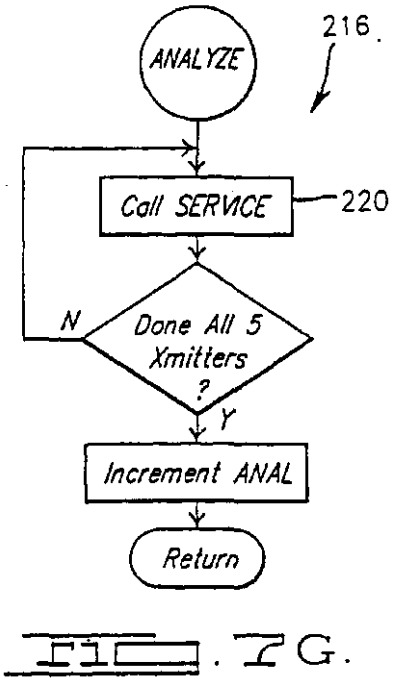


FIG. 7G.

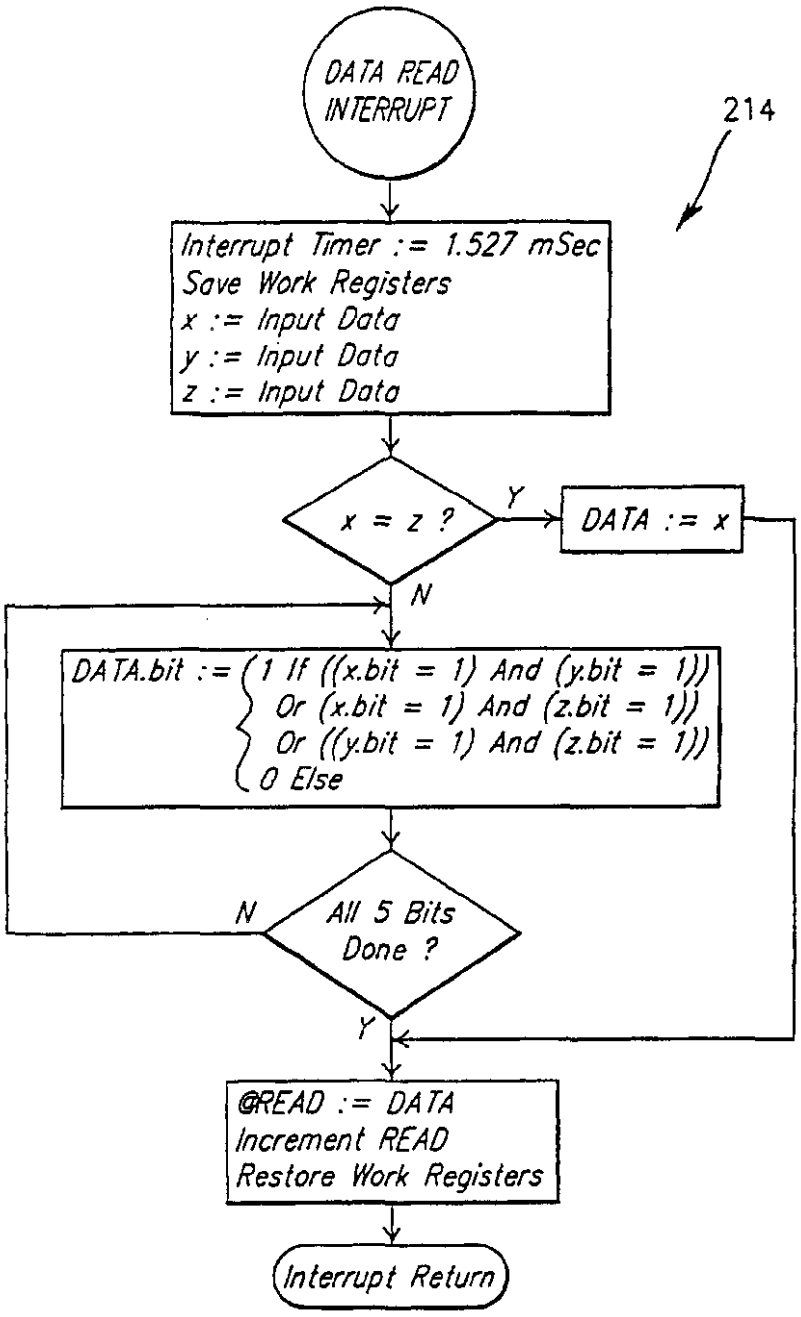

DATA READ INTERRUPT
214
Interrupt Timer := 1.527 mSec
Save Work Registers
x := Input Data
y := Input Data
z := Input Data
x = z ?
Y
DATA := x
N
DATA.bit := 1 If ((x.bit = 1) And (y.bit = 1)) Or (x.bit = 1) And (z.bit = 1)) Or ((y.bit = 1) And (z.bit = 1)) 0 Else
N
All 5 Bits Done ?
Y
@READ := DATA
Increment READ
Restore Work Registers
Interrupt Return

FIG. 7E.

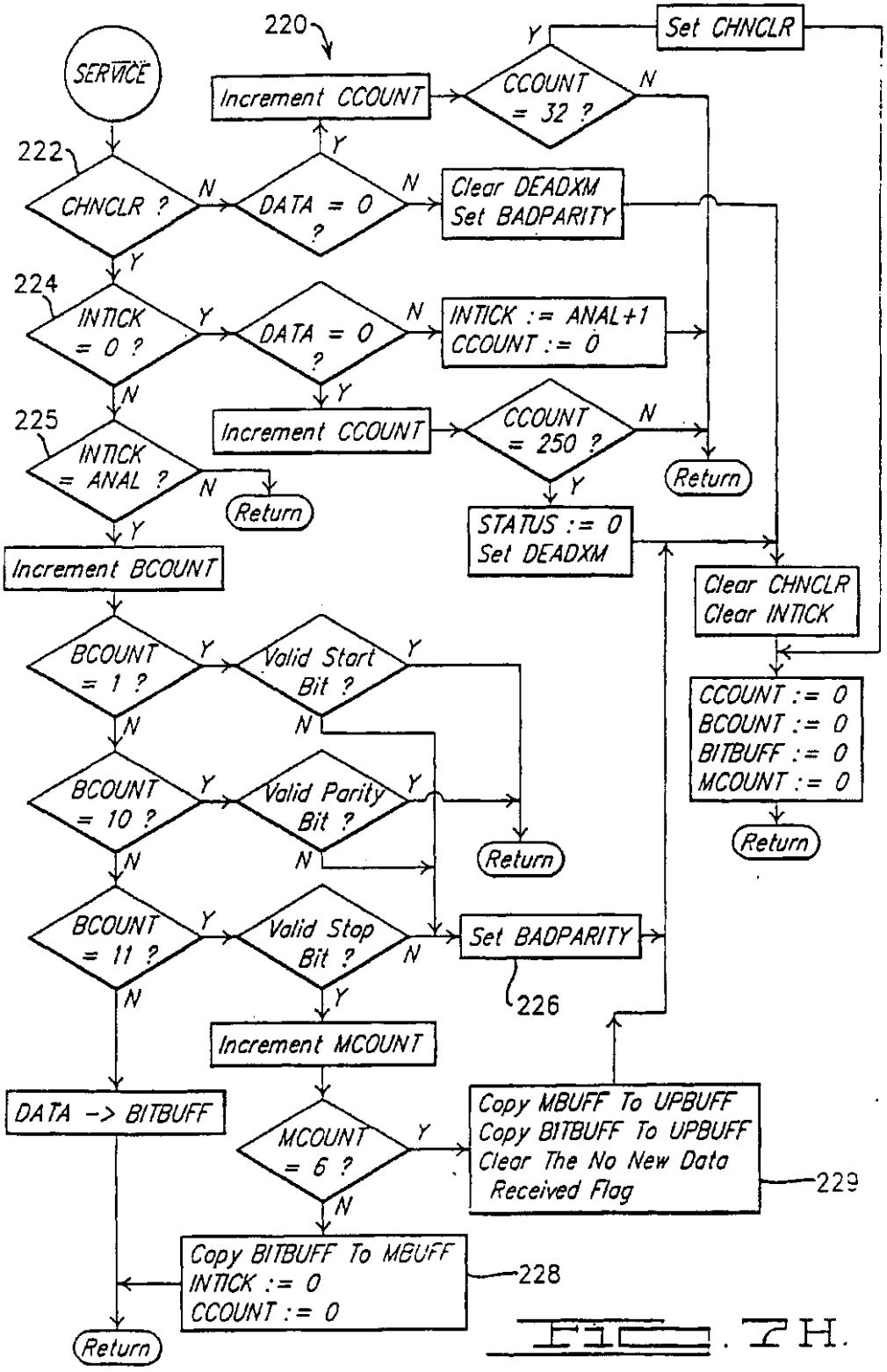

SERVICE
220
Increment CCOUNT
CCOUNT = 32 ?
Y
N
Set CHNCLR
222
CHNCLR ?
N
DATA = 0 ?
N
Clear DEADXM
Set BADPARITY
Y
Y
224
INTICK = 0 ?
Y
DATA = 0 ?
N
INTICK := ANAL+1
CCOUNT := 0
N
Y
225
INTICK = ANAL ?
N
Return
Y
Increment CCOUNT
CCOUNT = 250 ?
N
Return
Y
STATUS := 0
Set DEADXM
Increment BCOUNT
Clear CHNCLR
Clear INTICK
BCOUNT = 1 ?
Y
Valid Start Bit ?
Y
N
N
CCOUNT := 0
BCOUNT := 0
BITBUFF := 0
MCOUNT := 0
BCOUNT = 10 ?
Y
Valid Parity Bit ?
Y
Return
N
N
Return
BCOUNT = 11 ?
Y
Valid Stop Bit ?
N
Set BADPARITY
226
N
Y
Increment MCOUNT
DATA -> BITBUFF
MCOUNT = 6 ?
Y
Copy MBUFF To UPBUFF
Copy BITBUFF To UPBUFF
Clear The No New Data Received Flag
229
N
Copy BITBUFF To MBUFF
INTICK := 0
CCOUNT := 0
228
Return

FIG. 7H.

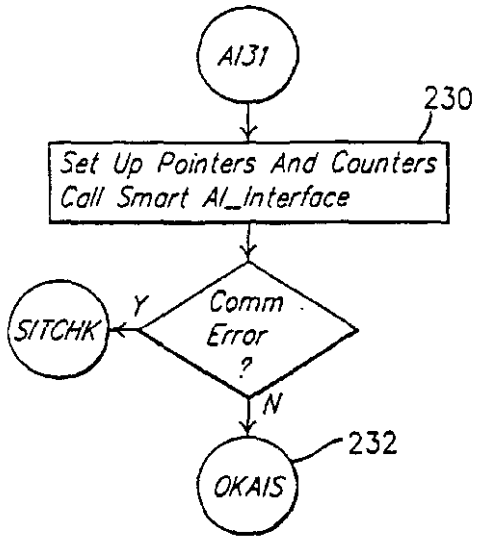

AI31
230
Set Up Pointers And Counters
Call Smart AI_Interface
SITCHK
Y
Comm Error ?
N
232
OKAIS

FIG. 7I.

OKAIS
Set Up Channel Counter
Point To Channel 1
A
Is This Transmitter Missing ?
Clear Database Count
Mask Off Database Byte Count
Mask Off Applicable Error Bits
Was No New Data Received ?
234
Y
N
Buffer Transmitter Status
Is The Transmitter In Option A Mode ?
Y
Flag "Bad PV Data"
B
N
Is The Transmitter In Output Mode ?
Y
Flag "Bad PV Data"
N
C

FIG. 7J.

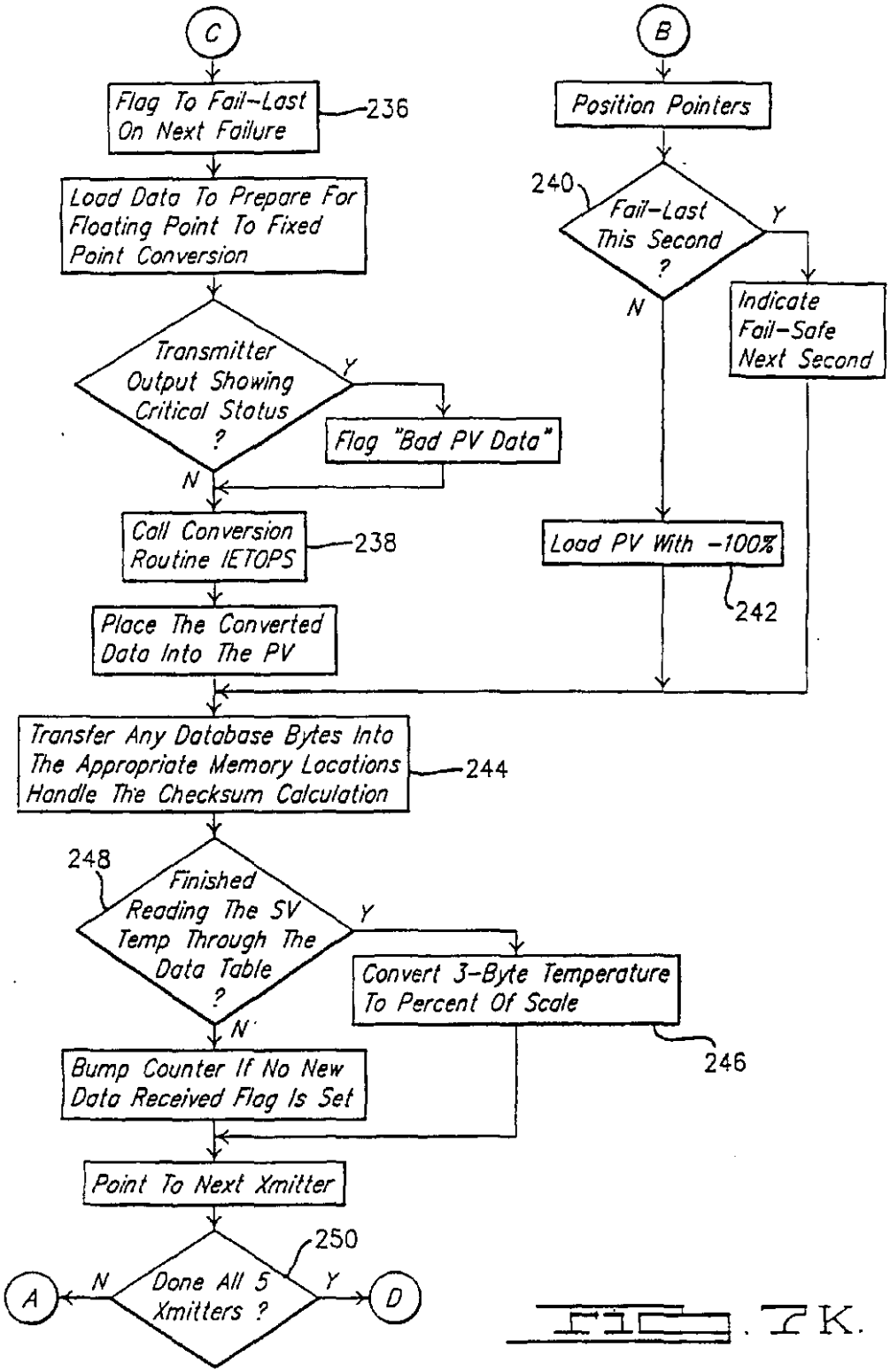

C
Flag To Fail-Last On Next Failure
236
Load Data To Prepare For Floating Point To Fixed Point Conversion
Transmitter Output Showing Critical Status ?
Y
Flag "Bad PV Data"
N
Call Conversion Routine IETOPS
238
Place The Converted Data Into The PV
B
Position Pointers
240
Fail-Last This Second ?
Y
N
Indicate Fail-Safe Next Second
Load PV With -100%
242
Transfer Any Database Bytes Into The Appropriate Memory Locations Handle The Checksum Calculation
244
248
Finished Reading The SV Temp Through The Data Table ?
Y
N
Convert 3-Byte Temperature To Percent Of Scale
246
Bump Counter If No New Data Received Flag Is Set
Point To Next Xmitter
250
Done All 5 Xmitters ?
N
Y
A
D

FIG. 7K.

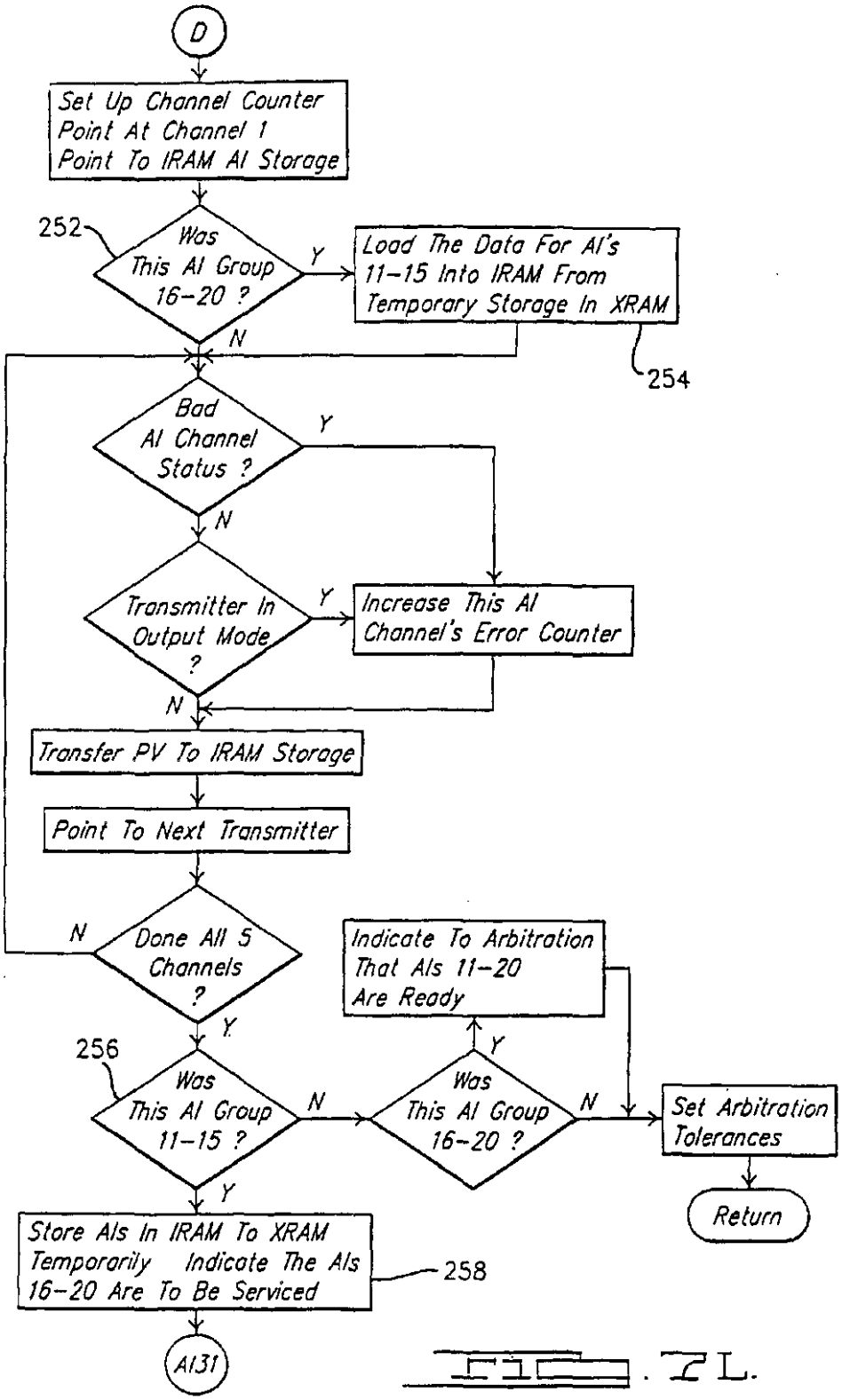

D
Set Up Channel Counter
Point At Channel 1
Point To IRAM AI Storage
252
Was This AI Group 16-20 ?
Y
N
Load The Data For AI's 11-15 Into IRAM From Temporary Storage In XRAM
254
Bad AI Channel Status ?
Y
N
Transmitter In Output Mode ?
Y
N
Increase This AI Channel's Error Counter
Transfer PV To IRAM Storage
Point To Next Transmitter
Done All 5 Channels ?
N
Y
256
Was This AI Group 11-15 ?
N
Y
Was This AI Group 16-20 ?
Y
N
Indicate To Arbitration That AIs 11-20 Are Ready
Set Arbitration Tolerances
Return
Store AIs In IRAM To XRAM Temporarily Indicate The AIs 16-20 Are To Be Serviced
258
AI31

FIG. 2L.

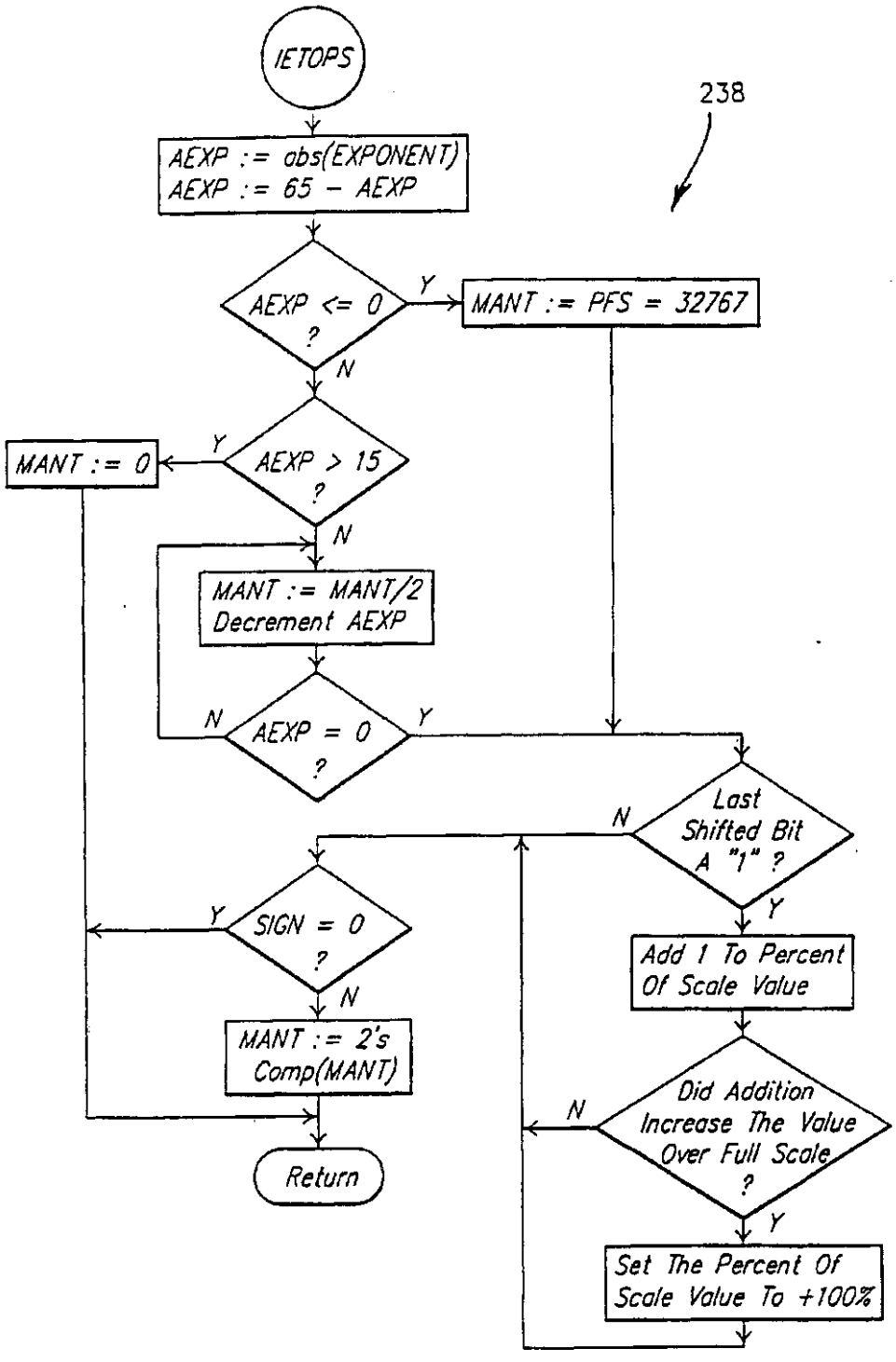

IETOPS
238
AEXP := abs(EXPONENT)
AEXP := 65 - AEXP
AEXP <= 0 ?
Y
MANT := PFS = 32767
N
AEXP > 15 ?
Y
MANT := 0
N
MANT := MANT/2
Decrement AEXP
AEXP = 0 ?
N
Y
Last Shifted Bit A "1" ?
N
Y
SIGN = 0 ?
Y
N
Add 1 To Percent Of Scale Value
MANT := 2's Comp(MANT)
Did Addition Increase The Value Over Full Scale ?
N
Y
Set The Percent Of Scale Value To +100%
Return

FIG. 7M.

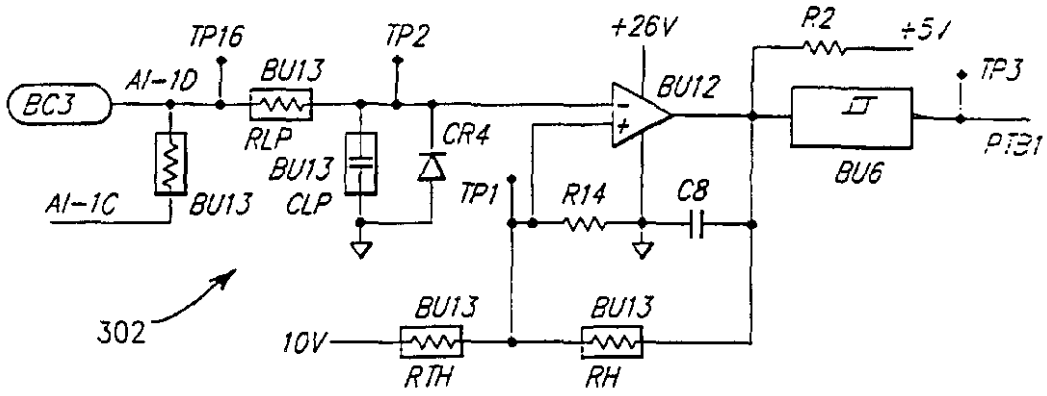

TP16
TP2
+26V
R2
+5V
AI-1D
BU13
BU12
TP3
BC3
RLP
CR4
BU6
PT31
BU13
AI-1C
BU13
CLP
TP1
R14
C8
302
BU13
BU13
10V
RTH
RH

FIG. 3A.

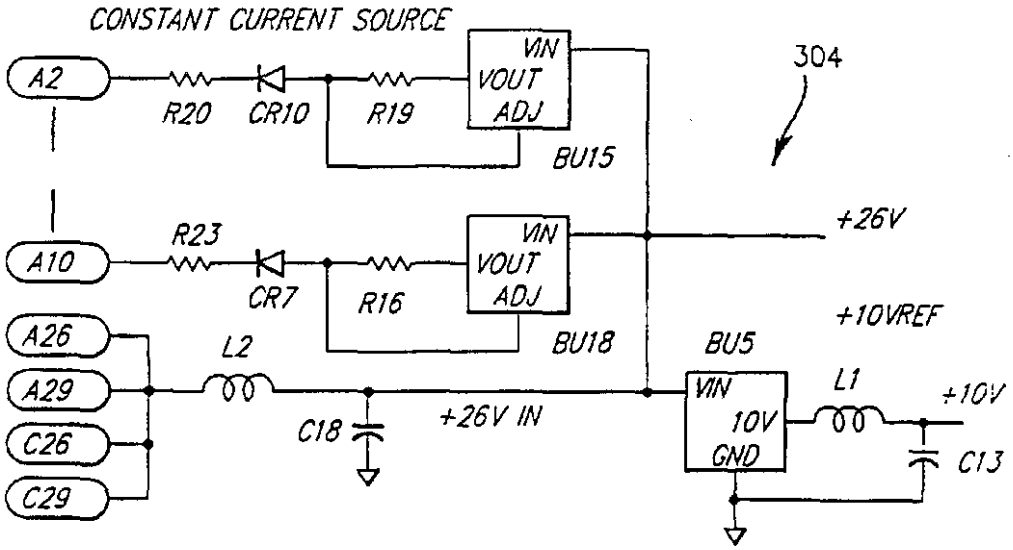

CONSTANT CURRENT SOURCE
304
A2
VIN
VOUT
ADJ
R20
CR10
R19
BU15
A10
R23
+26V
CR7
R16
BU18
+10VREF
A26
L2
BU5
A29
VIN
L1
+10V
+26V IN
C18
10V
C26
GND
C13
C29

FIG. 3C.

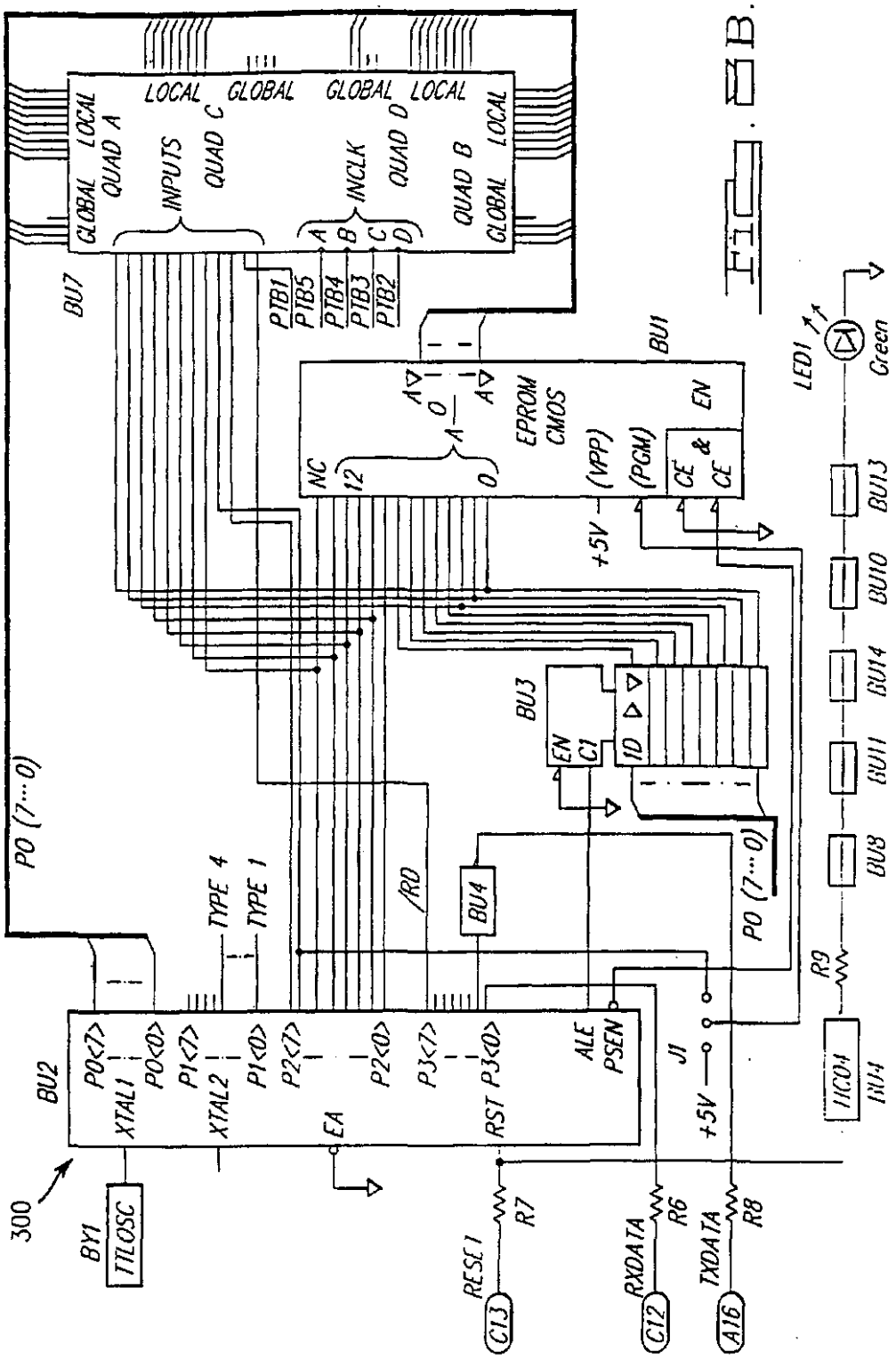

300
BY1
TTLOSC
BU2
XTAL1
XTAL2
EA
RST
ALE
PSEN
P0<7>
P0<0>
P1<7>
P1<0>
P2<7>
P2<0>
P3<7>
P3<0>
TYPE 4
TYPE 1
/RD
BU4
PO (7…0)
BU3
EN
C1
1D
BU1
EPROM
CMOS
NC
12
0
(VPP)
(PGM)
CE
&
EN
+5V
BU7
GLOBAL
LOCAL
QUAD A
INPUTS
QUAD C
INCLK
QUAD D
QUAD B
PTB1
PTB5
PTB4
PTB3
PTB2
J1
R7
R6
R8
RESET
RXDATA
TXDATA
C13
C12
A16
R9
BU8
BU11
BU14
BU10
BU13
LED1
Green

FIG. 8B.

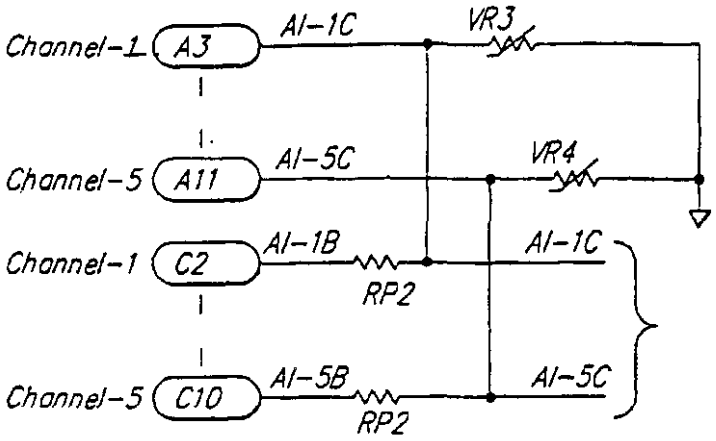

Channel-1
A3
AI-1C
VR3
Channel-5
A11
AI-5C
VR4
Channel-1
C2
AI-1B
RP2
AI-1C
Channel-5
C10
AI-5B
RP2
AI-5C

FIG. 5D.

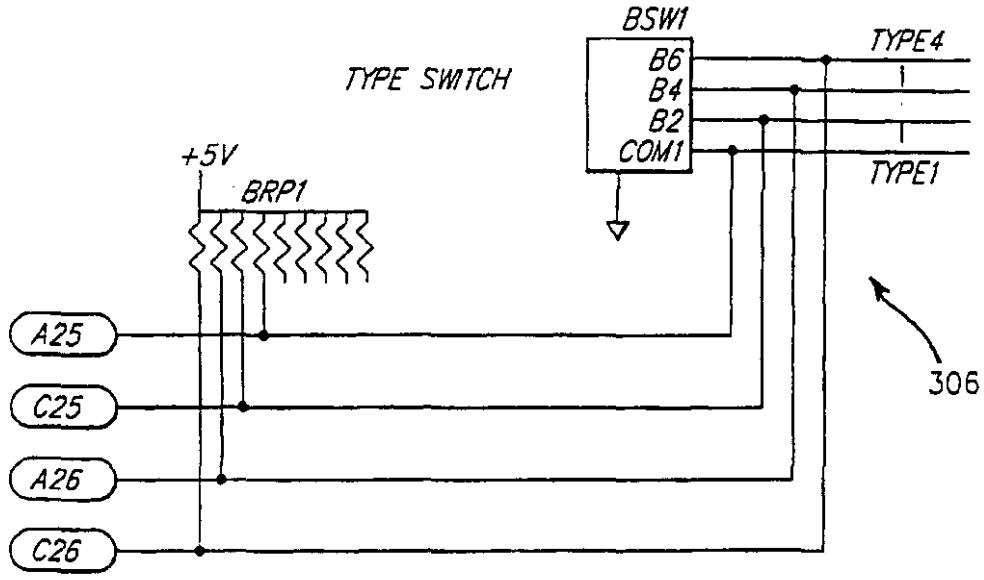

TYPE SWITCH
BSW1
B6
B4
B2
COM1
TYPE4
TYPE1
+5V
BRP1
A25
C25
A26
C26
306

FIG. 5E.

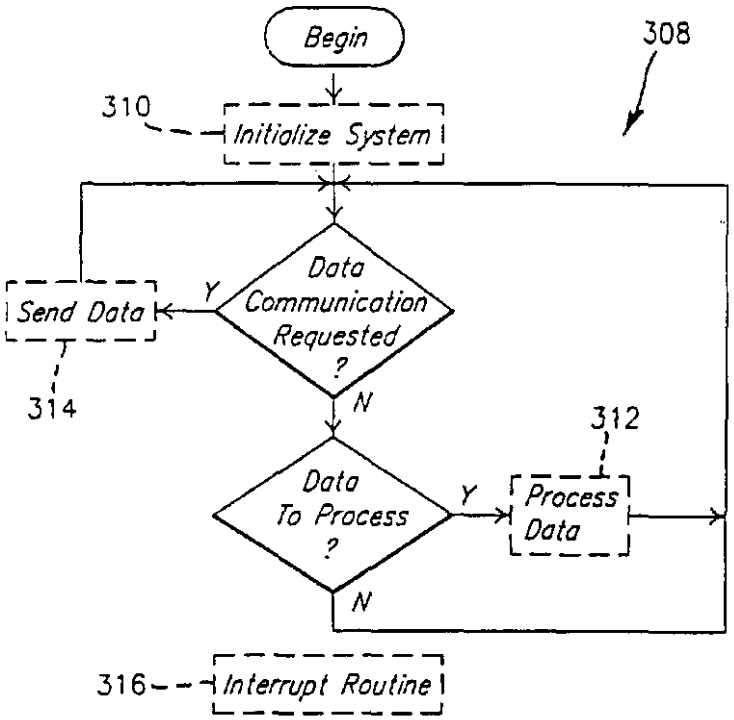

Begin
308
310
Initialize System
Data Communication Requested ?
Y
Send Data
314
N
312
Data To Process ?
Y
Process Data
N
316
Interrupt Routine

FIG. F.

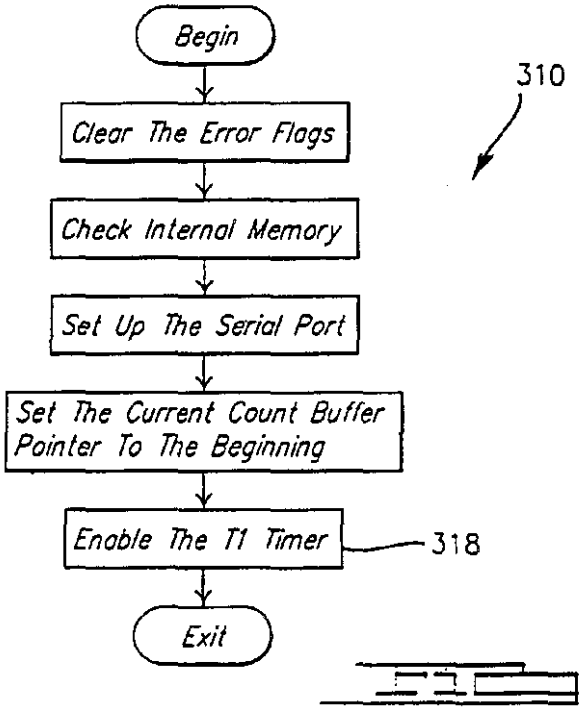

Begin
310
Clear The Error Flags
Check Internal Memory
Set Up The Serial Port
Set The Current Count Buffer Pointer To The Beginning
Enable The T1 Timer
318
Exit

FIG. G.

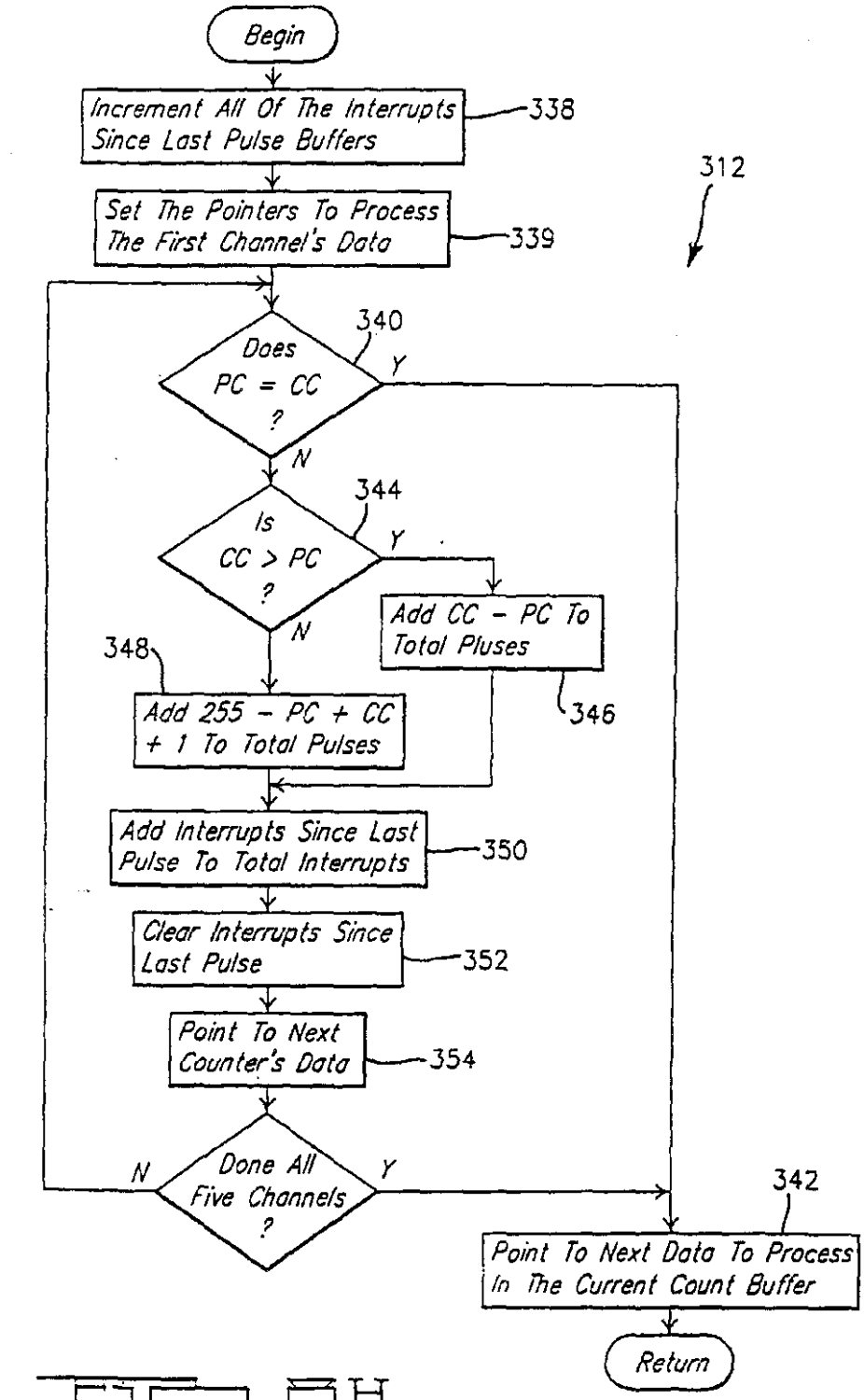

Begin
Increment All Of The Interrupts Since Last Pulse Buffers
338
312
Set The Pointers To Process The First Channel's Data
339
340
Does PC = CC ?
Y
N
344
Is CC > PC ?
Y
N
Add CC - PC To Total Pluses
346
348
Add 255 - PC + CC + 1 To Total Pulses
Add Interrupts Since Last Pulse To Total Interrupts
350
Clear Interrupts Since Last Pulse
352
Point To Next Counter's Data
354
Done All Five Channels ?
N
Y
342
Point To Next Data To Process In The Current Count Buffer
Return

FIG. 8H.

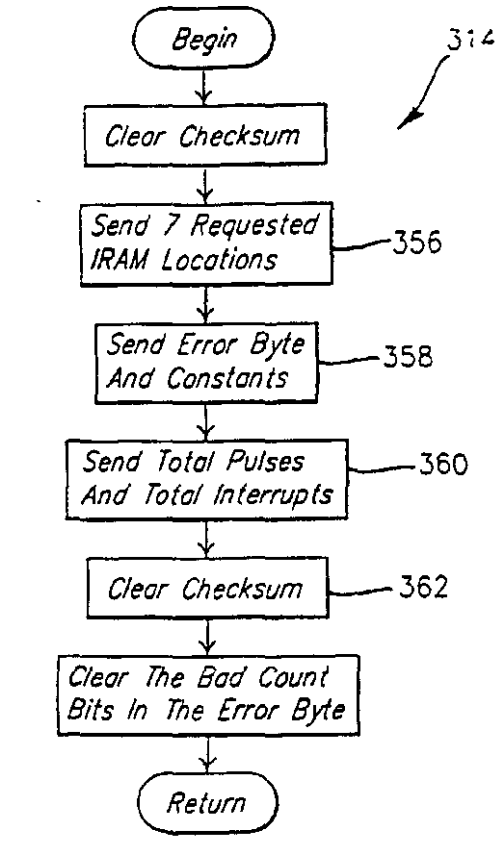

Begin
314
Clear Checksum
Send 7 Requested IRAM Locations
356
Send Error Byte And Constants
358
Send Total Pulses And Total Interrupts
360
Clear Checksum
362
Clear The Bad Count Bits In The Error Byte
Return

FIG. I.

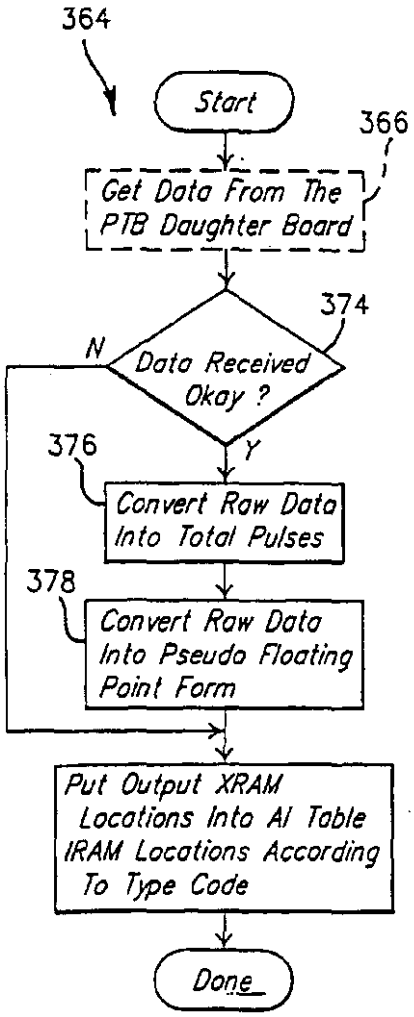

364
Start
366
Get Data From The PTB Daughter Board
374
Data Received Okay ?
N
Y
376
Convert Raw Data Into Total Pulses
378
Convert Raw Data Into Pseudo Floating Point Form
Put Output XRAM Locations Into AI Table IRAM Locations According To Type Code
Done

FIG. K.

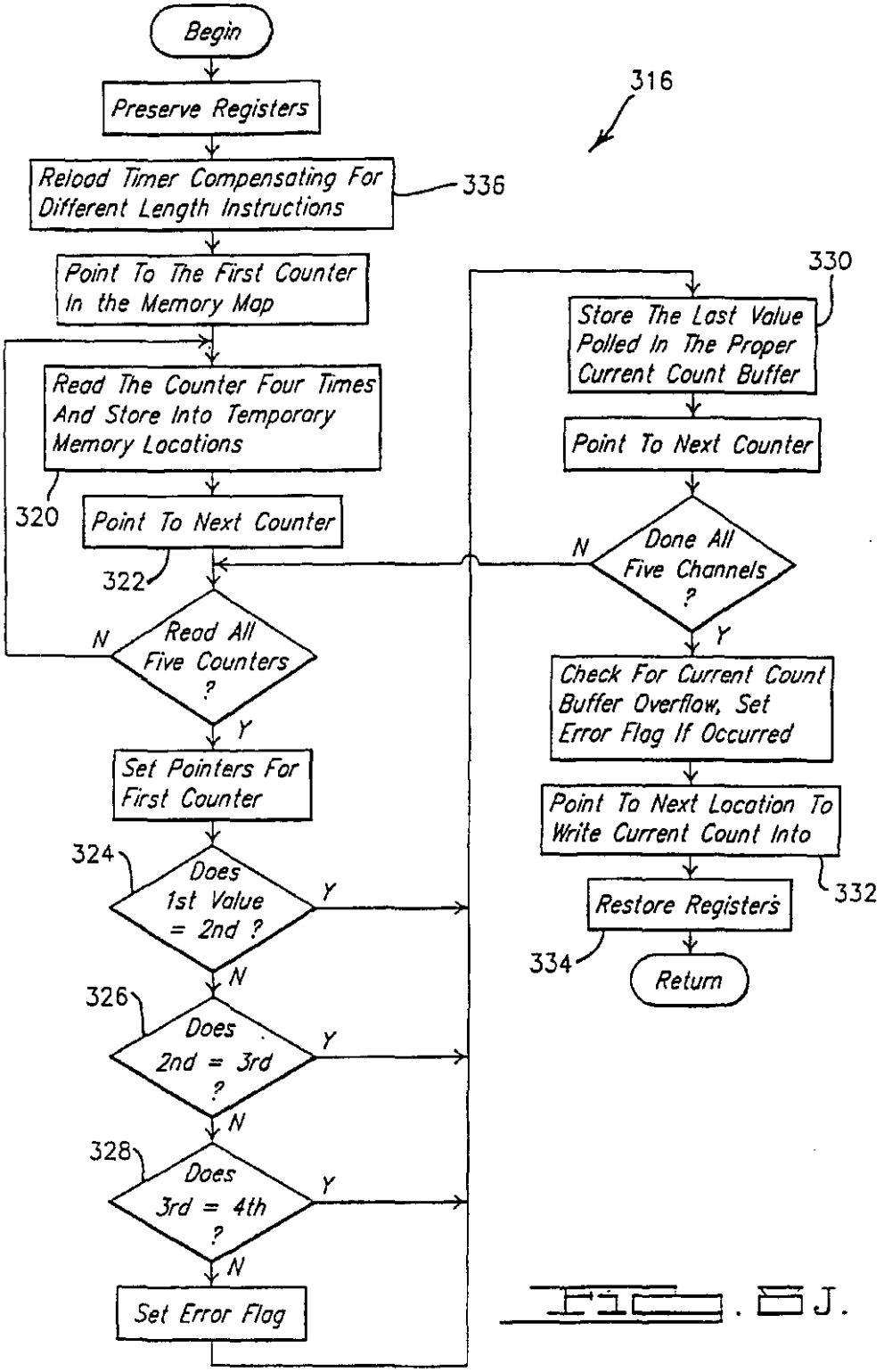

Begin
Preserve Registers
316
Reload Timer Compensating For Different Length Instructions
336
Point To The First Counter In the Memory Map
Read The Counter Four Times And Store Into Temporary Memory Locations
320
Point To Next Counter
322
N
Read All Five Counters ?
Y
Set Pointers For First Counter
324
Does 1st Value = 2nd ?
Y
N
326
Does 2nd = 3rd ?
Y
N
328
Does 3rd = 4th ?
Y
N
Set Error Flag
330
Store The Last Value Polled In The Proper Current Count Buffer
Point To Next Counter
N
Done All Five Channels ?
Y
Check For Current Count Buffer Overflow, Set Error Flag If Occurred
Point To Next Location To Write Current Count Into
332
Restore Registers
334
Return

FIG. 3J.

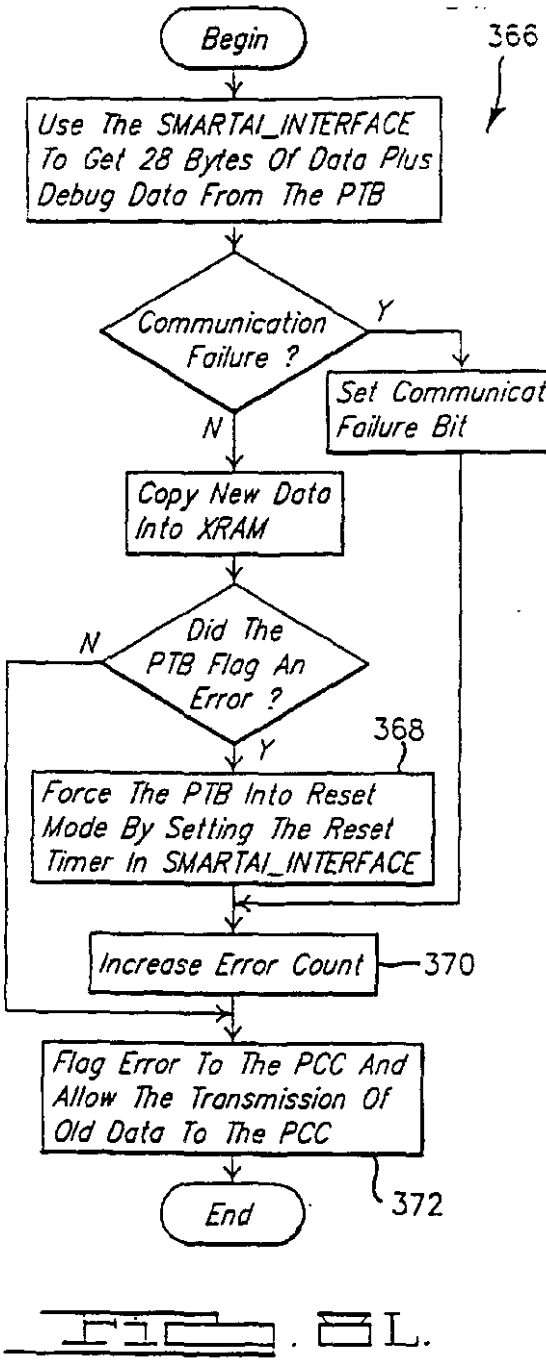

Begin
366
Use The SMARTAI_INTERFACE To Get 28 Bytes Of Data Plus Debug Data From The PTB
Communication Failure ?
Y
N
Set Communication Failure Bit
Copy New Data Into XRAM
Did The PTB Flag An Error ?
N
Y
368
Force The PTB Into Reset Mode By Setting The Reset Timer In SMARTAI_INTERFACE
Increase Error Count
370
Flag Error To The PCC And Allow The Transmission Of Old Data To The PCC
End
372

FIG. L.

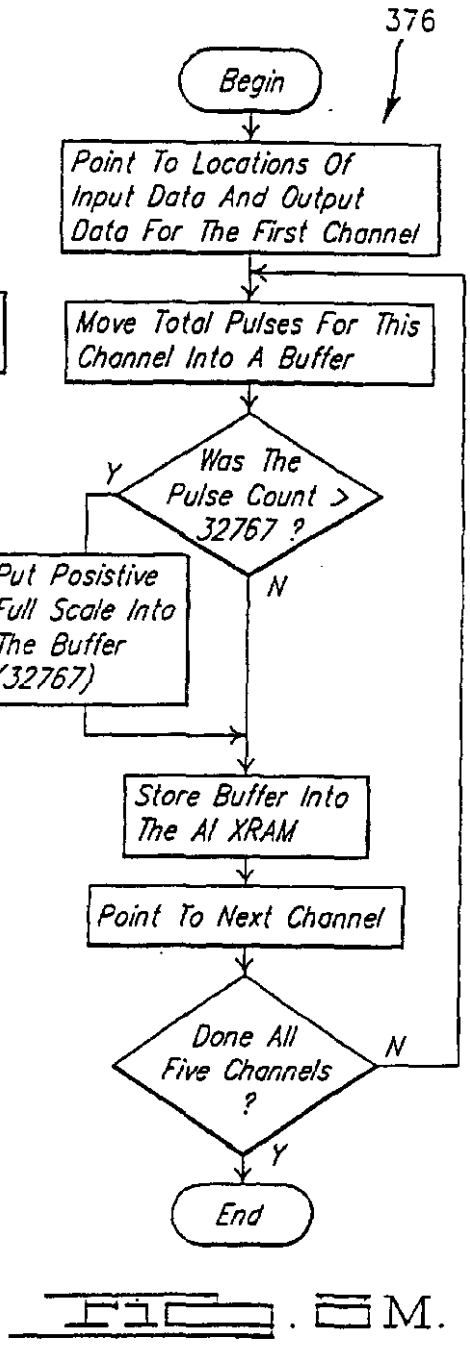

376
Begin
Point To Locations Of Input Data And Output Data For The First Channel
Move Total Pulses For This Channel Into A Buffer
Was The Pulse Count > 32767 ?
Y
N
Put Posistive Full Scale Into The Buffer (32767)
Store Buffer Into The AI XRAM
Point To Next Channel
Done All Five Channels ?
N
Y
End

FIG. M.

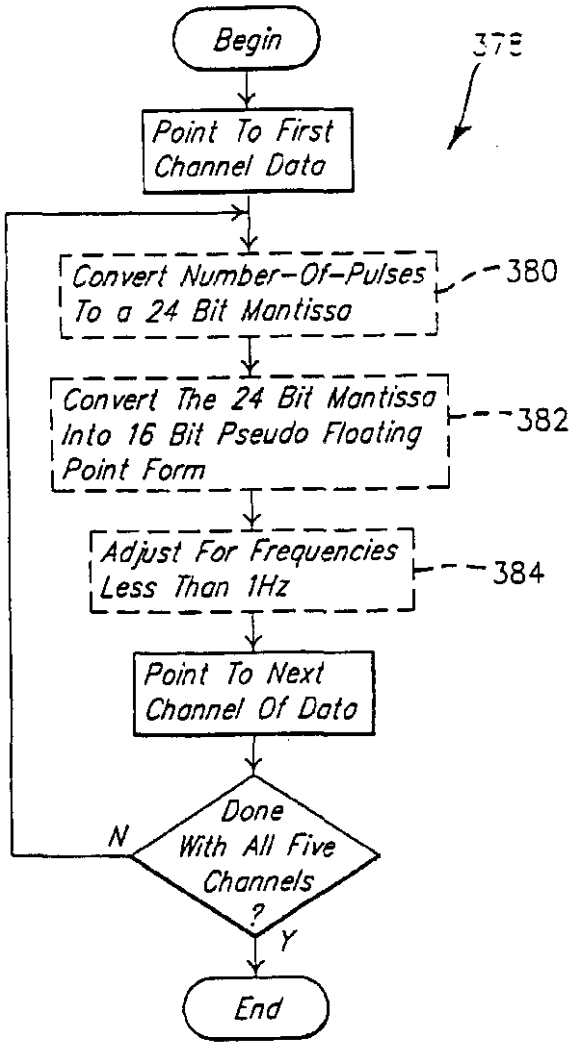

Begin
Point To First Channel Data
Convert Number-Of-Pulses To a 24 Bit Mantissa
380
Convert The 24 Bit Mantissa Into 16 Bit Pseudo Floating Point Form
382
Adjust For Frequencies Less Than 1Hz
384
Point To Next Channel Of Data
Done With All Five Channels ?
N
Y
End

FIG. 8N.

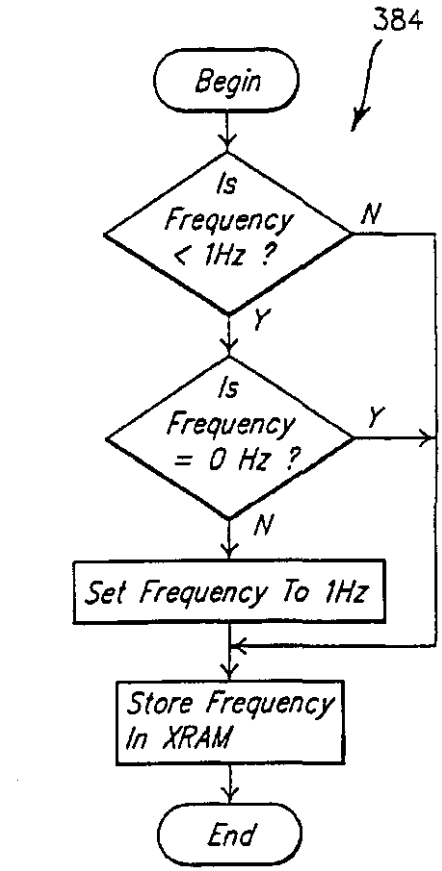

384
Begin
Is Frequency < 1Hz ?
N
Y
Is Frequency = 0 Hz ?
Y
N
Set Frequency To 1Hz
Store Frequency In XRAM
End

FIG. 8Q.

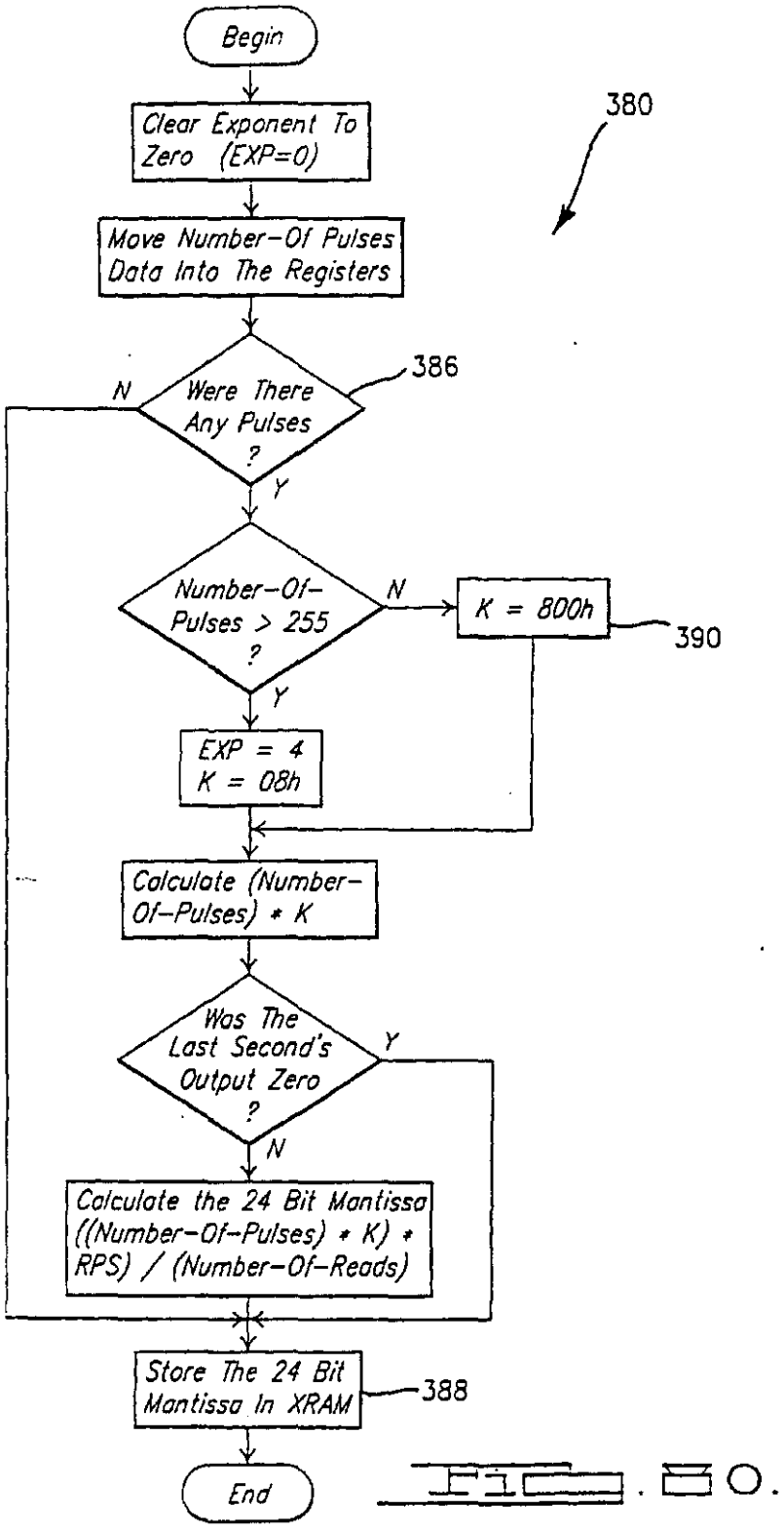

Begin
Clear Exponent To Zero (EXP=0)
380
Move Number-Of Pulses Data Into The Registers
386
N
Were There Any Pulses ?
Y
Number-Of-Pulses > 255 ?
N
K = 800h
390
Y
EXP = 4
K = 08h
Calculate (Number-Of-Pulses) * K
Was The Last Second's Output Zero ?
Y
N
Calculate the 24 Bit Mantissa ((Number-Of-Pulses) * K) * RPS) / (Number-Of-Reads)
Store The 24 Bit Mantissa In XRAM
388
End

Fig. 30.

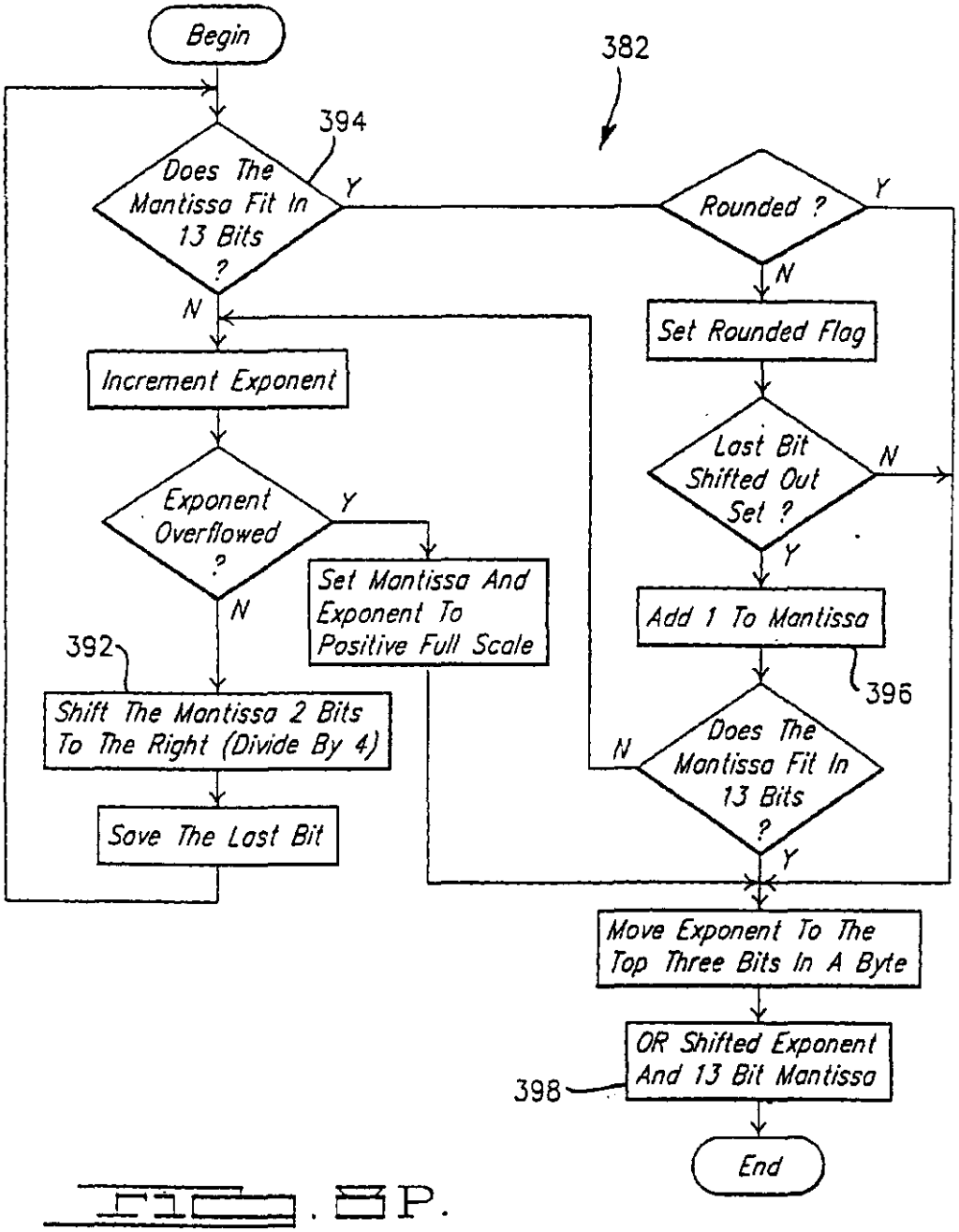

Begin
382
394
Does The Mantissa Fit In 13 Bits ?
Y
N
Rounded ?
Y
N
Set Rounded Flag
Increment Exponent
Last Bit Shifted Out Set ?
N
Y
Exponent Overflowed ?
Y
N
Set Mantissa And Exponent To Positive Full Scale
Add 1 To Mantissa
396
392
Shift The Mantissa 2 Bits To The Right (Divide By 4)
Does The Mantissa Fit In 13 Bits ?
N
Y
Save The Last Bit
Move Exponent To The Top Three Bits In A Byte
OR Shifted Exponent And 13 Bit Mantissa
398
End

FIG. 8P.

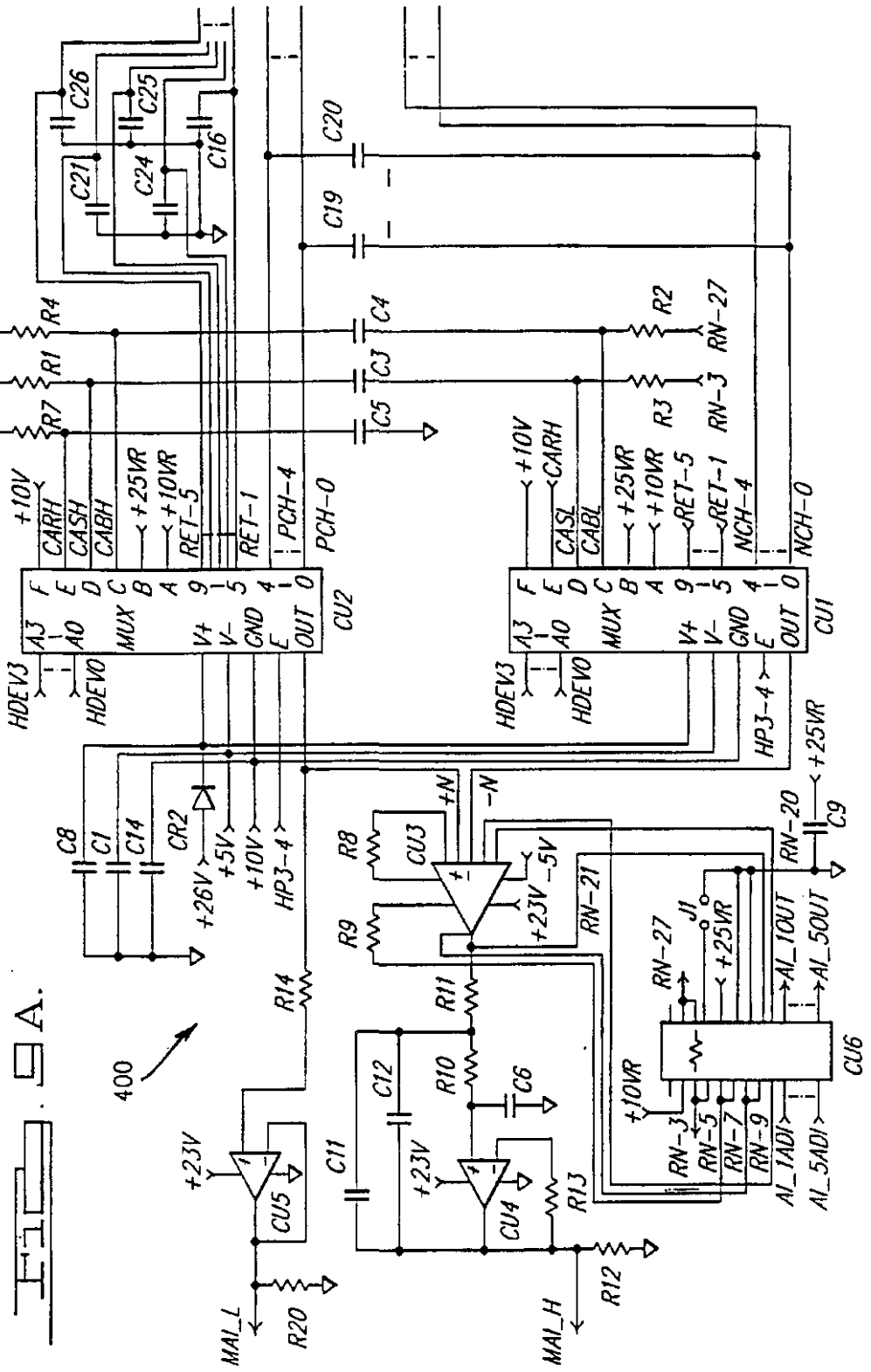

400
CU1
CU2
CU3
CU4
CU5
CU6
HDEV3
HDEV0
MUX
V+
V-
GND
OUT
CARH
CASH
CABH
CASL
CABL
RET-5
RET-1
PCH-4
PCH-0
NCH-4
NCH-0
HP3-4
+10V
+10VR
+25VR
+26V
+5V
-5V
+23V
+N
-N
RN-3
RN-27
RN-21
RN-20
RN-5
RN-7
RN-9
AI_1ADI
AI_5ADI
AI_1OUT
AI_5OUT
J1
MAI_L
MAI_H
R1
R2
R3
R4
R7
R8
R9
R10
R11
R12
R13
R14
R20
C1
C3
C4
C5
C6
C8
C9
C11
C12
C14
C16
C19
C20
C21
C24
C25
C26
CR2

FIG. 9A.

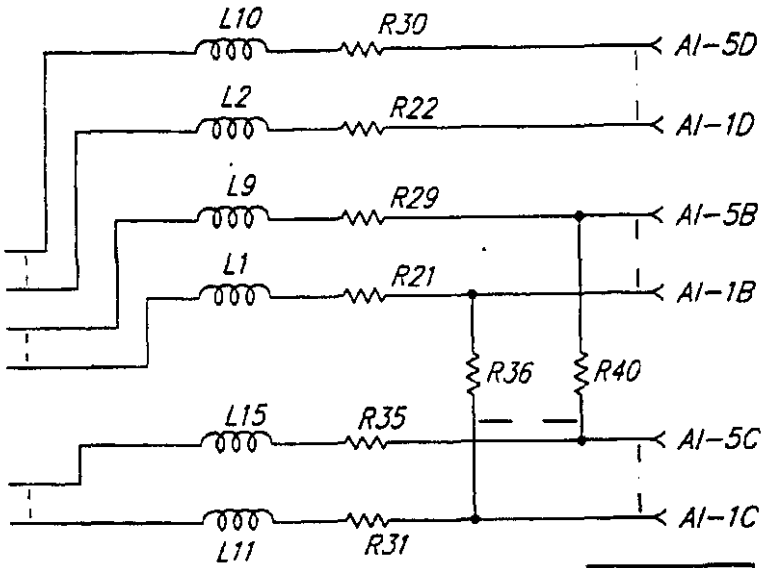


FIG. 3B.

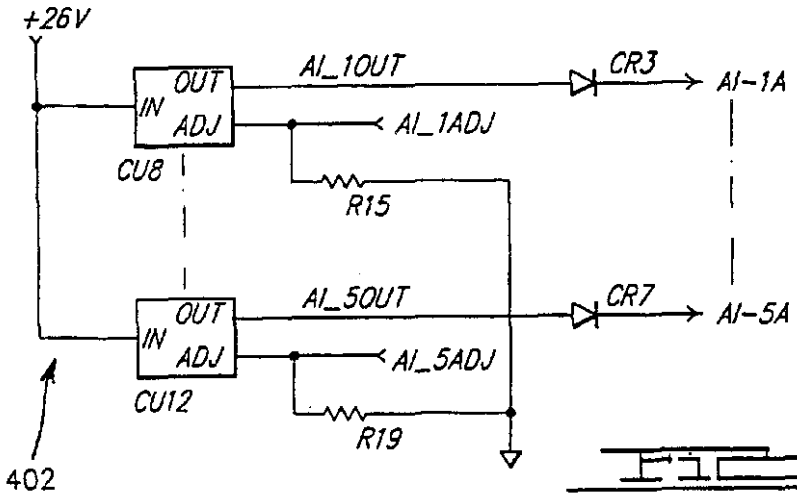


FIG. 3C.

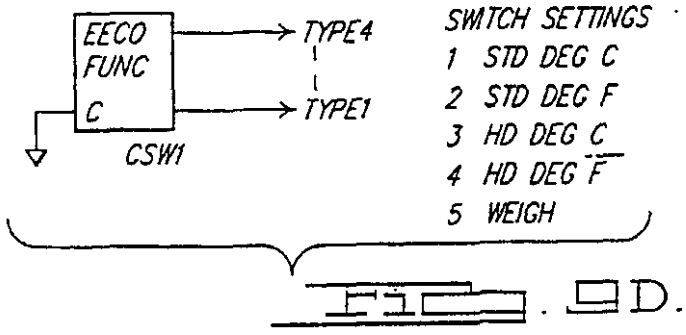


FIG. 3D.

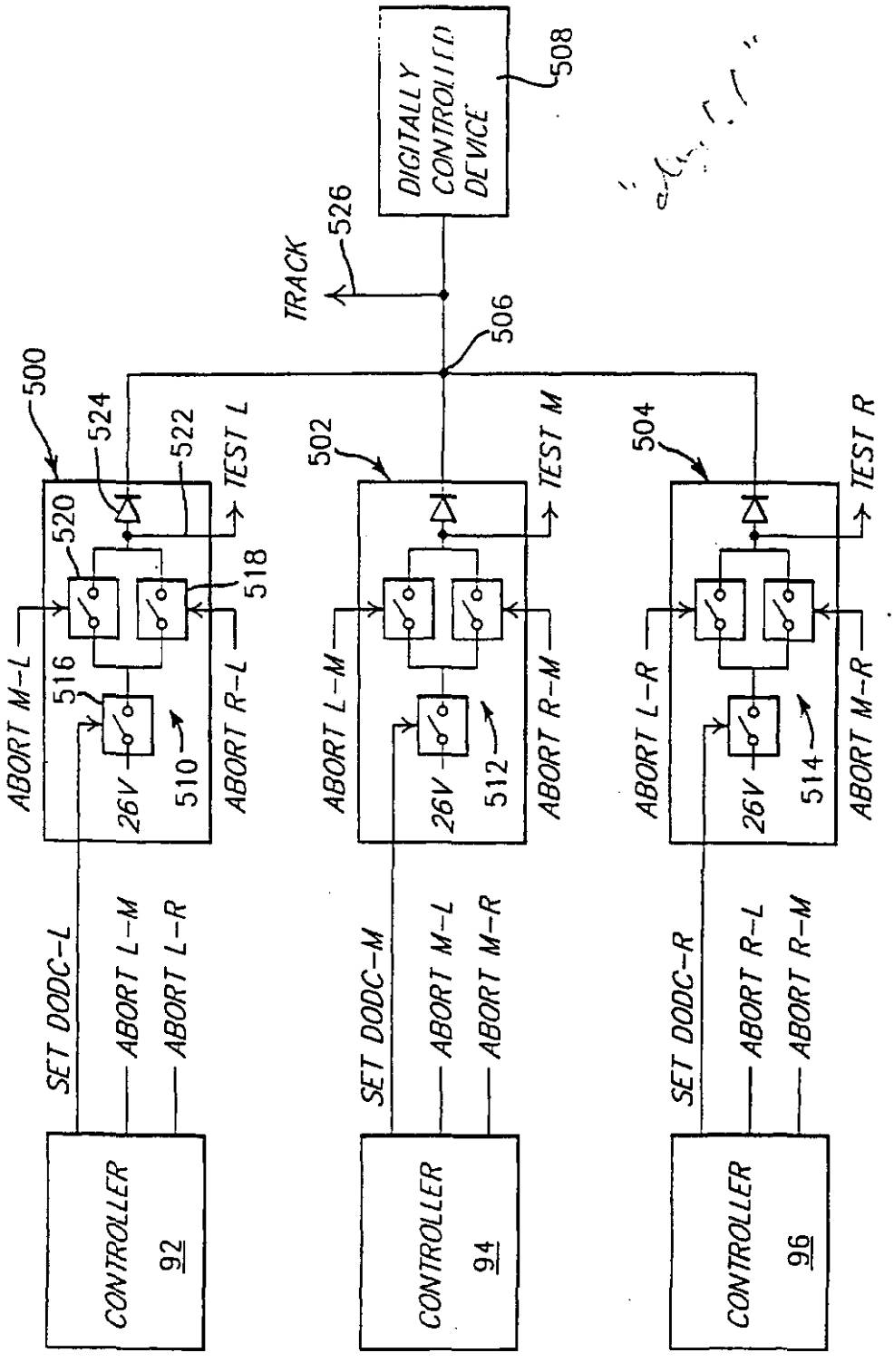

DIGITALLY CONTROLLED DEVICE
508
TRACK
526
506
500
502
504
524
522
520
518
516
510
512
514
TEST L
TEST M
TEST R
ABORT M-L
ABORT L-M
ABORT R-L
ABORT L-R
ABORT R-M
ABORT M-R
26V
SET DODC-L
SET DODC-M
SET DODC-R
CONTROLLER 92
CONTROLLER 94
CONTROLLER 96

Fig. 1A.

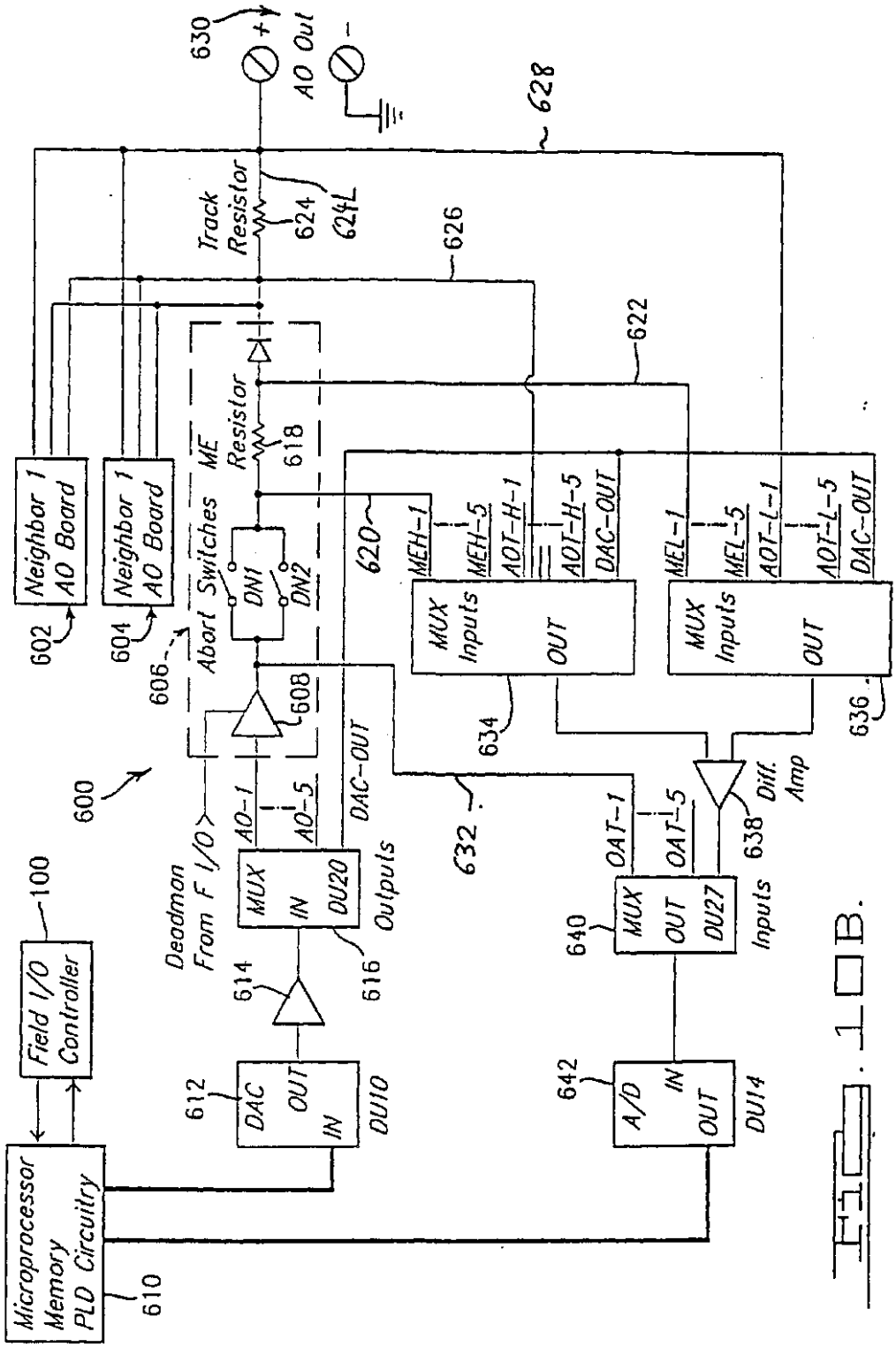

630
AO Out
628
Track Resistor
624
624L
626
622
ME Resistor
618
Neighbor 1 AO Board
602
Neighbor 1 AO Board
604
606
Abort Switches
DN1
DN2
608
620
MEH-1
MEH-5
AOT-H-1
AOT-H-5
DAC-OUT
MEL-1
MEL-5
AOT-L-1
AOT-L-5
DAC-OUT
MUX Inputs
OUT
634
MUX Inputs
OUT
636
600
Deadman From F I/O
AO-1
AO-5
DAC-OUT
632
OAT-1
OAT-5
Diff. Amp
638
MUX IN DU20
Outputs
616
614
MUX OUT DU27
Inputs
640
100
Field I/O Controller
612
DAC OUT IN
DU10
642
A/D IN OUT
DU14
Microprocessor Memory PLD Circuitry
610

FIG. 10B.

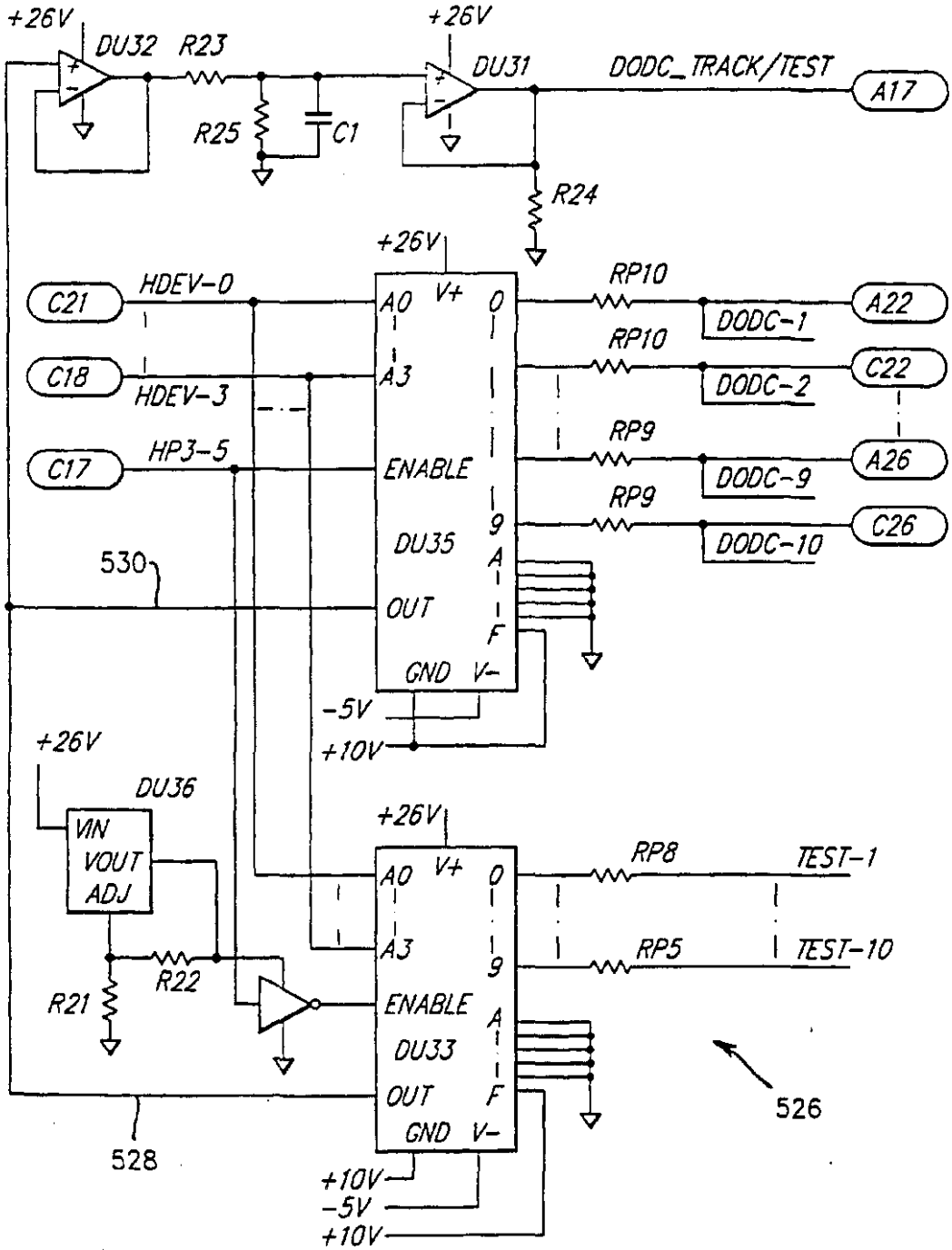

+26V
DU32
R23
R25
C1
+26V
DU31
DODC_TRACK/TEST
A17
R24
+26V
C21
HDEV-0
C18
HDEV-3
C17
HP3-5
A0
V+
A3
ENABLE
DU35
OUT
GND
V-
0
9
A
F
RP10
DODC-1
A22
RP10
DODC-2
C22
RP9
DODC-9
A26
RP9
DODC-10
C26
530
-5V
+10V
+26V
DU36
VIN
VOUT
ADJ
R21
R22
+26V
A0
V+
A3
ENABLE
DU33
OUT
GND
V-
RP8
TEST-1
RP5
TEST-10
526
528
+10V
-5V
+10V

FIG. 11B.

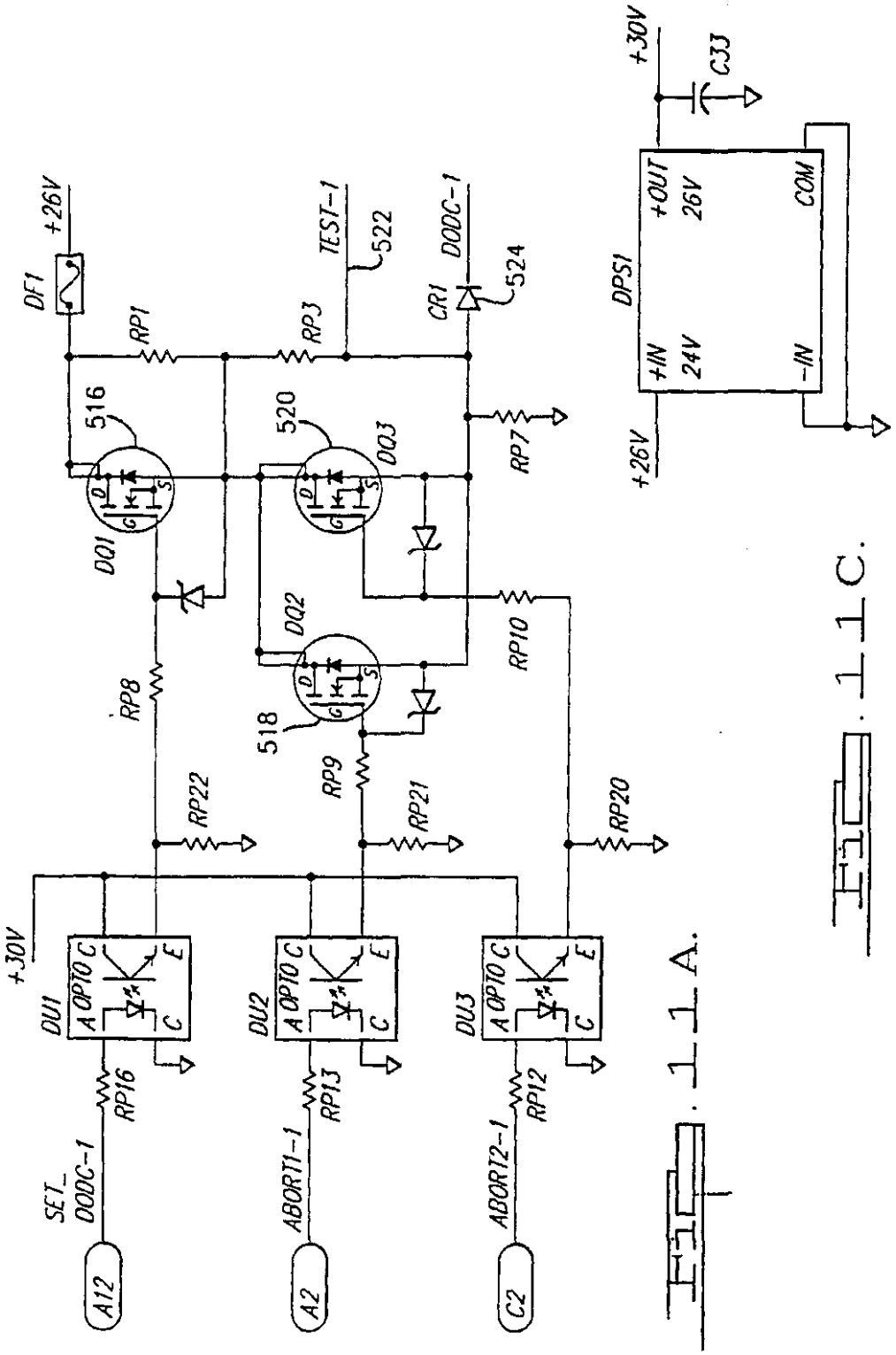


FIG. 11A.

FIG. 11C.

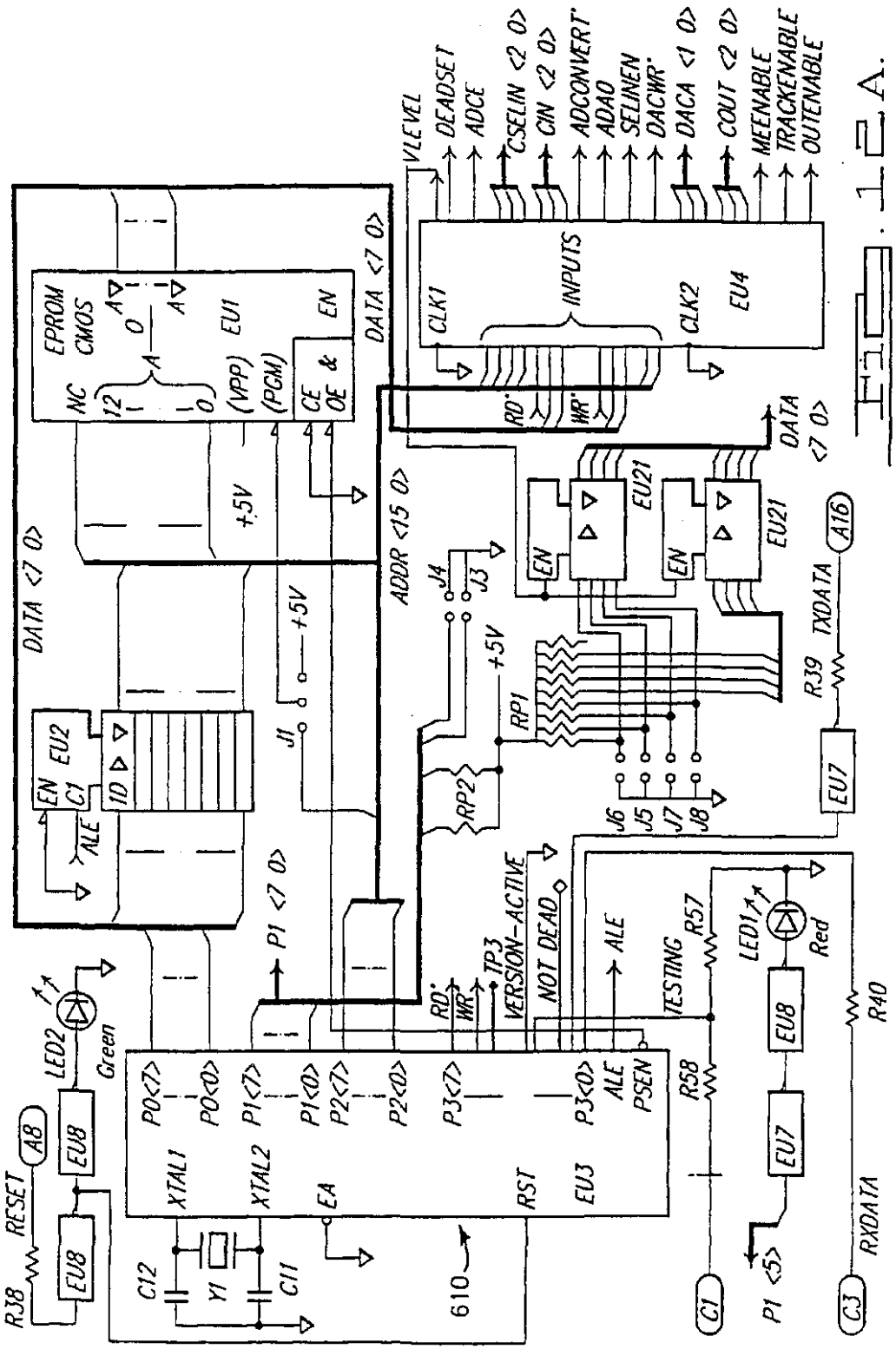

VLEVEL
DEADSET
ADCE
CSELIN <2 0>
CIN <2 0>
ADCONVERT
ADAO
SELINEN
DACWR
DACA <1 0>
COUT <2 0>
MEENABLE
TRACKENABLE
OUTENABLE
CLK1
INPUTS
CLK2
EU4
RD
WR
DATA <7 0>
EPROM CMOS
EU1
NC
12
(VPP)
(PGM)
CE
OE
EN
+5V
ADDR <15 0>
EU21
J4
J3
J1
RP1
RP2
J6
J5
J7
J8
R39
TXDATA
A16
EU7
EU2
EN
C1
ID
ALE
P1 <7 0>
TP3
VERSION-ACTIVE
NOT DEAD
ALE
TESTING
R57
LED1
Red
EU8
R40
LED2
Green
P0<7>
P0<0>
P1<7>
P1<0>
P2<7>
P2<0>
P3<7>
P3<0>
ALE
PSEN
R58
XTAL1
XTAL2
EA
RST
EU3
A8
RESET
R38
C12
Y1
C11
610
C1
P1 <5>
C3
RXDATA

Fig. 12A.

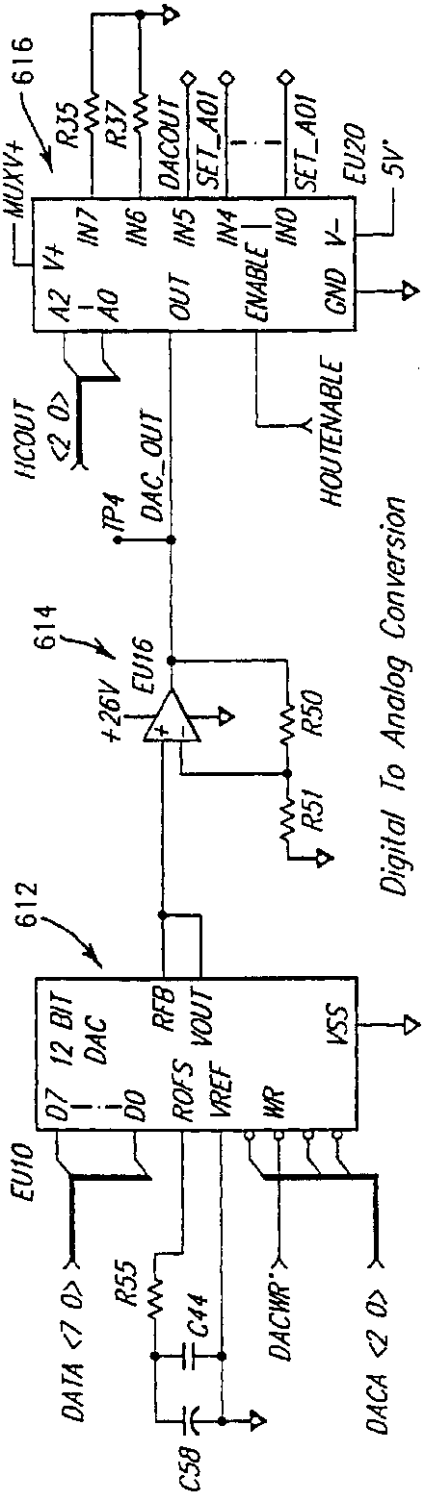


Fig. 12B.

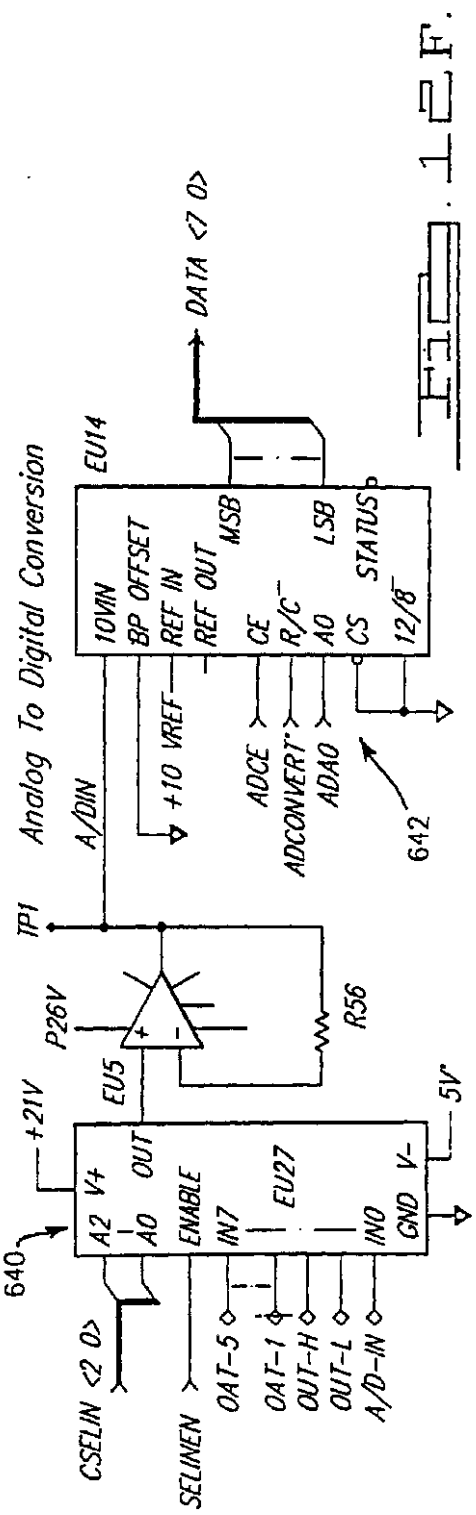


Fig. 12F.

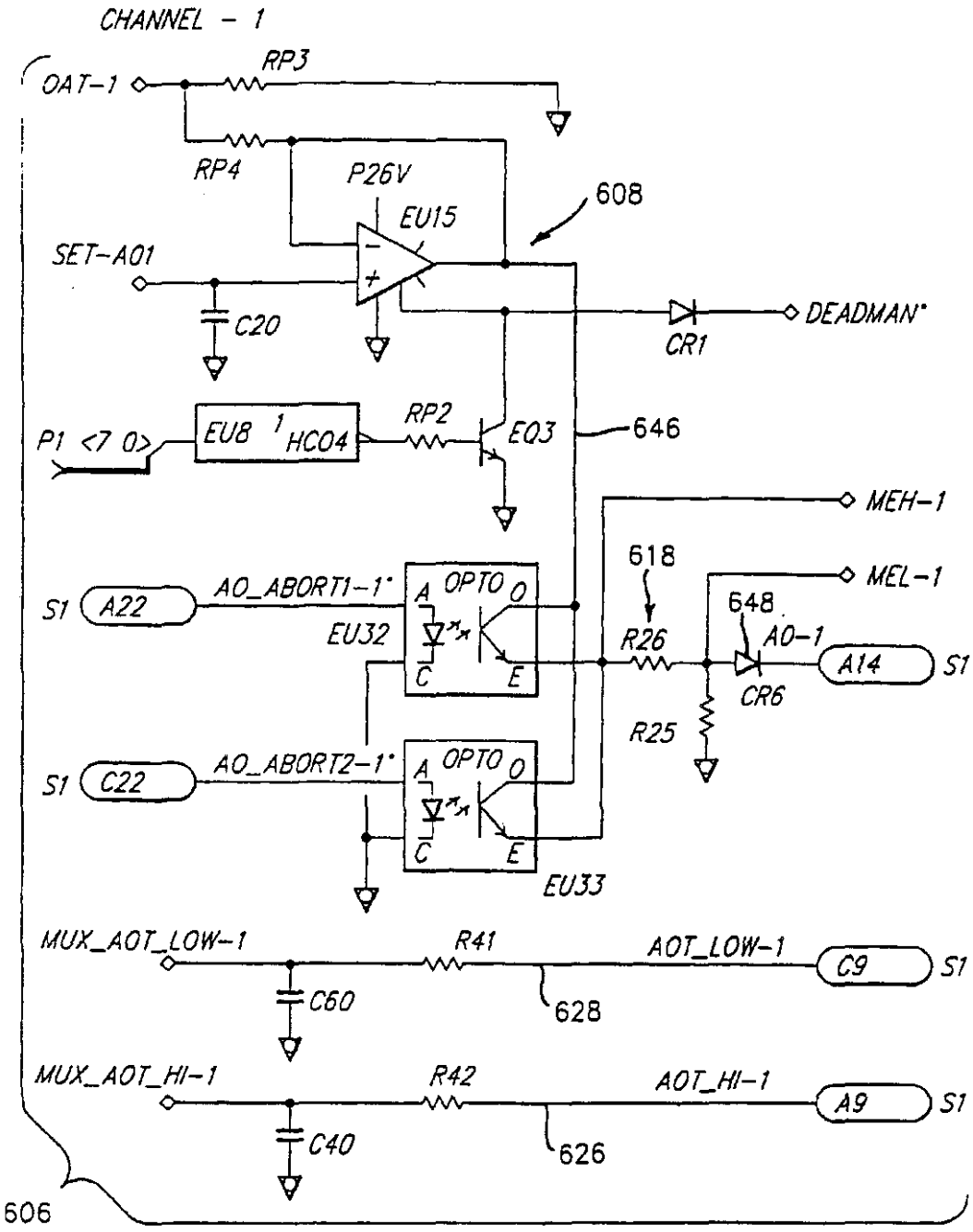

CHANNEL - 1
OAT-1
RP3
RP4
P26V
EU15
608
SET-A01
C20
DEADMAN*
CR1
P1 <7 0>
EU8 1 HC04
RP2
EO3
646
MEH-1
618
MEL-1
S1
A22
AO_ABORT1-1*
A OPTO O
EU32
C E
R26
648
AO-1
A14
S1
CR6
R25
S1
C22
AO_ABORT2-1*
A OPTO O
C E
EU33
MUX_AOT_LOW-1
R41
AOT_LOW-1
C9
S1
C60
628
MUX_AOT_HI-1
R42
AOT_HI-1
A9
S1
C40
626
606

FIG. 12C.

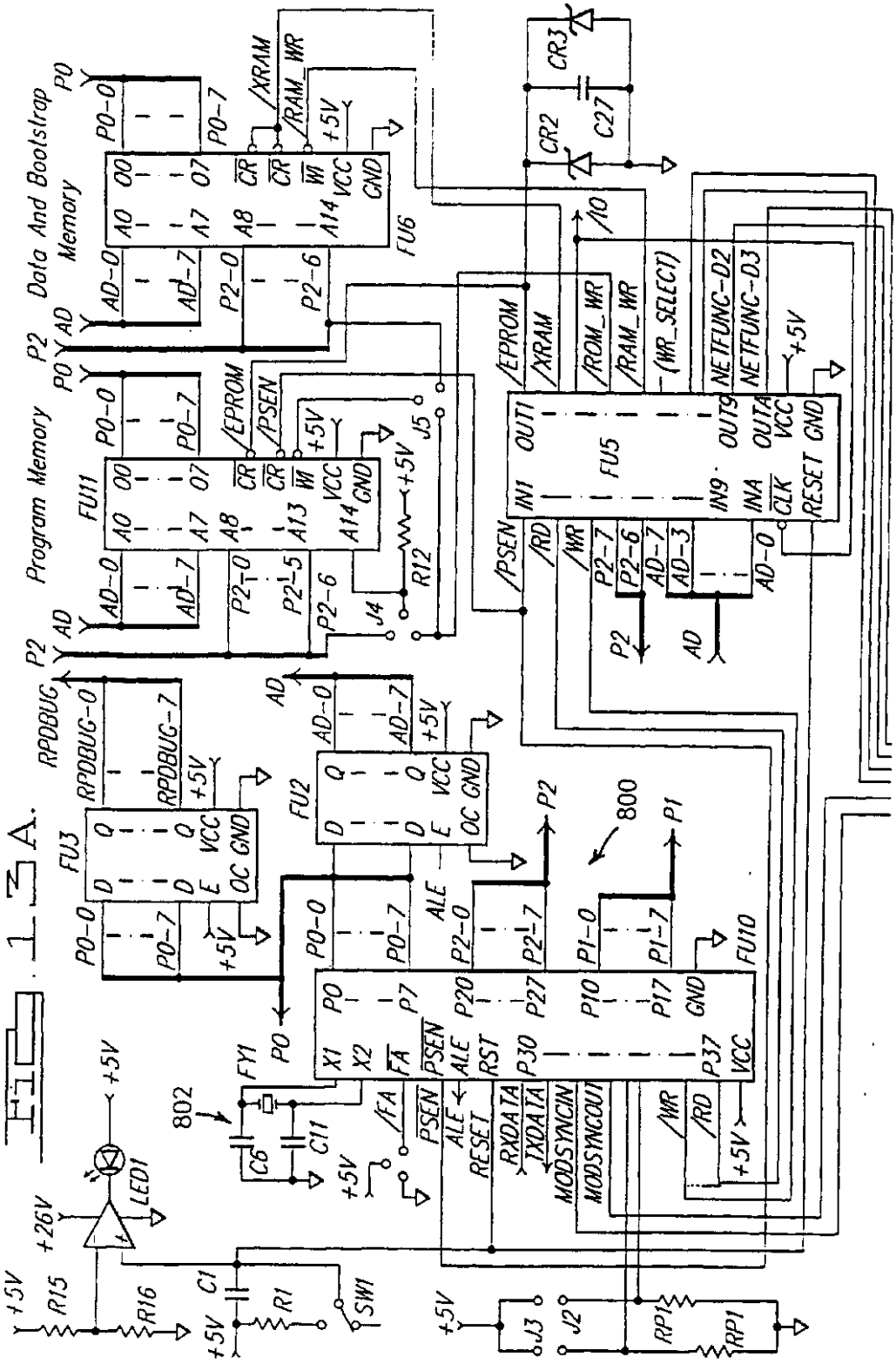

Data And Bootstrap Memory
Program Memory
FU6
FU11
FU5
FU3
FU2
FU10
FY1
800
802
LED1
+26V
+5V
R15
R16
C1
R1
SW1
J3
J2
RP1
RP1
C6
C11
CR2
CR3
C27
J4
J5
R12
P0
P2
AD
P1
RPDBUG
RPDBUG-0
RPDBUG-7
P0-0
P0-7
AD-0
AD-7
P2-0
P2-6
P2-5
P2-7
P1-0
P1-7
/XRAM
/RAM_WR
/EPROM
/PSEN
/ROM_WR
(WR_SELECT)
NETFUNC-D2
NETFUNC-D3
/RD
/WR
/FA
/ALE
RXDATA
TXDATA
MODSYNCIN
MODSYNCOUT
RESET
OUT1
OUT9
OUTA
IN1
IN9
INA
CLK
RESET
ALE
RST
P30
P37
P20
P27
P10
P17
GND
VCC
X1
X2
FA
PSEN

FIG. 13A.

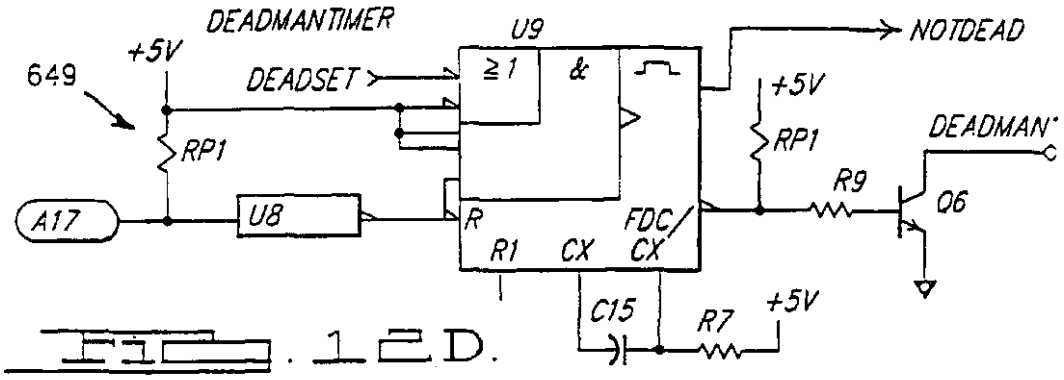


Fig. 12D.

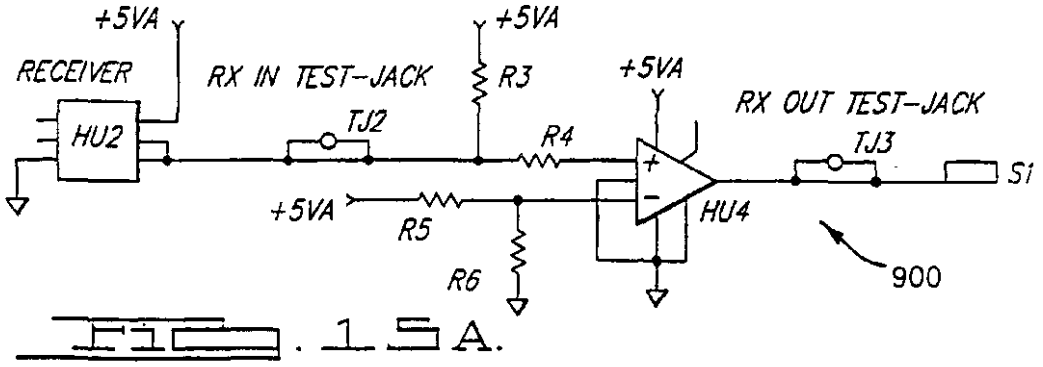


Fig. 15A.

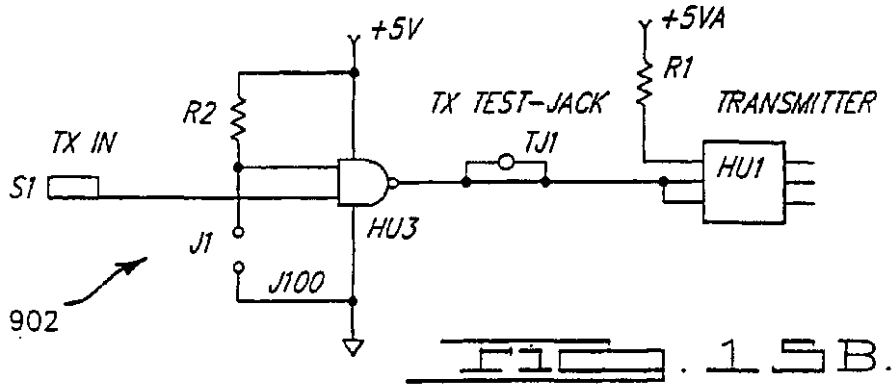


Fig. 15B.

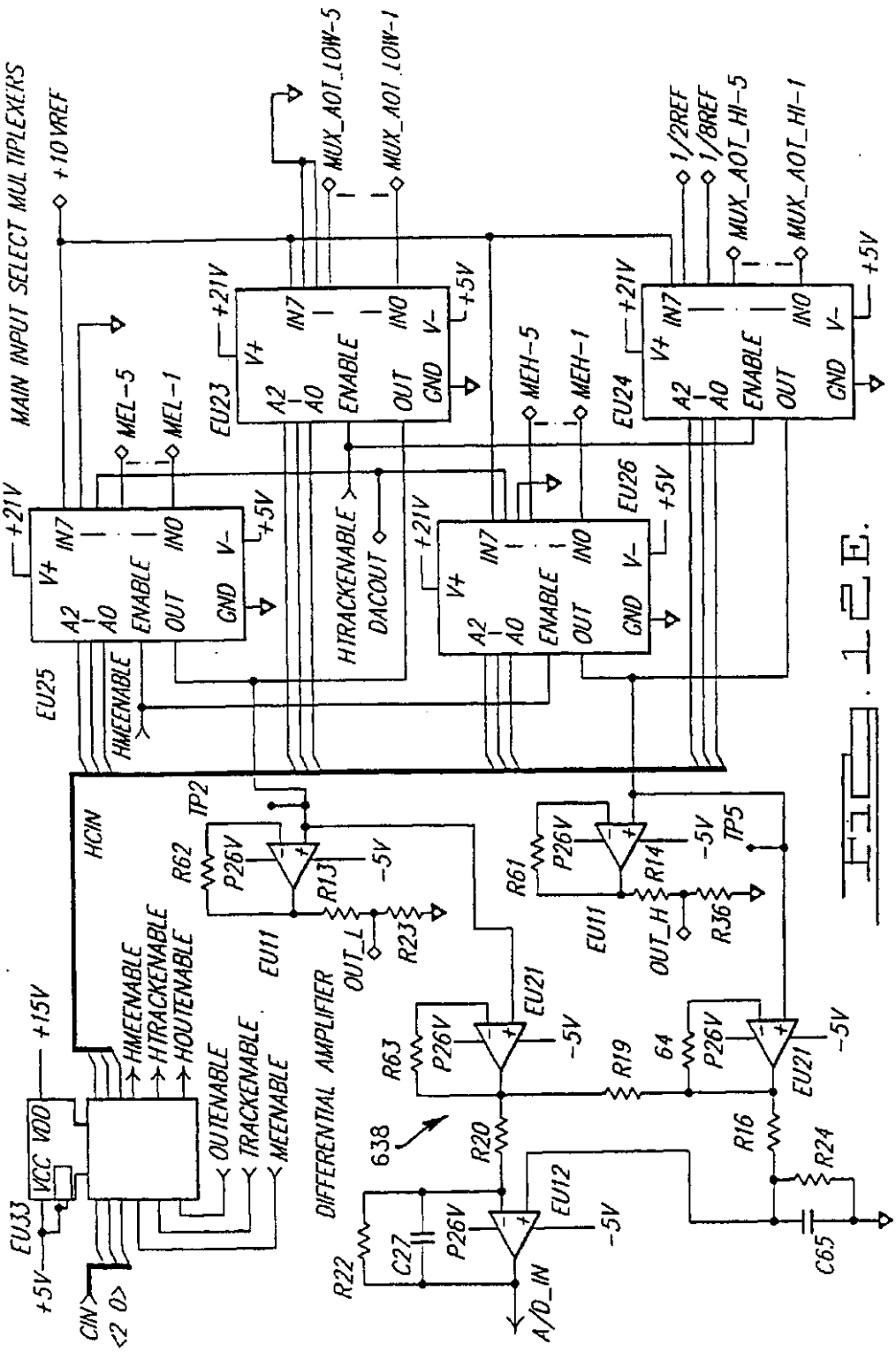

MAIN INPUT SELECT MULTIPLEXERS
+10 VREF
MUX_AOI_LOW-5
MUX_AOI_LOW-1
1/2REF
1/8REF
MUX_AOI_HI-5
MUX_AOI_HI-1
MEL-5
MEL-1
MEH-5
MEH-1
EU23
EU24
EU25
EU26
+21V
+5V
HTRACKENABLE
DACOUT
HMEENABLE
HCIN
HMEENABLE
HTRACKENABLE
HOUTENABLE
OUTENABLE
TRACKENABLE
MEENABLE
+15V
EU33
VCC VDD
+5V
CIN
<2 0>
TP2
TP5
R62
R61
P26V
R13
R14
-5V
EU11
OUT_L
OUT_H
R23
R36
DIFFERENTIAL AMPLIFIER
638
R63
R19
64
R20
R16
R24
R22
C27
C65
EU12
EU21
A/D_IN

Fig. 12E.

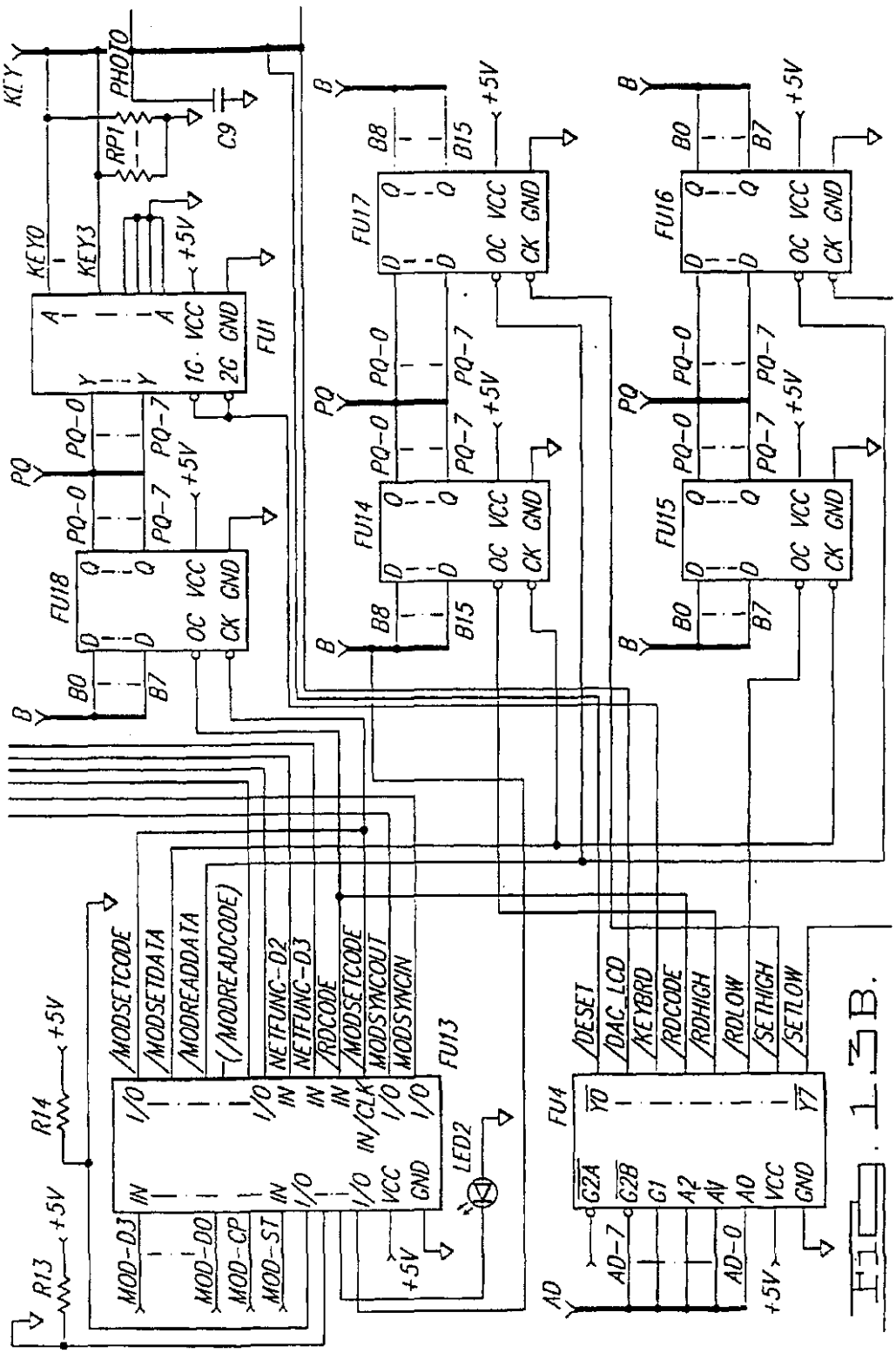

KEY
PHOTO
C9
RP1
KEY0
KEY3
+5V
FU11
A
Y
1G
2G
VCC
GND
PQ
PQ-0
PQ-7
FU18
FU14
FU17
FU15
FU16
D
Q
OC
CK
B
B0
B7
B8
B15
R14
R13
/MODSETCODE
/MODSETDATA
/MODREADDATA
/(/MODREADCODE)
NETFUNC-02
NETFUNC-03
/RDCODE
/MODSETCODE
MODSYNCOUT
MODSYNCIN
FU13
I/O
IN
IN/CLK
MOD-D3
MOD-D0
MOD-CP
MOD-ST
LED2
FU4
/DESET
/DAC_LCD
/KEYBRD
/RDCODE
/RDHIGH
/RDLOW
/SETHIGH
/SETLOW
Y0
Y7
G2A
G2B
G1
A2
A1
A0
AD
AD-7
AD-0

Fig. 13B.

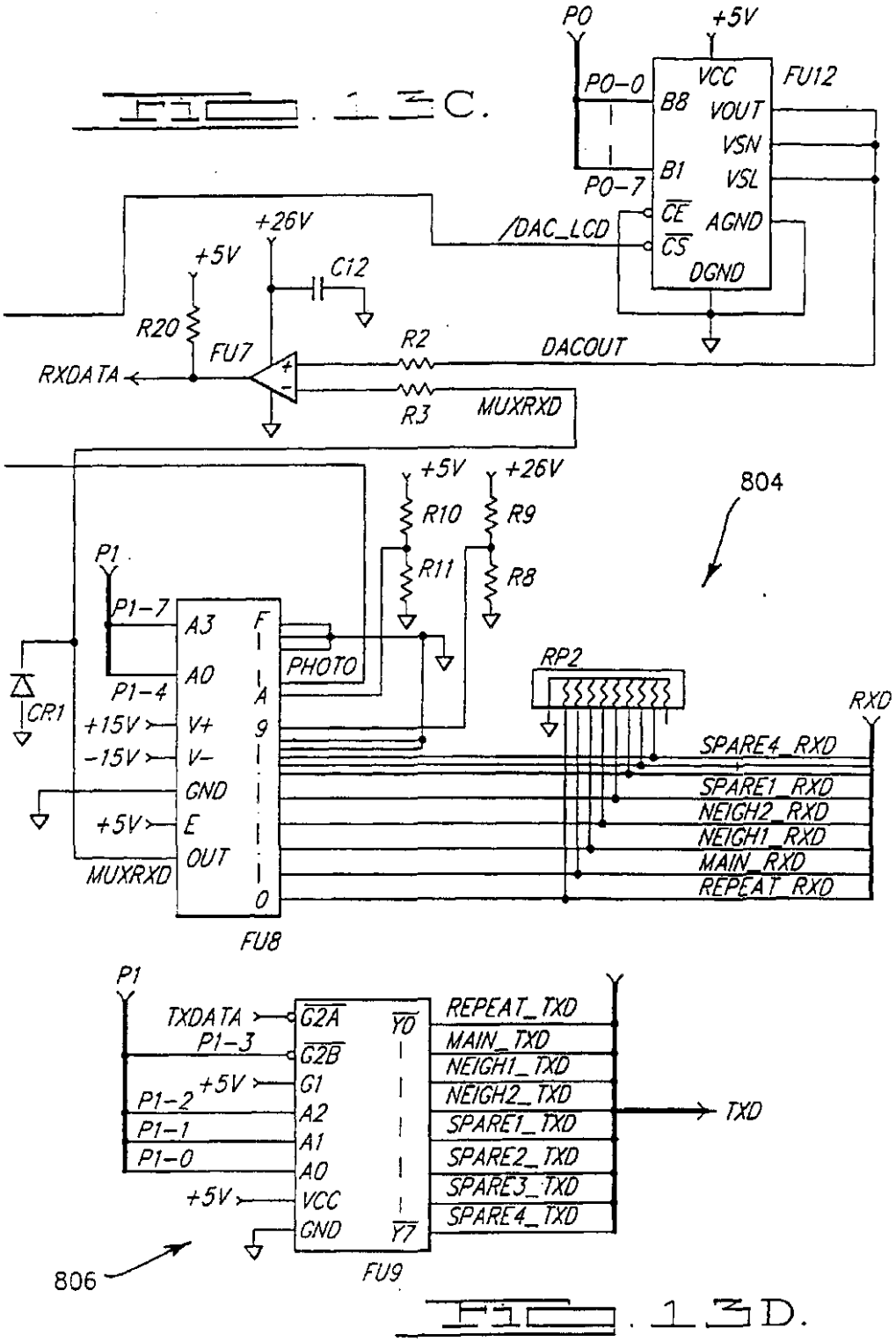

P0
+5V
P0-0
P0-7
VCC
B8
B1
FU12
VOUT
VSN
VSL
/DAC_LCD
CE
CS
AGND
DGND
+26V
+5V
C12
R20
FU7
RXDATA
R2
DACOUT
R3
MUXRXD
+5V
+26V
R10
R9
R11
R8
804
P1
P1-7
P1-4
A3
A0
F
A
9
0
PHOTO
CR1
+15V
-15V
V+
V-
GND
+5V
E
OUT
MUXRXD
FU8
RP2
RXD
SPARE4_RXD
SPARE1_RXD
NEIGH2_RXD
NEIGH1_RXD
MAIN_RXD
REPEAT_RXD
P1
TXDATA
P1-3
P1-2
P1-1
P1-0
+5V
G2A
G2B
G1
A2
A1
A0
VCC
GND
Y0
Y7
REPEAT_TXD
MAIN_TXD
NEIGH1_TXD
NEIGH2_TXD
SPARE1_TXD
SPARE2_TXD
SPARE3_TXD
SPARE4_TXD
TXD
806
FU9

FIG. 13D.

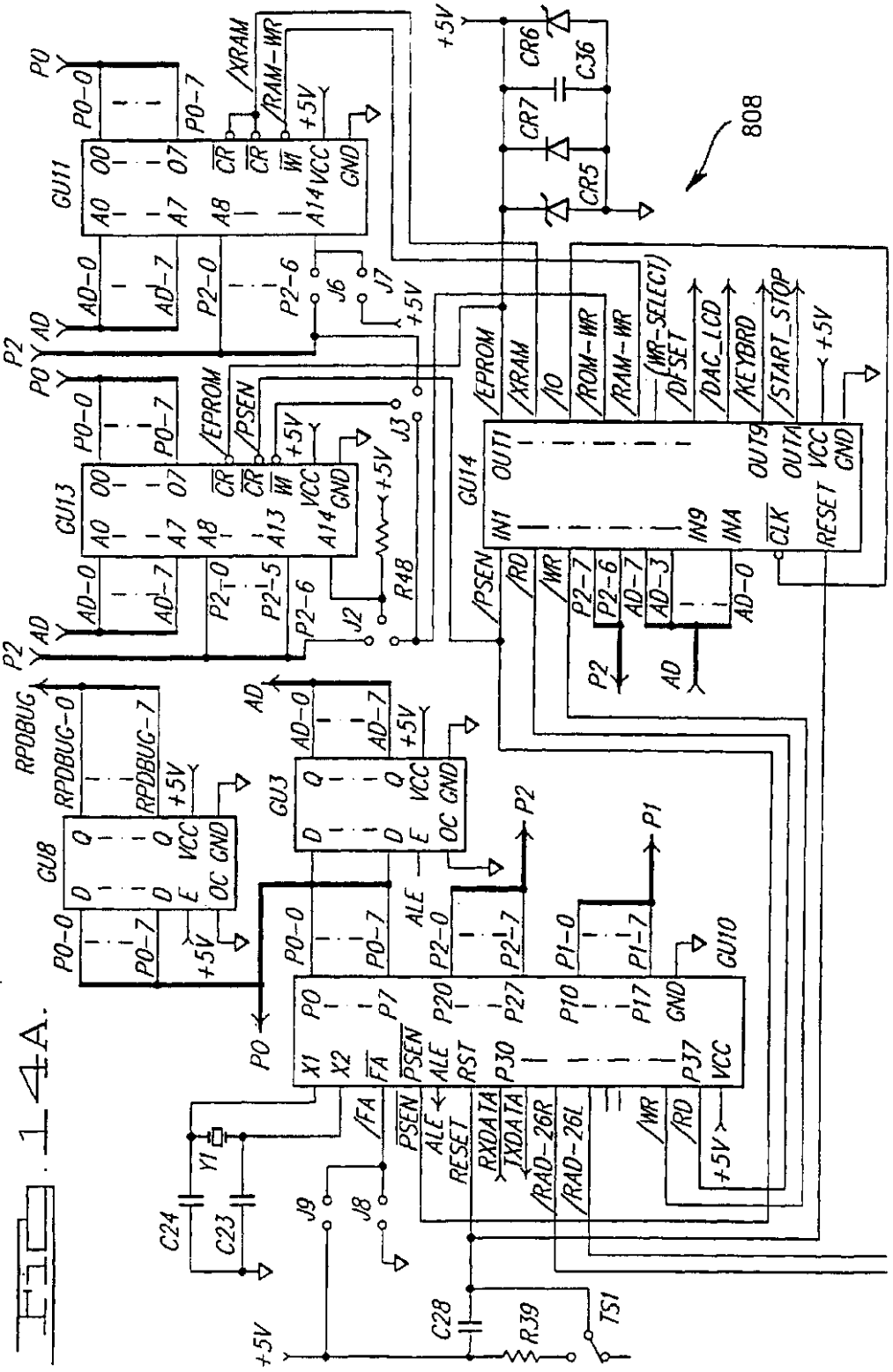

GU11
GU13
GU14
GU8
GU3
GU10
808
+5V
P0
P2
AD
P1
RPDBUG
/XRAM
/RAM-WR
/EPROM
/PSEN
/IO
/ROM-WR
/(WR-SELECT)
/DFSET
/DAC_LCD
/KEYBRD
/START_STOP
/RD
/WR
/FA
/RXDATA
/TXDATA
/RAD-26R
/RAD-26L
/RESET
/RAD-26L
C24
C23
C28
R39
TS1
Y1
J9
J8
J2
J3
J6
J7
R48
CR5
CR6
CR7
C36

FIG. 14A.

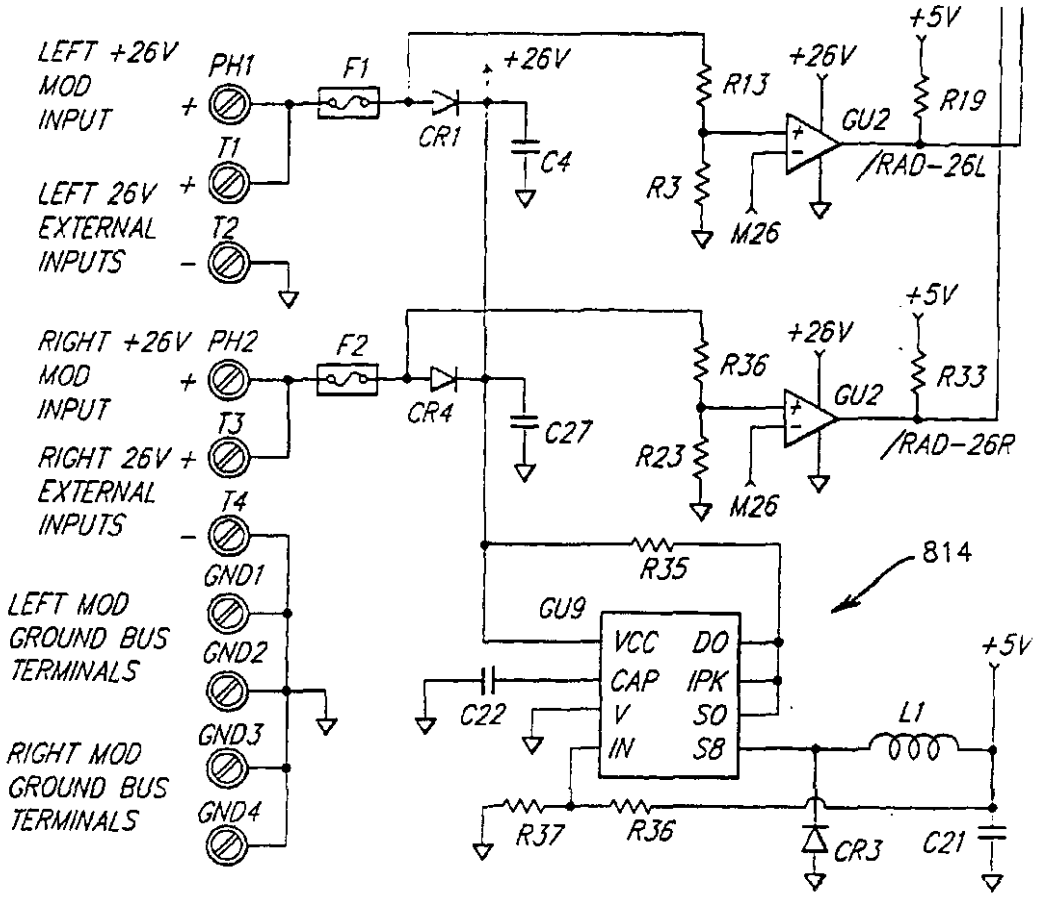


FIG. 14B.

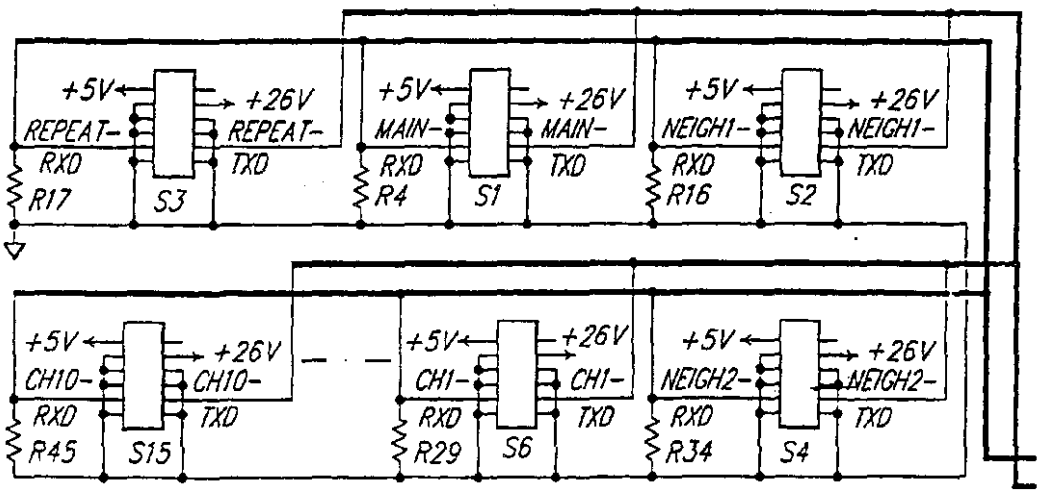


FIG. 14C.

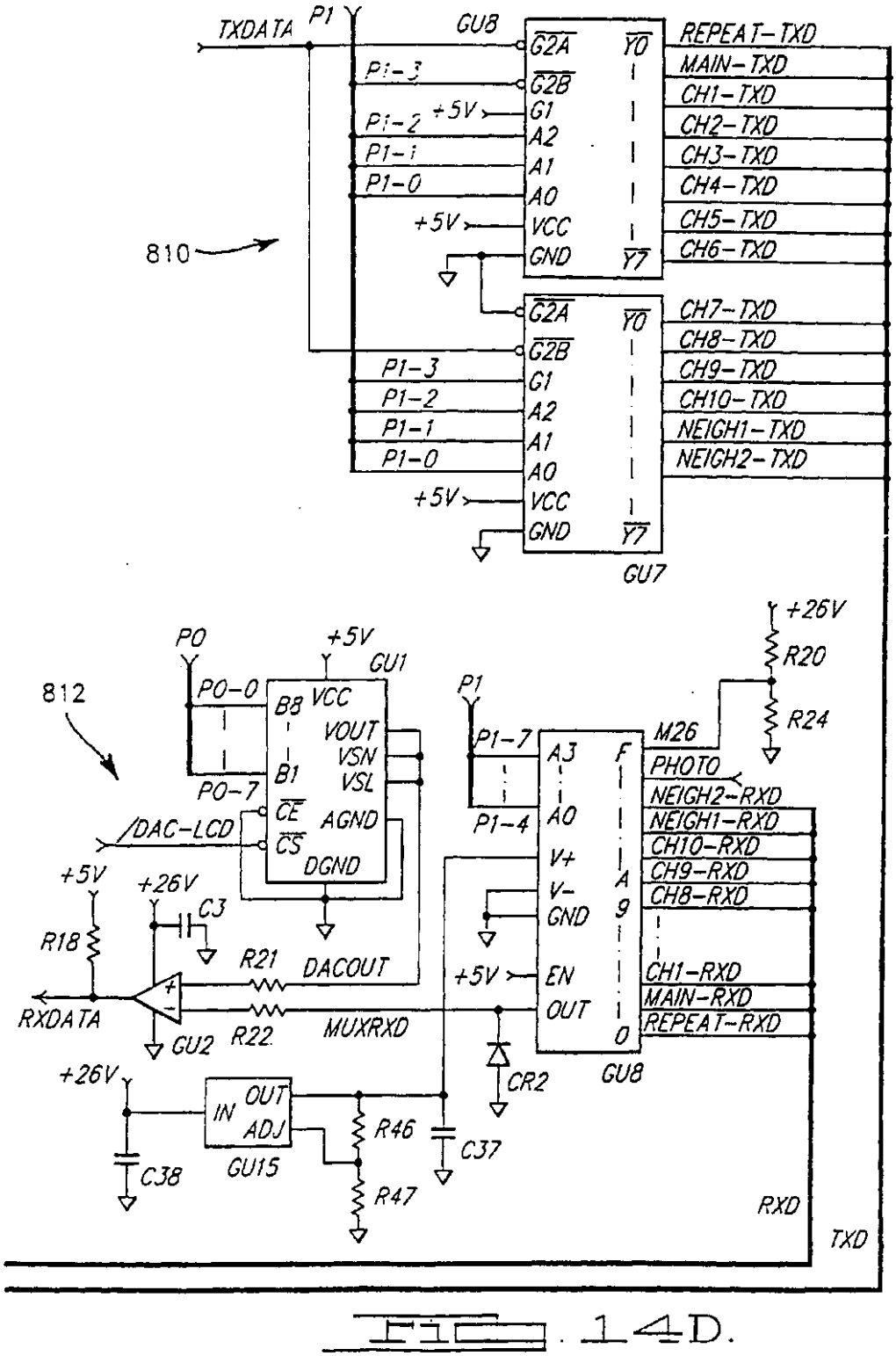

TXDATA
P1
GU8
G2A
G2B
G1
A2
A1
A0
VCC
GND
Y0
Y7
+5V
P1-3
P1-2
P1-1
P1-0
REPEAT-TXD
MAIN-TXD
CH1-TXD
CH2-TXD
CH3-TXD
CH4-TXD
CH5-TXD
CH6-TXD
CH7-TXD
CH8-TXD
CH9-TXD
CH10-TXD
NEIGH1-TXD
NEIGH2-TXD
GU7
810
812
P0
P0-0
P0-7
GU1
B8
B1
VCC
VOUT
VSN
VSL
AGND
DGND
CE
CS
/DAC-LCD
+26V
R20
R24
M26
PHOTO
NEIGH2-RXD
NEIGH1-RXD
CH10-RXD
CH9-RXD
CH8-RXD
CH1-RXD
MAIN-RXD
REPEAT-RXD
P1-7
P1-4
A3
A0
V+
V-
GND
EN
OUT
F
A
9
0
GU8
CR2
C3
R18
R21
DACOUT
RXDATA
R22
MUXRXD
GU2
C38
GU15
IN
OUT
ADJ
R46
R47
C37
RXD
TXD

FIG. 14D.

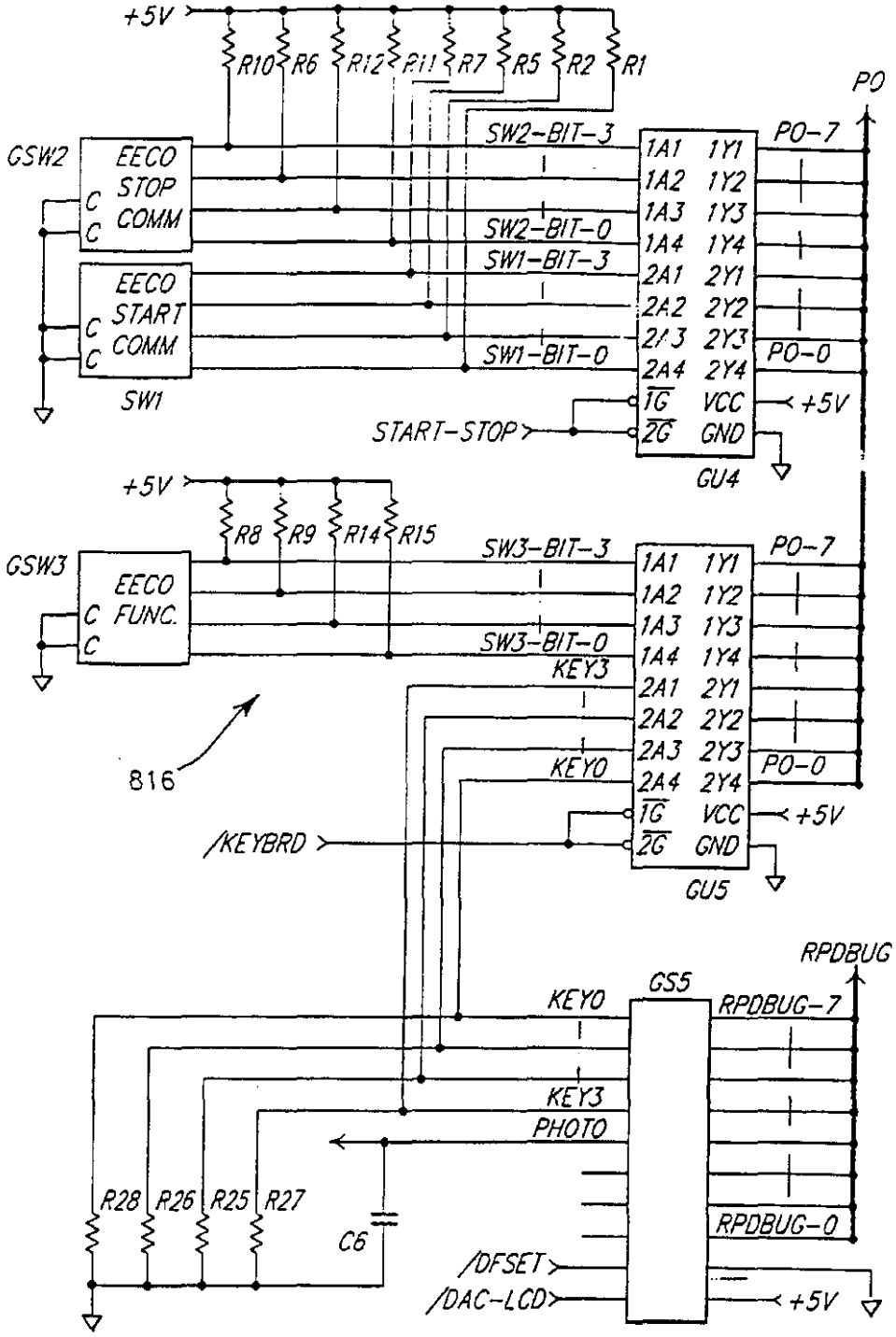

+5V
R10
R6
R12
R11
R7
R5
R2
R1
PO
GSW2
EECO
STOP
COMM
EECO
START
COMM
SW1
SW2-BIT-3
SW2-BIT-0
SW1-BIT-3
SW1-BIT-0
1A1 1Y1
1A2 1Y2
1A3 1Y3
1A4 1Y4
2A1 2Y1
2A2 2Y2
2A3 2Y3
2A4 2Y4
1G VCC
2G GND
PO-7
PO-0
+5V
START-STOP
GU4
+5V
R8
R9
R14
R15
GSW3
EECO
FUNC.
SW3-BIT-3
SW3-BIT-0
KEY3
KEY0
PO-7
PO-0
+5V
816
/KEYBRD
GU5
RPDBUG
GS5
KEY0
KEY3
PHOTO
RPDBUG-7
RPDBUG-0
R28
R26
R25
R27
C6
/DFSET
/DAC-LCD
+5V

FIG. 14E.

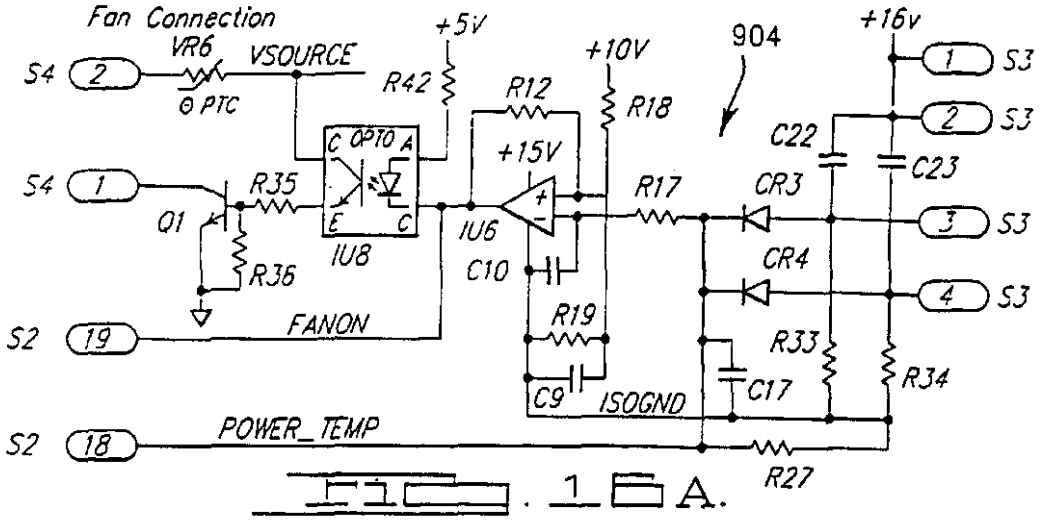


FIG. 16A.

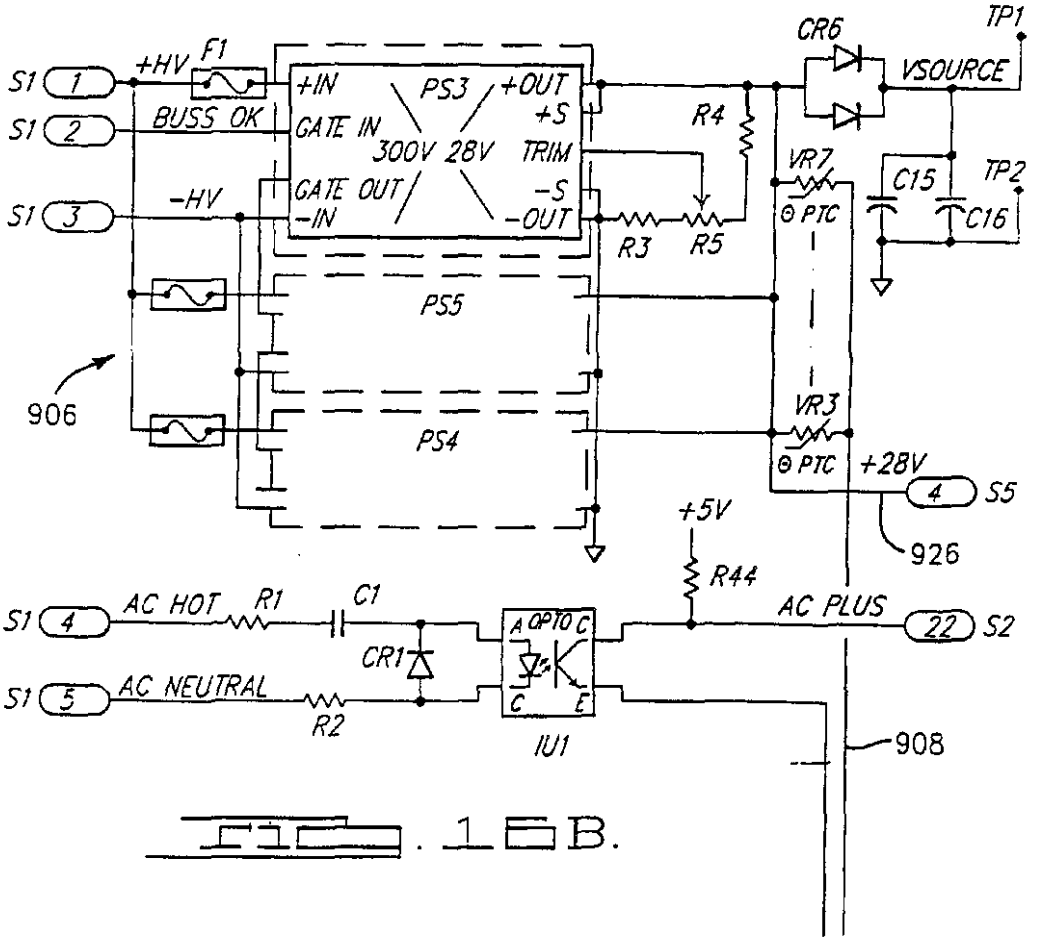


FIG. 16B.

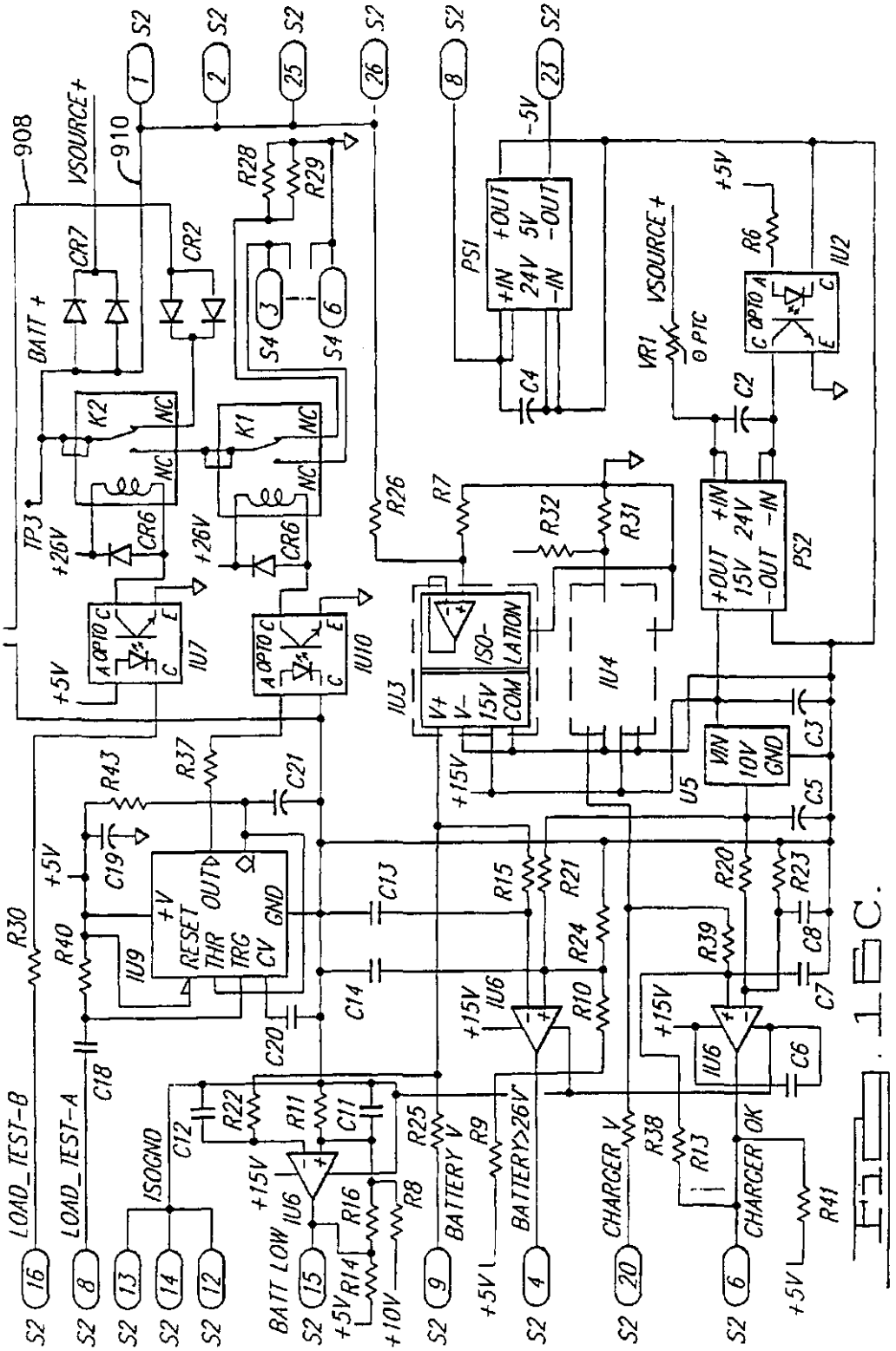

908
910
VSOURCE+
BATT +
CR7
CR2
K2
K1
NC
TP3
+26V
CR6
IU7
IU10
OPTO
+5V
R28
R29
S4
S2
R26
R7
R32
R31
IU3
ISO-LATION
V+
V-
15V
COM
IU4
+15V
PS1
+IN
24V
-IN
+OUT
5V
-OUT
-5V
C4
VR1
PTC
C2
R6
IU2
PS2
+OUT
15V
-OUT
+IN
24V
-IN
VIN
10V
GND
U5
C3
C5
R43
R37
C21
C19
R30
R40
IU9
+V
RESET
THR
TRG
CV
OUT
GND
C13
C14
C20
C18
R15
R21
R24
R10
IU6
R20
R23
R39
C8
C7
C6
LOAD_TEST-B
LOAD_TEST-A
ISOGND
C12
R22
R11
C11
R16
R14
R8
R25
R9
R38
R13
R41
BATT LOW
BATTERY V
BATTERY>26V
CHARGER V
CHARGER OK
+10V

FIG. 16C.

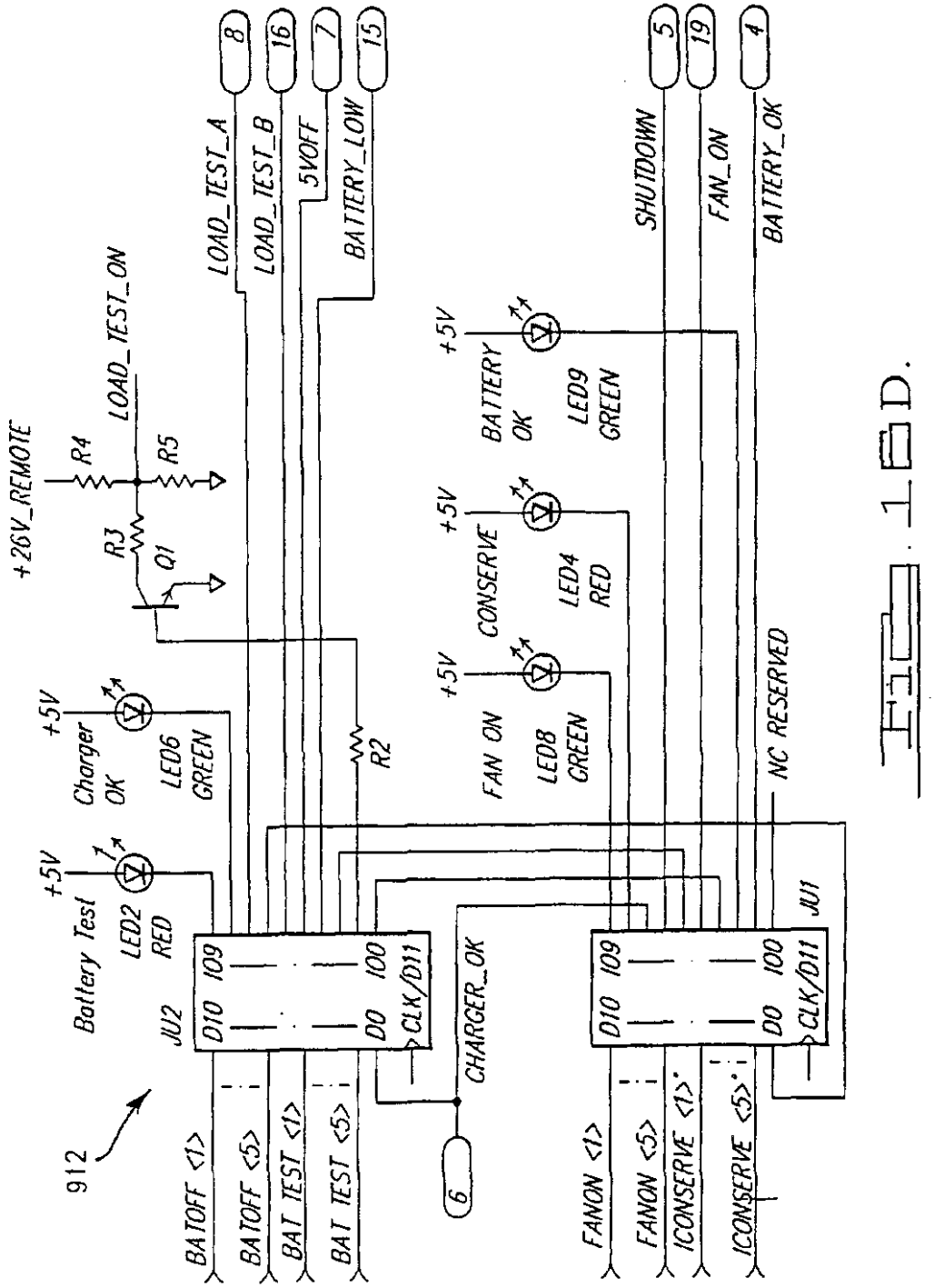

+26V_REMOTE
R4
LOAD_TEST_ON
R5
RJ
Q1
LOAD_TEST_A
LOAD_TEST_B
5VOFF
BATTERY_LOW
8
16
7
15
+5V
Charger OK
LED6 GREEN
R2
+5V
Battery Test
LED2 RED
JU2
D10 I09
D0 I00
CLK/D11
912
BATOFF <1>
BATOFF <5>
BAT TEST <1>
BAT TEST <5>
CHARGER_OK
6
+5V
FAN ON
LED8 GREEN
+5V
CONSERVE
LED4 RED
+5V
BATTERY OK
LED9 GREEN
SHUTDOWN
FAN_ON
BATTERY_OK
5
19
4
NC RESERVED
JU1
D10 I09
D0 I00
CLK/D11
FANON <1>
FANON <5>
ICONSERVE <1>'
ICONSERVE <5>'

Fig. 16D.

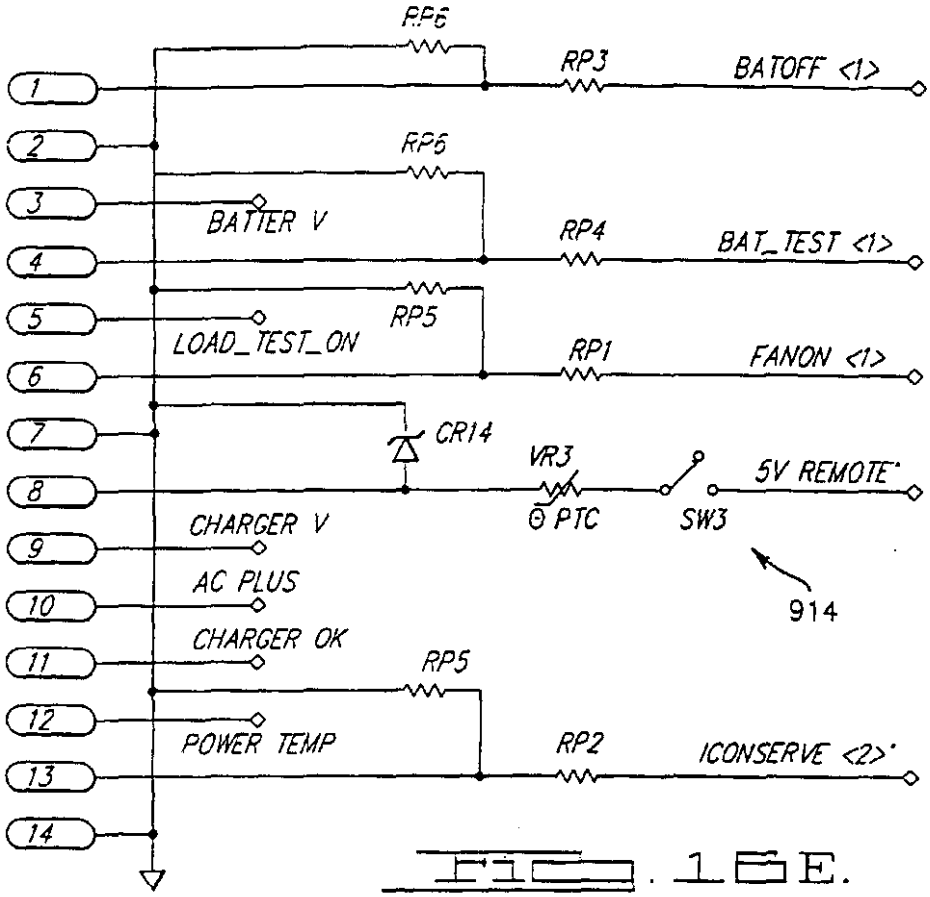


FIG. 15E.

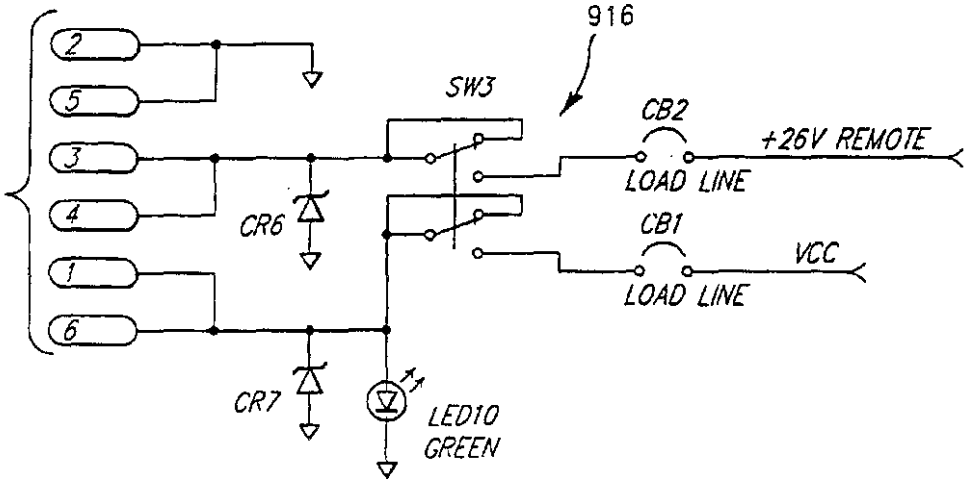


FIG. 15F.

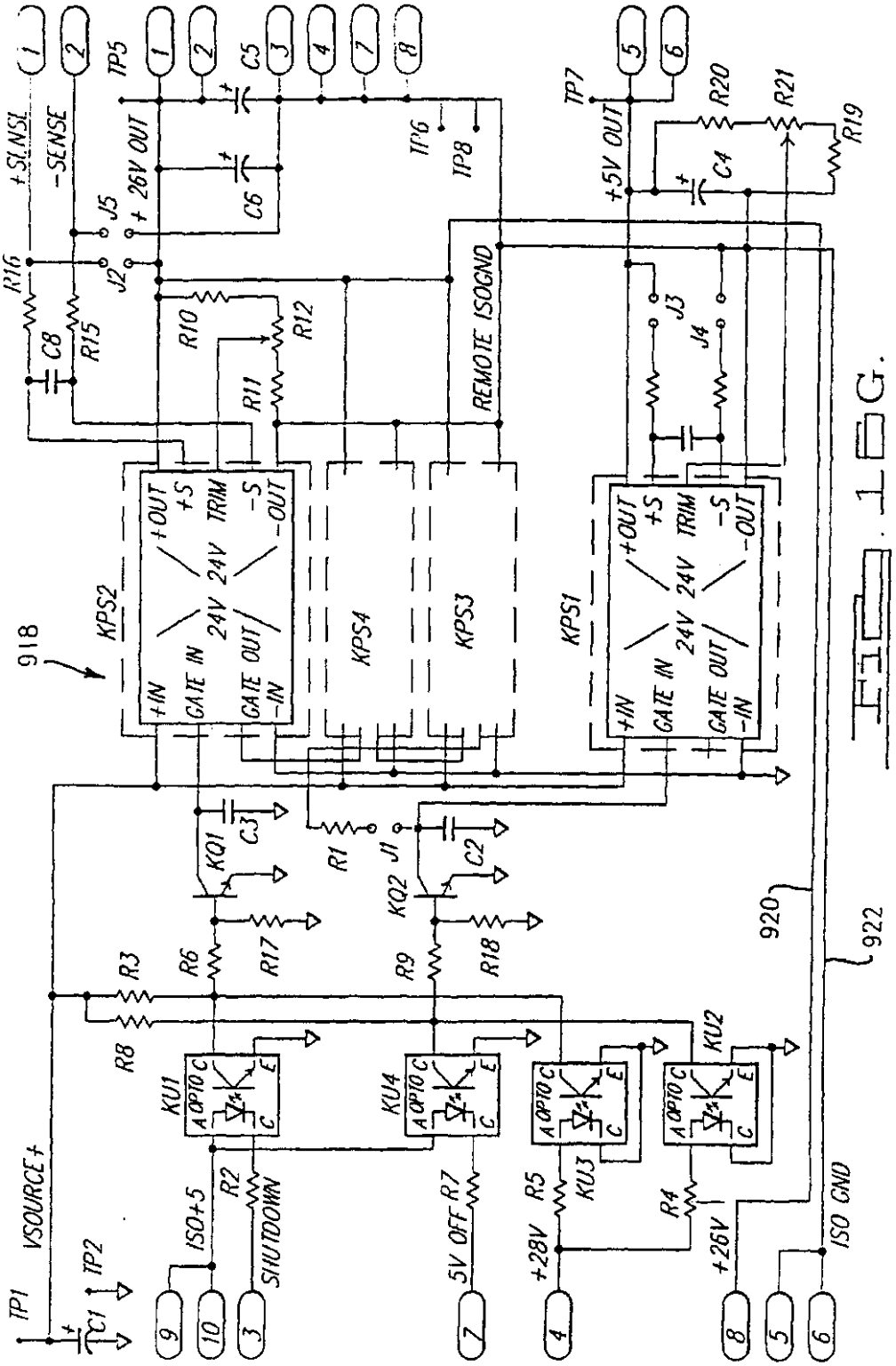

+SENSE
-SENSE
+26V OUT
TP5
TP6
TP8
TP7
+5V OUT
J5
J2
J3
J4
R16
R15
C8
R10
R11
R12
C5
C6
C4
R20
R21
R19
REMOTE ISOGND
KPS2
KPS4
KPS3
KPS1
+OUT
+S
TRIM
-S
-OUT
24V
+IN
GATE IN
GATE OUT
-IN
918
KQ1
KQ2
C3
C2
R1
J1
R3
R6
R17
R9
R18
R8
KU1
KU4
KU3
KU2
OPTO
A
C
E
ISO+5
R2
SHUTDOWN
5V OFF
R7
+28V
R5
R4
+26V
ISO GND
920
922
TP1
VSOURCE+
TP2
C1
9
10
3
7
4
8
5
6

FIG. 16

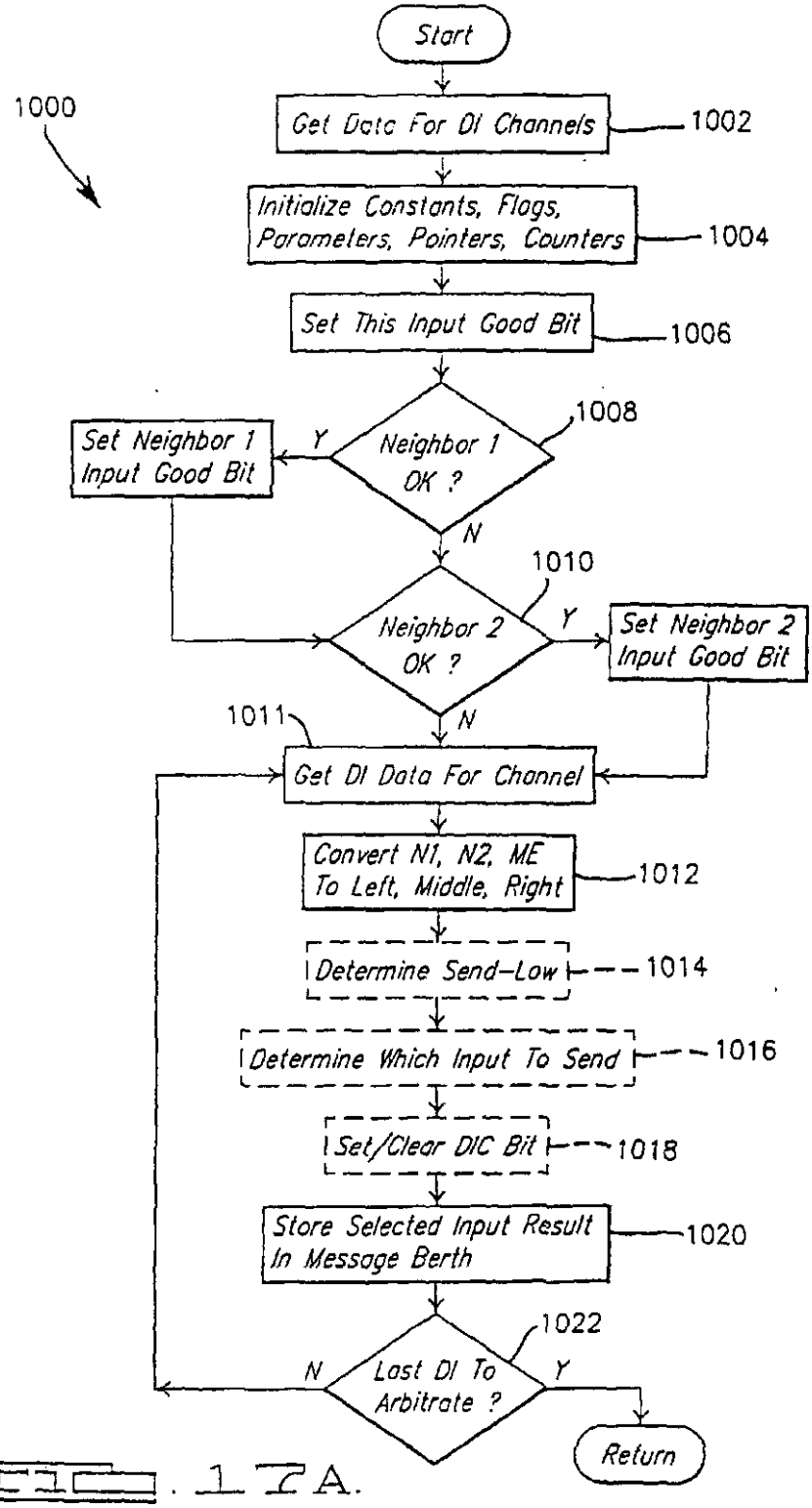

1000
Start
Get Data For DI Channels
1002
Initialize Constants, Flags, Parameters, Pointers, Counters
1004
Set This Input Good Bit
1006
Neighbor 1 OK ?
1008
Y
Set Neighbor 1 Input Good Bit
N
Neighbor 2 OK ?
1010
Y
Set Neighbor 2 Input Good Bit
N
1011
Get DI Data For Channel
Convert N1, N2, ME To Left, Middle, Right
1012
Determine Send-Low
1014
Determine Which Input To Send
1016
Set/Clear DIC Bit
1018
Store Selected Input Result In Message Berth
1020
Last DI To Arbitrate ?
1022
N
Y
Return

FIG. 17A.

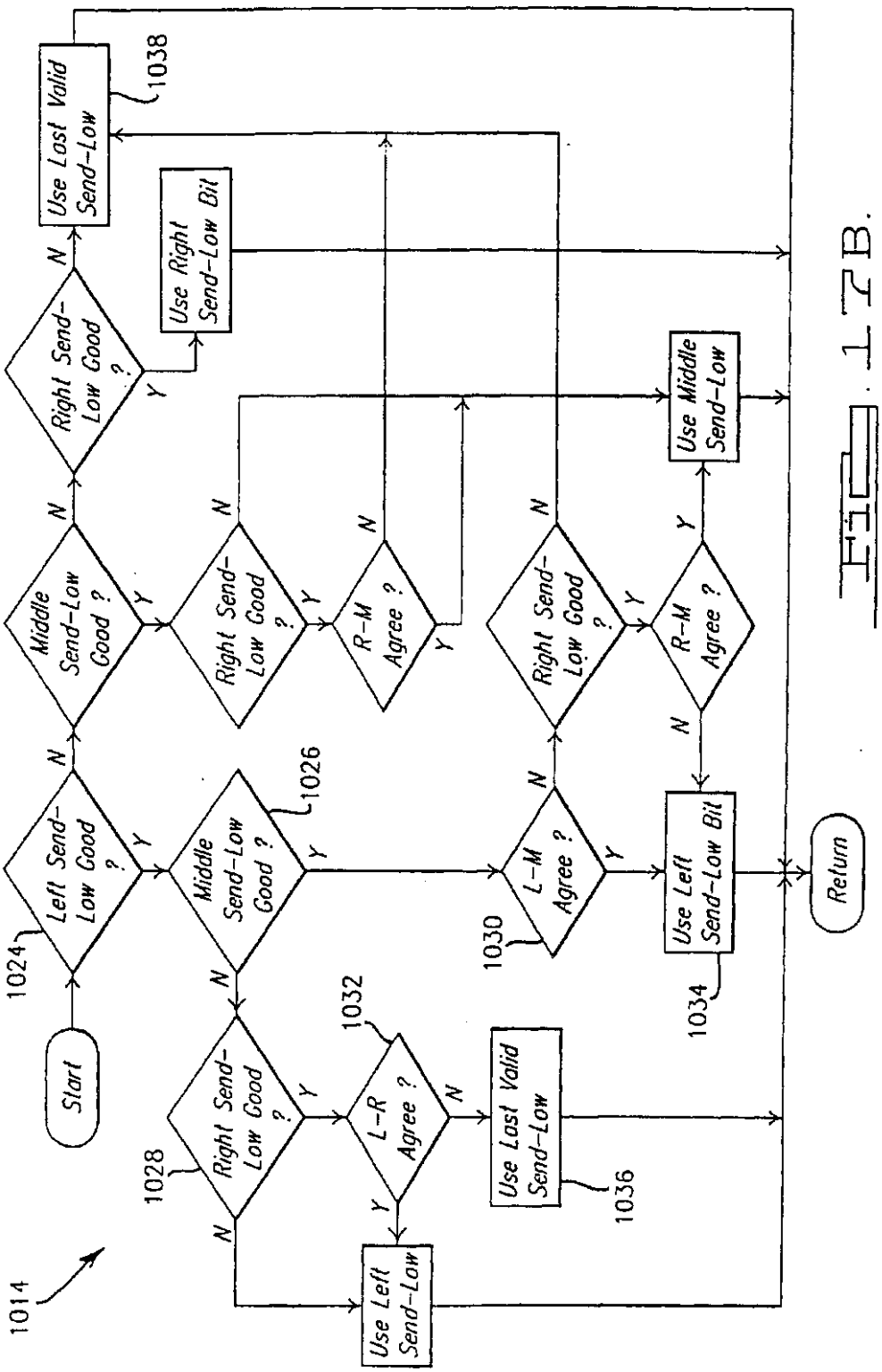

Start
Left Send-Low Good ?
Middle Send-Low Good ?
Right Send-Low Good ?
L-R Agree ?
Use Left Send-Low
Use Last Valid Send-Low
L-M Agree ?
Use Left Send-Low Bit
Right Send-Low Good ?
R-M Agree ?
Use Middle Send-Low
Middle Send-Low Good ?
Right Send-Low Good ?
R-M Agree ?
Use Right Send-Low Bit
Use Last Valid Send-Low
Return
1014
1024
1026
1028
1030
1032
1034
1036
1038
Y
N

Fig. 17B.

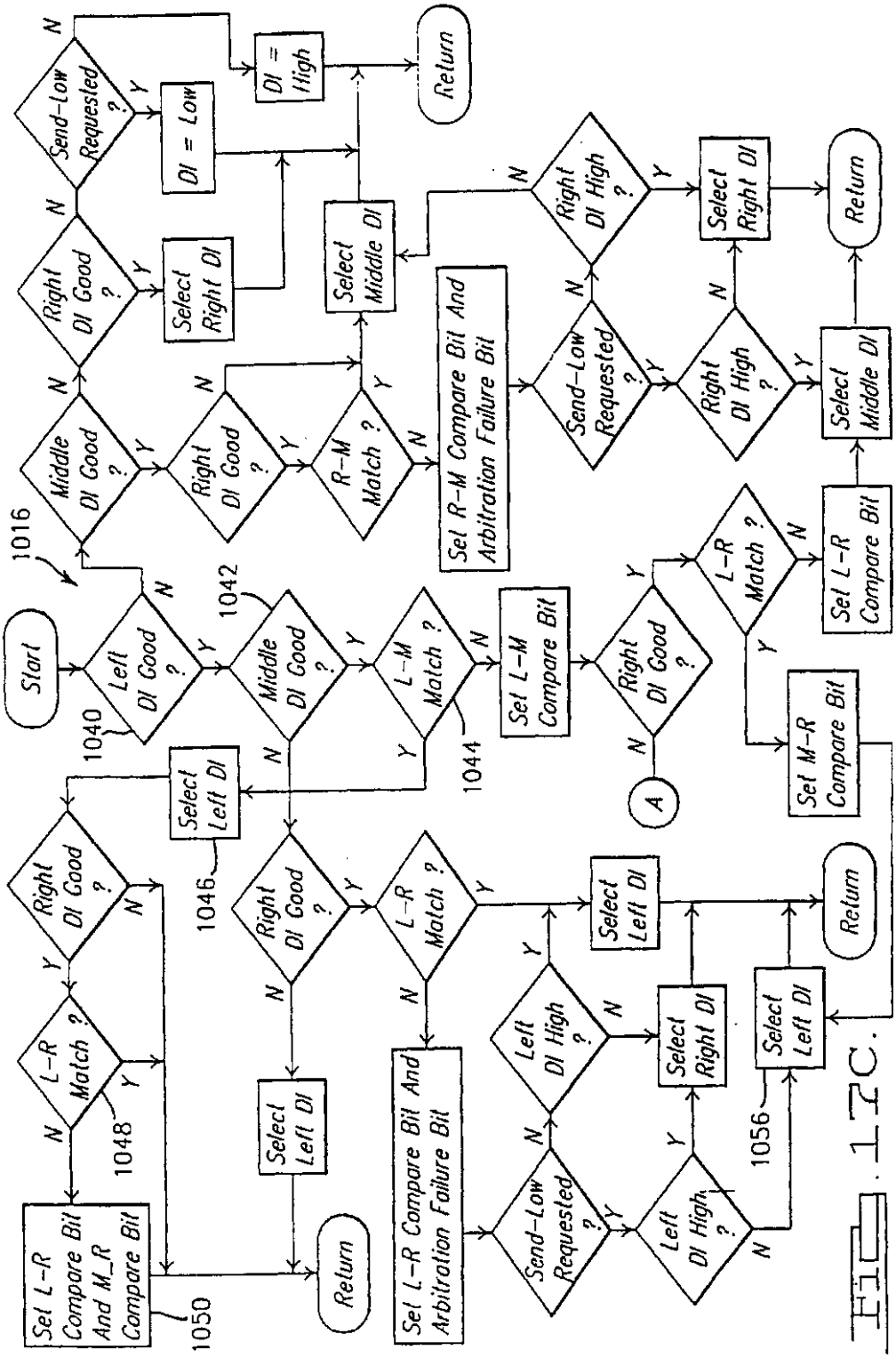

Start
1016
Left DI Good ?
1040
Middle DI Good ?
1042
L-M Match ?
1044
Set L-M Compare Bit
Right DI Good ?
A
L-R Match ?
Sel L-R Compare Bit
Sel M-R Compare Bit
Right DI Good ?
L-R Match ?
1046
Select Left DI
Return
Sel L-R Compare Bit And M_R Compare Bit
1048
1050
Sel L-R Compare Bit And Arbitration Failure Bit
Send-Low Requested ?
Left DI High ?
Select Right DI
Select Left DI
1056
Middle DI Good ?
Right DI Good ?
R-M Match ?
Sel R-M Compare Bit And Arbitration Failure Bit
Send-Low Requested ?
Right DI High ?
Select Middle DI
Select Right DI
DI = High
DI = Low
Send-Low Requested ?
Right DI Good ?
Middle DI Good ?

FIG. 17C.

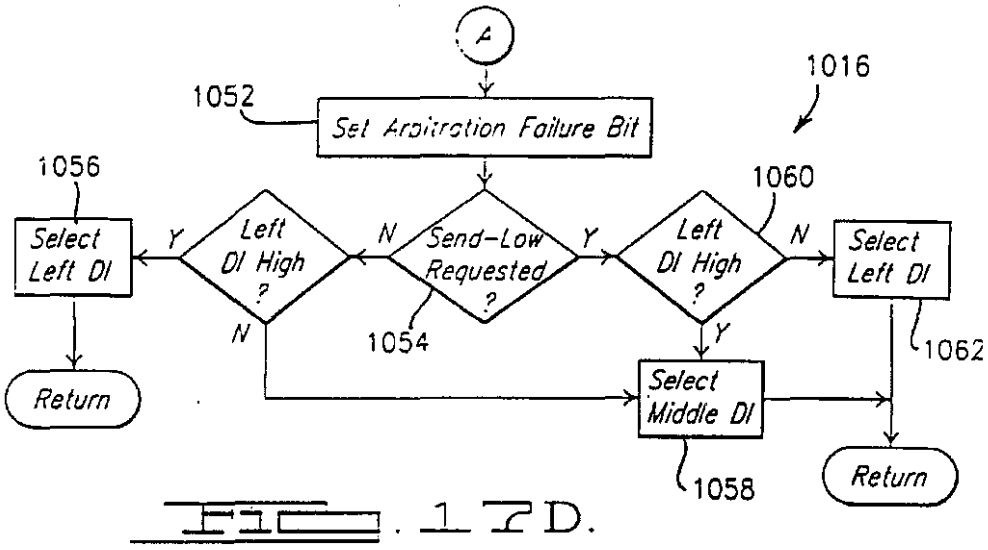


FIG. 17D.

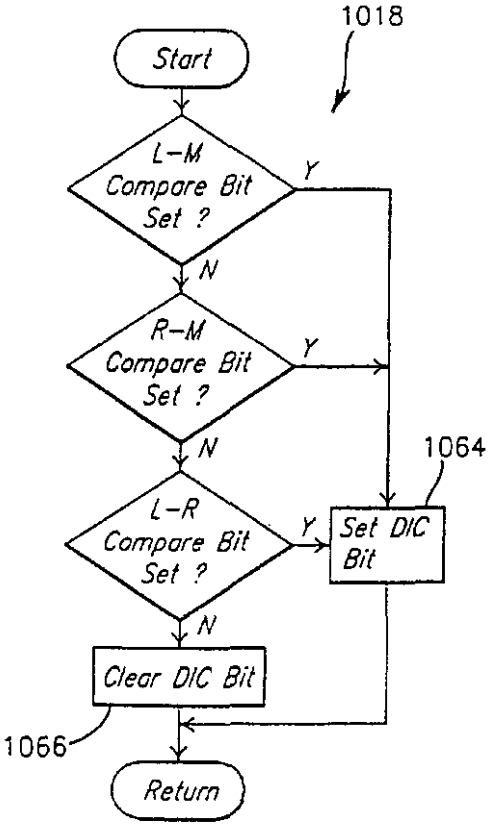


FIG. 17E.

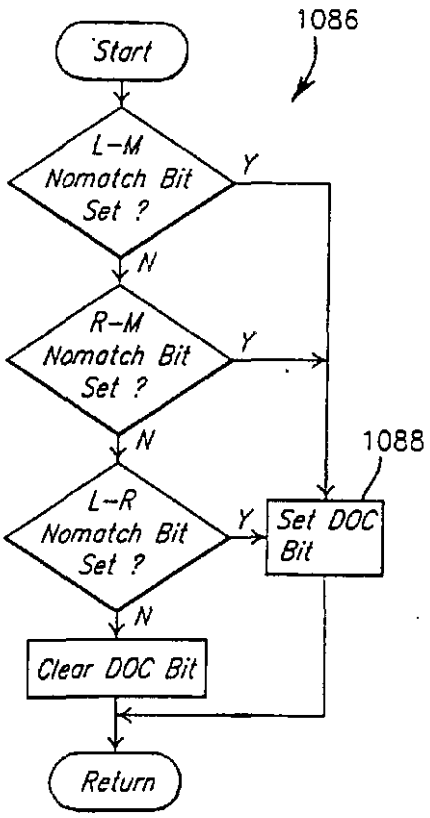


FIG. 17I.

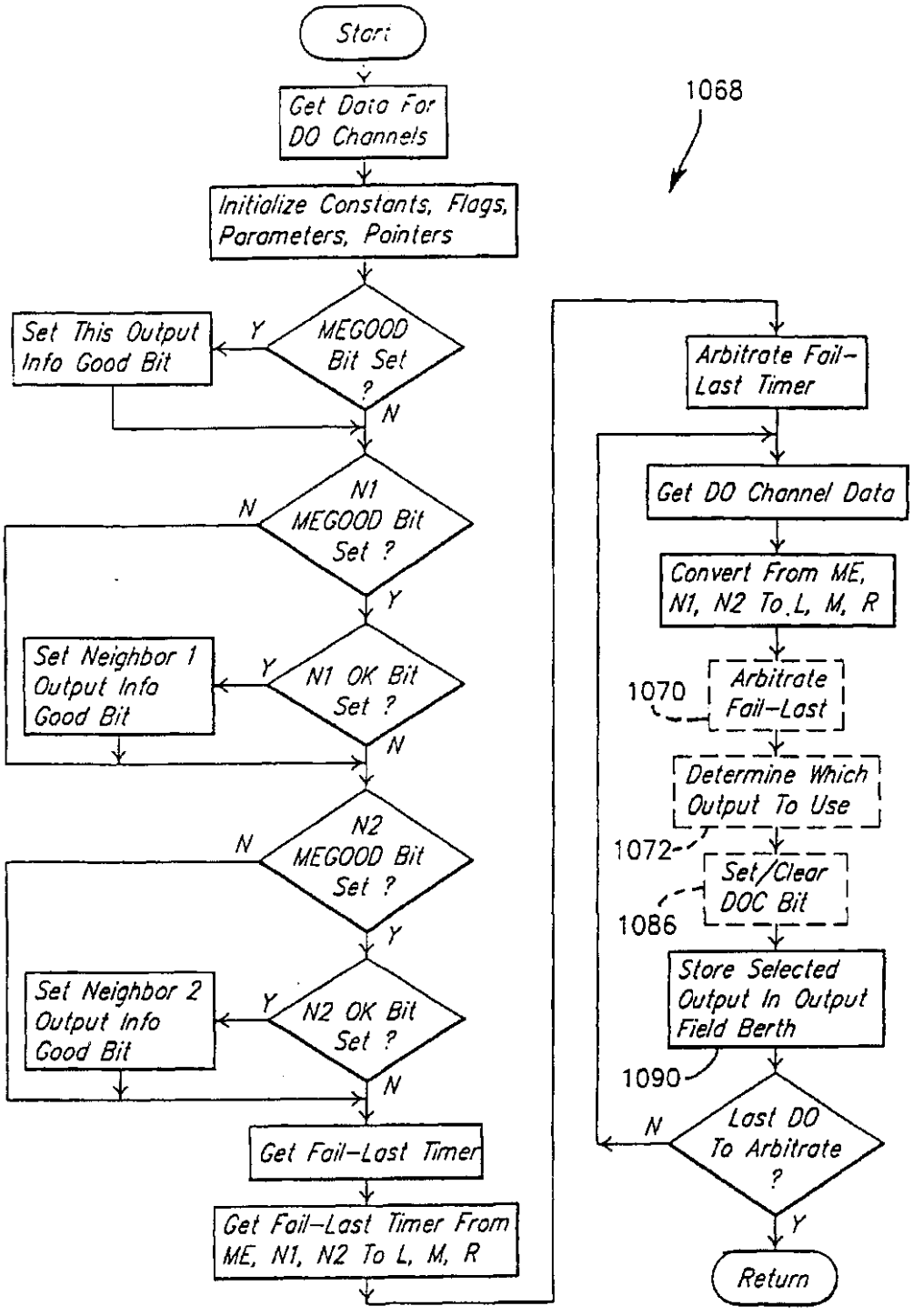

Start
Get Data For DO Channels
Initialize Constants, Flags, Parameters, Pointers
MEGOOD Bit Set ?
Y
Set This Output Info Good Bit
N
N1 MEGOOD Bit Set ?
N
Y
N1 OK Bit Set ?
Y
Set Neighbor 1 Output Info Good Bit
N
N2 MEGOOD Bit Set ?
N
Y
N2 OK Bit Set ?
Y
Set Neighbor 2 Output Info Good Bit
N
Get Fail-Last Timer
Get Fail-Last Timer From ME, N1, N2 To L, M, R
1068
Arbitrate Fail-Last Timer
Get DO Channel Data
Convert From ME, N1, N2 To L, M, R
1070
Arbitrate Fail-Last
Determine Which Output To Use
1072
Set/Clear DOC Bit
1086
Store Selected Output In Output Field Berth
1090
Last DO To Arbitrate ?
N
Y
Return

FIG. 17F.

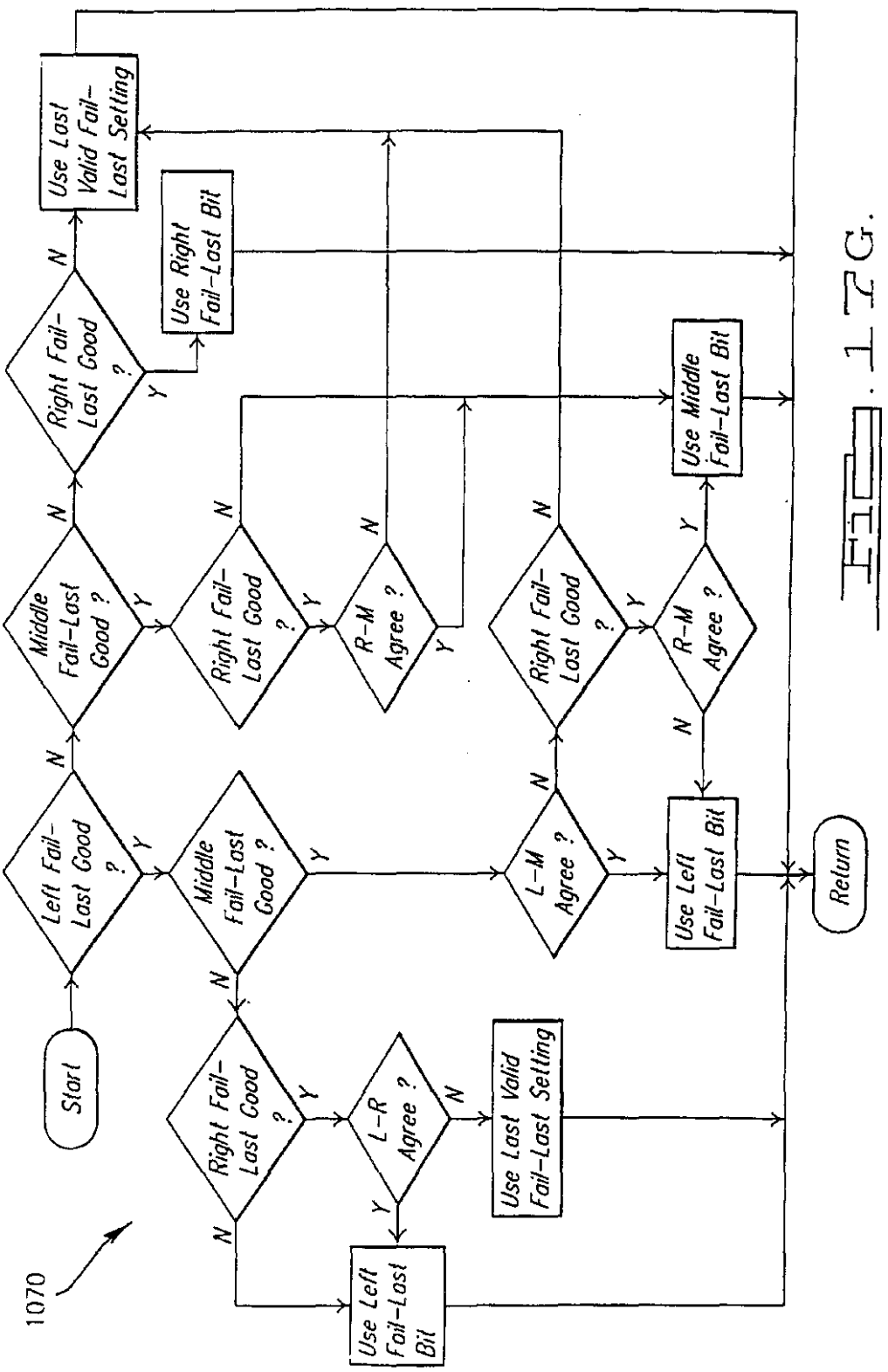

1070
Start
Left Fail-Last Good ?
Middle Fail-Last Good ?
Right Fail-Last Good ?
L-R Agree ?
Use Left Fail-Last Bit
Use Last Valid Fail-Last Setting
L-M Agree ?
Use Left Fail-Last Bit
Middle Fail-Last Good ?
Right Fail-Last Good ?
R-M Agree ?
Use Middle Fail-Last Bit
Right Fail-Last Good ?
R-M Agree ?
Use Middle Fail-Last Bit
Right Fail-Last Good ?
Use Right Fail-Last Bit
Use Last Valid Fail-Last Setting
Return
Y
N

Fig. 17G.

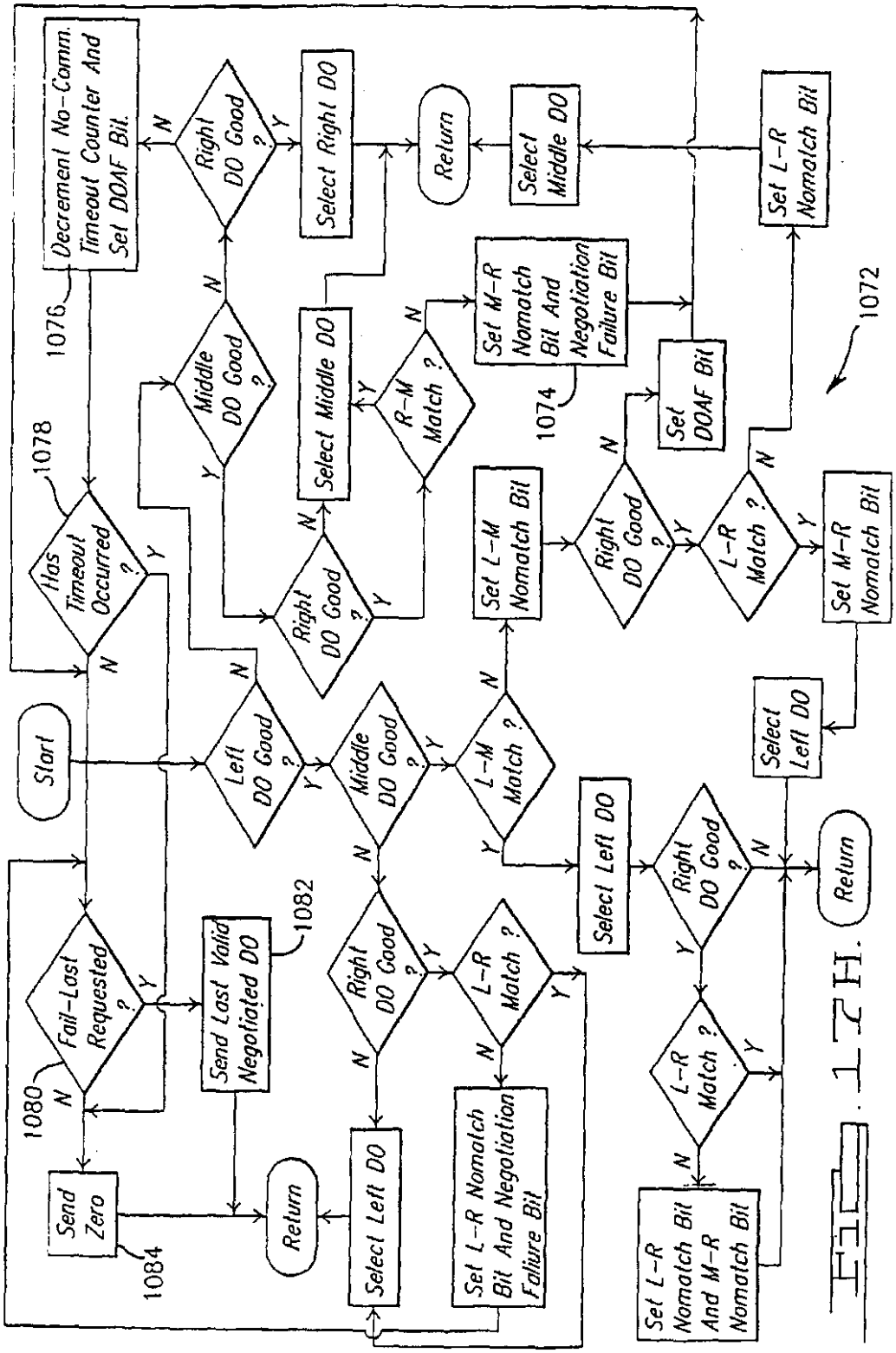

Start
Left DO Good ?
Middle DO Good ?
Right DO Good ?
L-M Match ?
L-R Match ?
Has Timeout Occurred ?
Fail-Last Requested ?
Send Last Valid Negotiated DO
Send Zero
Return
Select Left DO
Select Middle DO
Select Right DO
Set L-R Nomatch Bit And Negotiation Failure Bit
Set L-M Nomatch Bit
Set M-R Nomatch Bit
Set M-R Nomatch Bit And Negotiation Failure Bit
Set L-R Nomatch Bit And M-R Nomatch Bit
Set DOAF Bit
Decrement No-Comm. Timeout Counter And Set DOAF Bit.
1072
1074
1076
1078
1080
1082
1084

FIG. 17H.

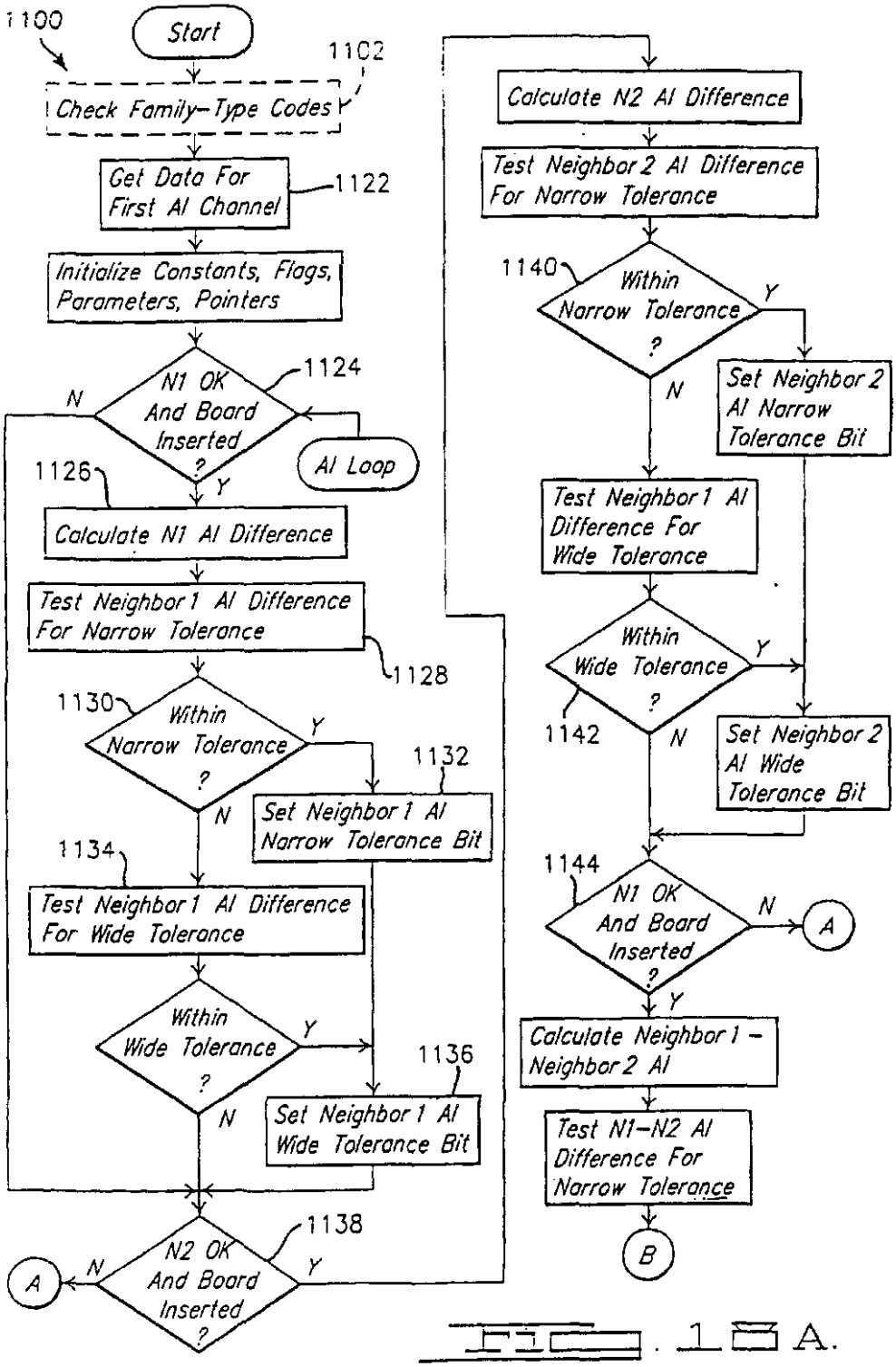

1100
Start
1102
Check Family-Type Codes
Get Data For First AI Channel
1122
Initialize Constants, Flags, Parameters, Pointers
1124
N1 OK And Board Inserted ?
N
Y
AI Loop
1126
Calculate N1 AI Difference
Test Neighbor 1 AI Difference For Narrow Tolerance
1128
1130
Within Narrow Tolerance ?
Y
N
1132
Set Neighbor 1 AI Narrow Tolerance Bit
1134
Test Neighbor 1 AI Difference For Wide Tolerance
Within Wide Tolerance ?
Y
N
1136
Set Neighbor 1 AI Wide Tolerance Bit
1138
N2 OK And Board Inserted ?
N
Y
A
Calculate N2 AI Difference
Test Neighbor 2 AI Difference For Narrow Tolerance
1140
Within Narrow Tolerance ?
Y
N
Set Neighbor 2 AI Narrow Tolerance Bit
Test Neighbor 1 AI Difference For Wide Tolerance
Within Wide Tolerance ?
Y
1142
N
Set Neighbor 2 AI Wide Tolerance Bit
1144
N1 OK And Board Inserted ?
N
A
Y
Calculate Neighbor 1 - Neighbor 2 AI
Test N1-N2 AI Difference For Narrow Tolerance
B

FIG. 1A.

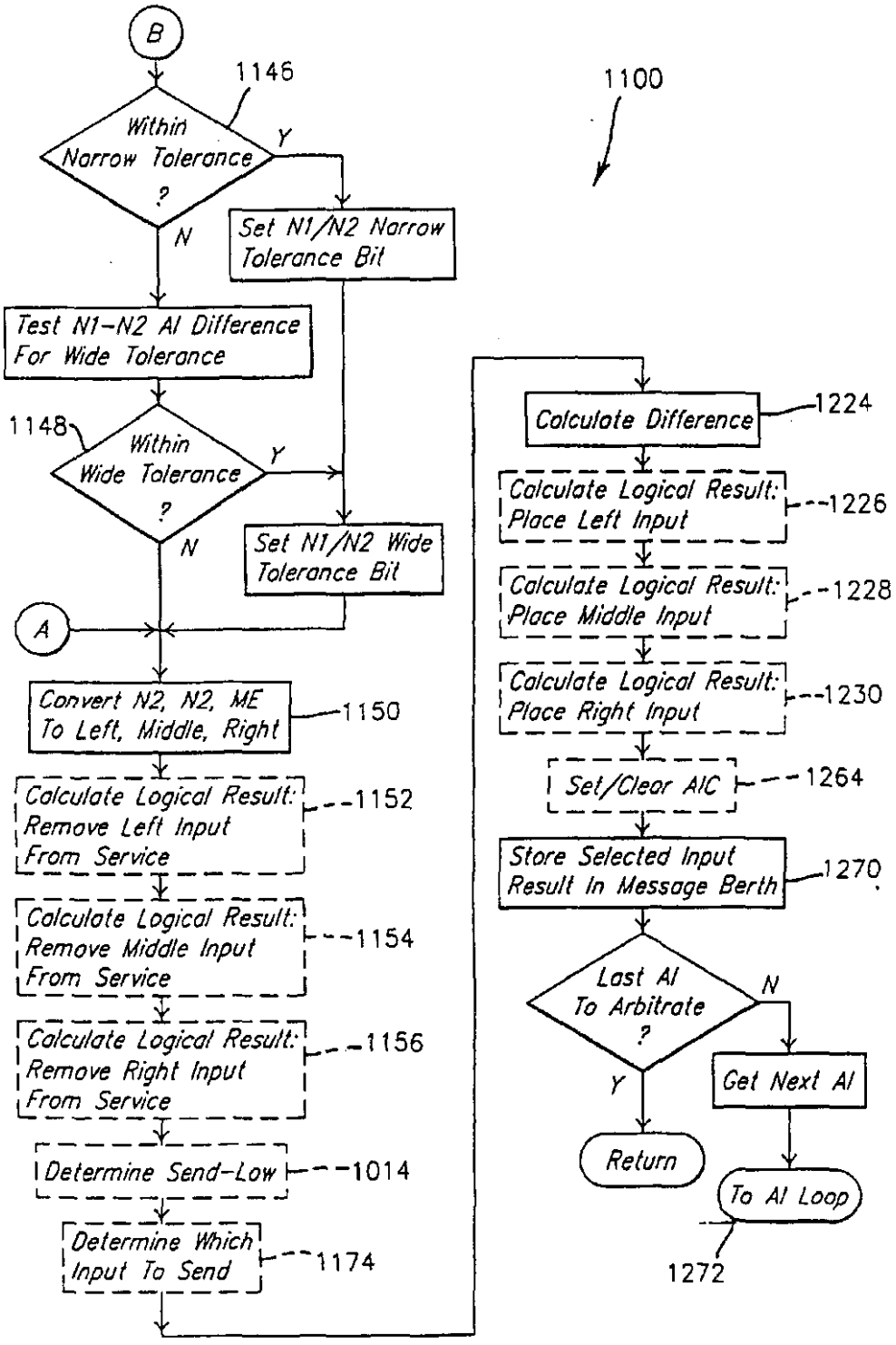

B
1146
Within Narrow Tolerance ?
Y
N
Set N1/N2 Narrow Tolerance Bit
Test N1-N2 AI Difference For Wide Tolerance
1148
Within Wide Tolerance ?
Y
N
Set N1/N2 Wide Tolerance Bit
A
Convert N2, N2, ME To Left, Middle, Right
1150
Calculate Logical Result: Remove Left Input From Service
1152
Calculate Logical Result: Remove Middle Input From Service
1154
Calculate Logical Result: Remove Right Input From Service
1156
Determine Send-Low
1014
Determine Which Input To Send
1174
1100
Calculate Difference
1224
Calculate Logical Result: Place Left Input
1226
Calculate Logical Result: Place Middle Input
1228
Calculate Logical Result: Place Right Input
1230
Set/Clear AIC
1264
Store Selected Input Result In Message Berth
1270
Last AI To Arbitrate ?
N
Y
Get Next AI
Return
To AI Loop
1272

FIG. 1B.

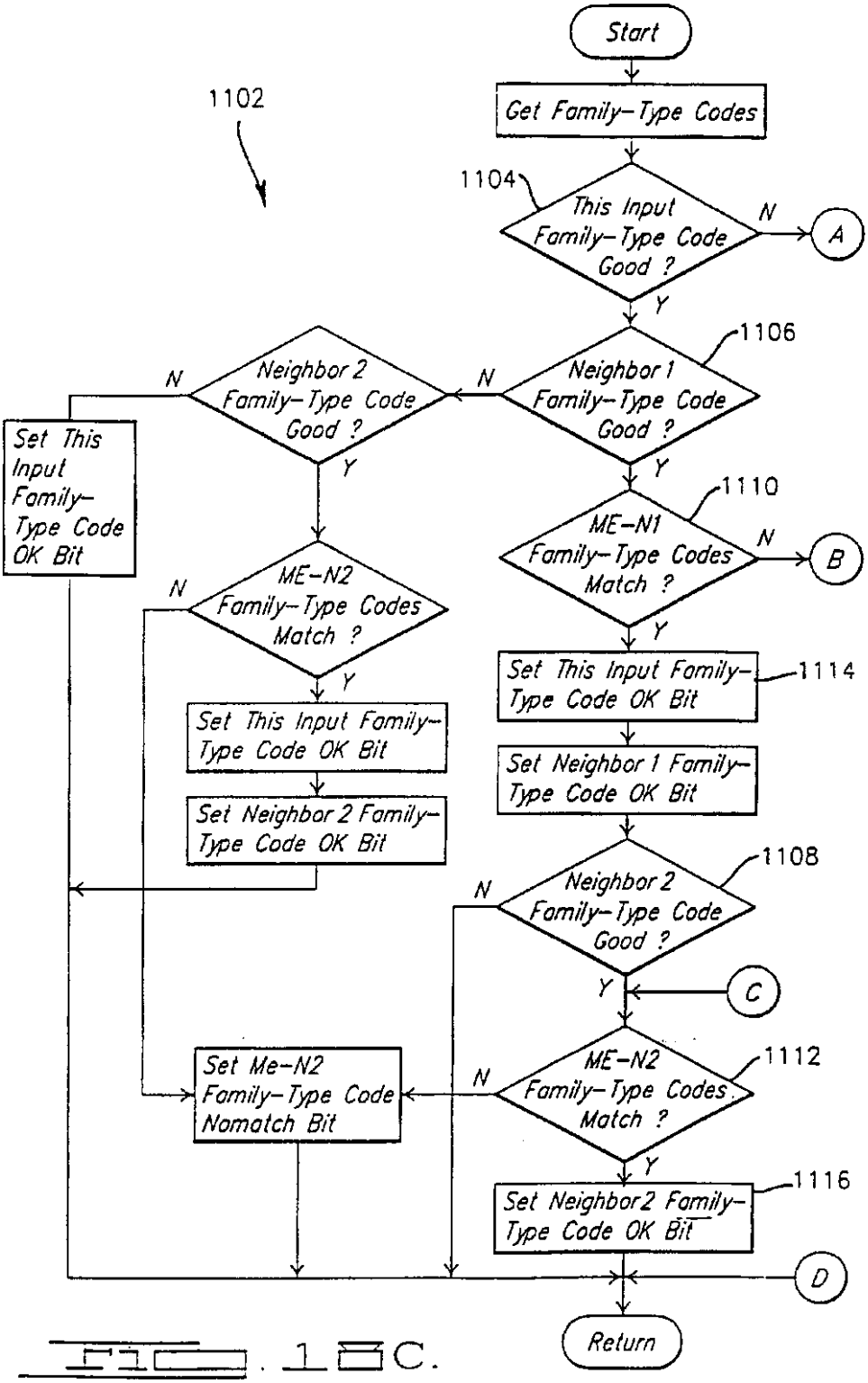

Start
1102
Get Family-Type Codes
1104
This Input Family-Type Code Good ?
N
A
Y
1106
Neighbor 1 Family-Type Code Good ?
N
Neighbor 2 Family-Type Code Good ?
N
Set This Input Family-Type Code OK Bit
Y
Y
1110
ME-N1 Family-Type Codes Match ?
N
B
Y
ME-N2 Family-Type Codes Match ?
N
Y
Set This Input Family-Type Code OK Bit
1114
Set This Input Family-Type Code OK Bit
Set Neighbor 1 Family-Type Code OK Bit
Set Neighbor 2 Family-Type Code OK Bit
1108
Neighbor 2 Family-Type Code Good ?
N
Y
C
Set Me-N2 Family-Type Code Nomatch Bit
ME-N2 Family-Type Codes Match ?
1112
N
Y
1116
Set Neighbor2 Family-Type Code OK Bit
D
Return

FIG. 18C.

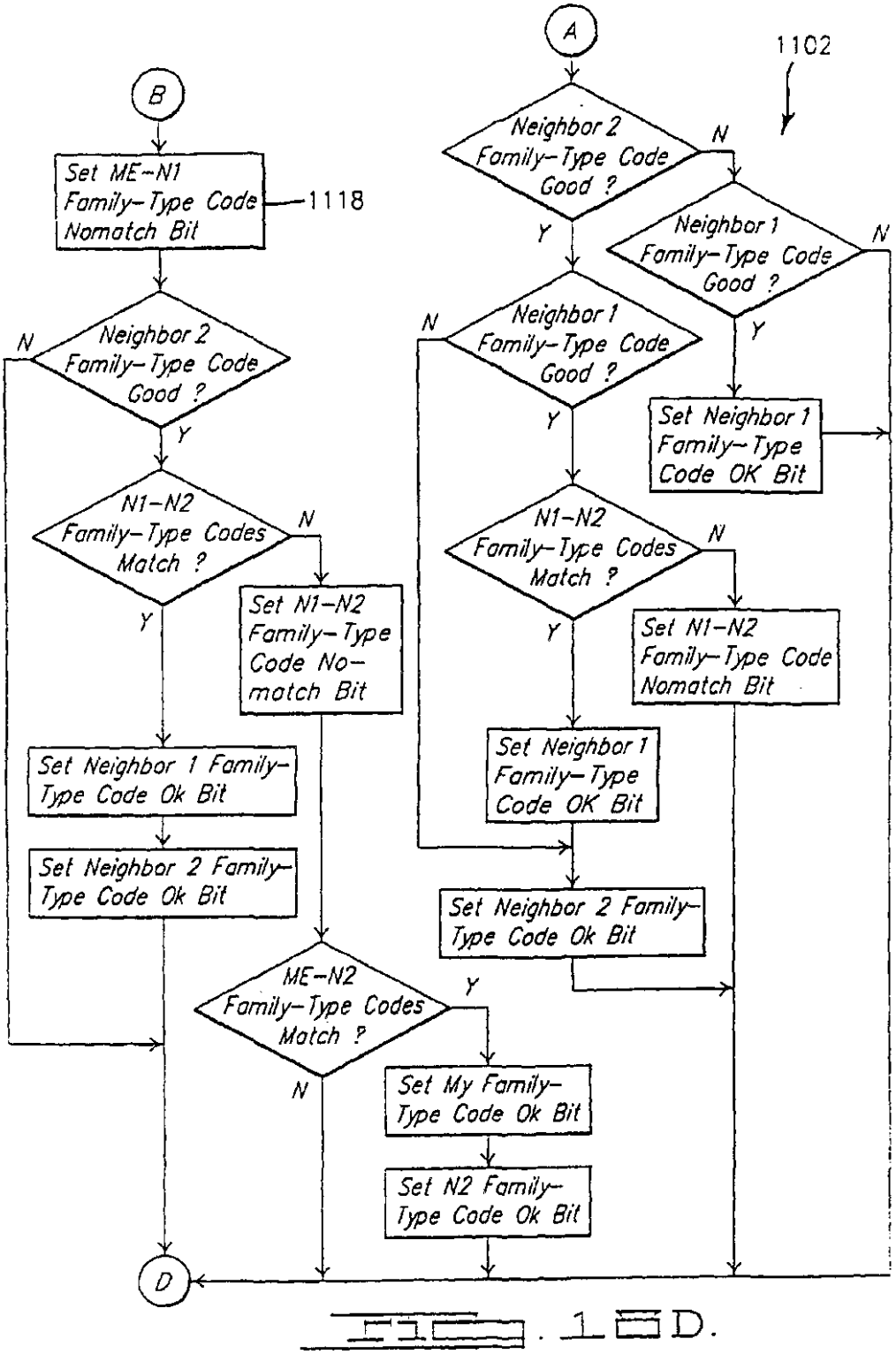

A
B
D
1102
1118
Neighbor 2 Family-Type Code Good ?
Neighbor 1 Family-Type Code Good ?
Set ME-N1 Family-Type Code Nomatch Bit
Set Neighbor 1 Family-Type Code OK Bit
N1-N2 Family-Type Codes Match ?
Set N1-N2 Family-Type Code No-match Bit
Set N1-N2 Family-Type Code Nomatch Bit
Set Neighbor 1 Family-Type Code Ok Bit
Set Neighbor 2 Family-Type Code Ok Bit
ME-N2 Family-Type Codes Match ?
Set My Family-Type Code Ok Bit
Set N2 Family-Type Code Ok Bit
Y
N

FIG. 18D.

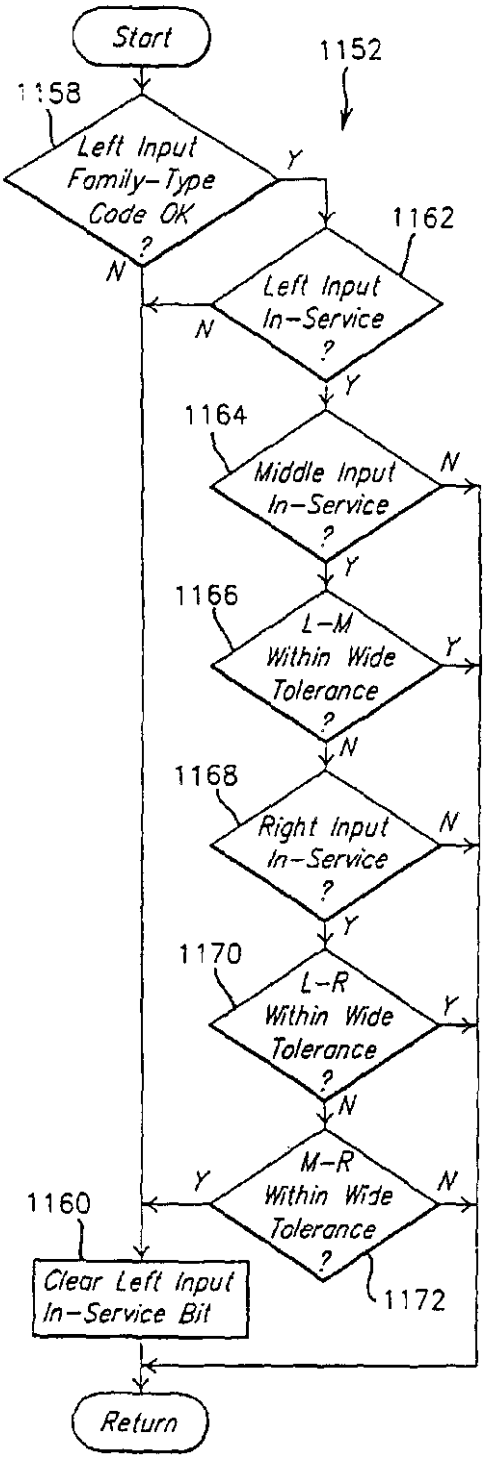


Fig. 18E.

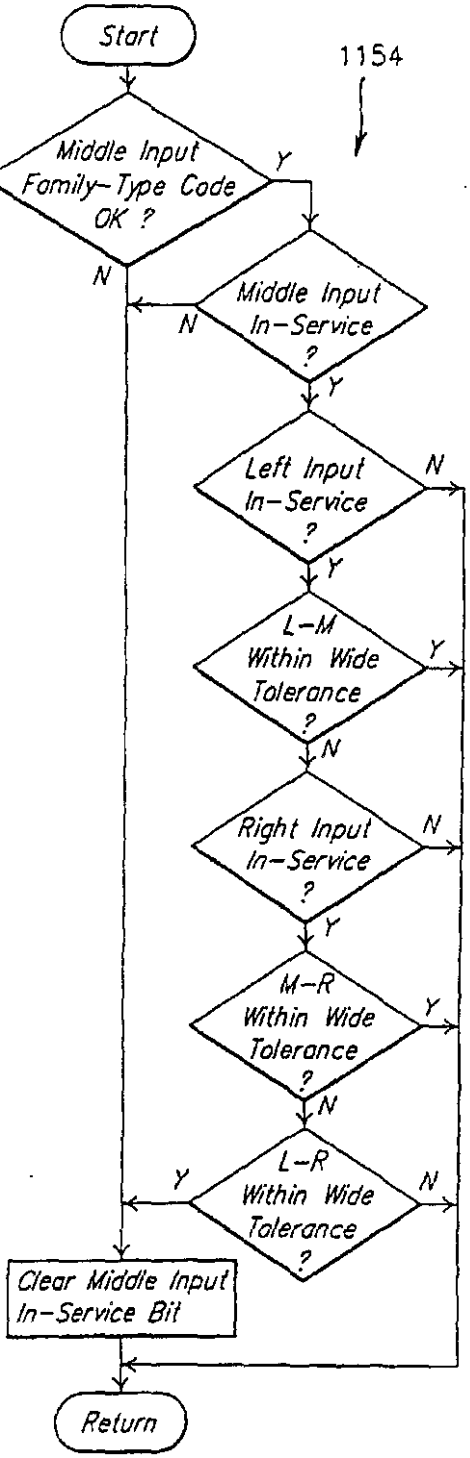


Fig. 18F.

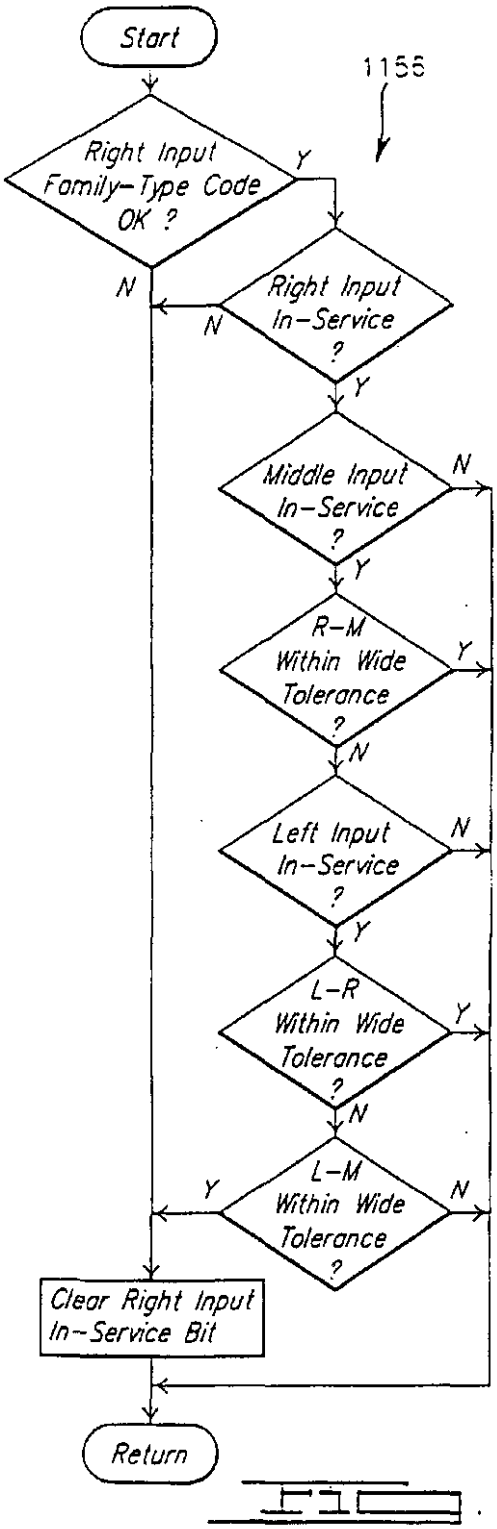


FIG. 16G.

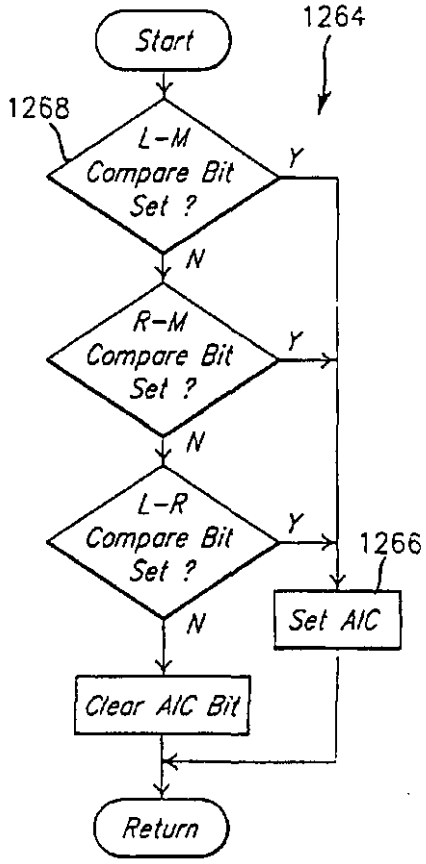
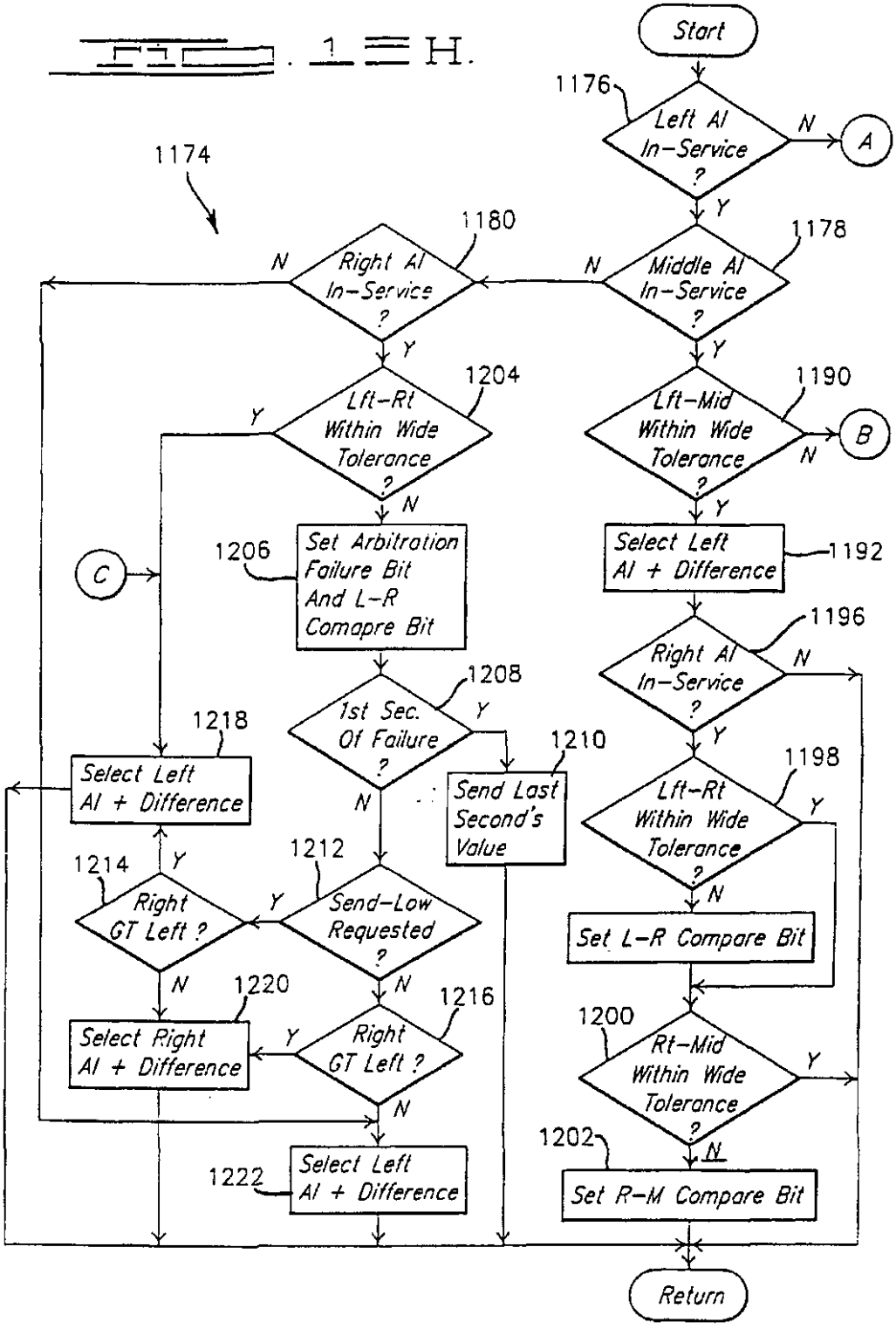


FIG. 16N.

FIG. 13H.


Start
1176
Left AI In-Service ?
N
A
Y
1174
1180
Right AI In-Service ?
N
1178
Middle AI In-Service ?
N
Y
Y
1204
Lft-Rt Within Wide Tolerance ?
Y
N
1190
Lft-Mid Within Wide Tolerance ?
N
B
Y
C
1206
Set Arbitration Failure Bit And L-R Comapre Bit
Select Left AI + Difference
1192
1196
Right AI In-Service ?
N
Y
1208
1st Sec. Of Failure ?
Y
N
1218
Select Left AI + Difference
1210
Send Last Second's Value
1198
Lft-Rt Within Wide Tolerance ?
Y
N
1214
Right GT Left ?
Y
N
1212
Send-Low Requested ?
Y
N
Set L-R Compare Bit
1220
Select Right AI + Difference
1216
Right GT Left ?
Y
N
1200
Rt-Mid Within Wide Tolerance ?
Y
N
1202
Set R-M Compare Bit
1222
Select Left AI + Difference
Return

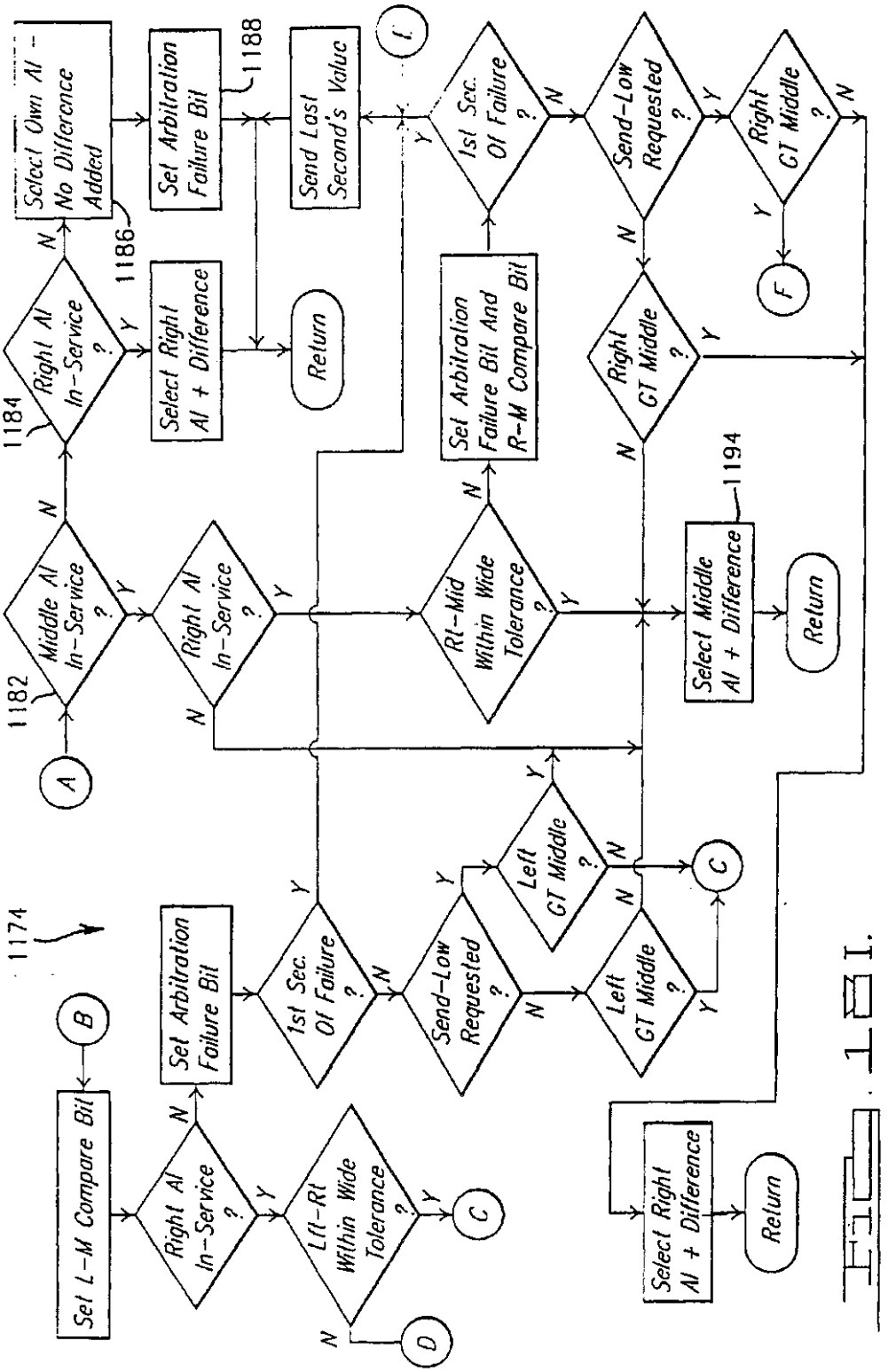

1174
Set L-M Compare Bit
B
Right Al In-Service ?
Lft-Rt Within Wide Tolerance ?
C
D
Set Arbitration Failure Bit
1st Sec. Of Failure ?
Send-Low Requested ?
Left GT Middle ?
Left GT Middle ?
C
Select Right Al + Difference
Return
A
1182
Middle Al In-Service ?
Right Al In-Service ?
Rt-Mid Within Wide Tolerance ?
Select Middle Al + Difference
1194
Return
1184
Right Al In-Service ?
Select Right Al + Difference
Return
1186
Select Own Al - No Difference Added
Set Arbitration Failure Bit
1188
Send Last Second's Value
E
1st Sec. Of Failure ?
Set Arbitration Failure Bit And R-M Compare Bit
Send-Low Requested ?
Right GT Middle ?
Right GT Middle ?
F
Y
N

FIG. 18I.

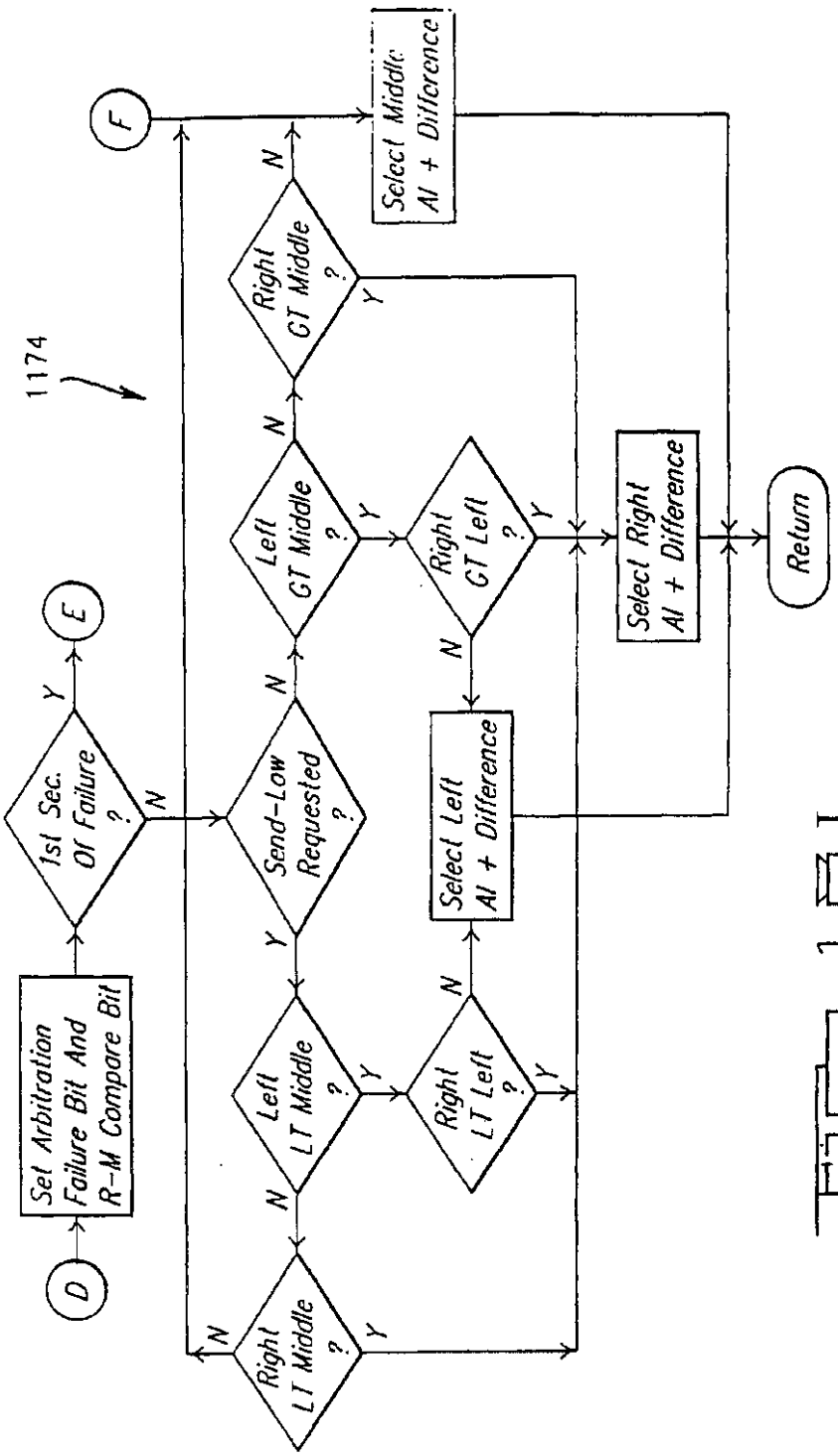

1174
D
Set Arbitration Failure Bit And R-M Compare Bit
1st Sec. Of Failure ?
Y
E
N
Send-Low Requested ?
Y
N
Left LT Middle ?
Y
N
Right LT Left ?
Y
N
Select Left Al + Difference
Right LT Middle ?
Y
N
Left GT Middle ?
Y
N
Right GT Left ?
Y
N
Select Right Al + Difference
Right GT Middle ?
Y
N
F
Select Middle Al + Difference
Return

Fig. 1BJ.

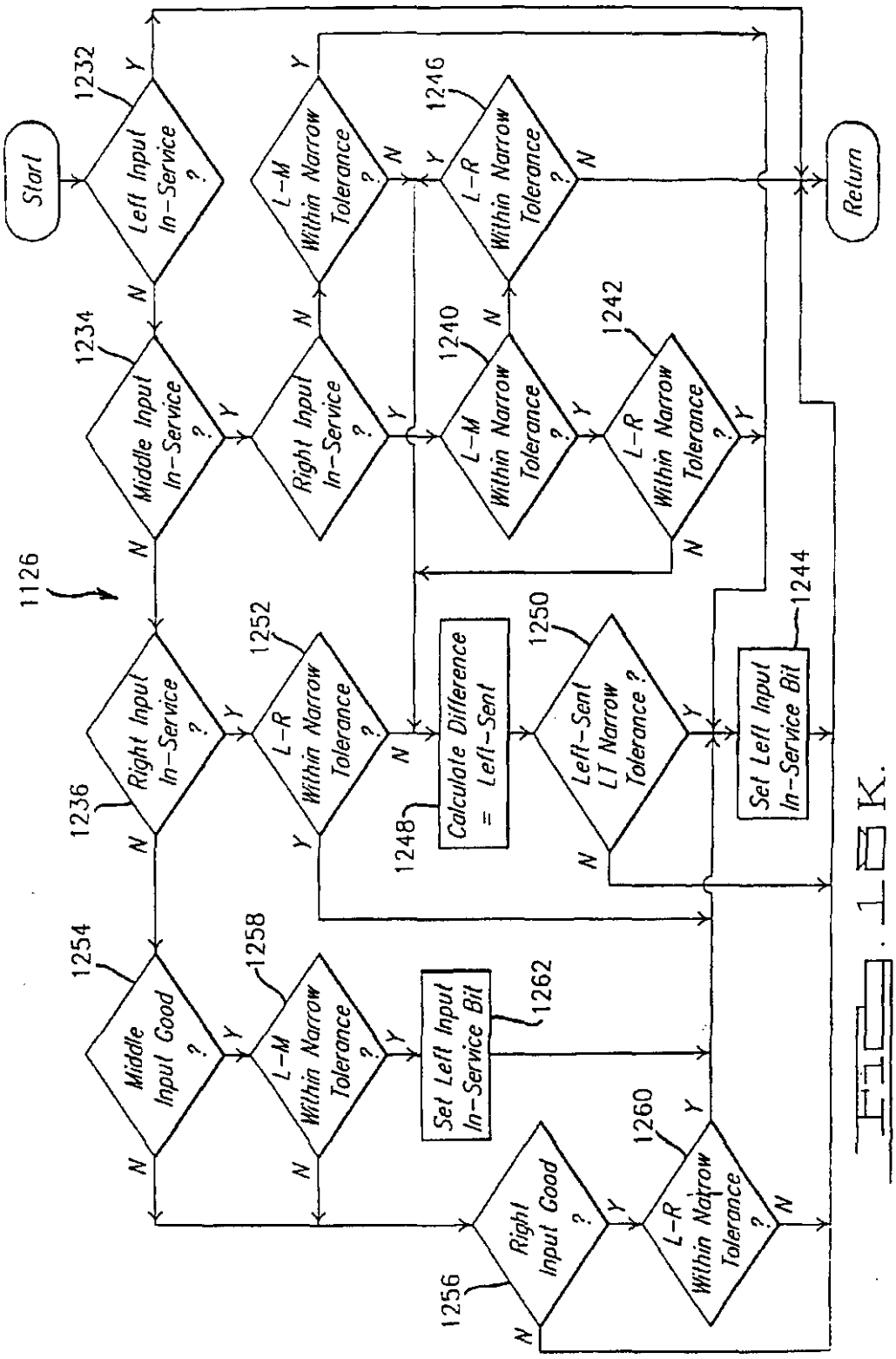

Start
Return
1126
1232
Left Input In-Service ?
1234
Middle Input In-Service ?
1236
Right Input In-Service ?
1254
Middle Input Good ?
1256
Right Input Good ?
1258
L-M Within Narrow Tolerance ?
1260
L-R Within Narrow Tolerance ?
1262
Set Left Input In-Service Bit
L-M Within Narrow Tolerance ?
1240
L-M Within Narrow Tolerance ?
1246
L-R Within Narrow Tolerance ?
Right Input In-Service ?
1242
L-R Within Narrow Tolerance ?
1252
L-R Within Narrow Tolerance ?
1248
Calculate Difference = Left-Sent
1250
Left-Sent LT Narrow Tolerance ?
1244
Set Left Input In-Service Bit
Y
N

FIG. 1K.

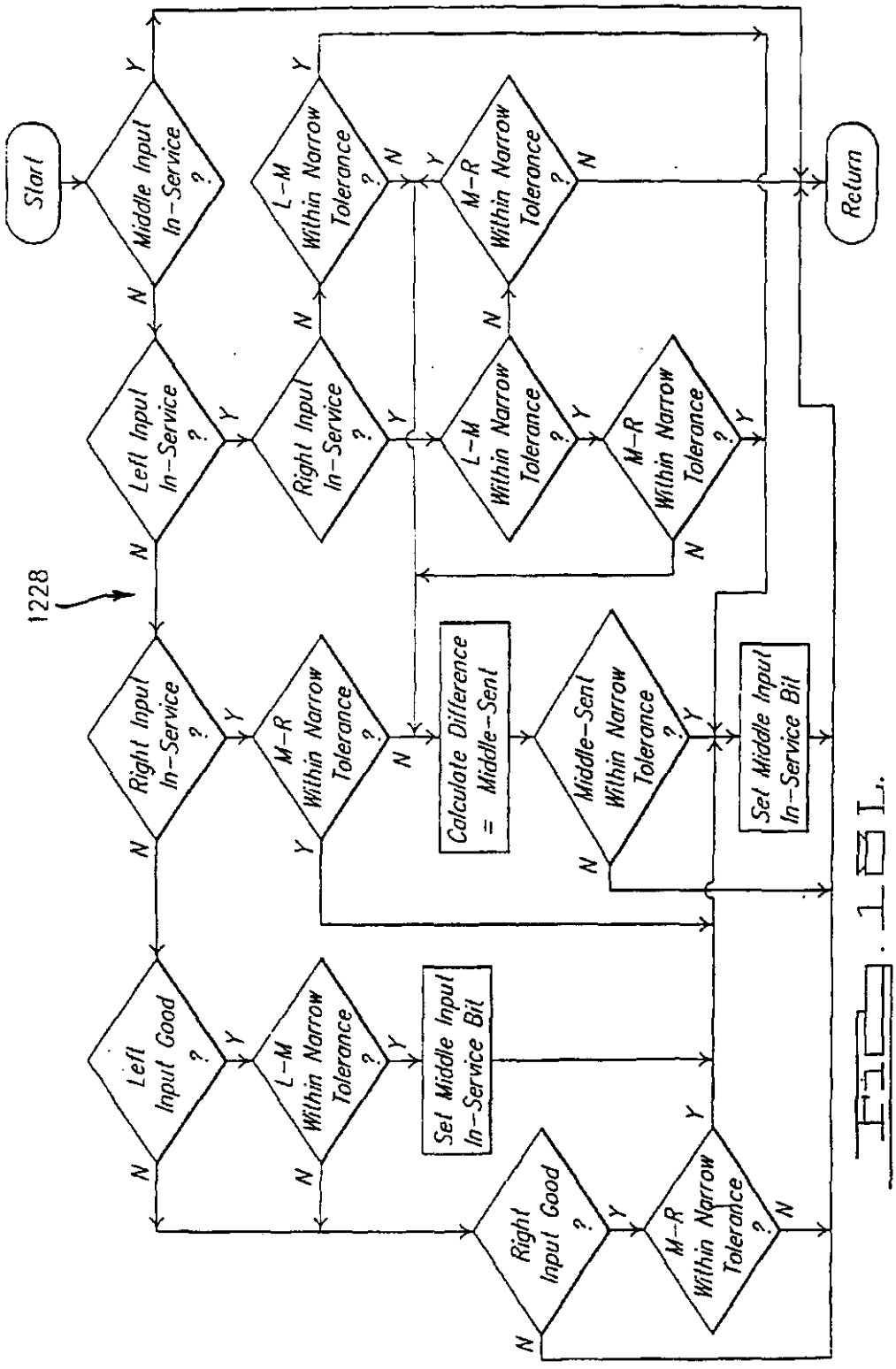

Start
Middle Input In-Service ?
Left Input In-Service ?
Right Input In-Service ?
L-M Within Narrow Tolerance ?
M-R Within Narrow Tolerance ?
L-M Within Narrow Tolerance ?
M-R Within Narrow Tolerance ?
1228
Right Input In-Service ?
M-R Within Narrow Tolerance ?
Calculate Difference = Middle-Sent
Middle-Sent Within Narrow Tolerance ?
Set Middle Input In-Service Bit
Left Input Good ?
L-M Within Narrow Tolerance ?
Set Middle Input In-Service Bit
Right Input Good ?
M-R Within Narrow Tolerance ?
Return
Y
N

Fig. 15L.

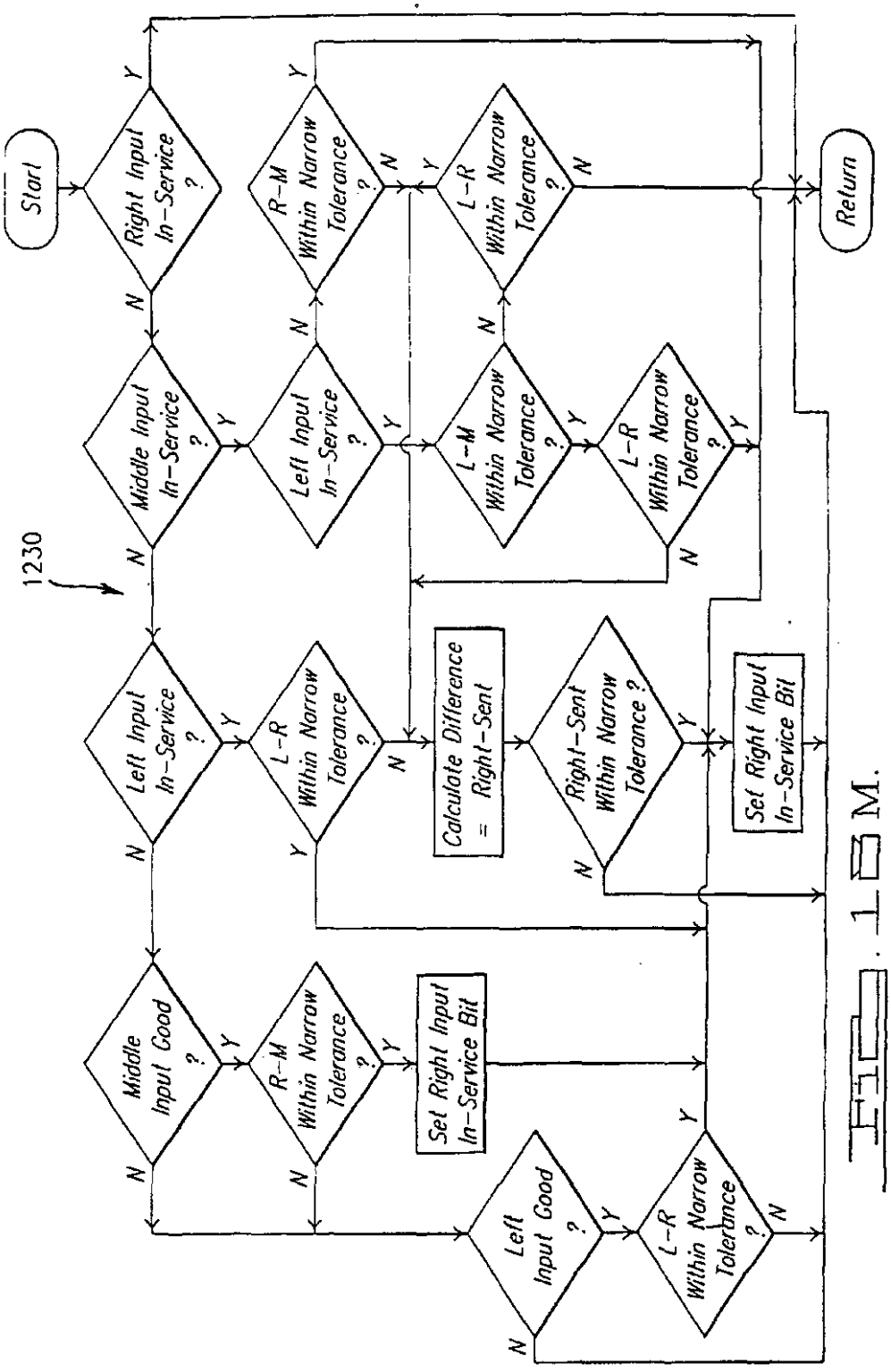

Start
Right Input In-Service ?
R-M Within Narrow Tolerance ?
L-R Within Narrow Tolerance ?
Return
1230
Middle Input In-Service ?
Left Input In-Service ?
L-M Within Narrow Tolerance ?
L-R Within Narrow Tolerance ?
Left Input In-Service ?
L-R Within Narrow Tolerance ?
Calculate Difference = Right-Sent
Right-Sent Within Narrow Tolerance ?
Set Right Input In-Service Bit
Middle Input Good ?
R-M Within Narrow Tolerance ?
Set Right Input In-Service Bit
Left Input Good ?
L-R Within Narrow Tolerance ?
Y
N

FIG. 18M.

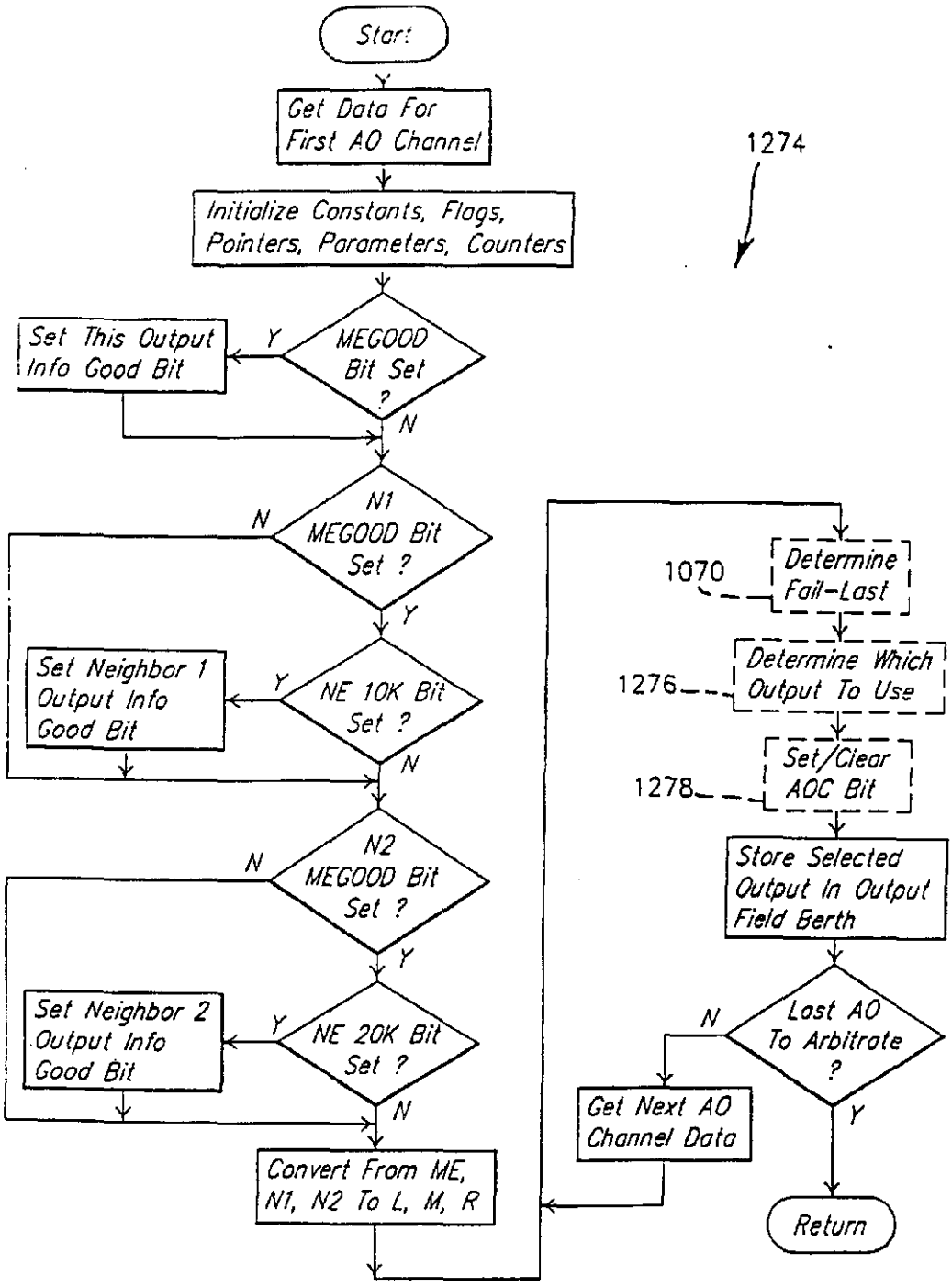

Start
Get Data For First AO Channel
Initialize Constants, Flags, Pointers, Parameters, Counters
1274
MEGOOD Bit Set ?
Y
Set This Output Info Good Bit
N
N1 MEGOOD Bit Set ?
N
Y
NE 10K Bit Set ?
Y
Set Neighbor 1 Output Info Good Bit
N
N2 MEGOOD Bit Set ?
N
Y
NE 20K Bit Set ?
Y
Set Neighbor 2 Output Info Good Bit
N
Convert From ME, N1, N2 To L, M, R
1070
Determine Fail-Last
1276
Determine Which Output To Use
1278
Set/Clear AOC Bit
Store Selected Output In Output Field Berth
Last AO To Arbitrate ?
N
Y
Get Next AO Channel Data
Return

Fig. 130.

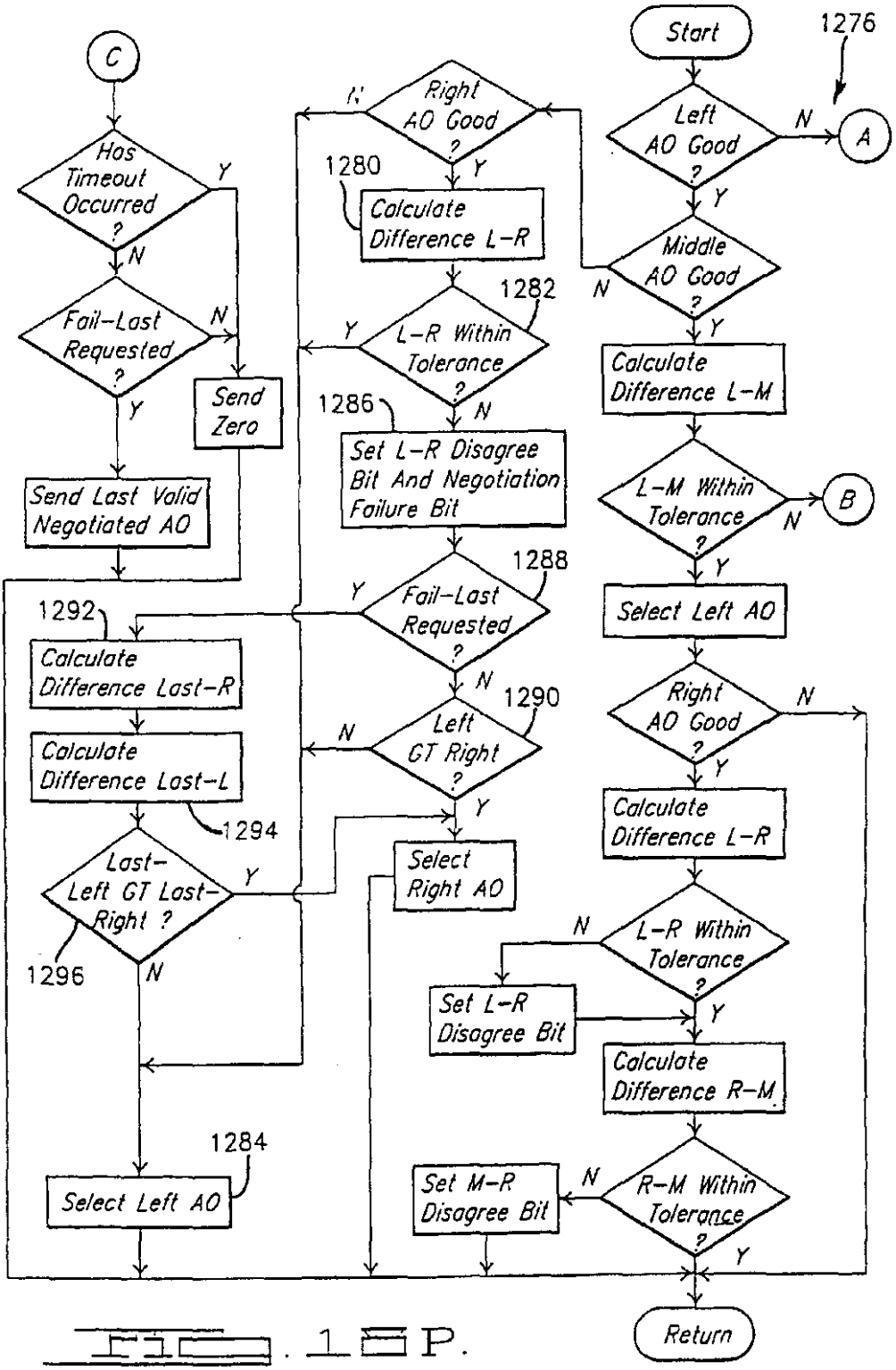

Start
1276
Left AO Good ?
N
A
Y
Middle AO Good ?
N
Y
Calculate Difference L-M
L-M Within Tolerance ?
N
B
Y
Select Left AO
Right AO Good ?
N
Y
Calculate Difference L-R
L-R Within Tolerance ?
N
Set L-R Disagree Bit
Y
Calculate Difference R-M
R-M Within Tolerance ?
N
Set M-R Disagree Bit
Y
Return
Right AO Good ?
1280
Y
Calculate Difference L-R
1282
L-R Within Tolerance ?
Y
N
1286
Set L-R Disagree Bit And Negotiation Failure Bit
1288
Fail-Last Requested ?
Y
N
1290
Left GT Right ?
N
Y
Select Right AO
C
Has Timeout Occurred ?
Y
N
Fail-Last Requested ?
N
Y
Send Zero
Send Last Valid Negotiated AO
1292
Calculate Difference Last-R
Calculate Difference Last-L
1294
Last-Left GT Last-Right ?
Y
1296
N
1284
Select Left AO

Fig. 15P.

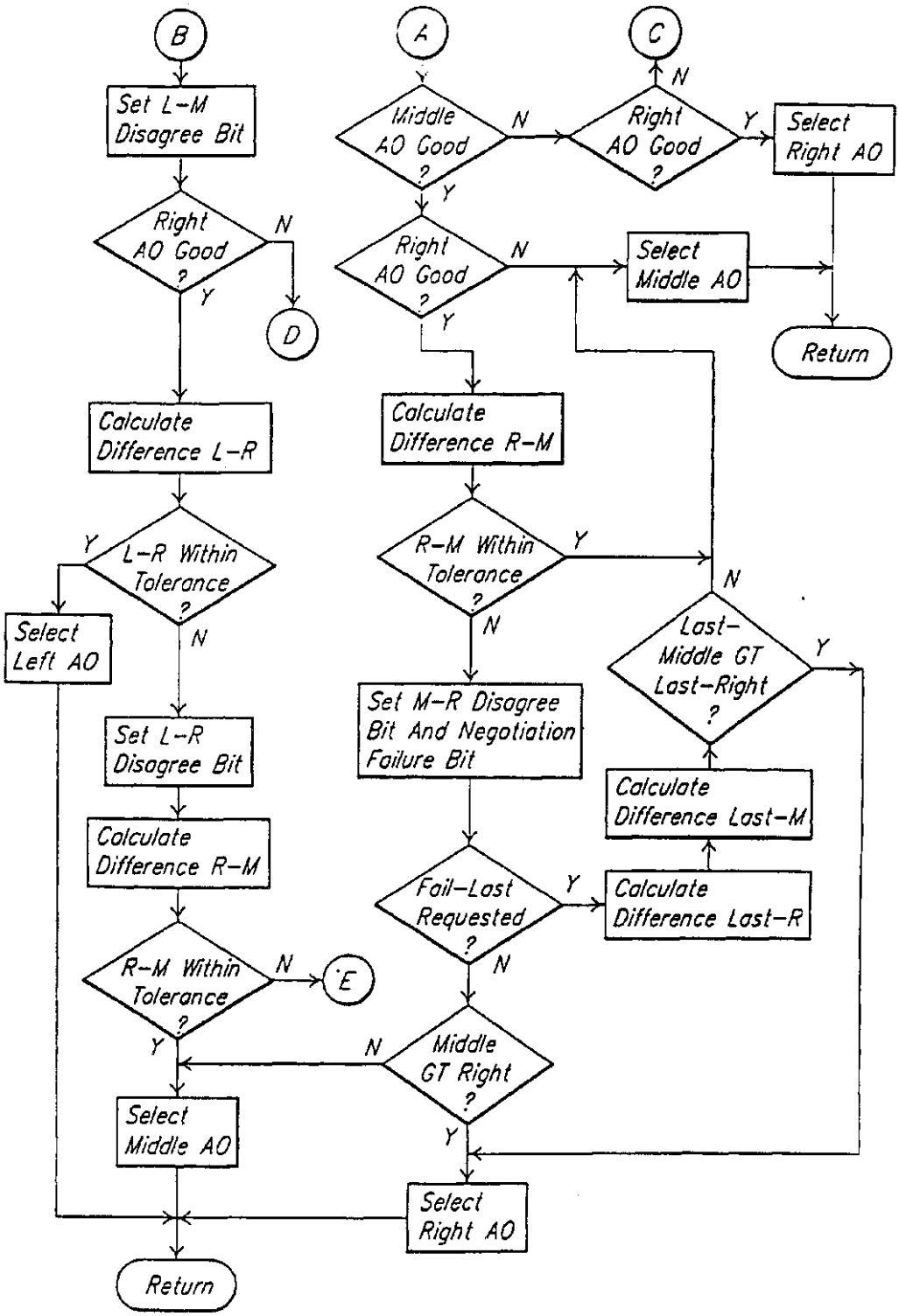

B
Set L-M Disagree Bit
Right AO Good ?
N
Y
D
Calculate Difference L-R
L-R Within Tolerance ?
Y
N
Select Left AO
Set L-R Disagree Bit
Calculate Difference R-M
R-M Within Tolerance ?
N
E
Y
Select Middle AO
Return
A
Middle AO Good ?
N
Y
C
Right AO Good ?
N
Y
Select Right AO
Right AO Good ?
N
Y
Select Middle AO
Return
Calculate Difference R-M
R-M Within Tolerance ?
Y
N
Set M-R Disagree Bit And Negotiation Failure Bit
Fail-Last Requested ?
Y
N
Calculate Difference Last-R
Calculate Difference Last-M
Last-Middle GT Last-Right ?
Y
N
Middle GT Right ?
N
Y
Select Right AO

FIG. 10Q.

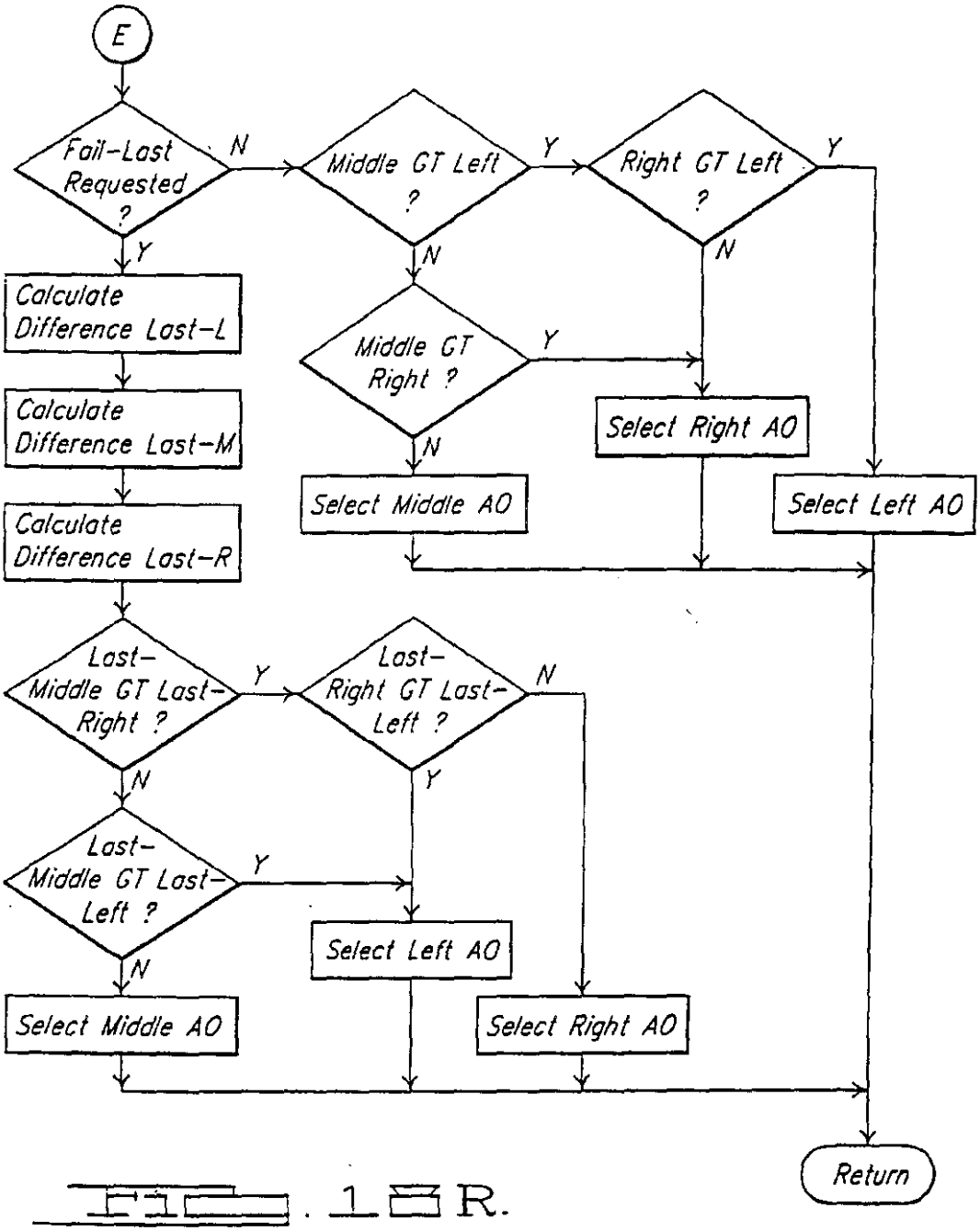

E
Fail-Last Requested ?
N
Y
Middle GT Left ?
Y
N
Right GT Left ?
Y
N
Calculate Difference Last-L
Calculate Difference Last-M
Calculate Difference Last-R
Middle GT Right ?
Y
N
Select Right AO
Select Middle AO
Select Left AO
Last-Middle GT Last-Right ?
Y
N
Last-Right GT Last-Left ?
N
Y
Last-Middle GT Last-Left ?
Y
N
Select Left AO
Select Middle AO
Select Right AO
Return

FIG. 18R.

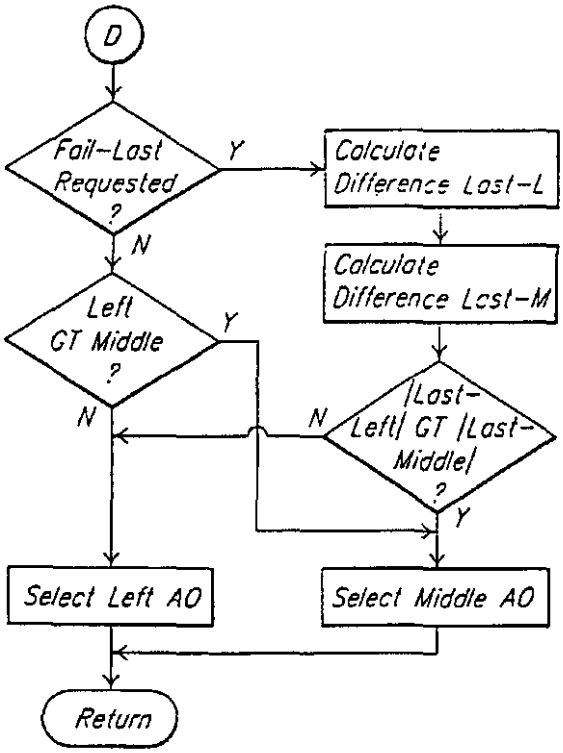


FIG. 18S.

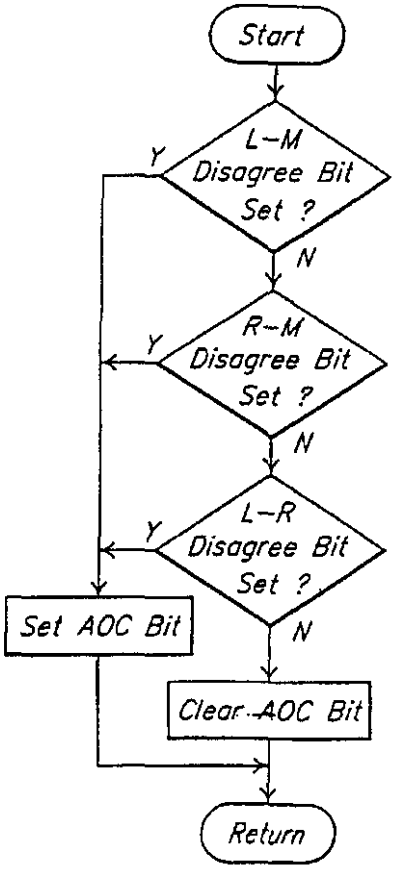


FIG. 18T.

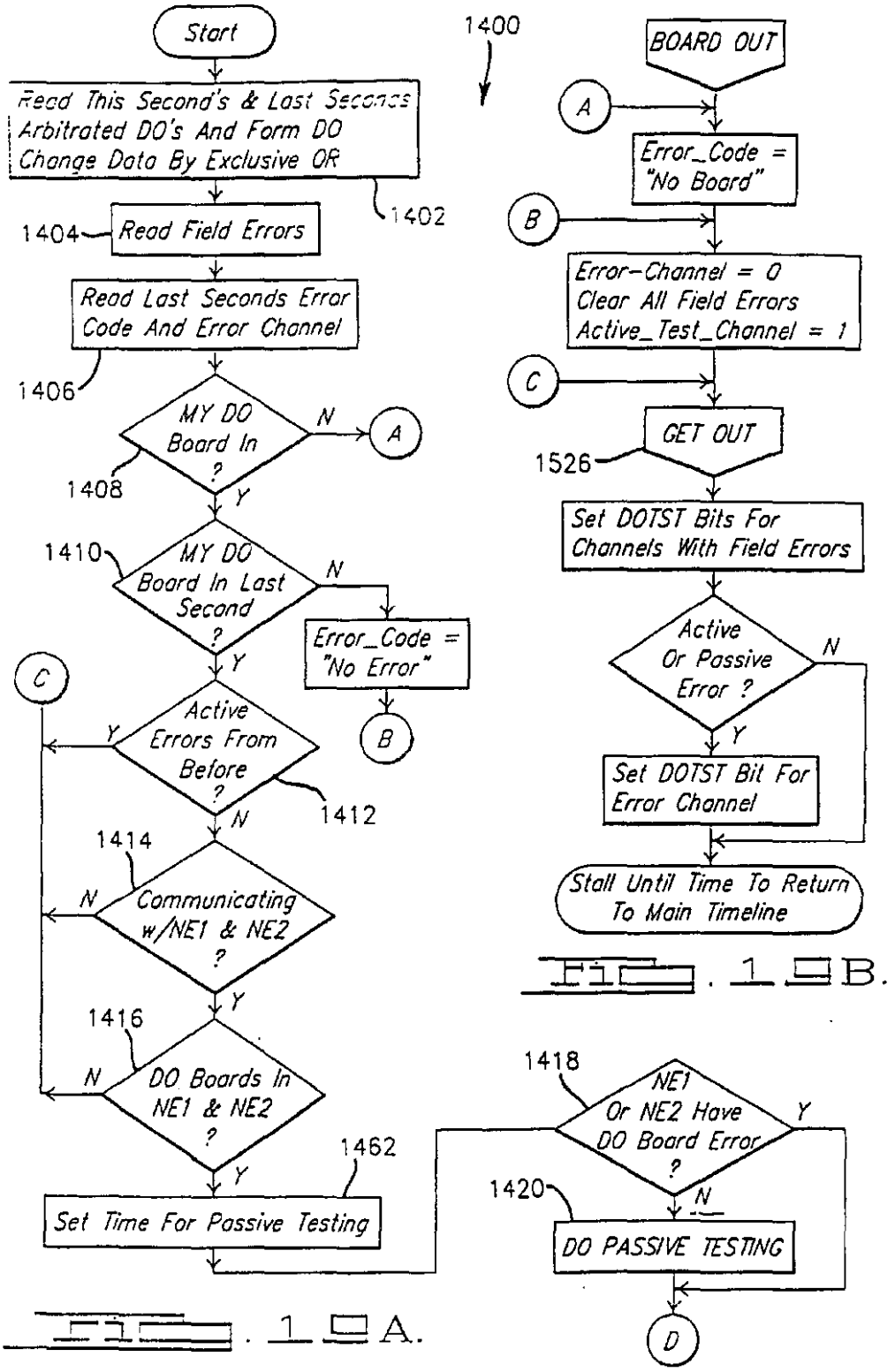


Fig. 10A.

Fig. 10B.

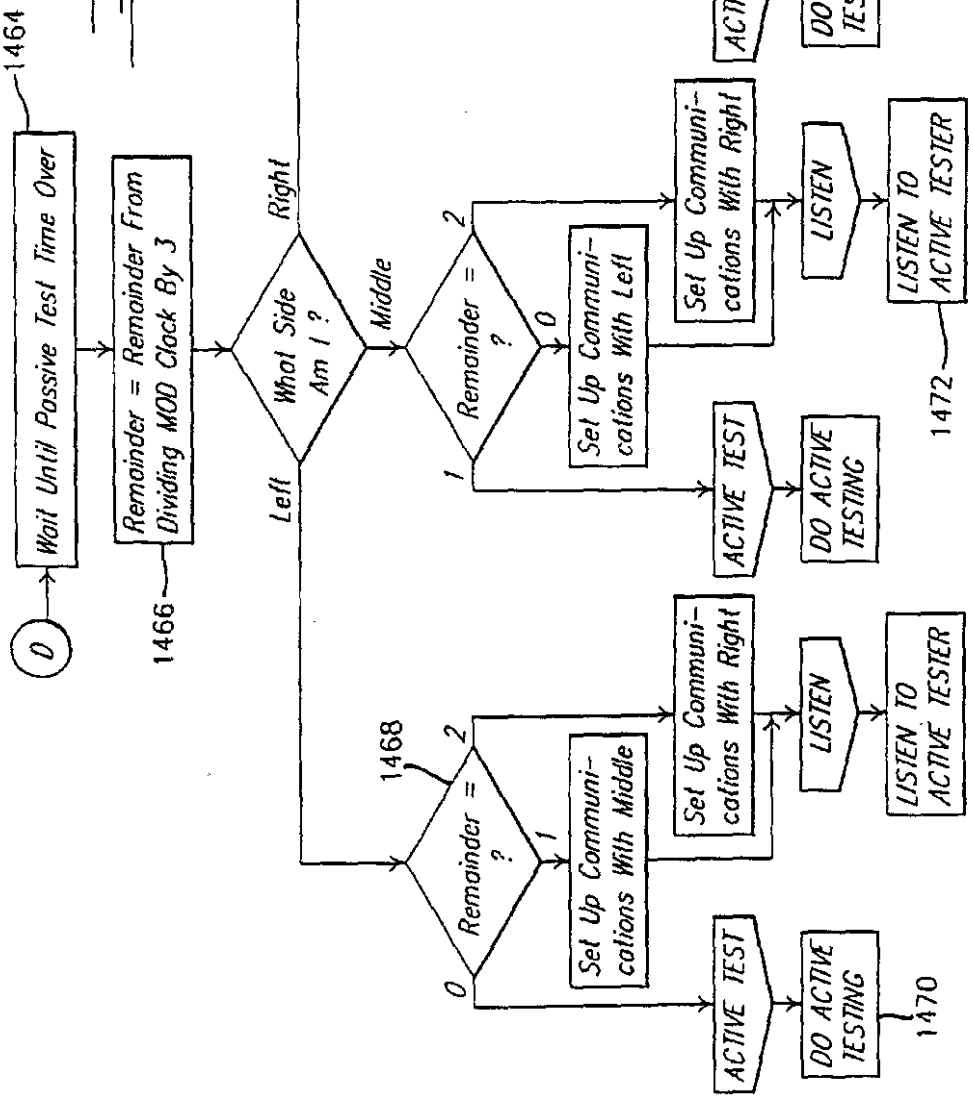

0
Wait Until Passive Test Time Over
1464
Remainder = Remainder From Dividing MOD Clock By 3
1466
What Side Am I ?
Left
Middle
Right
Remainder = ?
1468
0
1
2
Set Up Communi-cations With Middle
Set Up Communi-cations With Left
Set Up Communi-cations With Right
ACTIVE TEST
DO ACTIVE TESTING
LISTEN
LISTEN TO ACTIVE TESTER
1470
1472
1473

FIG. 15C.

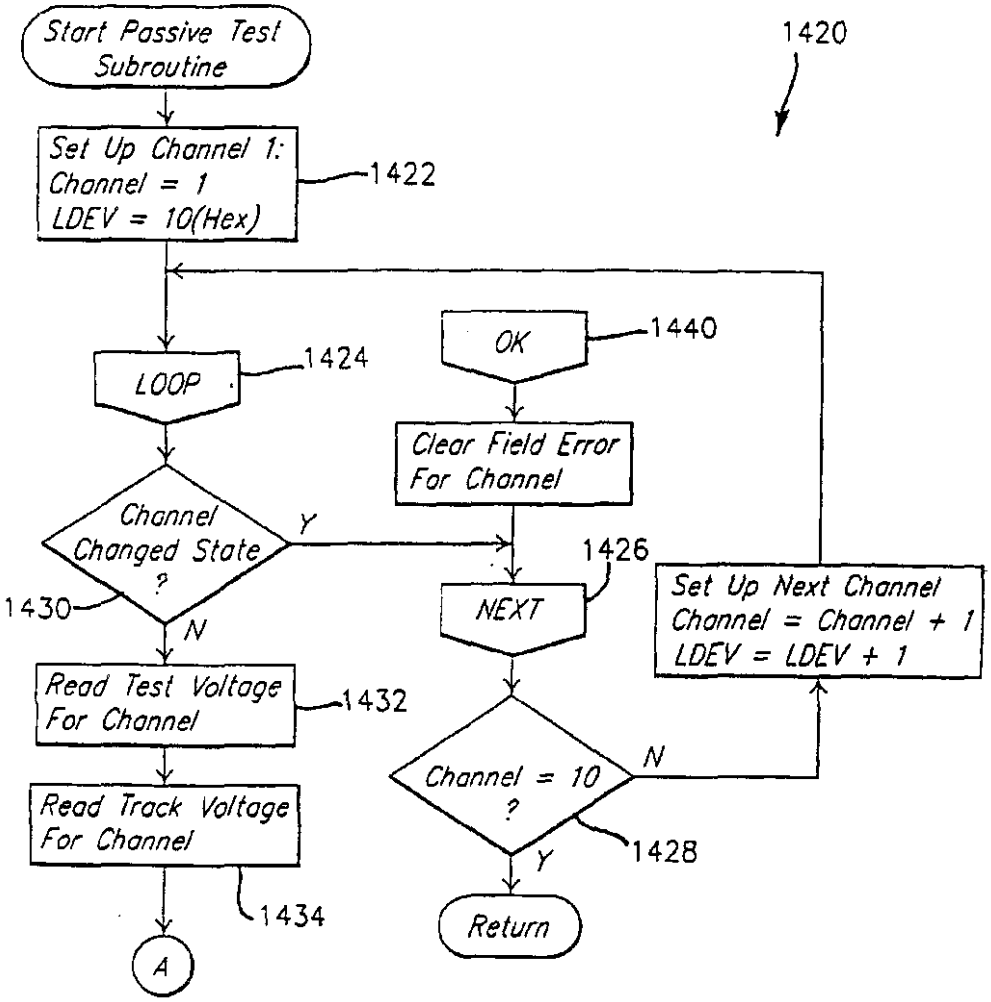

Start Passive Test Subroutine
1420
Set Up Channel 1:
Channel = 1
LDEV = 10(Hex)
1422
LOOP
1424
OK
1440
Clear Field Error For Channel
Channel Changed State ?
Y
1426
1430
N
NEXT
Set Up Next Channel
Channel = Channel + 1
LDEV = LDEV + 1
Read Test Voltage For Channel
1432
Channel = 10 ?
N
Read Track Voltage For Channel
1428
Y
1434
Return
A

FIG. 10D.

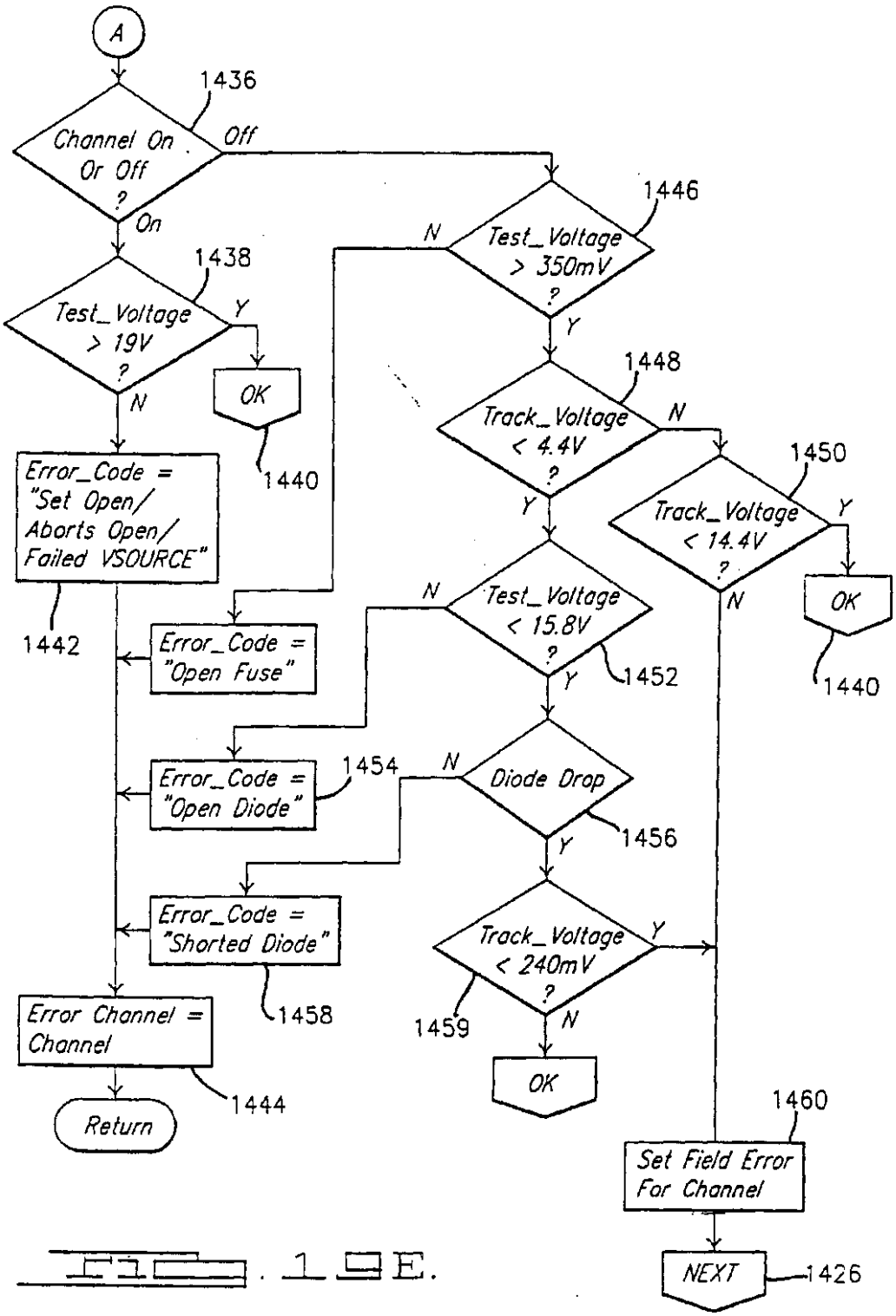

A
1436
Channel On Or Off ?
Off
On
1438
Test_Voltage > 19V ?
Y
N
OK
1440
Error_Code = "Set Open/ Aborts Open/ Failed VSOURCE"
1442
1446
Test_Voltage > 350mV ?
N
Y
Error_Code = "Open Fuse"
1448
Track_Voltage < 4.4V ?
N
Y
1450
Track_Voltage < 14.4V ?
Y
N
OK
1440
Test_Voltage < 15.8V ?
N
Y
1452
Error_Code = "Open Diode"
1454
Diode Drop
N
Y
1456
Error_Code = "Shorted Diode"
1458
Track_Voltage < 240mV ?
Y
N
1459
OK
Error Channel = Channel
1444
Return
1460
Set Field Error For Channel
NEXT
1426

FIG. 10E.

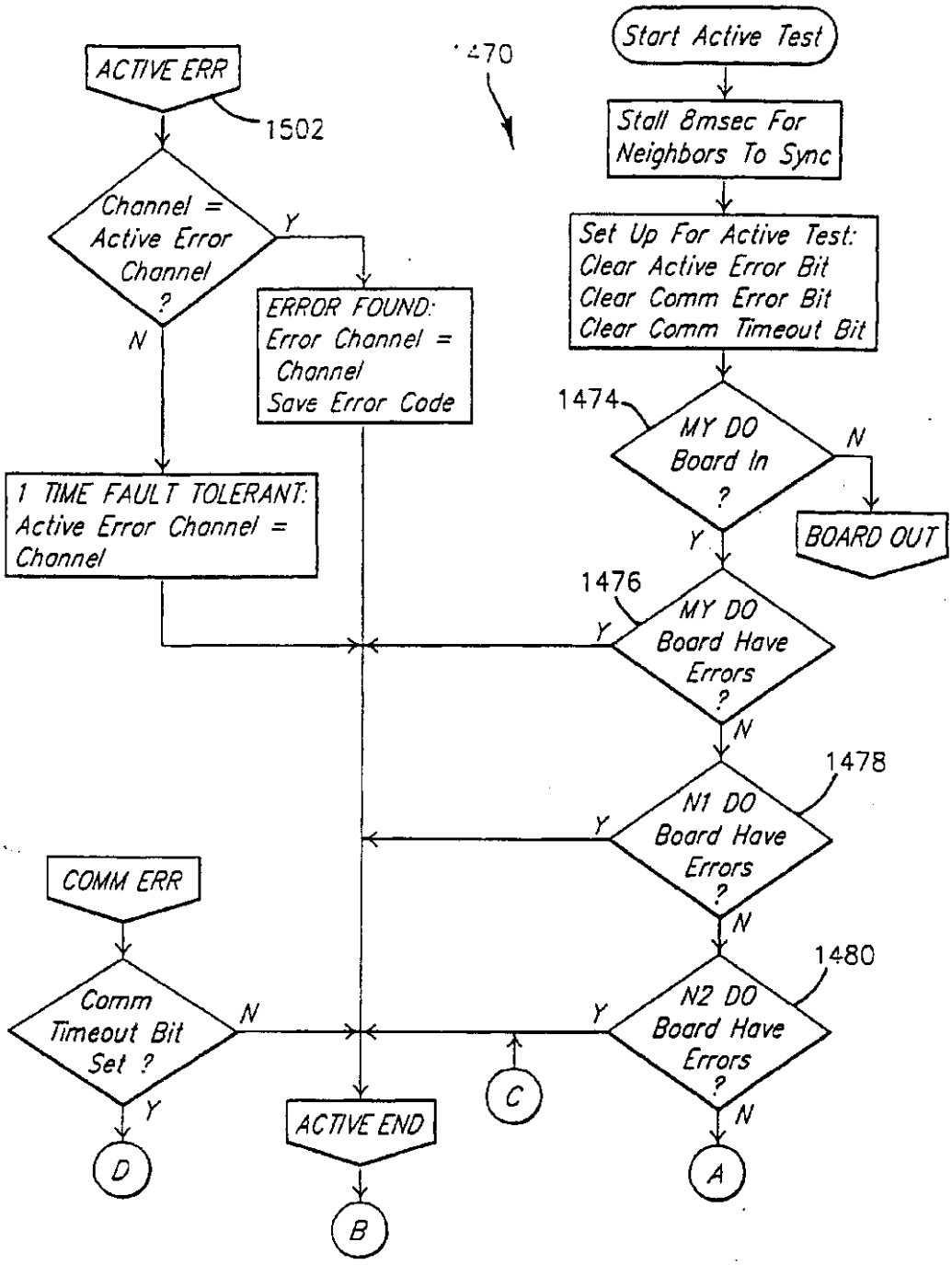

ACTIVE ERR
1502
Channel = Active Error Channel ?
Y
N
ERROR FOUND: Error Channel = Channel Save Error Code
1 TIME FAULT TOLERANT: Active Error Channel = Channel
COMM ERR
Comm Timeout Bit Set ?
N
Y
D
ACTIVE END
B
C
470
Start Active Test
Stall 8msec For Neighbors To Sync
Set Up For Active Test: Clear Active Error Bit Clear Comm Error Bit Clear Comm Timeout Bit
1474
MY DO Board In ?
N
BOARD OUT
Y
1476
MY DO Board Have Errors ?
Y
N
1478
N1 DO Board Have Errors ?
Y
N
1480
N2 DO Board Have Errors ?
Y
N
A

FIG. 19F.

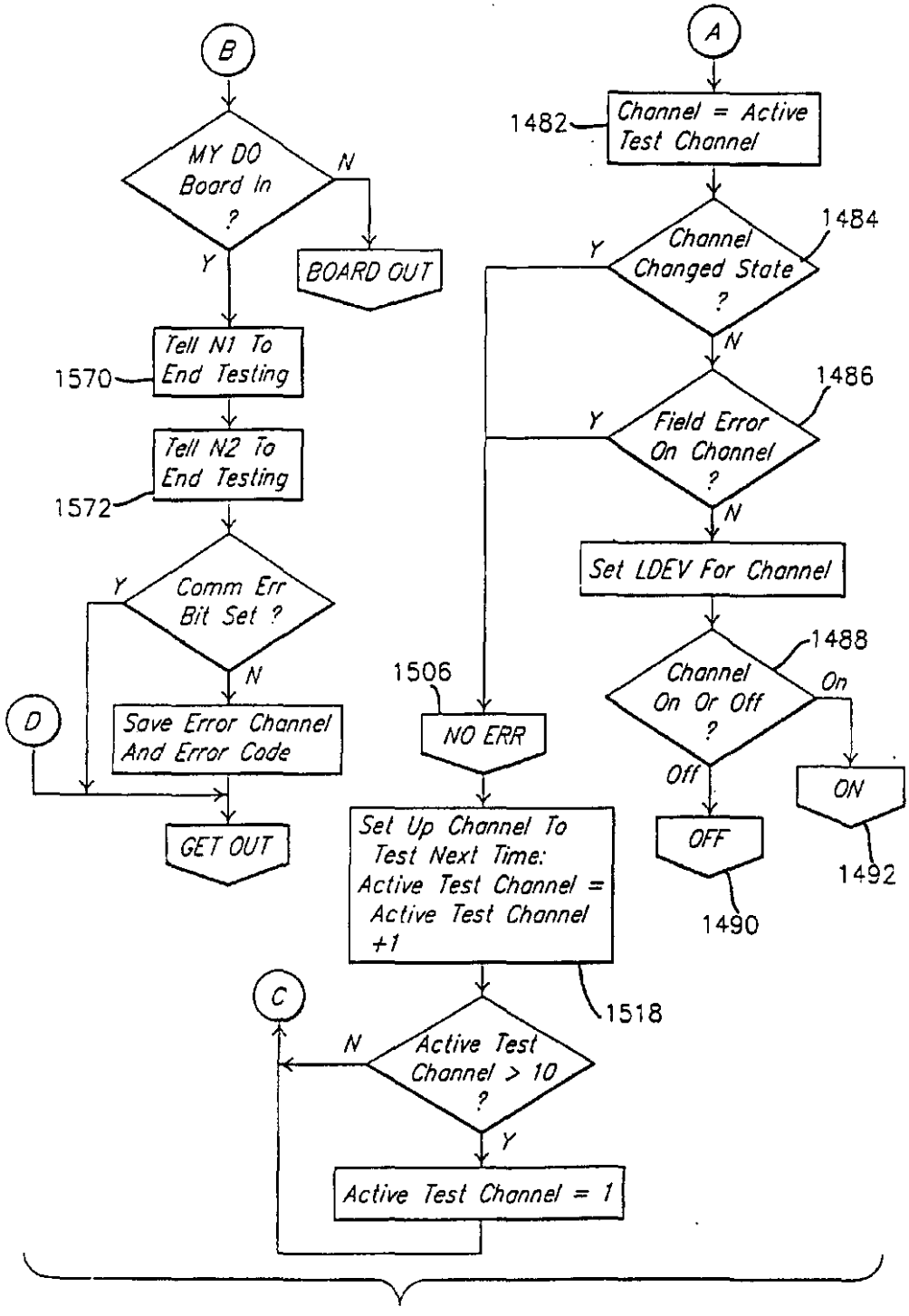

B
MY DO Board In ?
N
BOARD OUT
Y
1570
Tell N1 To End Testing
Tell N2 To End Testing
1572
Y
Comm Err Bit Set ?
N
D
Save Error Channel And Error Code
GET OUT
A
1482
Channel = Active Test Channel
1484
Y
Channel Changed State ?
N
1486
Y
Field Error On Channel ?
N
Set LDEV For Channel
1488
Channel On Or Off ?
On
Off
ON
1492
OFF
1490
1506
NO ERR
Set Up Channel To Test Next Time: Active Test Channel = Active Test Channel +1
1518
C
N
Active Test Channel > 10 ?
Y
Active Test Channel = 1

FIG. 10G.

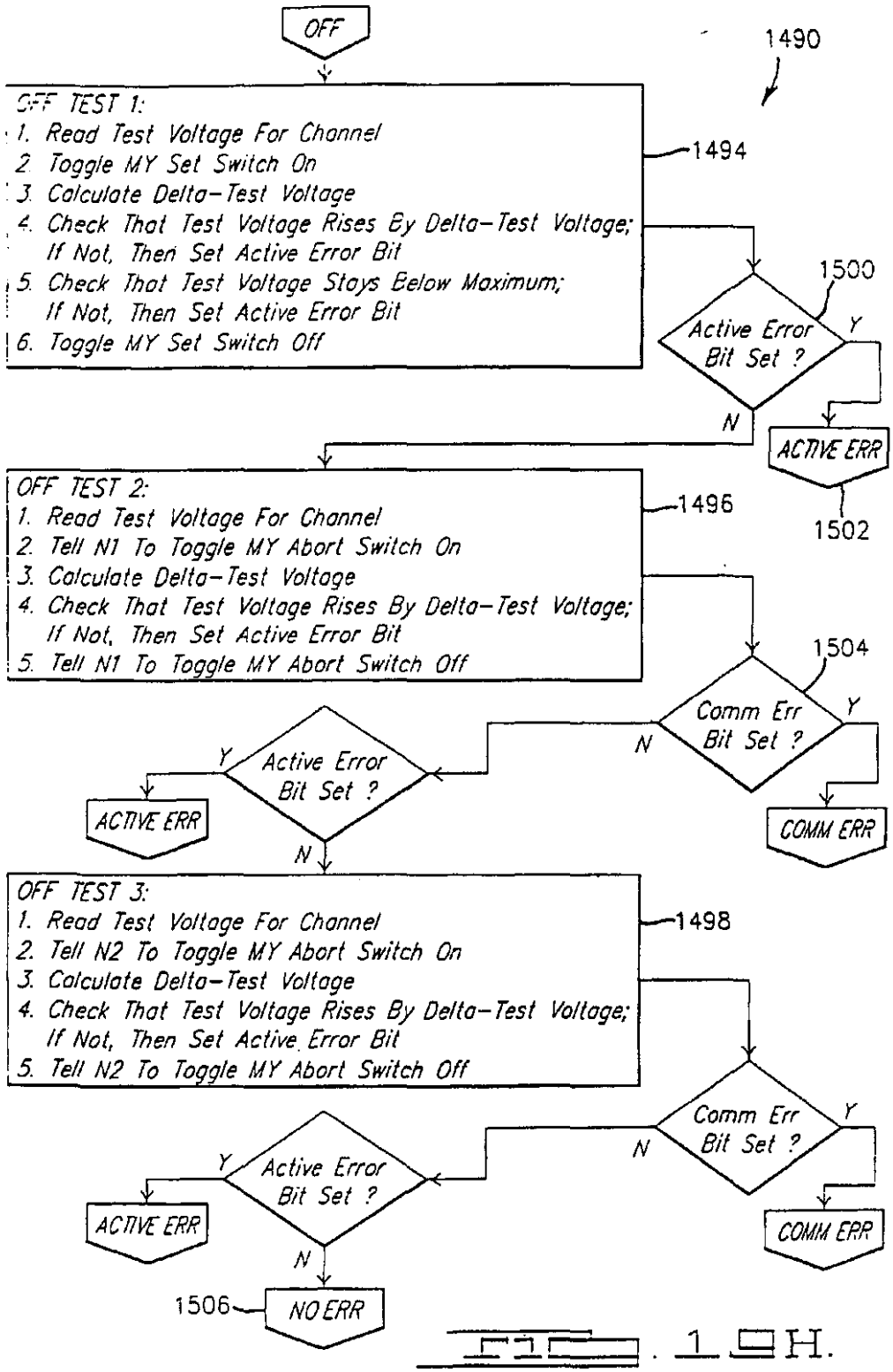

OFF
1490
OFF TEST 1:
1. Read Test Voltage For Channel
2. Toggle MY Set Switch On
3. Calculate Delta-Test Voltage
4. Check That Test Voltage Rises By Delta-Test Voltage; If Not, Then Set Active Error Bit
5. Check That Test Voltage Stays Below Maximum; If Not, Then Set Active Error Bit
6. Toggle MY Set Switch Off
1494
1500
Active Error Bit Set ?
Y
N
ACTIVE ERR
1502
OFF TEST 2:
1. Read Test Voltage For Channel
2. Tell N1 To Toggle MY Abort Switch On
3. Calculate Delta-Test Voltage
4. Check That Test Voltage Rises By Delta-Test Voltage; If Not, Then Set Active Error Bit
5. Tell N1 To Toggle MY Abort Switch Off
1496
1504
Comm Err Bit Set ?
Y
N
Active Error Bit Set ?
Y
N
ACTIVE ERR
COMM ERR
OFF TEST 3:
1. Read Test Voltage For Channel
2. Tell N2 To Toggle MY Abort Switch On
3. Calculate Delta-Test Voltage
4. Check That Test Voltage Rises By Delta-Test Voltage; If Not, Then Set Active Error Bit
5. Tell N2 To Toggle MY Abort Switch Off
1498
Comm Err Bit Set ?
Y
N
Active Error Bit Set ?
Y
N
ACTIVE ERR
COMM ERR
1506
NO ERR

FIG. 10H.

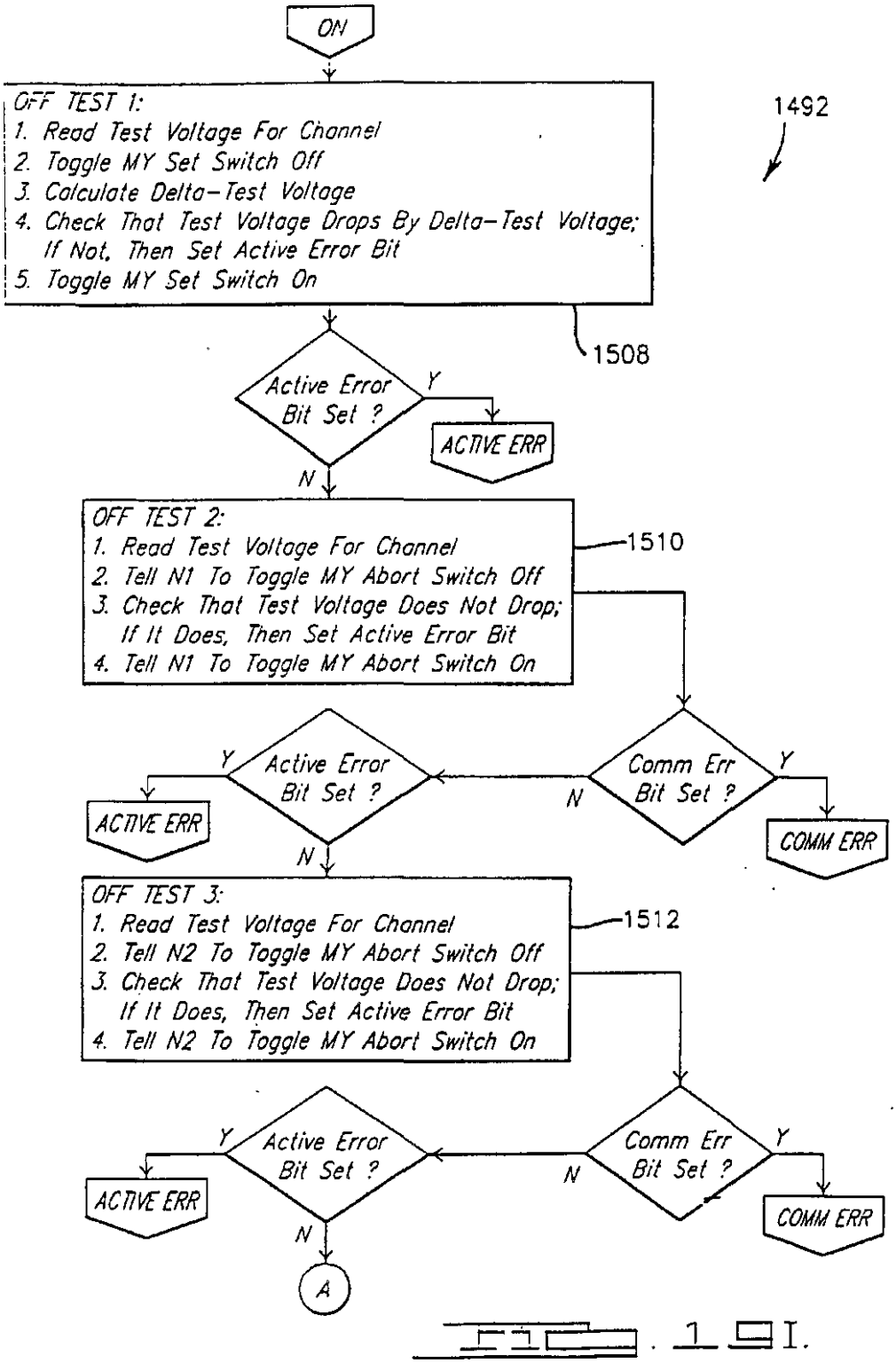

ON
OFF TEST 1:
1. Read Test Voltage For Channel
2. Toggle MY Set Switch Off
3. Calculate Delta-Test Voltage
4. Check That Test Voltage Drops By Delta-Test Voltage; If Not, Then Set Active Error Bit
5. Toggle MY Set Switch On
1492
1508
Active Error Bit Set ?
Y
ACTIVE ERR
N
OFF TEST 2:
1. Read Test Voltage For Channel
2. Tell N1 To Toggle MY Abort Switch Off
3. Check That Test Voltage Does Not Drop; If It Does, Then Set Active Error Bit
4. Tell N1 To Toggle MY Abort Switch On
1510
Active Error Bit Set ?
Y
ACTIVE ERR
N
Comm Err Bit Set ?
Y
COMM ERR
OFF TEST 3:
1. Read Test Voltage For Channel
2. Tell N2 To Toggle MY Abort Switch Off
3. Check That Test Voltage Does Not Drop; If It Does, Then Set Active Error Bit
4. Tell N2 To Toggle MY Abort Switch On
1512
Active Error Bit Set ?
Y
ACTIVE ERR
N
Comm Err Bit Set ?
Y
COMM ERR
N
A

FIG. 1I.

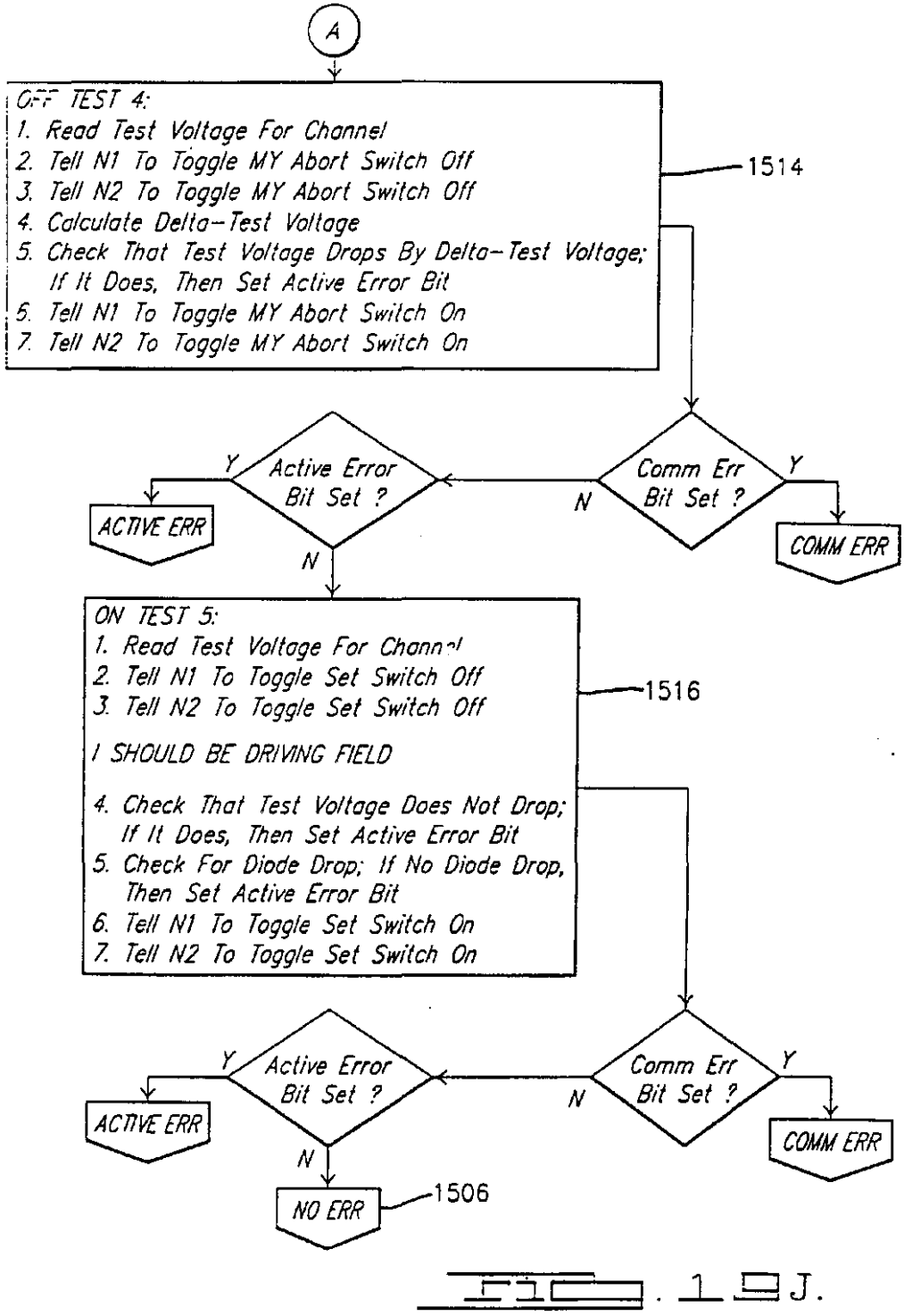

A
OFF TEST 4:
1. Read Test Voltage For Channel
2. Tell N1 To Toggle MY Abort Switch Off
3. Tell N2 To Toggle MY Abort Switch Off
4. Calculate Delta-Test Voltage
5. Check That Test Voltage Drops By Delta-Test Voltage; If It Does, Then Set Active Error Bit
6. Tell N1 To Toggle MY Abort Switch On
7. Tell N2 To Toggle MY Abort Switch On
1514
Comm Err Bit Set ?
Y
N
COMM ERR
Active Error Bit Set ?
Y
N
ACTIVE ERR
ON TEST 5:
1. Read Test Voltage For Channel
2. Tell N1 To Toggle Set Switch Off
3. Tell N2 To Toggle Set Switch Off
I SHOULD BE DRIVING FIELD
4. Check That Test Voltage Does Not Drop; If It Does, Then Set Active Error Bit
5. Check For Diode Drop; If No Diode Drop, Then Set Active Error Bit
6. Tell N1 To Toggle Set Switch On
7. Tell N2 To Toggle Set Switch On
1516
Comm Err Bit Set ?
Y
N
COMM ERR
Active Error Bit Set ?
Y
N
ACTIVE ERR
NO ERR
1506

Fig. 19J.

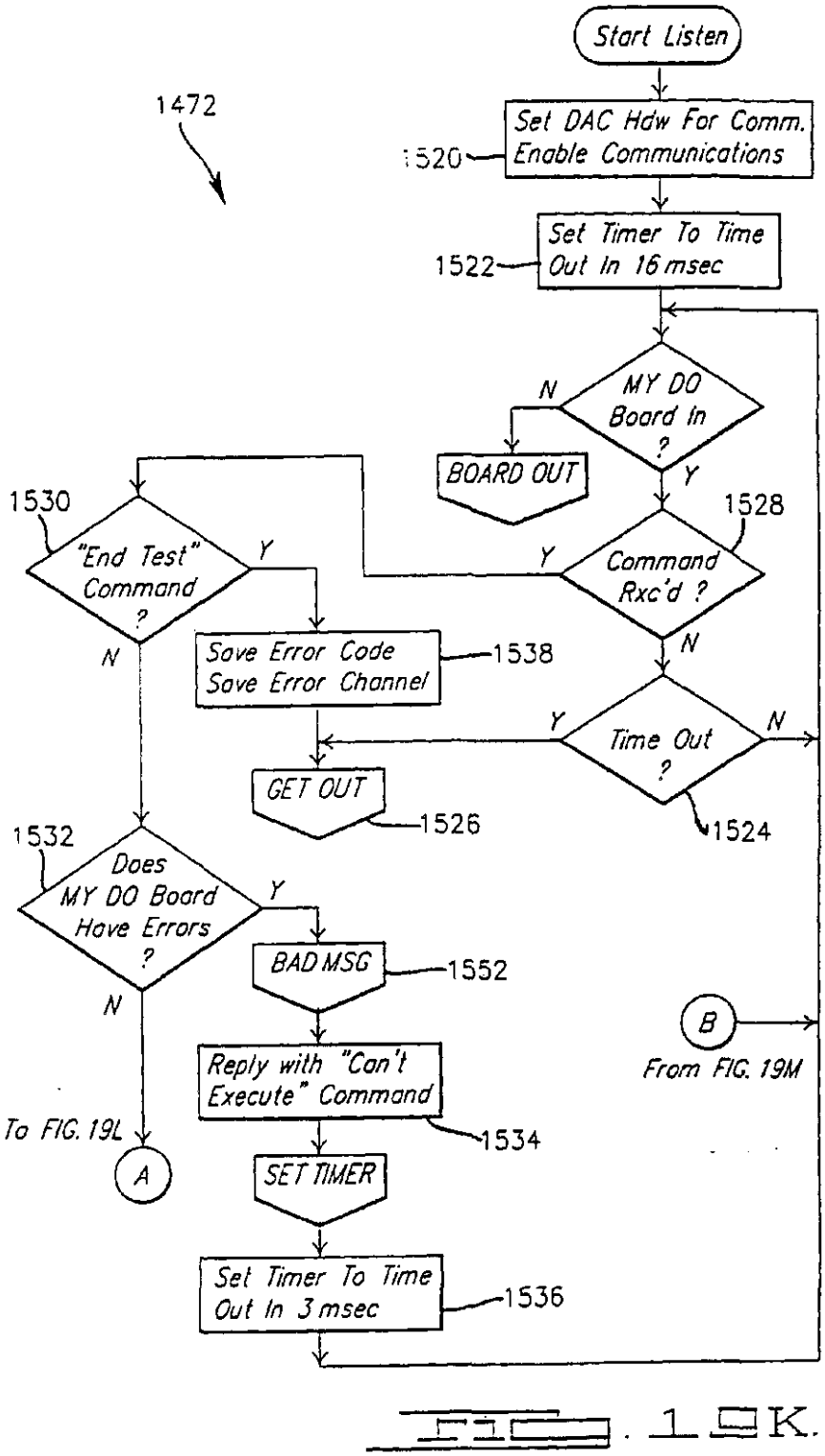

Start Listen
1472
Set DAC Hdw For Comm. Enable Communications
1520
Set Timer To Time Out In 16 msec
1522
MY DO Board In ?
N
Y
BOARD OUT
1528
Command Rxc'd ?
Y
N
1530
"End Test" Command ?
Y
N
Save Error Code Save Error Channel
1538
Time Out ?
Y
N
1524
GET OUT
1526
1532
Does MY DO Board Have Errors ?
Y
N
BAD MSG
1552
Reply with "Can't Execute" Command
1534
SET TIMER
Set Timer To Time Out In 3 msec
1536
B
From FIG. 19M
To FIG. 19L
A

FIG. 19K.

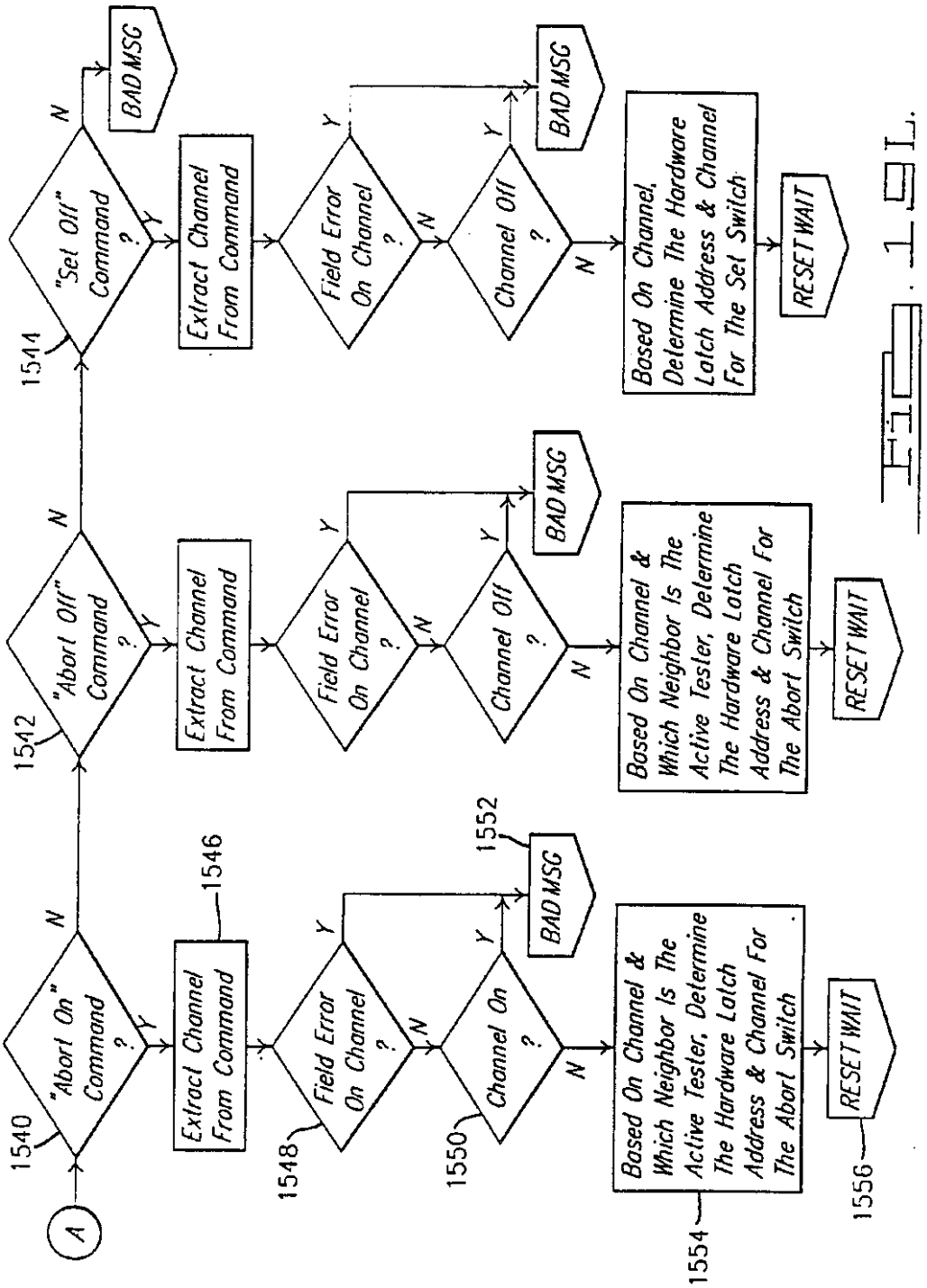

A
1540
"Abort On" Command ?
Y
N
1546
Extract Channel From Command
1548
Field Error On Channel ?
Y
N
1552
BAD MSG
1550
Channel On ?
Y
N
1554
Based On Channel & Which Neighbor Is The Active Tester, Determine The Hardware Latch Address & Channel For The Abort Switch
1556
RESET WAIT
1542
"Abort Off" Command ?
Y
N
Extract Channel From Command
Field Error On Channel ?
Y
N
BAD MSG
Channel Off ?
Y
N
Based On Channel & Which Neighbor Is The Active Tester, Determine The Hardware Latch Address & Channel For The Abort Switch
RESET WAIT
1544
"Set Off" Command ?
Y
N
BAD MSG
Extract Channel From Command
Field Error On Channel ?
Y
N
BAD MSG
Channel Off ?
Y
N
Based On Channel, Determine The Hardware Latch Address & Channel For The Set Switch
RESET WAIT

Fig. 15L.

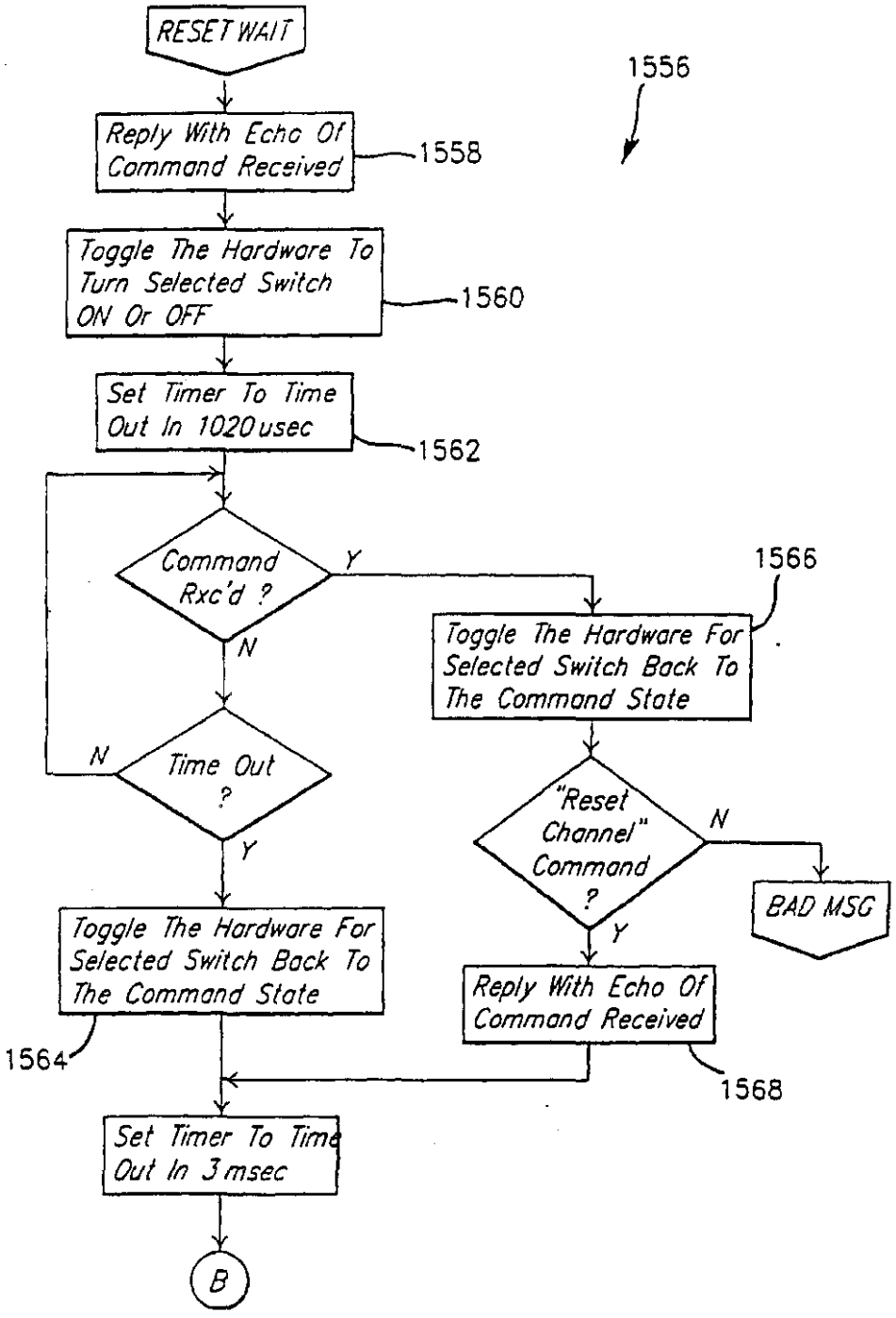

RESET WAIT
1556
Reply With Echo Of Command Received
1558
Toggle The Hardware To Turn Selected Switch ON Or OFF
1560
Set Timer To Time Out In 1020 usec
1562
Command Rxc'd ?
Y
N
1566
Toggle The Hardware For Selected Switch Back To The Command State
Time Out ?
N
Y
"Reset Channel" Command ?
N
Y
BAD MSG
Toggle The Hardware For Selected Switch Back To The Command State
1564
Reply With Echo Of Command Received
1568
Set Timer To Time Out In 3 msec
B

FIG. 15M.

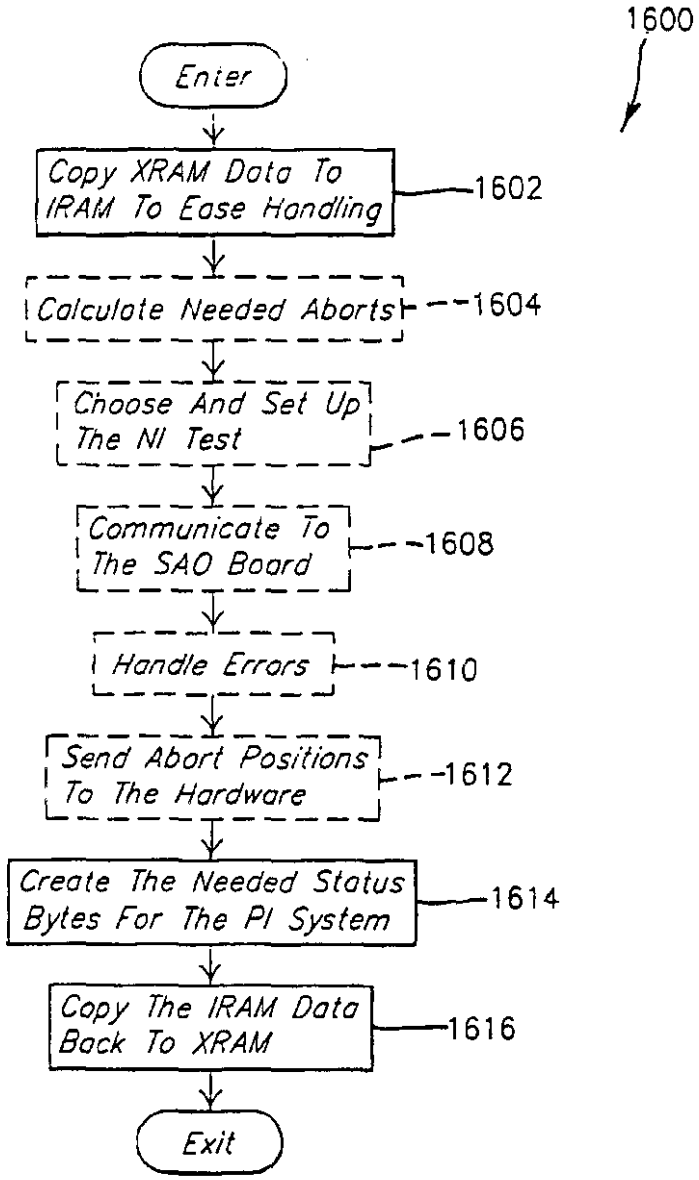

1600
Enter
Copy XRAM Data To IRAM To Ease Handling
1602
Calculate Needed Aborts
1604
Choose And Set Up The NI Test
1606
Communicate To The SAO Board
1608
Handle Errors
1610
Send Abort Positions To The Hardware
1612
Create The Needed Status Bytes For The PI System
1614
Copy The IRAM Data Back To XRAM
1616
Exit

FIG. 20A.

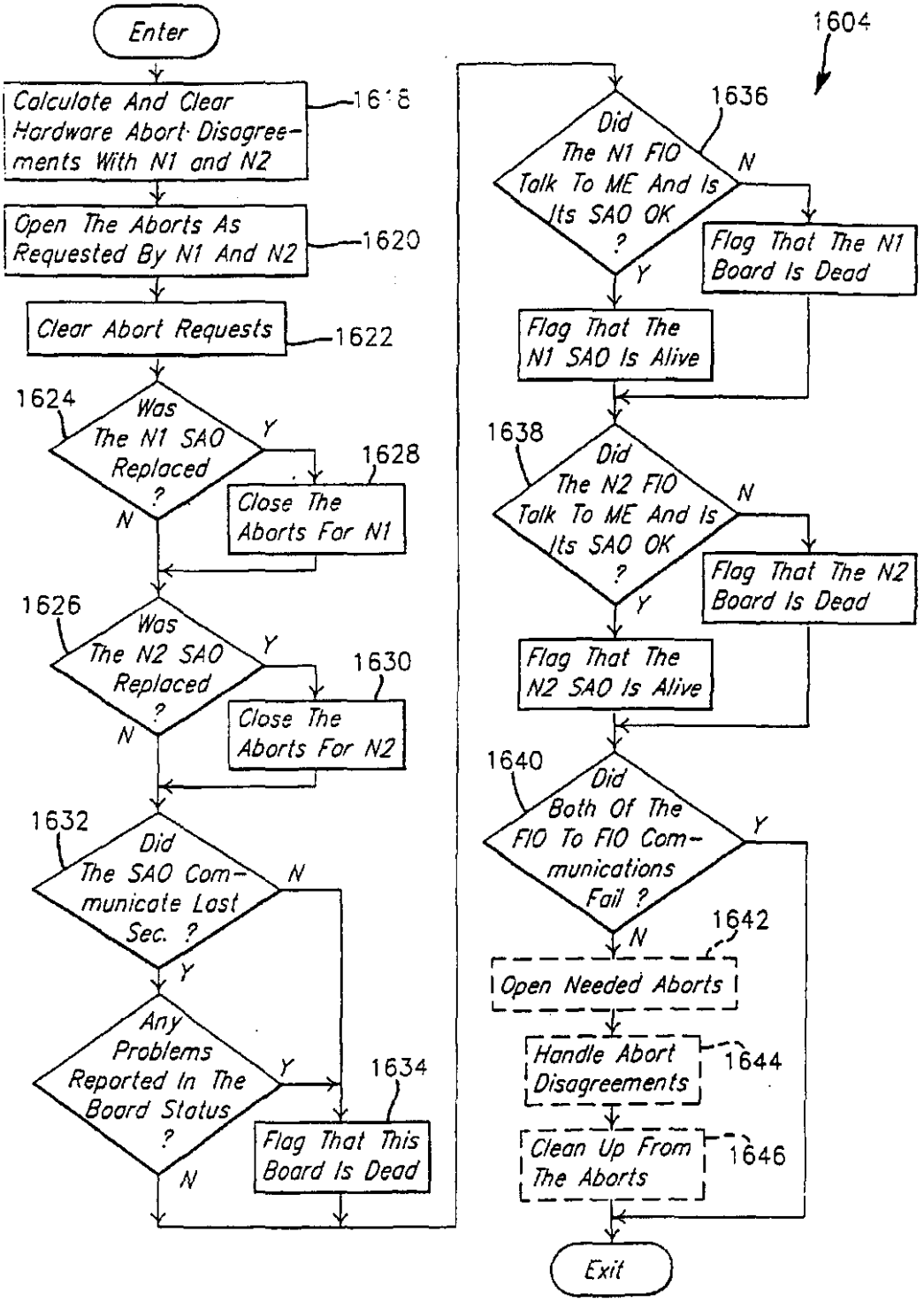
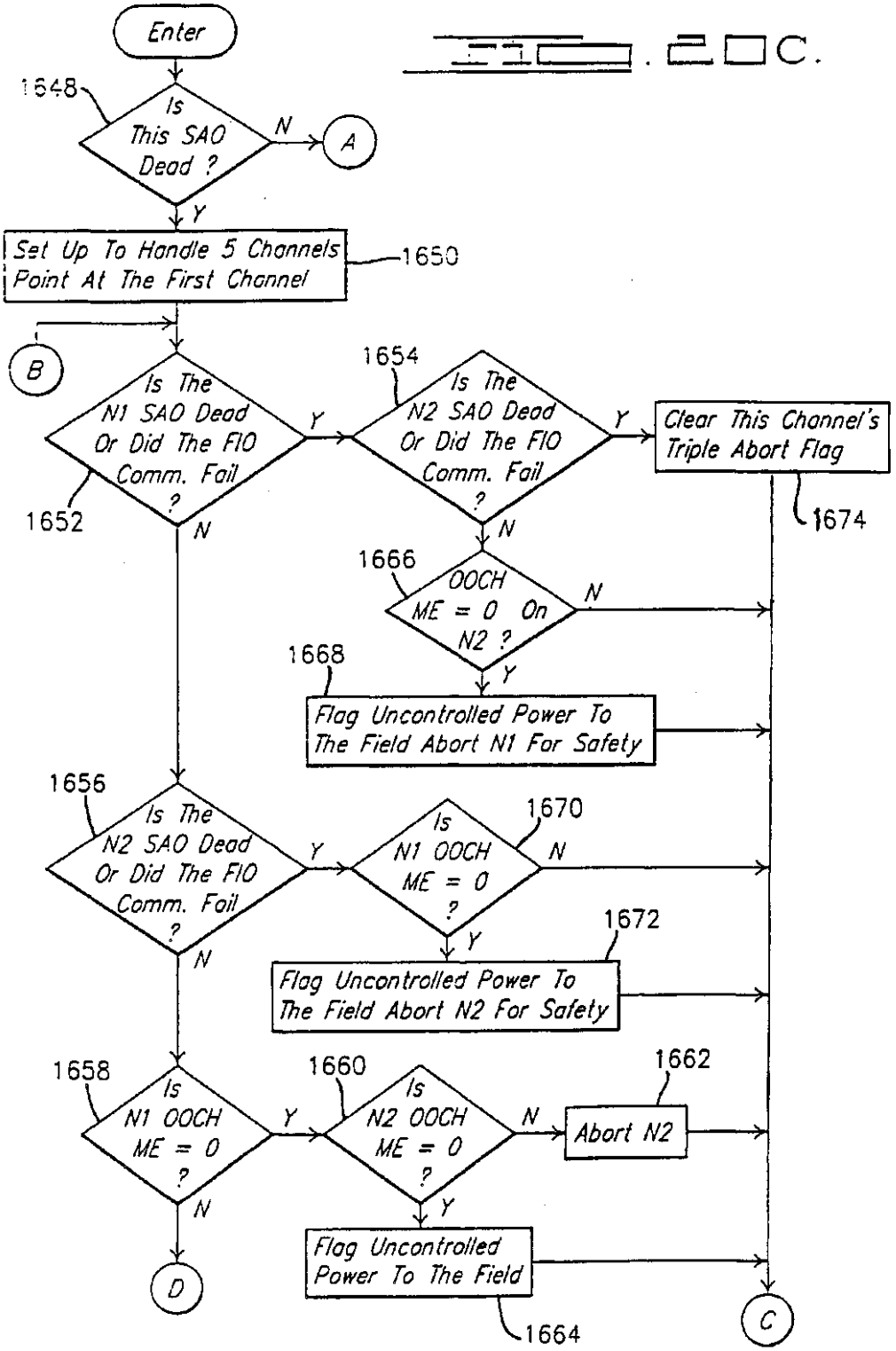

Enter
Calculate And Clear Hardware Abort Disagreements With N1 and N2
1618
Open The Aborts As Requested By N1 And N2
1620
Clear Abort Requests
1622
1624
Was The N1 SAO Replaced ?
Y
N
1628
Close The Aborts For N1
1626
Was The N2 SAO Replaced ?
Y
N
1630
Close The Aborts For N2
1632
Did The SAO Communicate Last Sec. ?
N
Y
Any Problems Reported In The Board Status ?
Y
N
1634
Flag That This Board Is Dead
1604
1636
Did The N1 FIO Talk To ME And Is Its SAO OK ?
N
Y
Flag That The N1 Board Is Dead
Flag That The N1 SAO Is Alive
1638
Did The N2 FIO Talk To ME And Is Its SAO OK ?
N
Y
Flag That The N2 Board Is Dead
Flag That The N2 SAO Is Alive
1640
Did Both Of The FIO To FIO Communications Fail ?
Y
N
1642
Open Needed Aborts
Handle Abort Disagreements
1644
Clean Up From The Aborts
1646
Exit

FIG. 20B.


Enter
1648
Is This SAO Dead ?
N
A
Y
Set Up To Handle 5 Channels Point At The First Channel
1650
B
Is The N1 SAO Dead Or Did The FIO Comm. Fail ?
Y
1654
Is The N2 SAO Dead Or Did The FIO Comm. Fail ?
Y
Clear This Channel's Triple Abort Flag
1674
1652
N
N
1666
OOCH ME = 0 On N2 ?
N
Y
1668
Flag Uncontrolled Power To The Field Abort N1 For Safety
1656
Is The N2 SAO Dead Or Did The FIO Comm. Fail ?
Y
1670
Is N1 OOCH ME = 0 ?
N
Y
1672
N
Flag Uncontrolled Power To The Field Abort N2 For Safety
1658
Is N1 OOCH ME = 0 ?
Y
1660
Is N2 OOCH ME = 0 ?
N
1662
Abort N2
N
Y
D
Flag Uncontrolled Power To The Field
1664
C

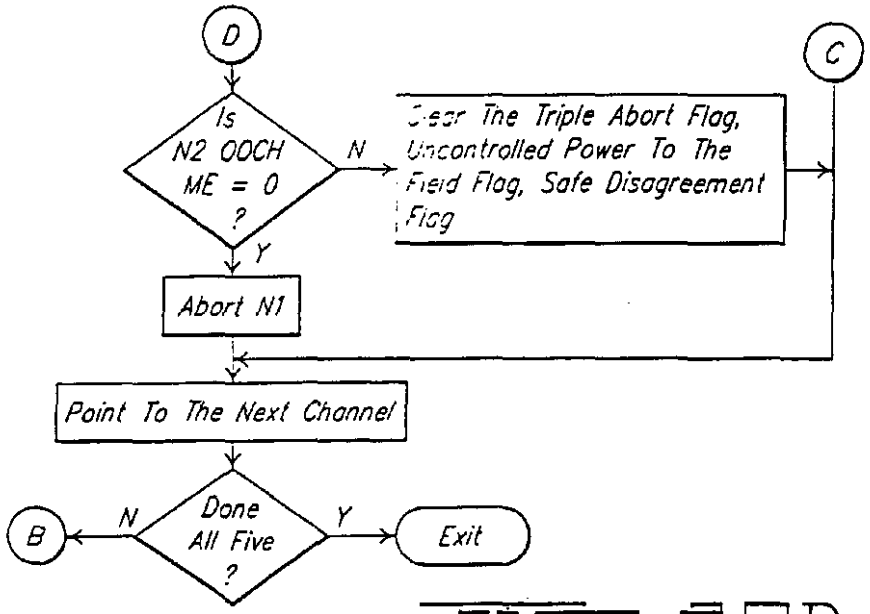

D
Is N2 OOCH ME = 0 ?
N
Clear The Triple Abort Flag, Uncontrolled Power To The Field Flag, Safe Disagreement Flag
C
Y
Abort N1
Point To The Next Channel
Done All Five ?
N
Y
B
Exit

FIG. 2D.

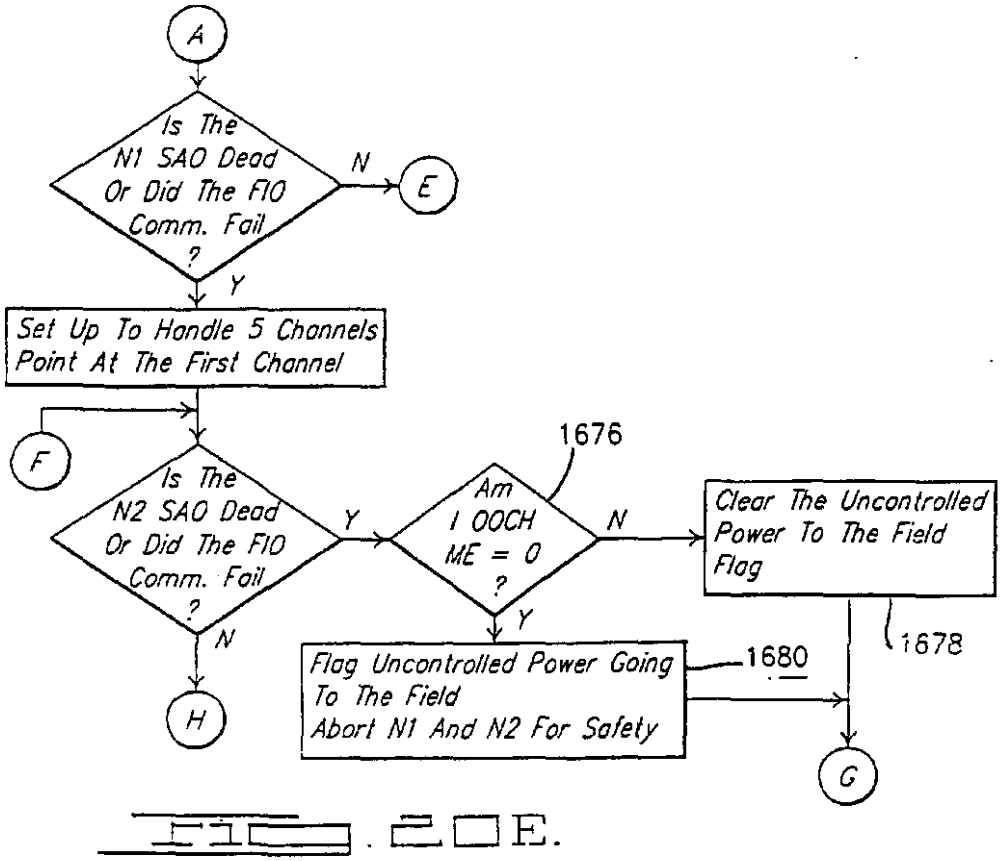

A
Is The N1 SAO Dead Or Did The FIO Comm. Fail ?
N
E
Y
Set Up To Handle 5 Channels Point At The First Channel
F
1676
Is The N2 SAO Dead Or Did The FIO Comm. Fail ?
Y
Am I OOCH ME = 0 ?
N
Clear The Uncontrolled Power To The Field Flag
1678
N
Y
Flag Uncontrolled Power Going To The Field Abort N1 And N2 For Safety
1680
H
G

FIG. 2E.

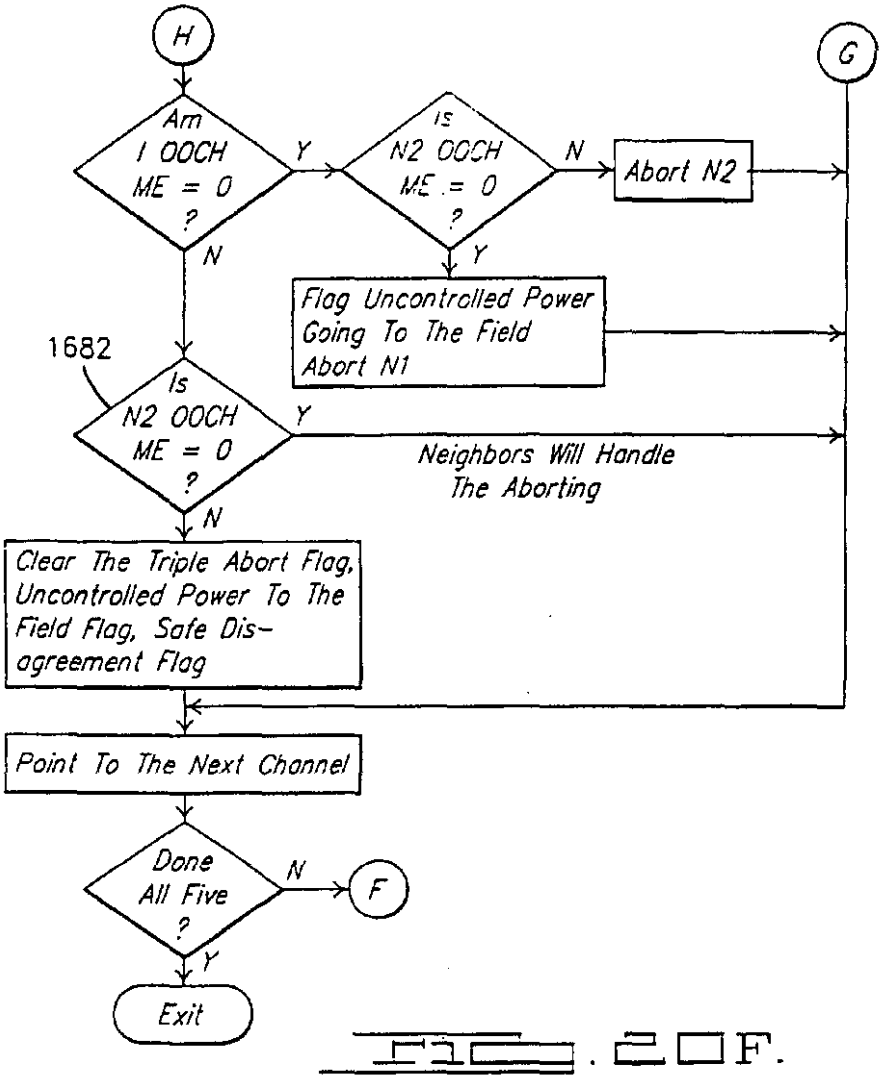


FIG. 2OF.

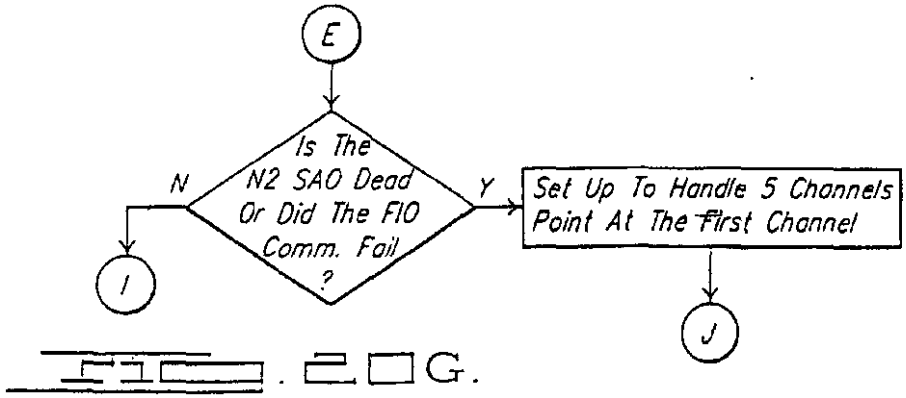


FIG. 2OG.

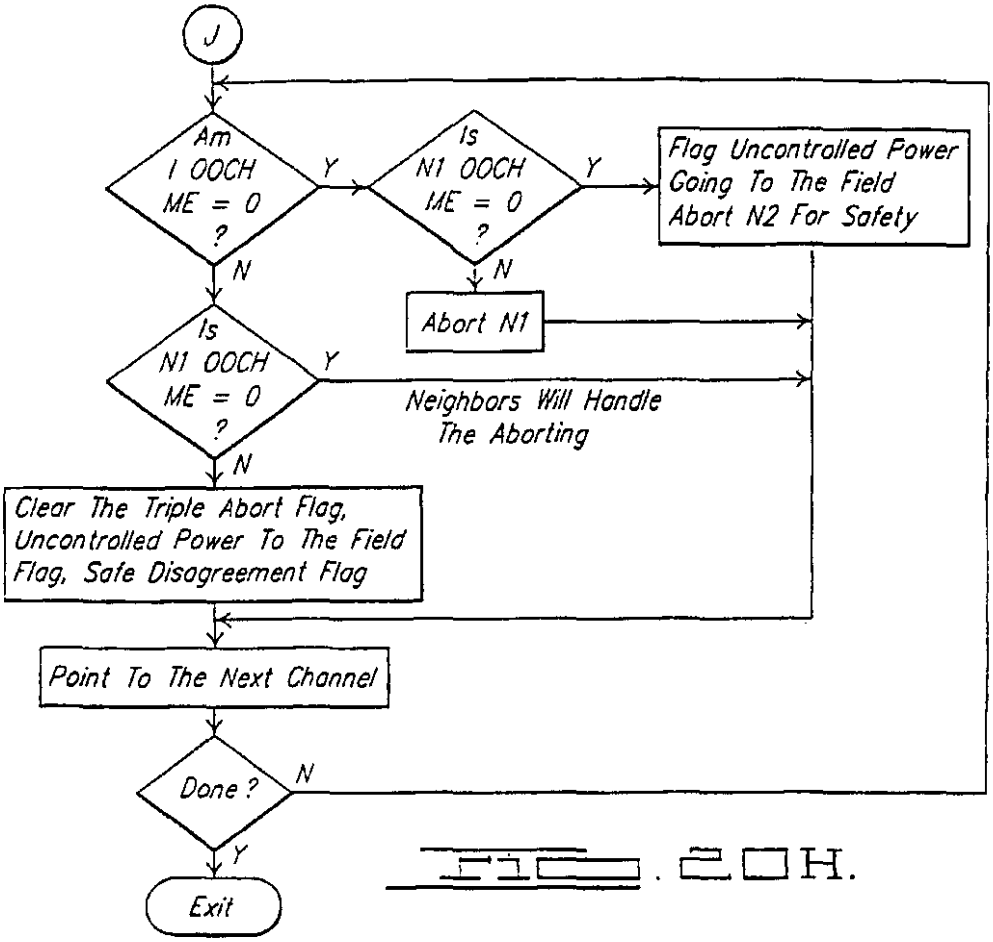


FIG. 20H.

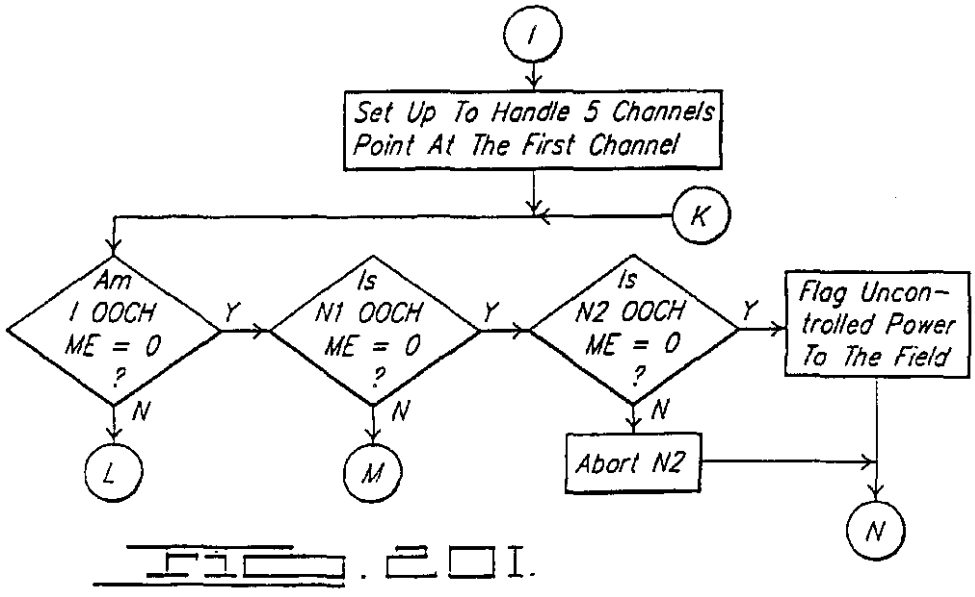


FIG. 20I.

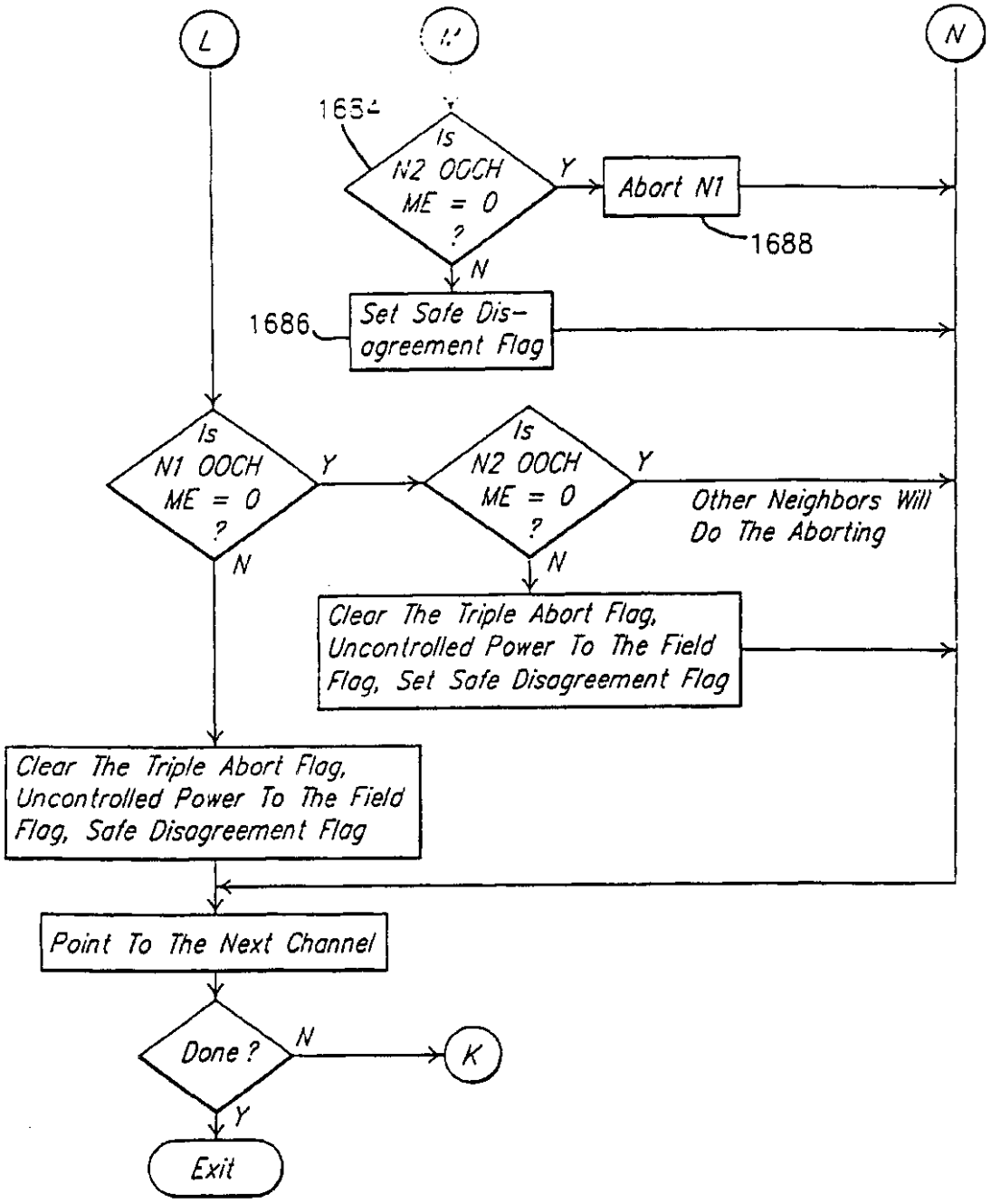

L
N
1634
Is N2 OOCH ME = 0 ?
Y
Abort N1
1688
N
1686
Set Safe Dis-ogreement Flag
Is N1 OOCH ME = 0 ?
Y
Is N2 OOCH ME = 0 ?
Y
Other Neighbors Will Do The Aborting
N
N
Clear The Triple Abort Flag, Uncontrolled Power To The Field Flag, Set Safe Disagreement Flag
Clear The Triple Abort Flag, Uncontrolled Power To The Field Flag, Safe Disagreement Flag
Point To The Next Channel
Done ?
N
K
Y
Exit

FIG. 20J.

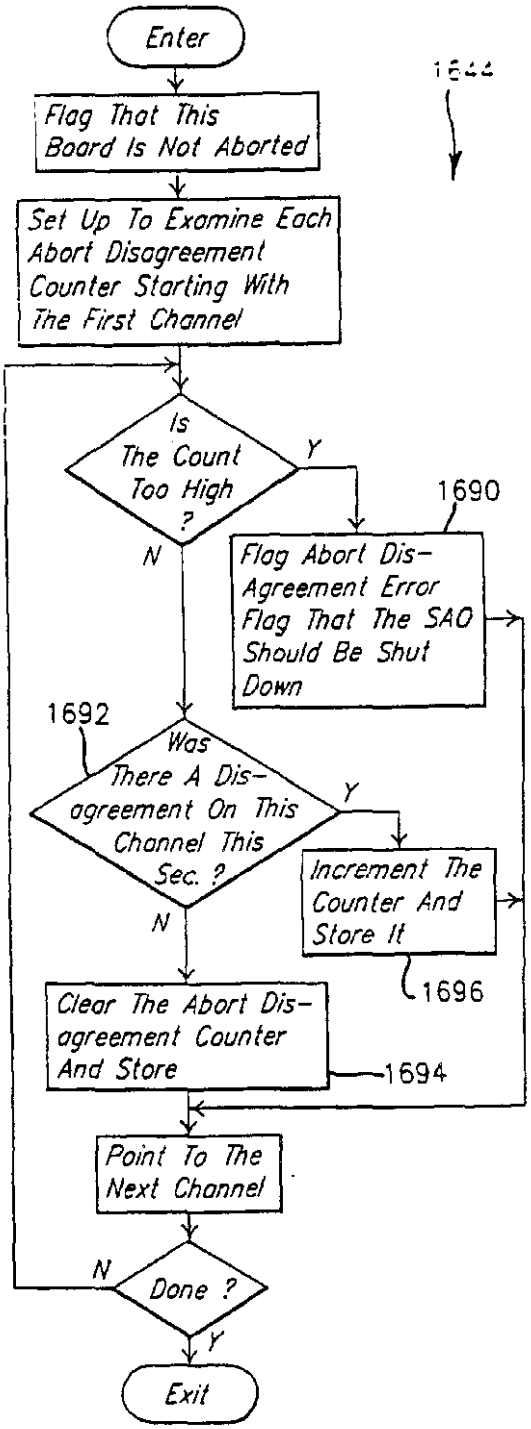


Fig. 20K.

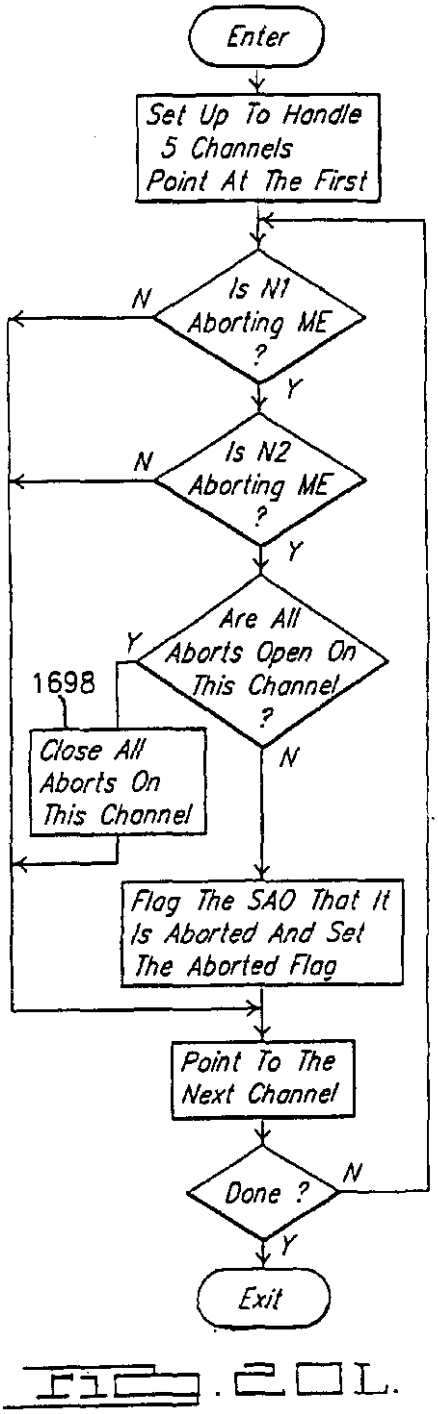


Fig. 20L.

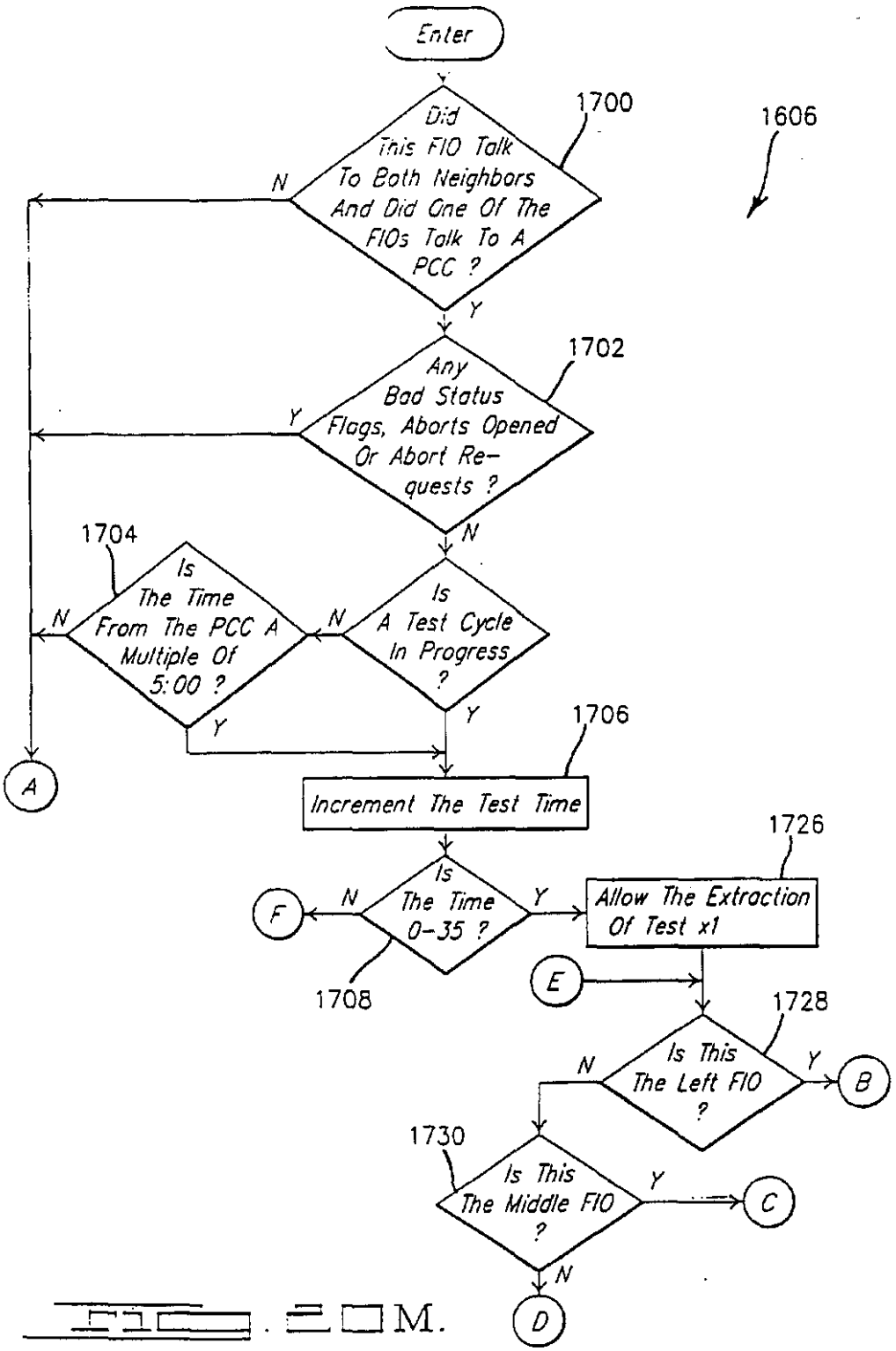

Enter
1700
Did This FIO Talk To Both Neighbors And Did One Of The FIOs Talk To A PCC ?
N
Y
1606
1702
Any Bad Status Flags, Aborts Opened Or Abort Requests ?
Y
N
1704
Is The Time From The PCC A Multiple Of 5:00 ?
N
Is A Test Cycle In Progress ?
N
Y
Y
1706
A
Increment The Test Time
1726
Is The Time 0-35 ?
N
Y
F
1708
Allow The Extraction Of Test x1
E
1728
Is This The Left FIO ?
N
Y
B
1730
Is This The Middle FIO ?
Y
C
N
D

FIG. 2M.

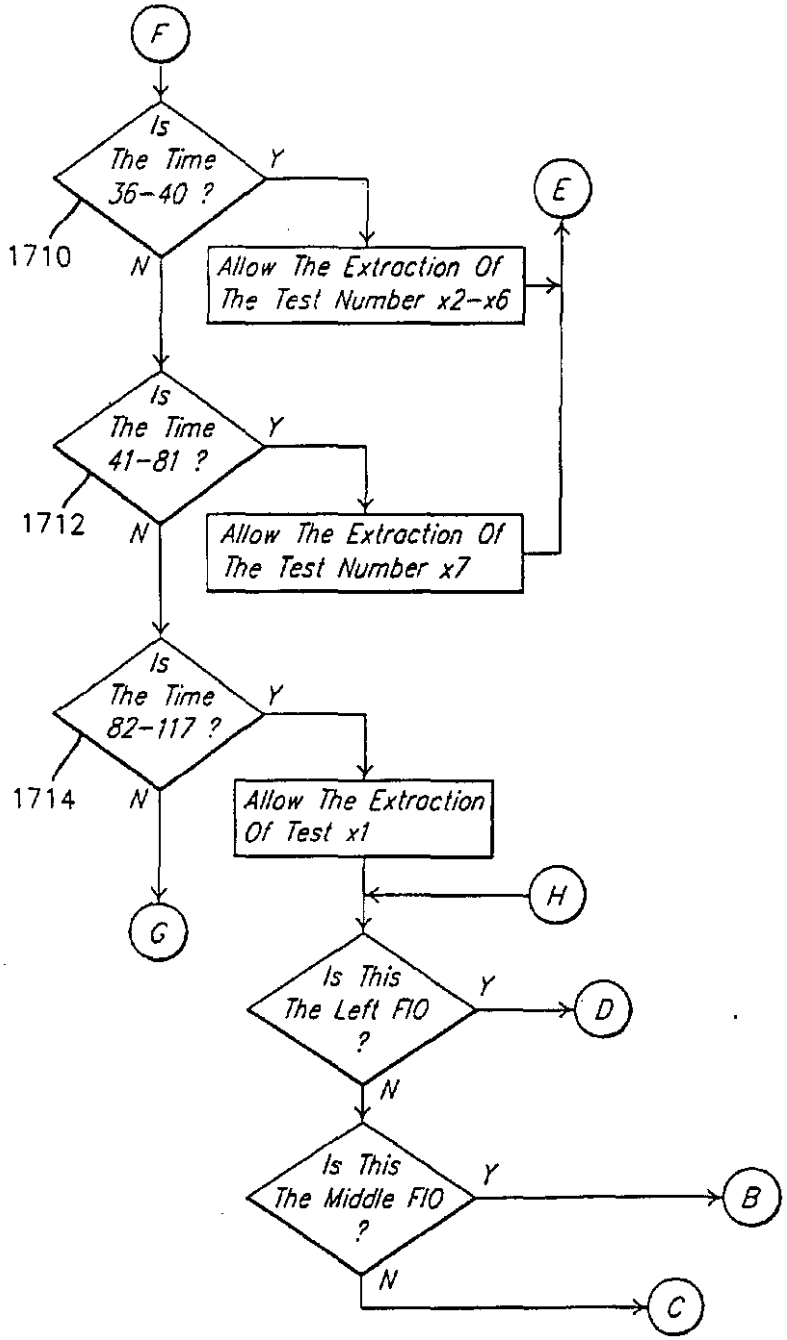

F
Is The Time 36-40 ?
Y
N
1710
Allow The Extraction Of The Test Number x2-x6
E
Is The Time 41-81 ?
Y
N
1712
Allow The Extraction Of The Test Number x7
Is The Time 82-117 ?
Y
N
1714
Allow The Extraction Of Test x1
G
H
Is This The Left FIO ?
Y
D
N
Is This The Middle FIO ?
Y
B
N
C

FIG. 20N.

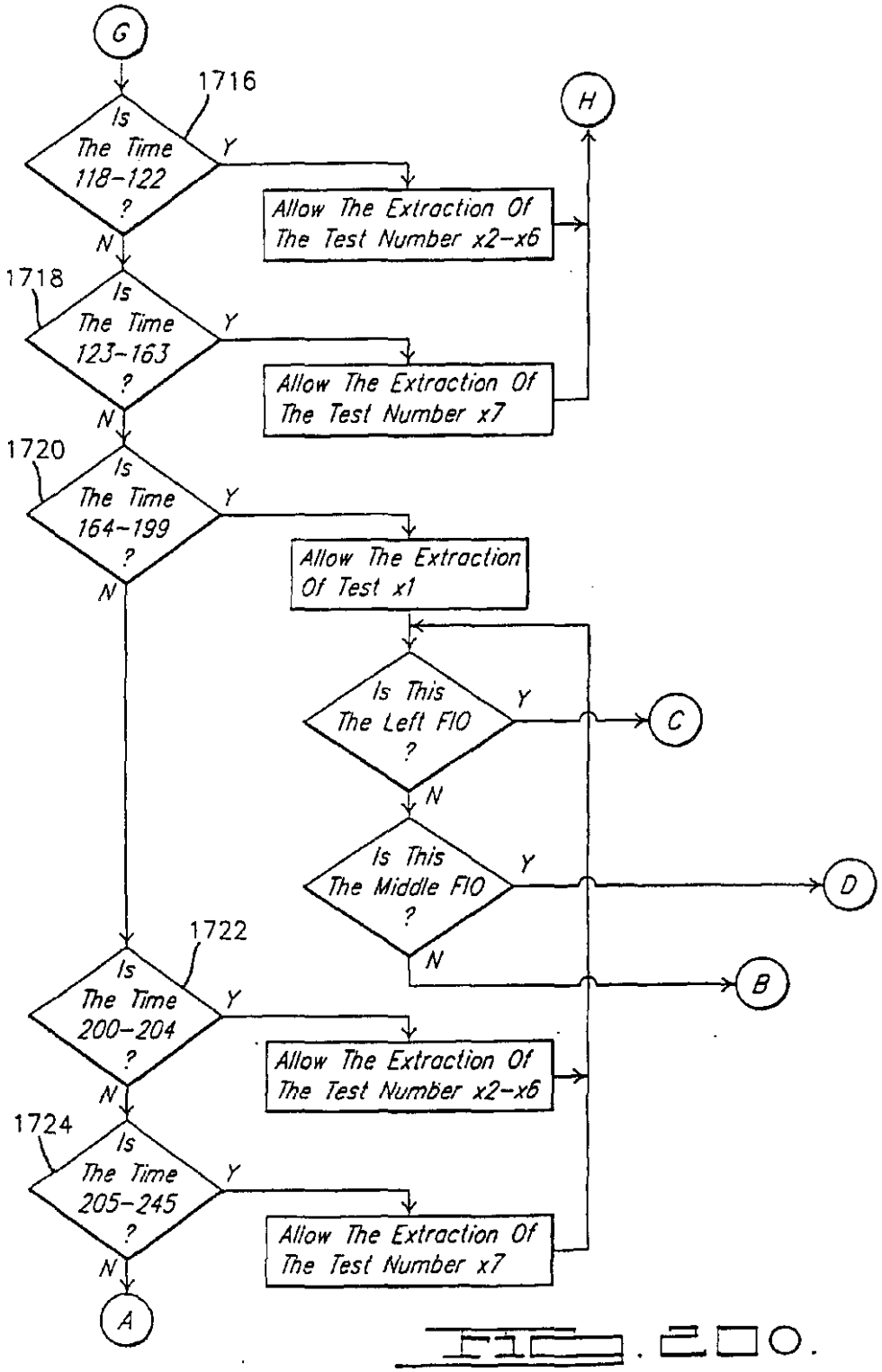

G
1716
Is The Time 118-122 ?
Y
N
H
Allow The Extraction Of The Test Number x2-x6
1718
Is The Time 123-163 ?
Y
N
Allow The Extraction Of The Test Number x7
1720
Is The Time 164-199 ?
Y
N
Allow The Extraction Of Test x1
Is This The Left FIO ?
Y
C
N
Is This The Middle FIO ?
Y
D
N
B
1722
Is The Time 200-204 ?
Y
N
Allow The Extraction Of The Test Number x2-x6
1724
Is The Time 205-245 ?
Y
N
Allow The Extraction Of The Test Number x7
A

Fig. 200.

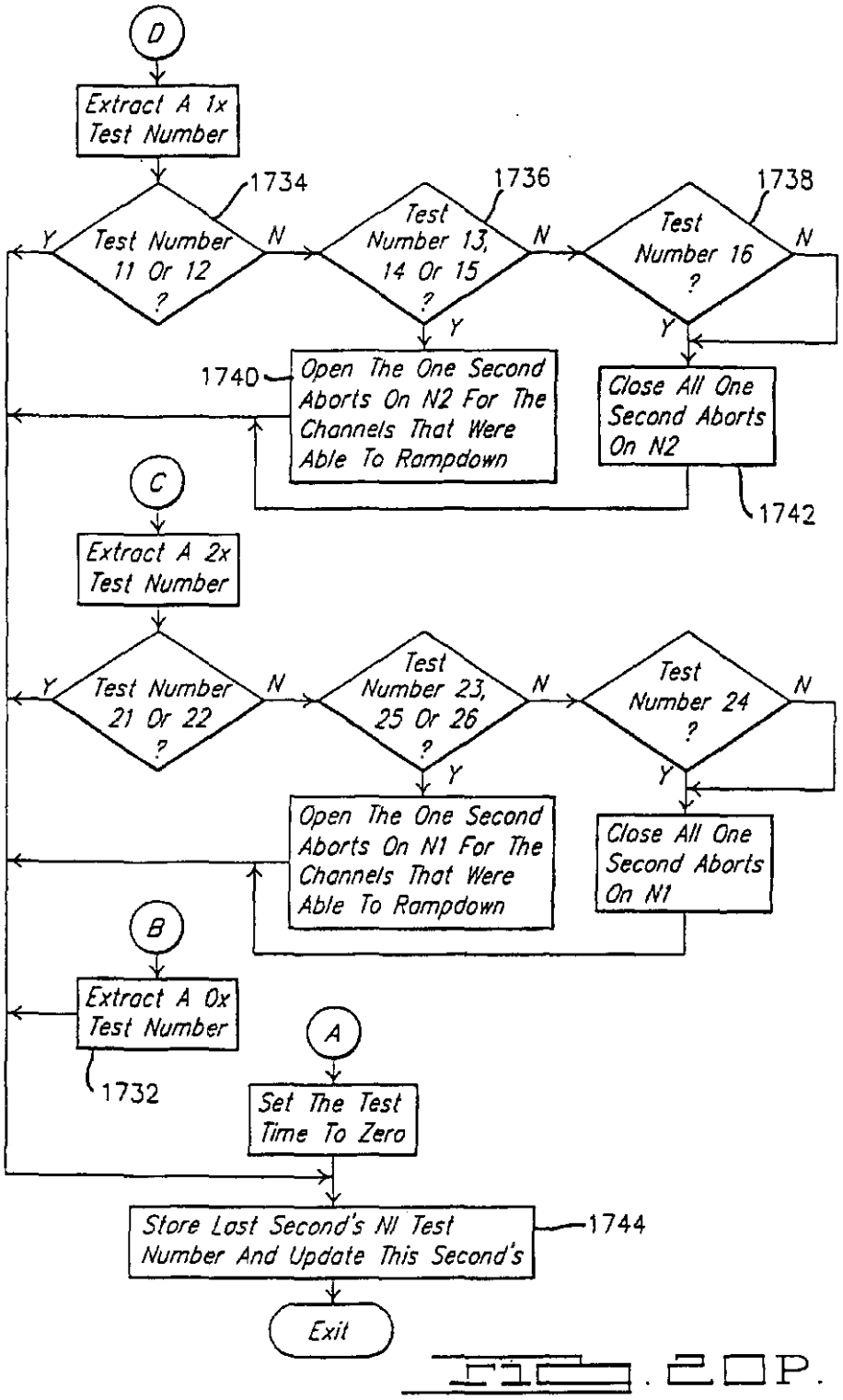

D
Extract A 1x Test Number
1734
Test Number 11 Or 12 ?
1736
Test Number 13, 14 Or 15 ?
1738
Test Number 16 ?
Y
N
1740
Open The One Second Aborts On N2 For The Channels That Were Able To Rampdown
Close All One Second Aborts On N2
1742
C
Extract A 2x Test Number
Test Number 21 Or 22 ?
Test Number 23, 25 Or 26 ?
Test Number 24 ?
Open The One Second Aborts On N1 For The Channels That Were Able To Rampdown
Close All One Second Aborts On N1
B
Extract A 0x Test Number
1732
A
Set The Test Time To Zero
Store Last Second's N1 Test Number And Update This Second's
1744
Exit

FIG. 20P.

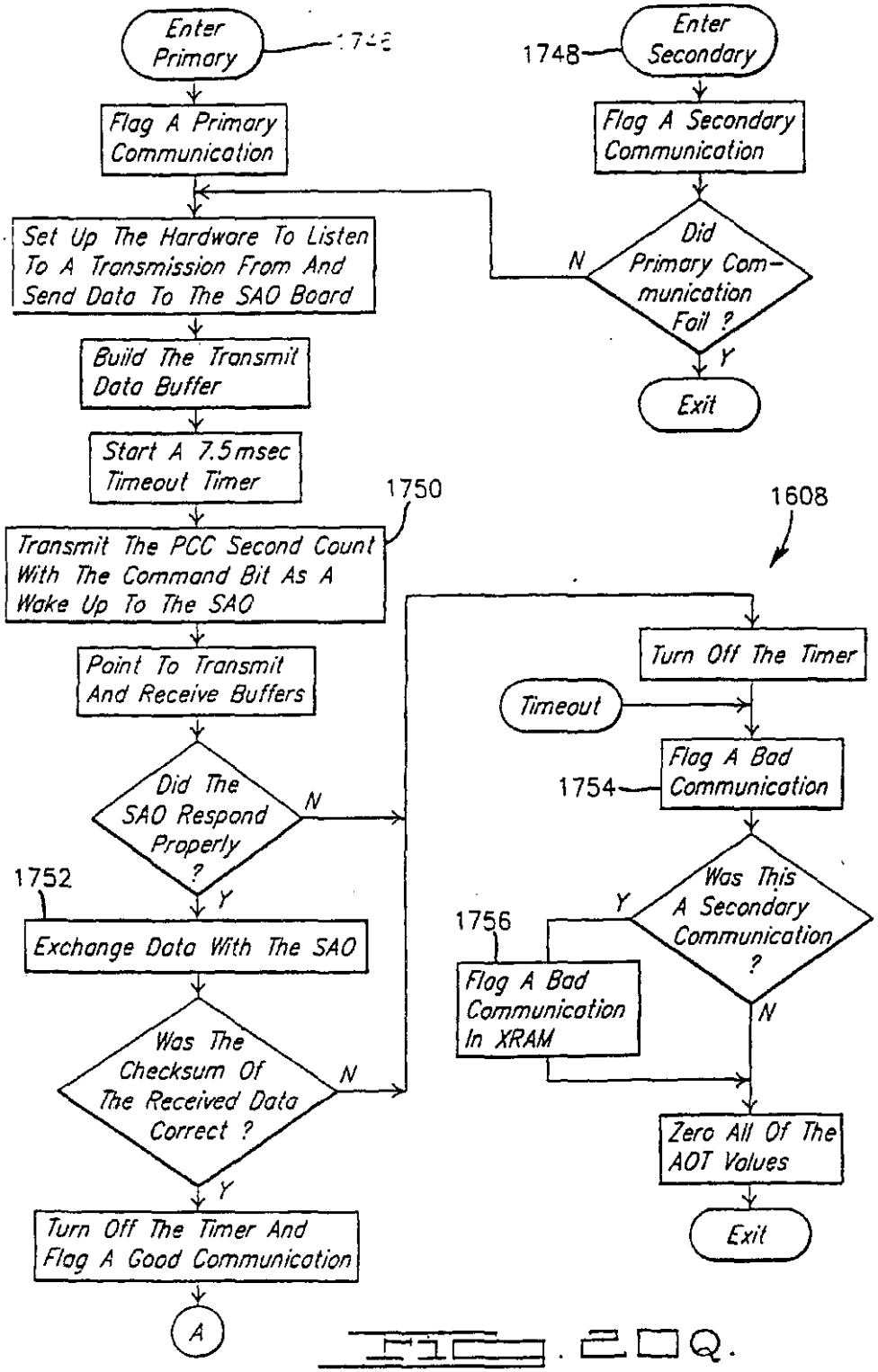

Enter Primary
1748
Enter Secondary
Flag A Primary Communication
Flag A Secondary Communication
Set Up The Hardware To Listen To A Transmission From And Send Data To The SAO Board
Did Primary Communication Fail ?
N
Y
Exit
Build The Transmit Data Buffer
Start A 7.5msec Timeout Timer
1750
1608
Transmit The PCC Second Count With The Command Bit As A Wake Up To The SAO
Turn Off The Timer
Timeout
Point To Transmit And Receive Buffers
Flag A Bad Communication
1754
Did The SAO Respond Properly ?
N
Y
1752
Was This A Secondary Communication ?
Y
1756
N
Exchange Data With The SAO
Flag A Bad Communication In XRAM
Was The Checksum Of The Received Data Correct ?
N
Zero All Of The AOT Values
Y
Exit
Turn Off The Timer And Flag A Good Communication
A

FIG. 20Q.

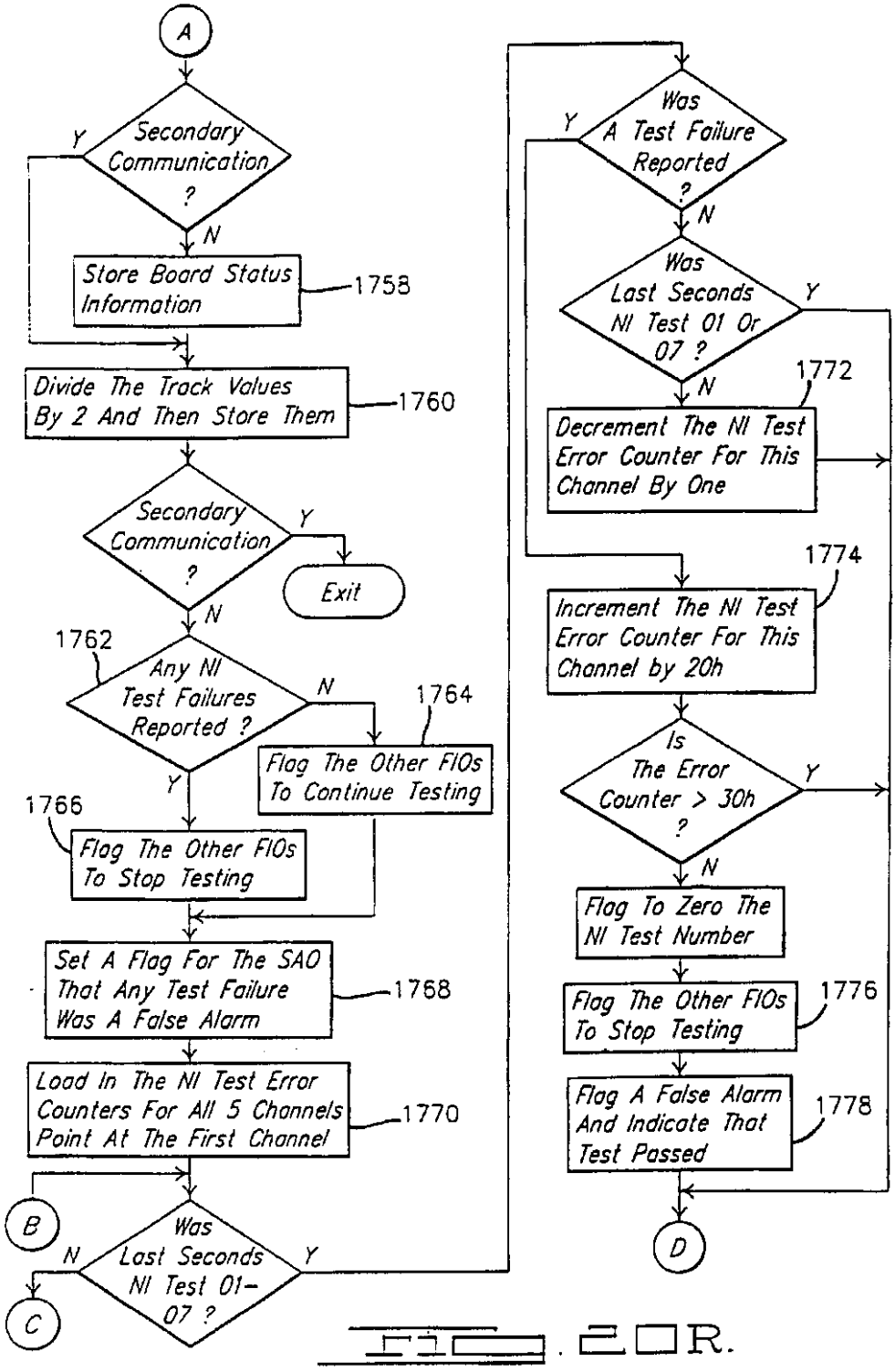
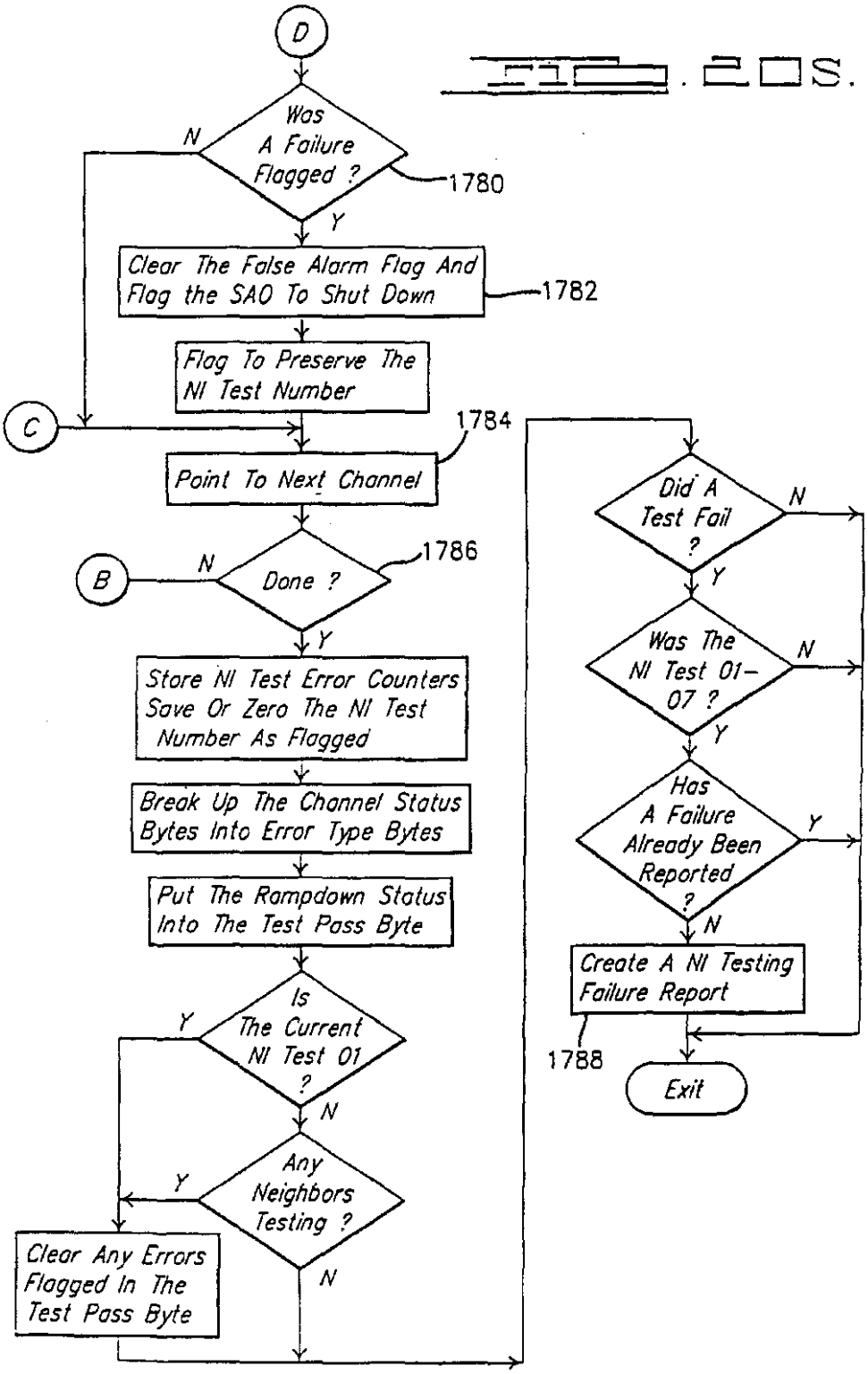

A
Secondary Communication ?
Y
N
Store Board Status Information
1758
Divide The Track Values By 2 And Then Store Them
1760
Secondary Communication ?
Y
Exit
N
1762
Any NI Test Failures Reported ?
N
1764
Flag The Other FIOs To Continue Testing
Y
1766
Flag The Other FIOs To Stop Testing
Set A Flag For The SAO That Any Test Failure Was A False Alarm
1768
Load In The NI Test Error Counters For All 5 Channels Point At The First Channel
1770
B
Was Last Seconds NI Test 01-07 ?
N
Y
C
Was A Test Failure Reported ?
Y
N
Was Last Seconds NI Test 01 Or 07 ?
Y
N
1772
Decrement The NI Test Error Counter For This Channel By One
1774
Increment The NI Test Error Counter For This Channel by 20h
Is The Error Counter > 30h ?
Y
N
Flag To Zero The NI Test Number
Flag The Other FIOs To Stop Testing
1776
Flag A False Alarm And Indicate That Test Passed
1778
D

FIG. 20R.

FIG. 20S.


D
Was A Failure Flagged ?
N
Y
1780
Clear The False Alarm Flag And Flag the SAO To Shut Down
1782
Flag To Preserve The NI Test Number
C
1784
Point To Next Channel
B
N
Done ?
1786
Y
Store NI Test Error Counters Save Or Zero The NI Test Number As Flagged
Break Up The Channel Status Bytes Into Error Type Bytes
Put The Rampdown Status Into The Test Pass Byte
Is The Current NI Test 01 ?
Y
N
Any Neighbors Testing ?
Y
N
Clear Any Errors Flagged In The Test Pass Byte
Did A Test Fail ?
N
Y
Was The NI Test 01-07 ?
N
Y
Has A Failure Already Been Reported ?
Y
N
Create A NI Testing Failure Report
1788
Exit

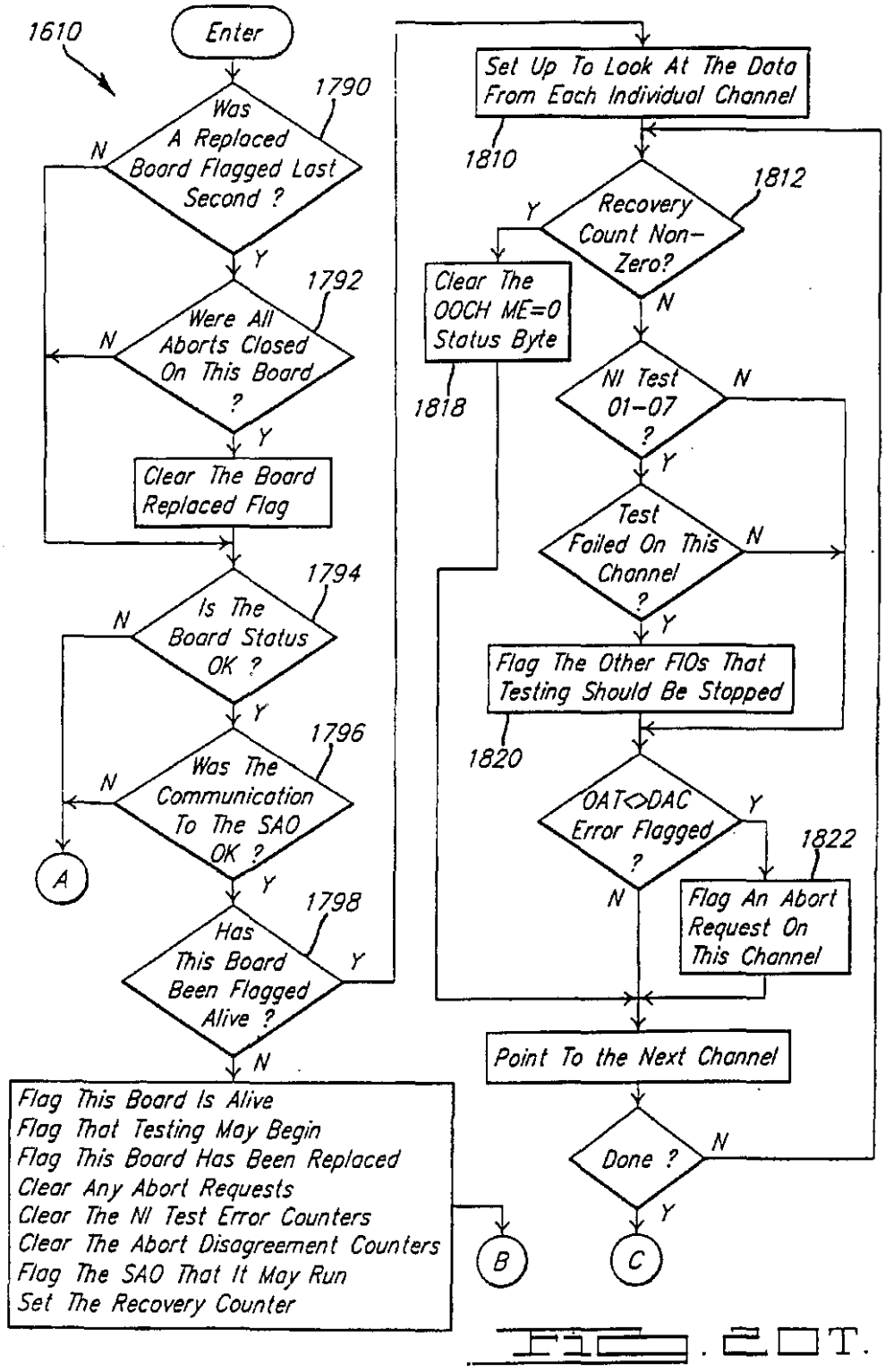

1610
Enter
1790
Was A Replaced Board Flagged Last Second ?
N
Y
1792
Were All Aborts Closed On This Board ?
N
Y
Clear The Board Replaced Flag
1794
Is The Board Status OK ?
N
Y
1796
Was The Communication To The SAO OK ?
N
Y
A
1798
Has This Board Been Flagged Alive ?
Y
N
Flag This Board Is Alive
Flag That Testing May Begin
Flag This Board Has Been Replaced
Clear Any Abort Requests
Clear The NI Test Error Counters
Clear The Abort Disagreement Counters
Flag The SAO That It May Run
Set The Recovery Counter
B
Set Up To Look At The Data From Each Individual Channel
1810
1812
Recovery Count Non-Zero?
Y
N
Clear The OOCH ME=0 Status Byte
1818
NI Test 01-07 ?
N
Y
Test Failed On This Channel ?
N
Y
Flag The Other FIOs That Testing Should Be Stopped
1820
OAT<>DAC Error Flagged ?
Y
N
1822
Flag An Abort Request On This Channel
Point To the Next Channel
Done ?
N
Y
C

Fig. 20T.

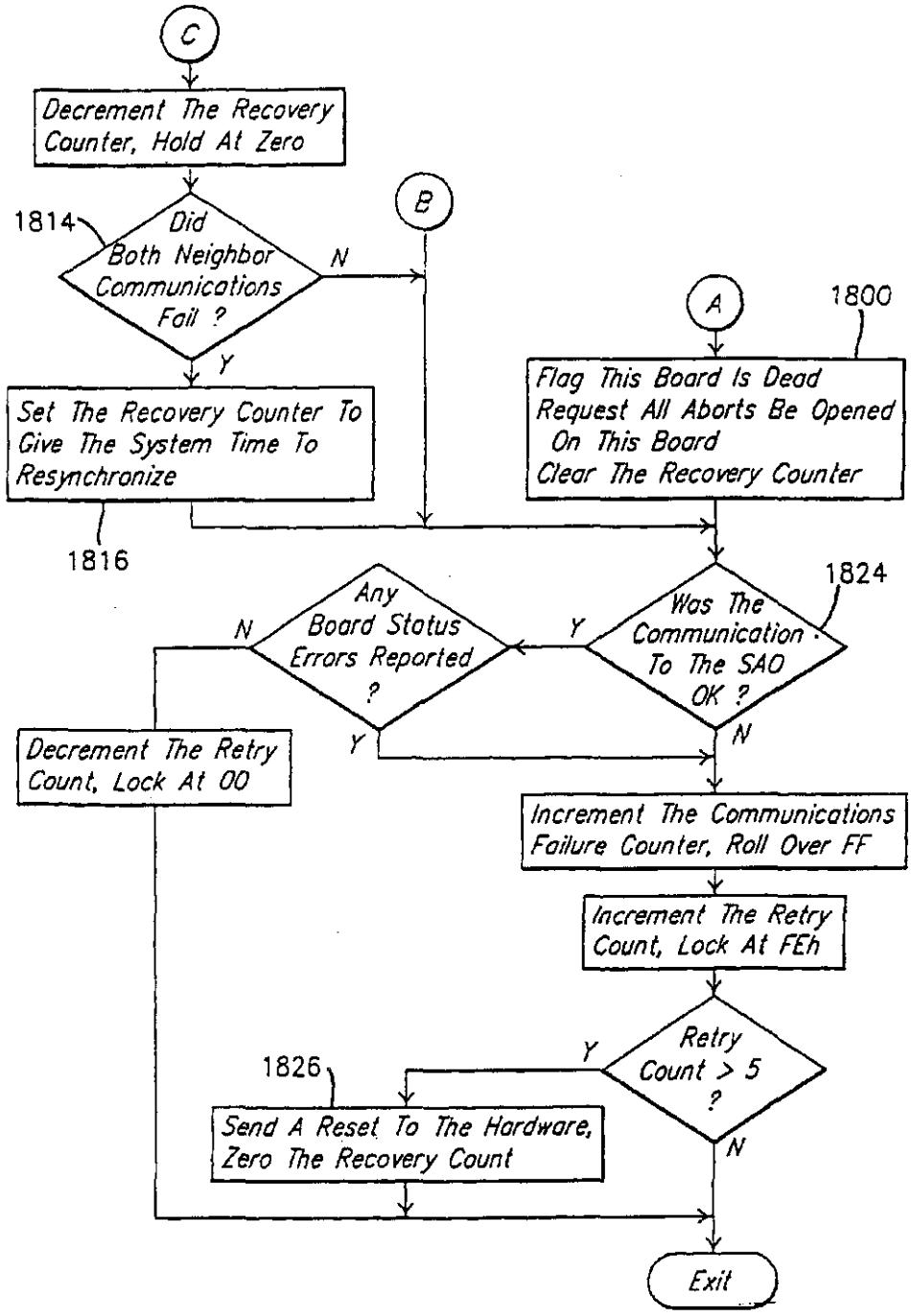

C
Decrement The Recovery Counter, Hold At Zero
1814
Did Both Neighbor Communications Fail ?
N
Y
B
Set The Recovery Counter To Give The System Time To Resynchronize
1816
A
1800
Flag This Board Is Dead
Request All Aborts Be Opened On This Board
Clear The Recovery Counter
1824
Was The Communication To The SAO OK ?
Y
N
Any Board Status Errors Reported ?
N
Y
Decrement The Retry Count, Lock At 00
Increment The Communications Failure Counter, Roll Over FF
Increment The Retry Count, Lock At FEh
Retry Count > 5 ?
Y
N
1826
Send A Reset To The Hardware, Zero The Recovery Count
Exit

FIG. 20U.

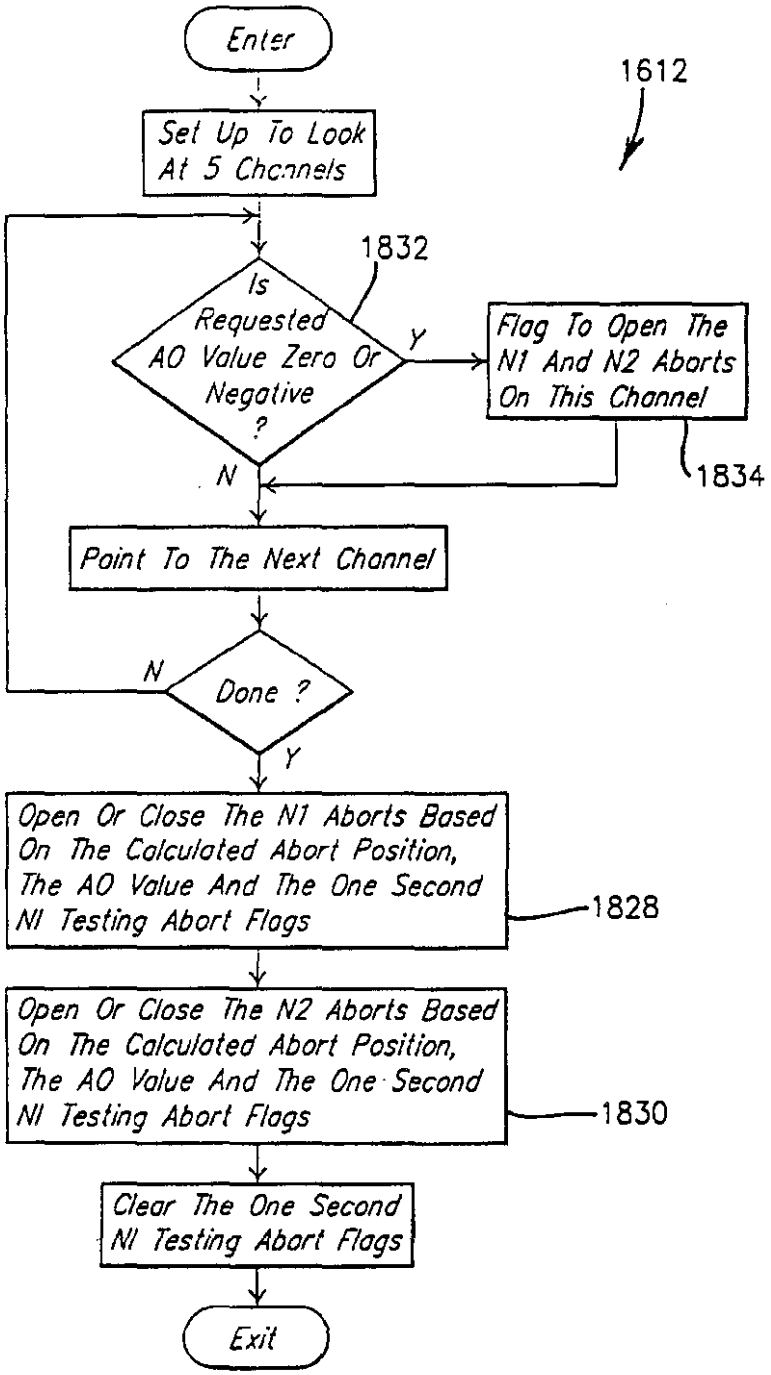
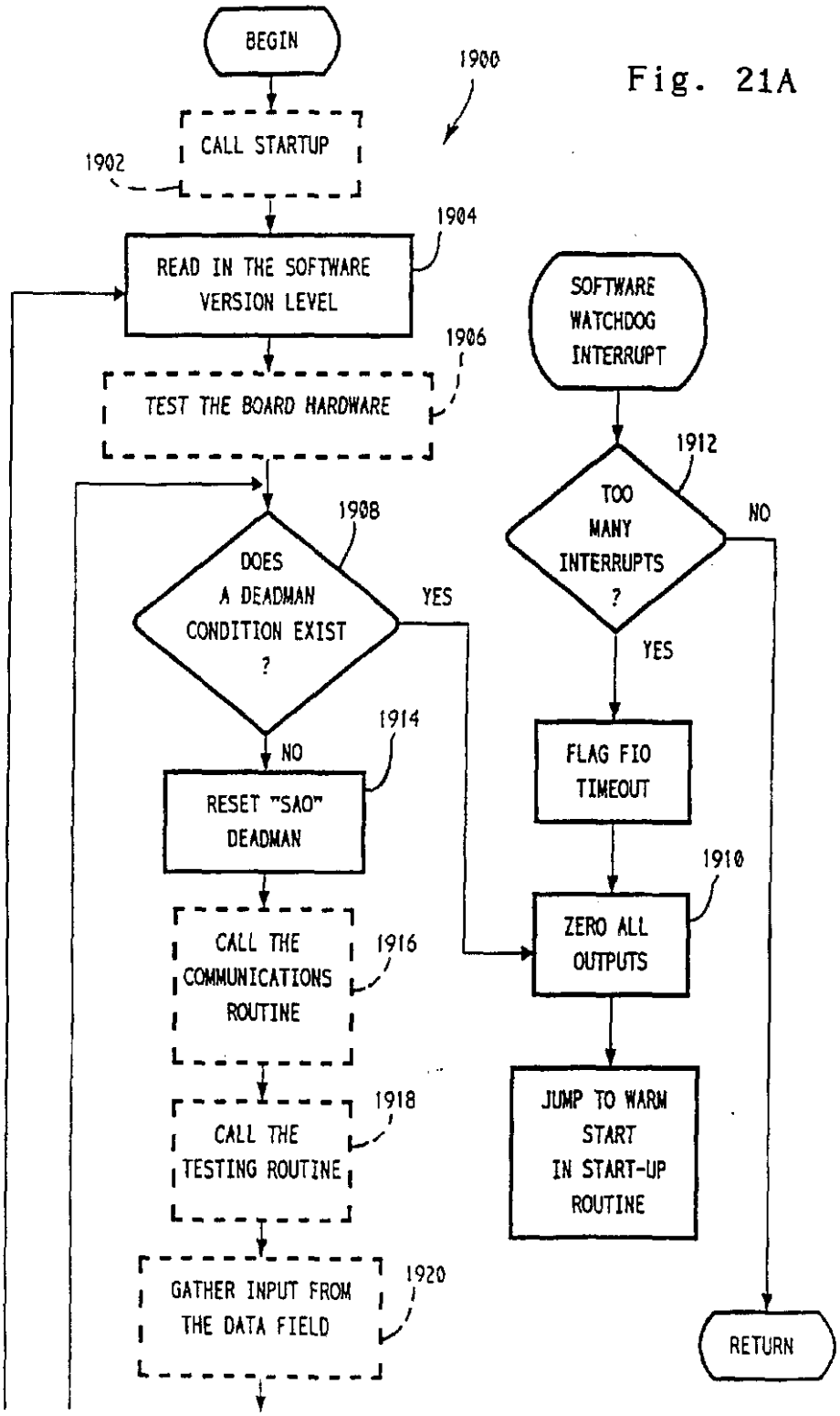

Enter
1612
Set Up To Look At 5 Channels
1832
Is Requested AO Value Zero Or Negative ?
Y
Flag To Open The N1 And N2 Aborts On This Channel
1834
N
Point To The Next Channel
N
Done ?
Y
Open Or Close The N1 Aborts Based On The Calculated Abort Position, The AO Value And The One Second N1 Testing Abort Flags
1828
Open Or Close The N2 Aborts Based On The Calculated Abort Position, The AO Value And The One Second N1 Testing Abort Flags
1830
Clear The One Second N1 Testing Abort Flags
Exit

Fig. 20V.

Fig. 21A


BEGIN
1900
CALL STARTUP
1902
1904
READ IN THE SOFTWARE VERSION LEVEL
1906
TEST THE BOARD HARDWARE
1908
DOES A DEADMAN CONDITION EXIST ?
YES
NO
1914
RESET "SAO" DEADMAN
CALL THE COMMUNICATIONS ROUTINE
1916
CALL THE TESTING ROUTINE
1918
GATHER INPUT FROM THE DATA FIELD
1920
SOFTWARE WATCHDOG INTERRUPT
1912
TOO MANY INTERRUPTS ?
NO
YES
FLAG FIO TIMEOUT
1910
ZERO ALL OUTPUTS
JUMP TO WARM START IN START-UP ROUTINE
RETURN

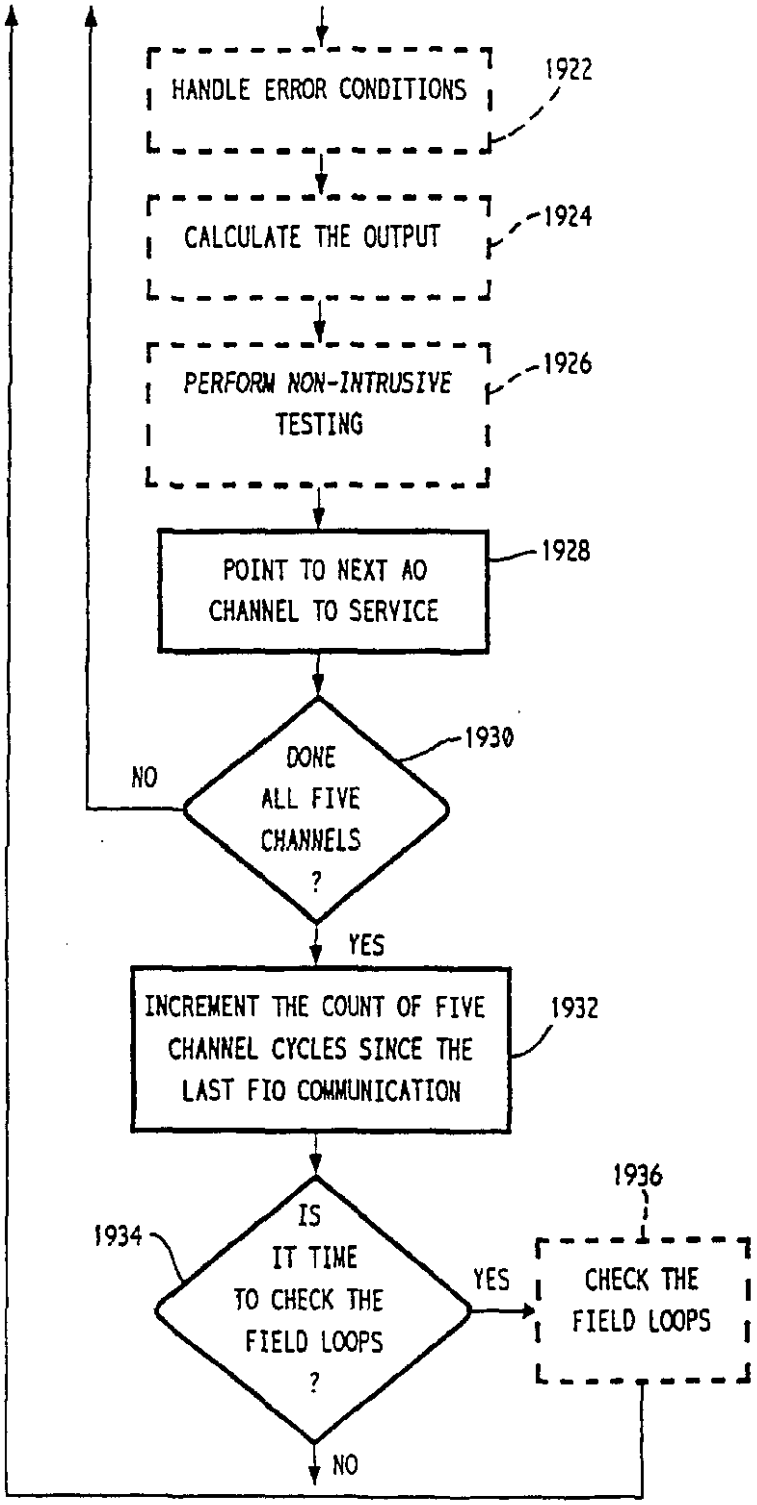

HANDLE ERROR CONDITIONS
1922
CALCULATE THE OUTPUT
1924
PERFORM NON-INTRUSIVE TESTING
1926
POINT TO NEXT AO CHANNEL TO SERVICE
1928
DONE ALL FIVE CHANNELS ?
1930
NO
YES
INCREMENT THE COUNT OF FIVE CHANNEL CYCLES SINCE THE LAST FIO COMMUNICATION
1932
IS IT TIME TO CHECK THE FIELD LOOPS ?
1934
YES
NO
CHECK THE FIELD LOOPS
1936

Fig. 21B

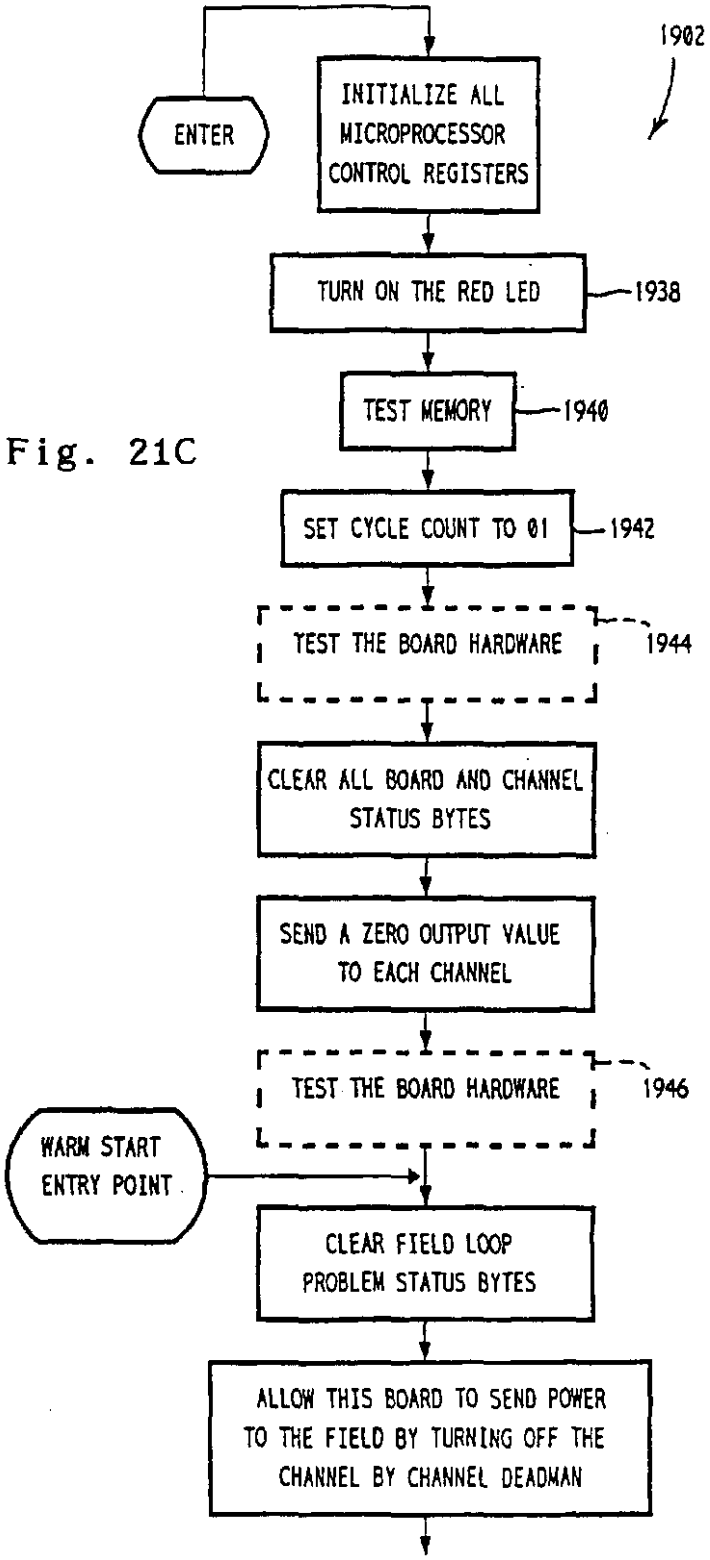

1902
ENTER
INITIALIZE ALL MICROPROCESSOR CONTROL REGISTERS
TURN ON THE RED LED
1938
TEST MEMORY
1940
SET CYCLE COUNT TO 01
1942
TEST THE BOARD HARDWARE
1944
CLEAR ALL BOARD AND CHANNEL STATUS BYTES
SEND A ZERO OUTPUT VALUE TO EACH CHANNEL
TEST THE BOARD HARDWARE
1946
WARM START ENTRY POINT
CLEAR FIELD LOOP PROBLEM STATUS BYTES
ALLOW THIS BOARD TO SEND POWER TO THE FIELD BY TURNING OFF THE CHANNEL BY CHANNEL DEADMAN

Fig. 21C

Fig. 21D

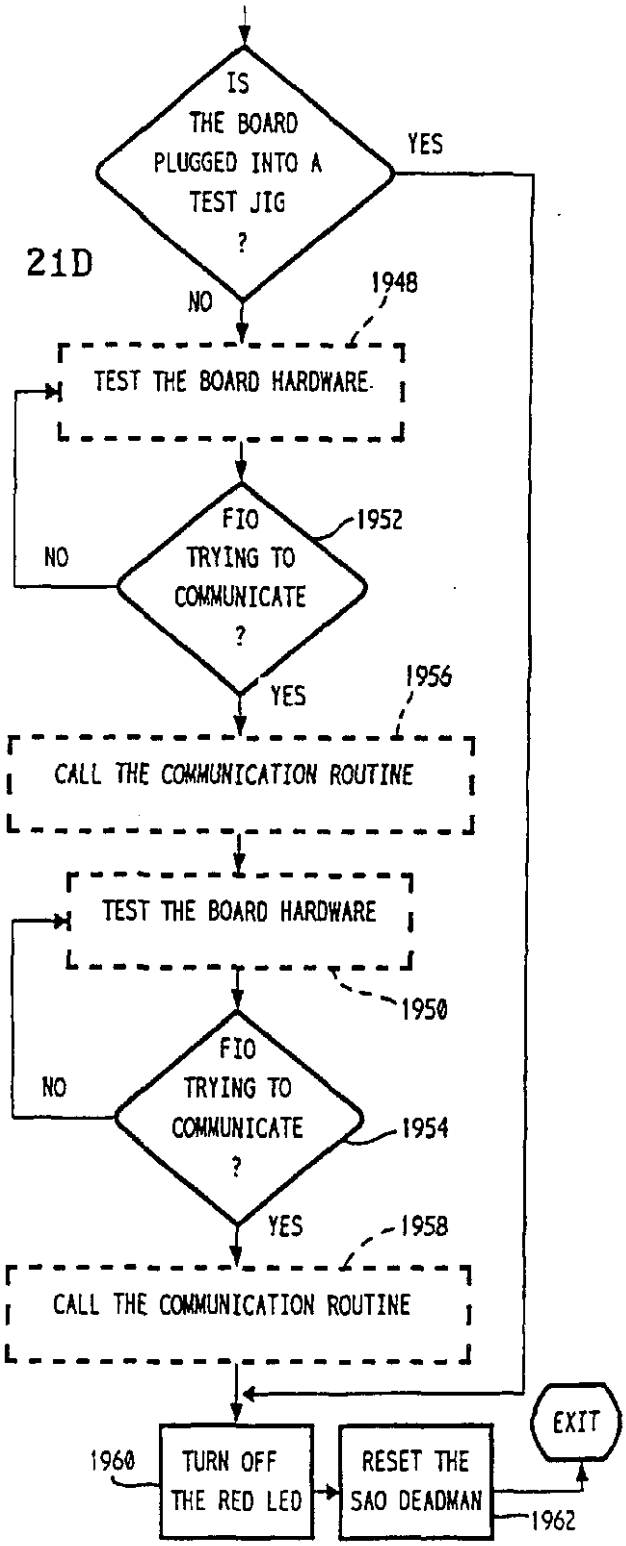

IS THE BOARD PLUGGED INTO A TEST JIG ?
YES
NO
1948
TEST THE BOARD HARDWARE
FIO TRYING TO COMMUNICATE ?
1952
NO
YES
1956
CALL THE COMMUNICATION ROUTINE
TEST THE BOARD HARDWARE
1950
FIO TRYING TO COMMUNICATE ?
NO
1954
YES
1958
CALL THE COMMUNICATION ROUTINE
EXIT
1960
TURN OFF THE RED LED
RESET THE SAO DEADMAN
1962

Fig. 21E

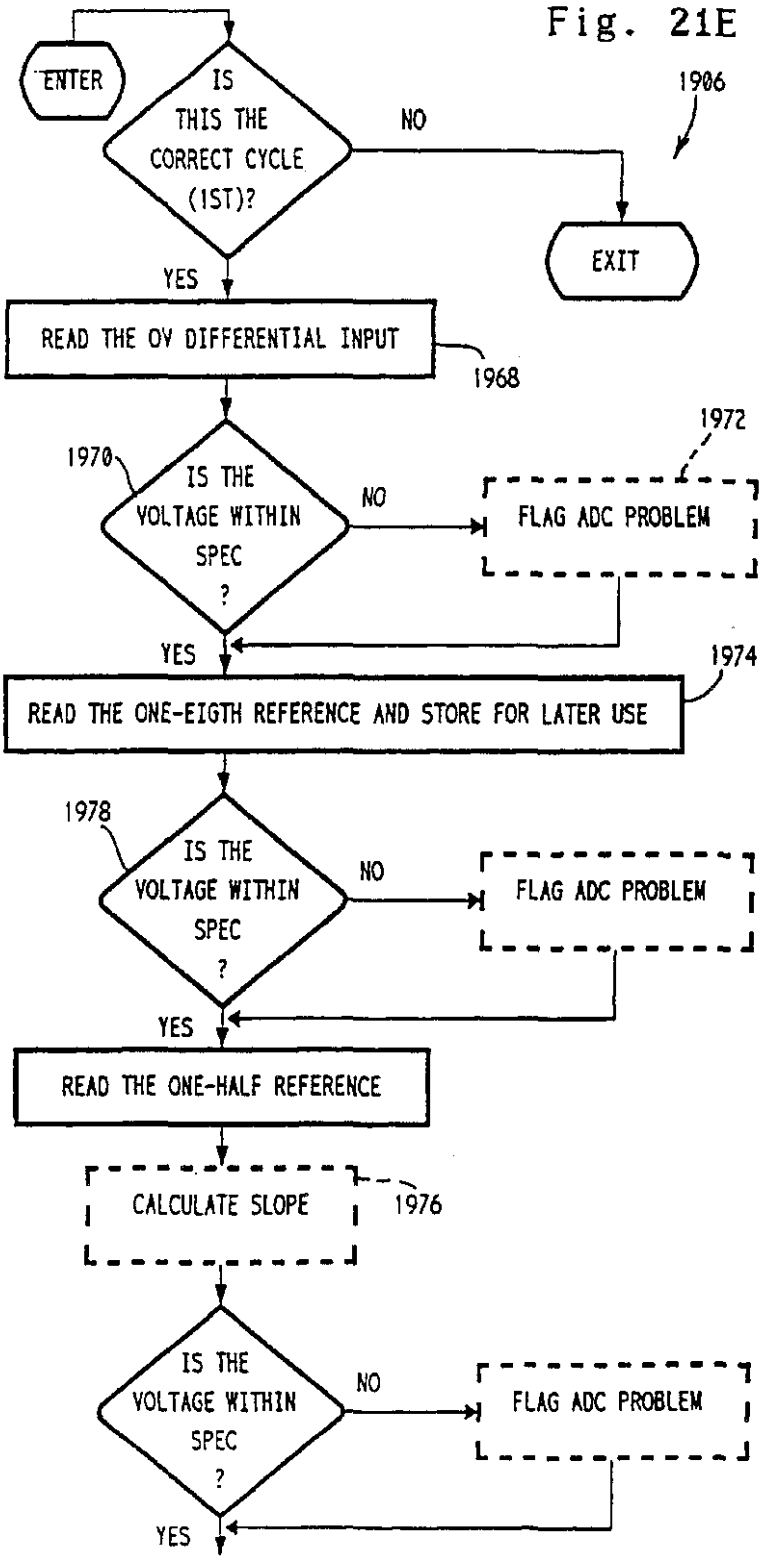
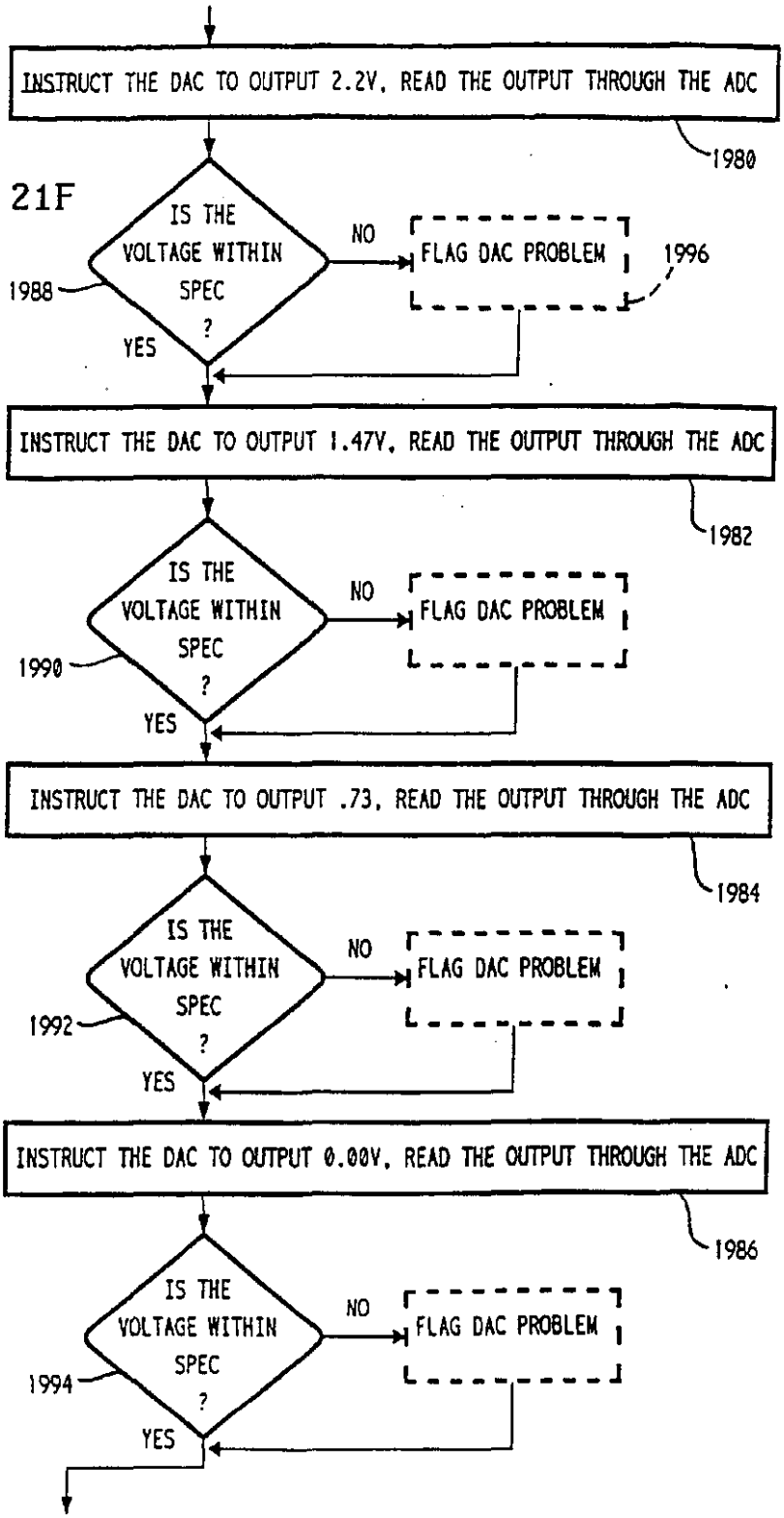

ENTER
IS THIS THE CORRECT CYCLE (1ST)?
NO
1906
EXIT
YES
READ THE OV DIFFERENTIAL INPUT
1968
1970
IS THE VOLTAGE WITHIN SPEC ?
NO
1972
FLAG ADC PROBLEM
YES
1974
READ THE ONE-EIGTH REFERENCE AND STORE FOR LATER USE
1978
IS THE VOLTAGE WITHIN SPEC ?
NO
FLAG ADC PROBLEM
YES
READ THE ONE-HALF REFERENCE
CALCULATE SLOPE
1976
IS THE VOLTAGE WITHIN SPEC ?
NO
FLAG ADC PROBLEM
YES

Fig. 21F


INSTRUCT THE DAC TO OUTPUT 2.2V, READ THE OUTPUT THROUGH THE ADC
1980
IS THE VOLTAGE WITHIN SPEC ?
1988
NO
FLAG DAC PROBLEM
1996
YES
INSTRUCT THE DAC TO OUTPUT 1.47V, READ THE OUTPUT THROUGH THE ADC
1982
IS THE VOLTAGE WITHIN SPEC ?
1990
NO
FLAG DAC PROBLEM
YES
INSTRUCT THE DAC TO OUTPUT .73, READ THE OUTPUT THROUGH THE ADC
1984
IS THE VOLTAGE WITHIN SPEC ?
1992
NO
FLAG DAC PROBLEM
YES
INSTRUCT THE DAC TO OUTPUT 0.00V, READ THE OUTPUT THROUGH THE ADC
1986
IS THE VOLTAGE WITHIN SPEC ?
1994
NO
FLAG DAC PROBLEM
YES

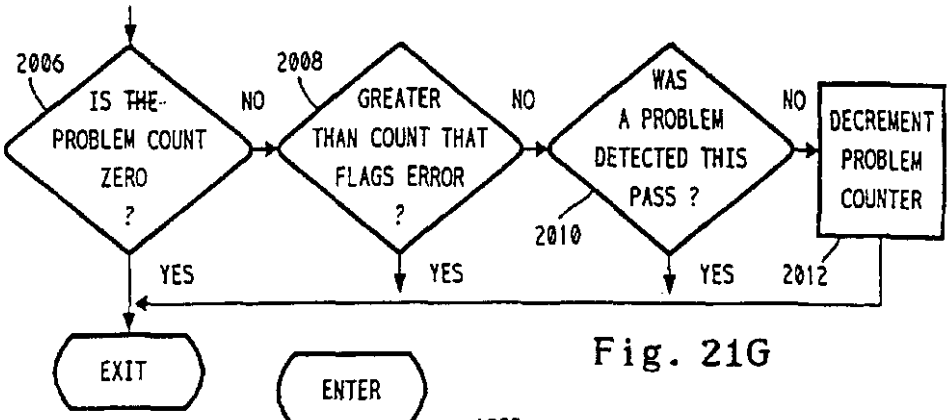


Fig. 21G

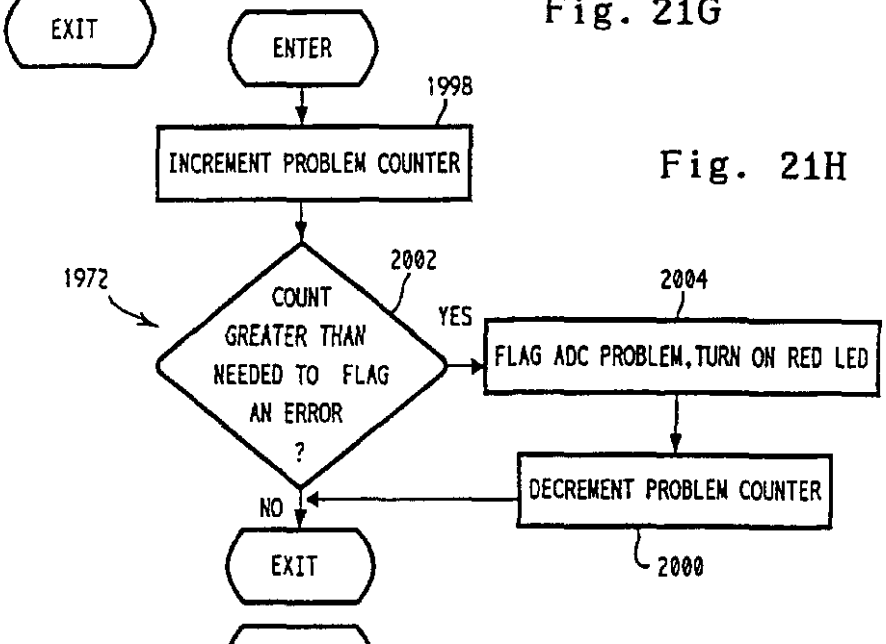


Fig. 21H

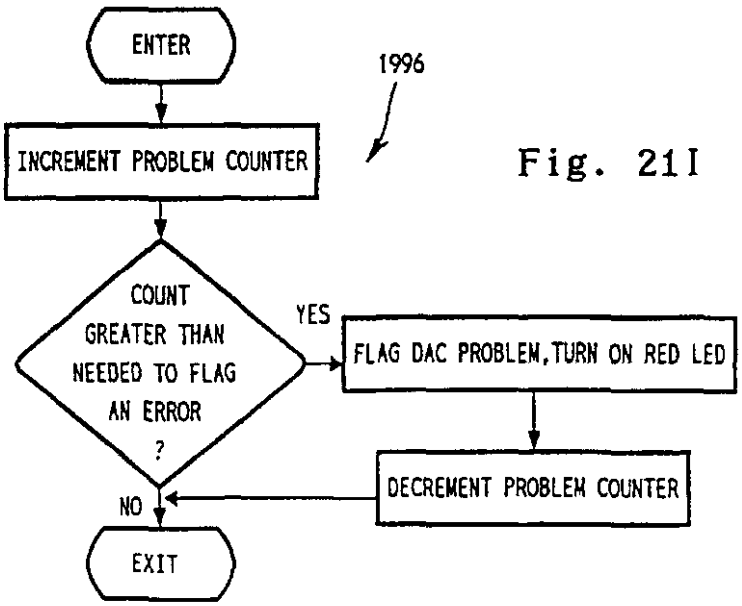


Fig. 21I

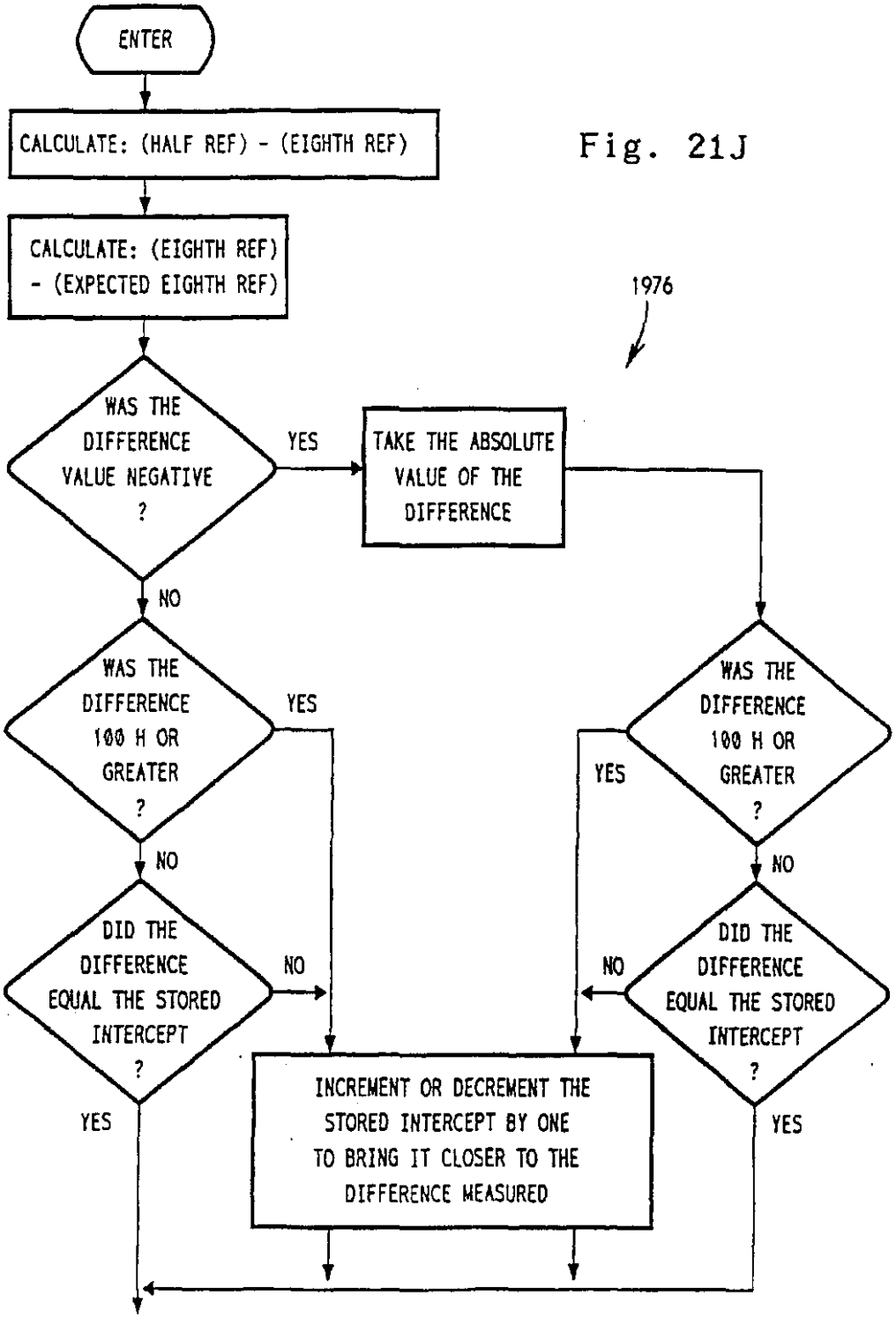

ENTER
CALCULATE: (HALF REF) - (EIGHTH REF)
CALCULATE: (EIGHTH REF) - (EXPECTED EIGHTH REF)
1976
WAS THE DIFFERENCE VALUE NEGATIVE ?
YES
TAKE THE ABSOLUTE VALUE OF THE DIFFERENCE
NO
WAS THE DIFFERENCE 100 H OR GREATER ?
YES
NO
DID THE DIFFERENCE EQUAL THE STORED INTERCEPT ?
NO
YES
WAS THE DIFFERENCE 100 H OR GREATER ?
YES
NO
DID THE DIFFERENCE EQUAL THE STORED INTERCEPT ?
NO
YES
INCREMENT OR DECREMENT THE STORED INTERCEPT BY ONE TO BRING IT CLOSER TO THE DIFFERENCE MEASURED

Fig. 21J

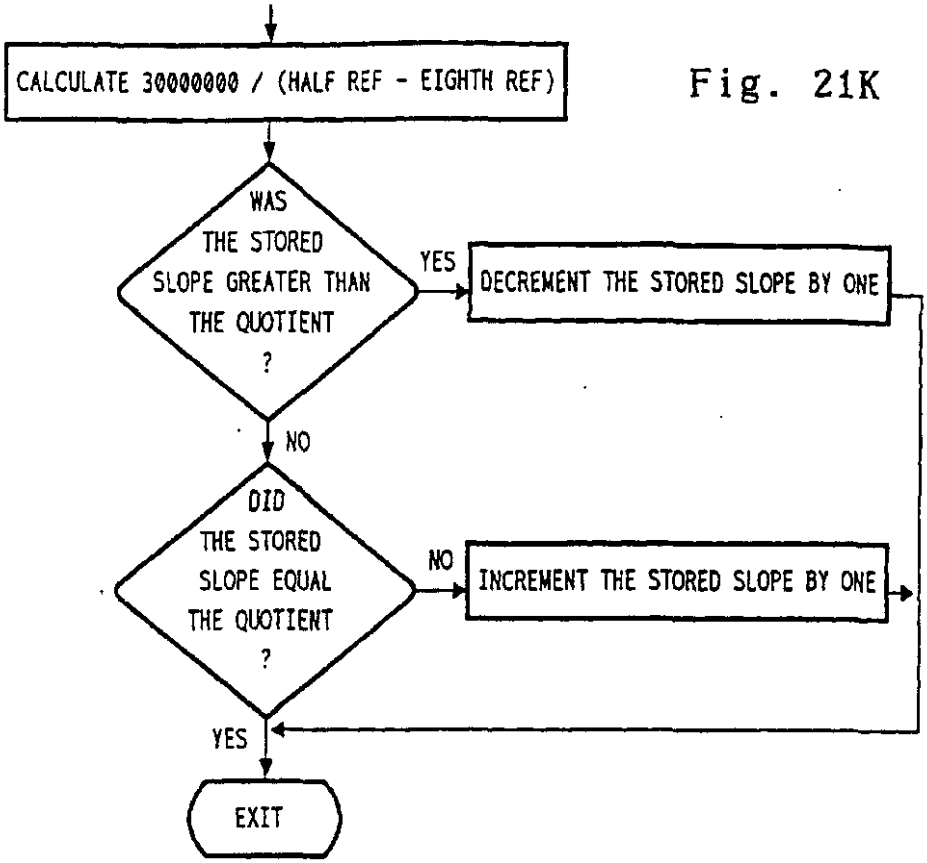


Fig. 21K

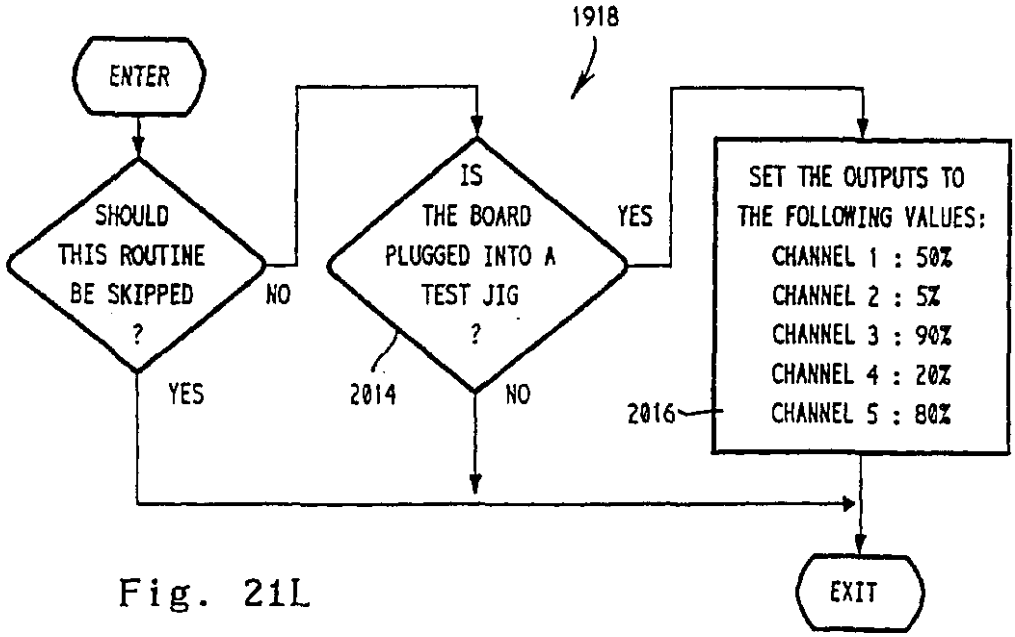


Fig. 21L

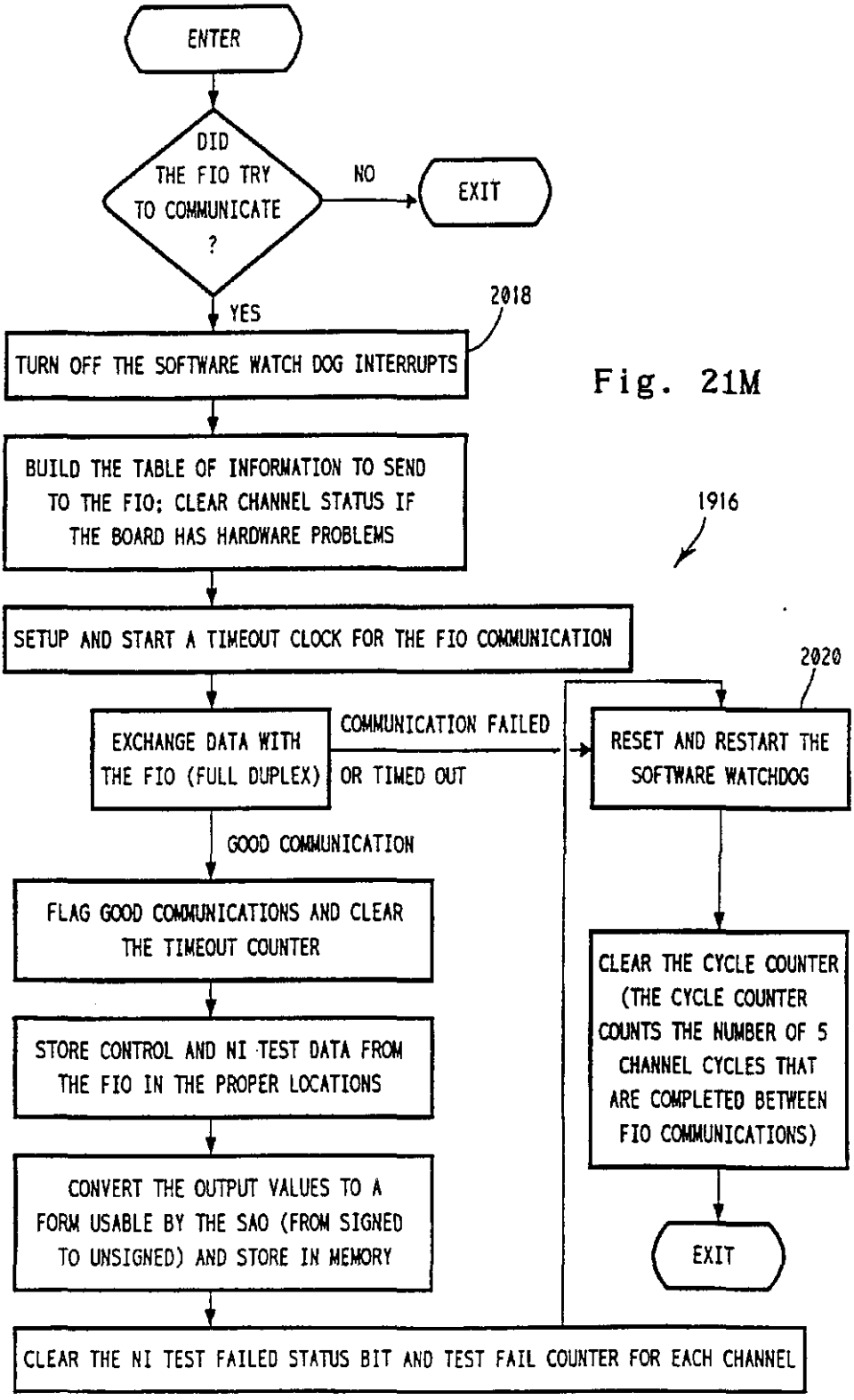

ENTER
DID THE FIO TRY TO COMMUNICATE ?
NO
EXIT
YES
2018
TURN OFF THE SOFTWARE WATCH DOG INTERRUPTS
BUILD THE TABLE OF INFORMATION TO SEND TO THE FIO; CLEAR CHANNEL STATUS IF THE BOARD HAS HARDWARE PROBLEMS
1916
SETUP AND START A TIMEOUT CLOCK FOR THE FIO COMMUNICATION
2020
EXCHANGE DATA WITH THE FIO (FULL DUPLEX)
COMMUNICATION FAILED OR TIMED OUT
RESET AND RESTART THE SOFTWARE WATCHDOG
GOOD COMMUNICATION
FLAG GOOD COMMUNICATIONS AND CLEAR THE TIMEOUT COUNTER
STORE CONTROL AND NI TEST DATA FROM THE FIO IN THE PROPER LOCATIONS
CONVERT THE OUTPUT VALUES TO A FORM USABLE BY THE SAO (FROM SIGNED TO UNSIGNED) AND STORE IN MEMORY
CLEAR THE CYCLE COUNTER (THE CYCLE COUNTER COUNTS THE NUMBER OF 5 CHANNEL CYCLES THAT ARE COMPLETED BETWEEN FIO COMMUNICATIONS)
EXIT
CLEAR THE NI TEST FAILED STATUS BIT AND TEST FAIL COUNTER FOR EACH CHANNEL

Fig. 21M

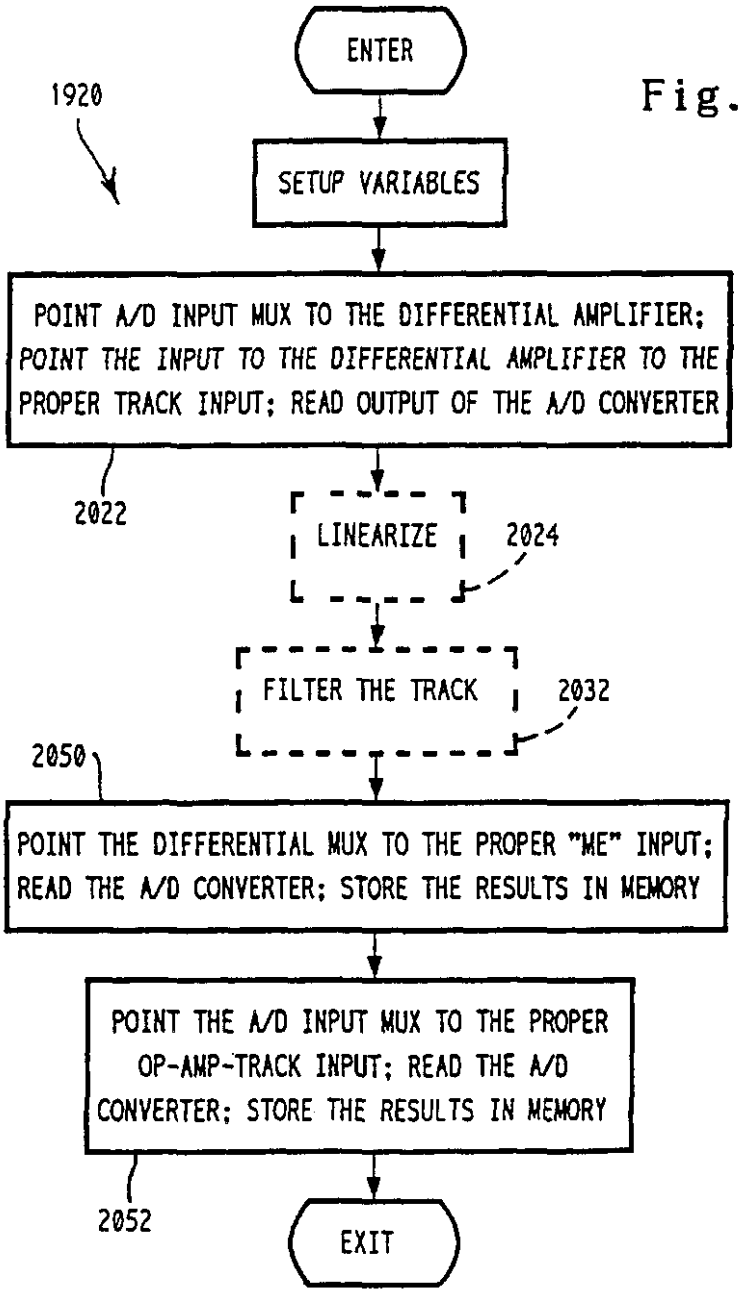

ENTER
1920
SETUP VARIABLES
POINT A/D INPUT MUX TO THE DIFFERENTIAL AMPLIFIER; POINT THE INPUT TO THE DIFFERENTIAL AMPLIFIER TO THE PROPER TRACK INPUT; READ OUTPUT OF THE A/D CONVERTER
2022
LINEARIZE
2024
FILTER THE TRACK
2032
2050
POINT THE DIFFERENTIAL MUX TO THE PROPER "ME" INPUT; READ THE A/D CONVERTER; STORE THE RESULTS IN MEMORY
POINT THE A/D INPUT MUX TO THE PROPER OP-AMP-TRACK INPUT; READ THE A/D CONVERTER; STORE THE RESULTS IN MEMORY
2052
EXIT

Fig. 21N

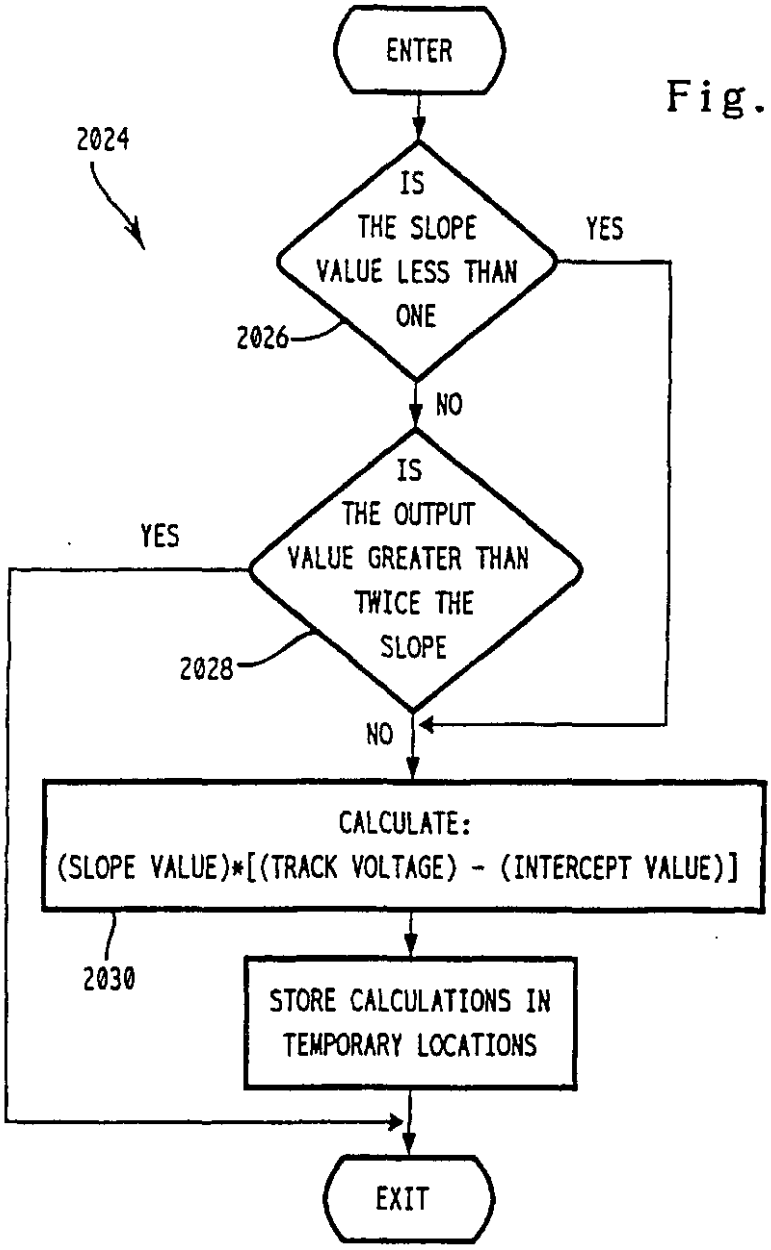
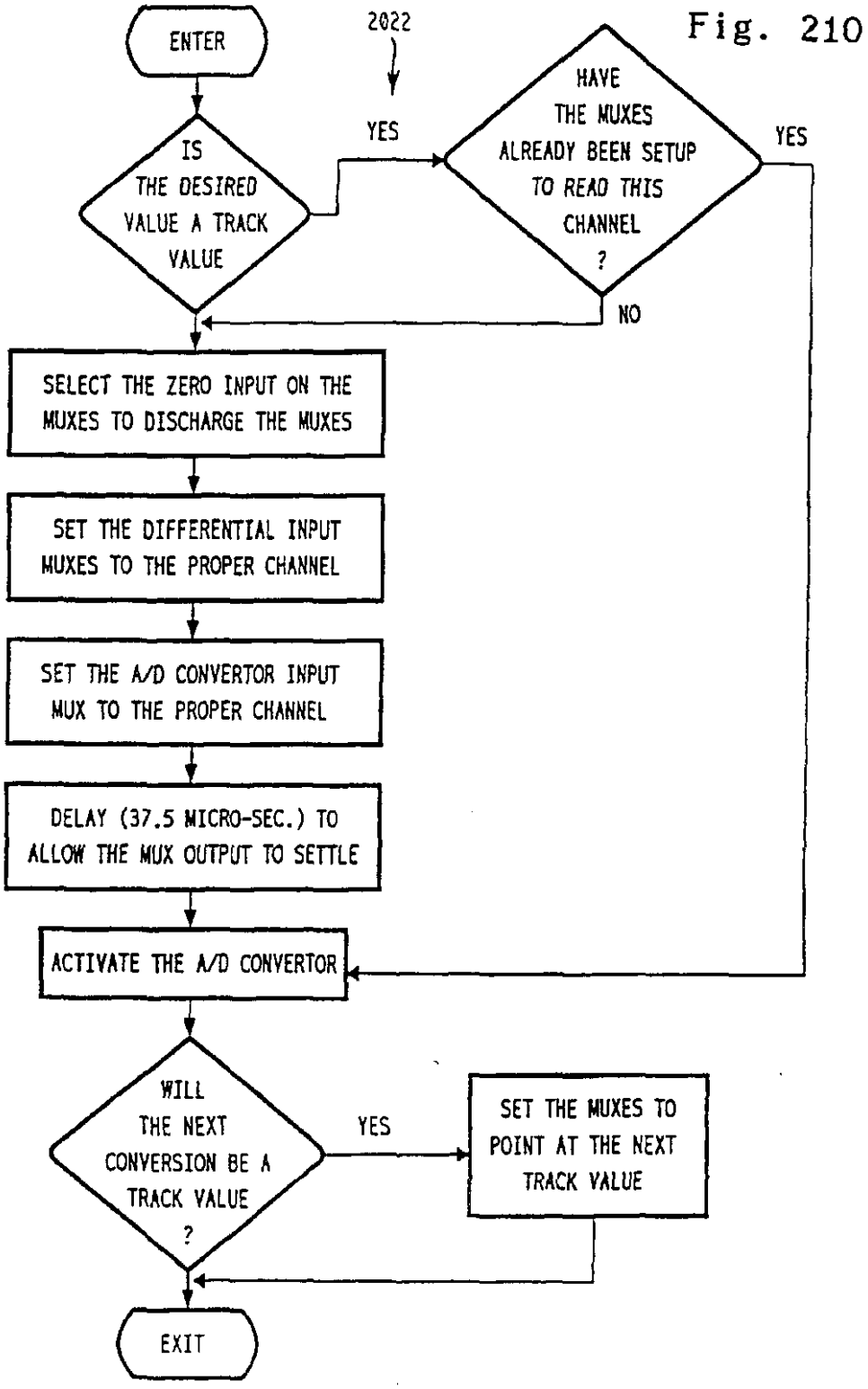

ENTER
2024
IS THE SLOPE VALUE LESS THAN ONE
YES
2026
NO
IS THE OUTPUT VALUE GREATER THAN TWICE THE SLOPE
YES
2028
NO
CALCULATE: (SLOPE VALUE)*[(TRACK VOLTAGE) - (INTERCEPT VALUE)]
2030
STORE CALCULATIONS IN TEMPORARY LOCATIONS
EXIT

Fig. 21P

Fig. 210


ENTER
2022
IS THE DESIRED VALUE A TRACK VALUE
YES
HAVE THE MUXES ALREADY BEEN SETUP TO READ THIS CHANNEL ?
YES
NO
SELECT THE ZERO INPUT ON THE MUXES TO DISCHARGE THE MUXES
SET THE DIFFERENTIAL INPUT MUXES TO THE PROPER CHANNEL
SET THE A/D CONVERTOR INPUT MUX TO THE PROPER CHANNEL
DELAY (37.5 MICRO-SEC.) TO ALLOW THE MUX OUTPUT TO SETTLE
ACTIVATE THE A/D CONVERTOR
WILL THE NEXT CONVERSION BE A TRACK VALUE ?
YES
SET THE MUXES TO POINT AT THE NEXT TRACK VALUE
EXIT

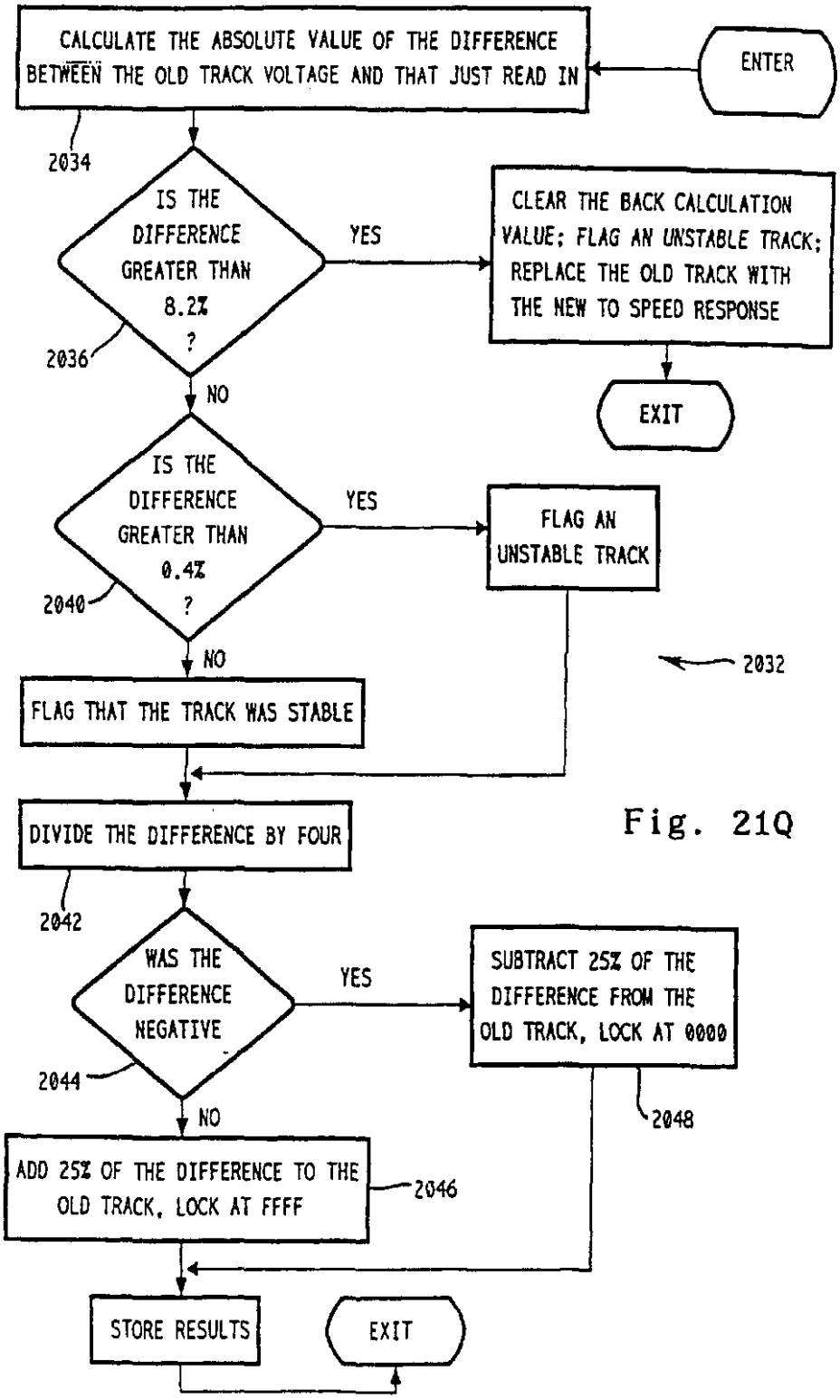

CALCULATE THE ABSOLUTE VALUE OF THE DIFFERENCE BETWEEN THE OLD TRACK VOLTAGE AND THAT JUST READ IN
ENTER
2034
IS THE DIFFERENCE GREATER THAN 8.2% ?
2036
YES
CLEAR THE BACK CALCULATION VALUE; FLAG AN UNSTABLE TRACK; REPLACE THE OLD TRACK WITH THE NEW TO SPEED RESPONSE
EXIT
NO
IS THE DIFFERENCE GREATER THAN 0.4% ?
2040
YES
FLAG AN UNSTABLE TRACK
NO
FLAG THAT THE TRACK WAS STABLE
2032
DIVIDE THE DIFFERENCE BY FOUR
2042
WAS THE DIFFERENCE NEGATIVE
2044
YES
SUBTRACT 25% OF THE DIFFERENCE FROM THE OLD TRACK, LOCK AT 0000
2048
NO
ADD 25% OF THE DIFFERENCE TO THE OLD TRACK, LOCK AT FFFF
2046
STORE RESULTS
EXIT

Fig. 21Q

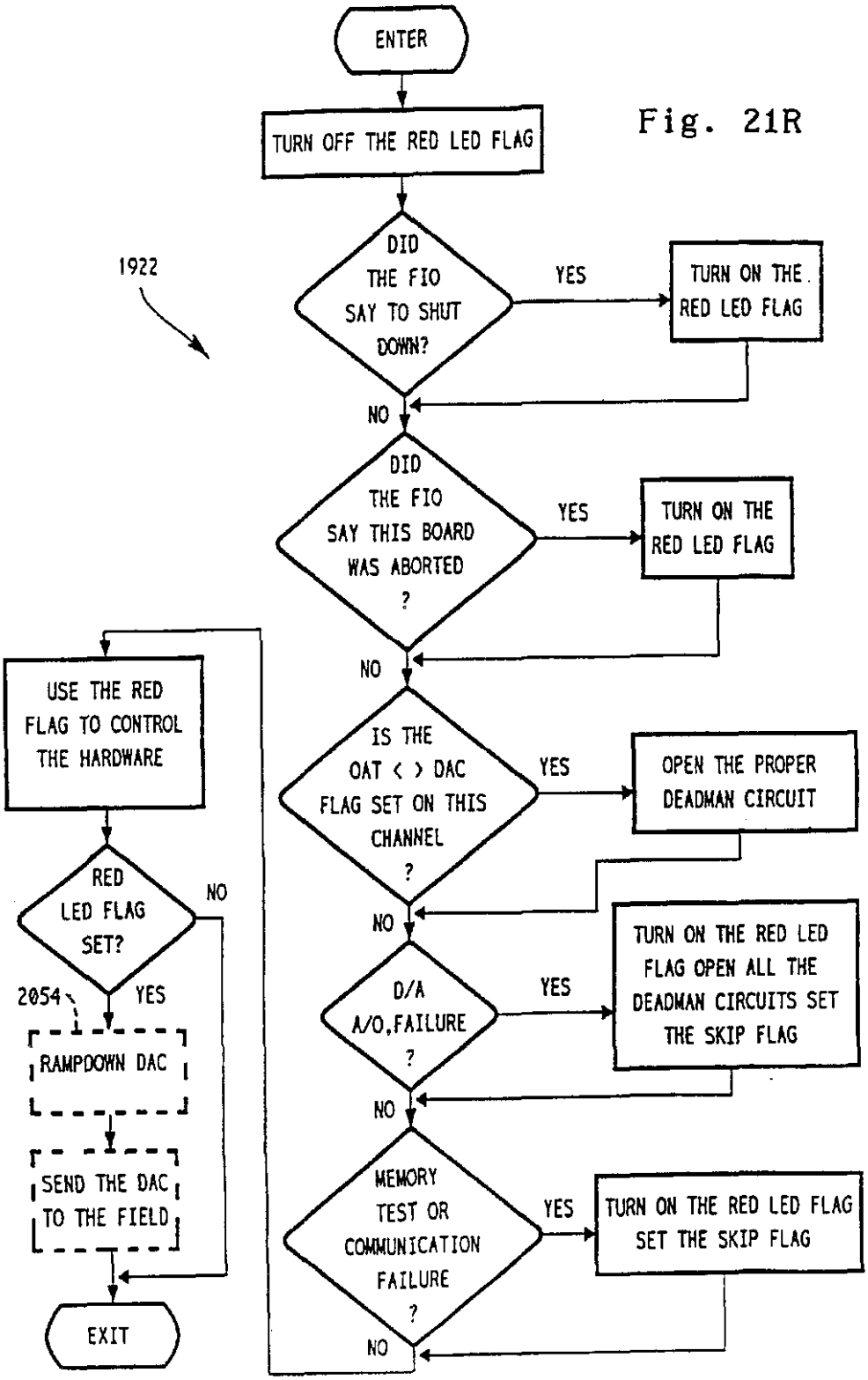

ENTER
TURN OFF THE RED LED FLAG
1922
DID THE FIO SAY TO SHUT DOWN?
YES
TURN ON THE RED LED FLAG
NO
DID THE FIO SAY THIS BOARD WAS ABORTED ?
YES
TURN ON THE RED LED FLAG
NO
IS THE OAT < > DAC FLAG SET ON THIS CHANNEL ?
YES
OPEN THE PROPER DEADMAN CIRCUIT
NO
D/A A/O,FAILURE ?
YES
TURN ON THE RED LED FLAG OPEN ALL THE DEADMAN CIRCUITS SET THE SKIP FLAG
NO
MEMORY TEST OR COMMUNICATION FAILURE ?
YES
TURN ON THE RED LED FLAG SET THE SKIP FLAG
NO
USE THE RED FLAG TO CONTROL THE HARDWARE
RED LED FLAG SET?
NO
2054
YES
RAMPDOWN DAC
SEND THE DAC TO THE FIELD
EXIT

Fig. 21R

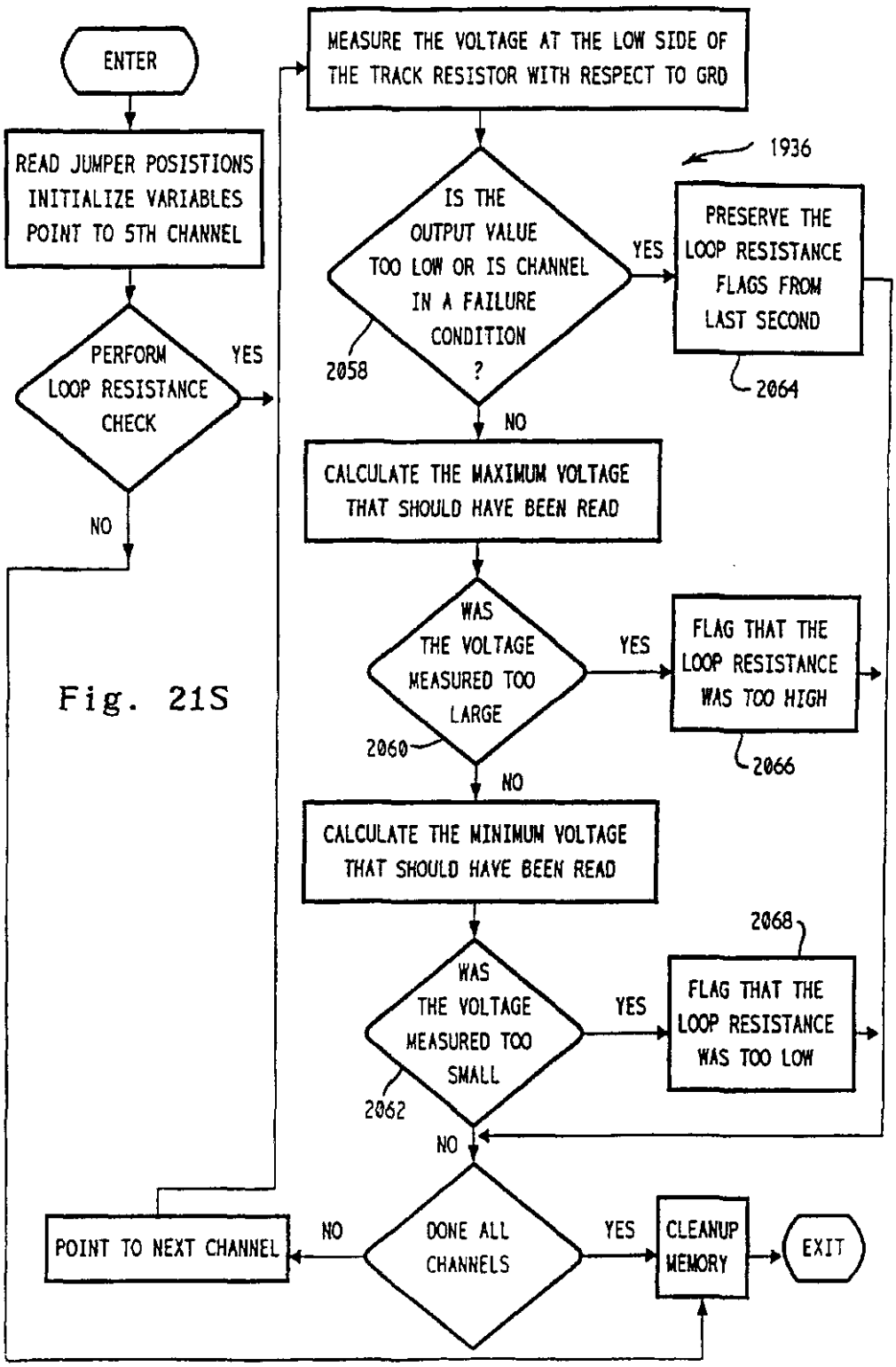

ENTER
READ JUMPER POSISTIONS
INITIALIZE VARIABLES
POINT TO 5TH CHANNEL
PERFORM LOOP RESISTANCE CHECK
YES
NO
MEASURE THE VOLTAGE AT THE LOW SIDE OF THE TRACK RESISTOR WITH RESPECT TO GRD
IS THE OUTPUT VALUE TOO LOW OR IS CHANNEL IN A FAILURE CONDITION ?
2058
YES
1936
PRESERVE THE LOOP RESISTANCE FLAGS FROM LAST SECOND
2064
NO
CALCULATE THE MAXIMUM VOLTAGE THAT SHOULD HAVE BEEN READ
WAS THE VOLTAGE MEASURED TOO LARGE
2060
YES
FLAG THAT THE LOOP RESISTANCE WAS TOO HIGH
2066
NO
CALCULATE THE MINIMUM VOLTAGE THAT SHOULD HAVE BEEN READ
2068
WAS THE VOLTAGE MEASURED TOO SMALL
2062
YES
FLAG THAT THE LOOP RESISTANCE WAS TOO LOW
NO
DONE ALL CHANNELS
NO
POINT TO NEXT CHANNEL
YES
CLEANUP MEMORY
EXIT

Fig. 21S

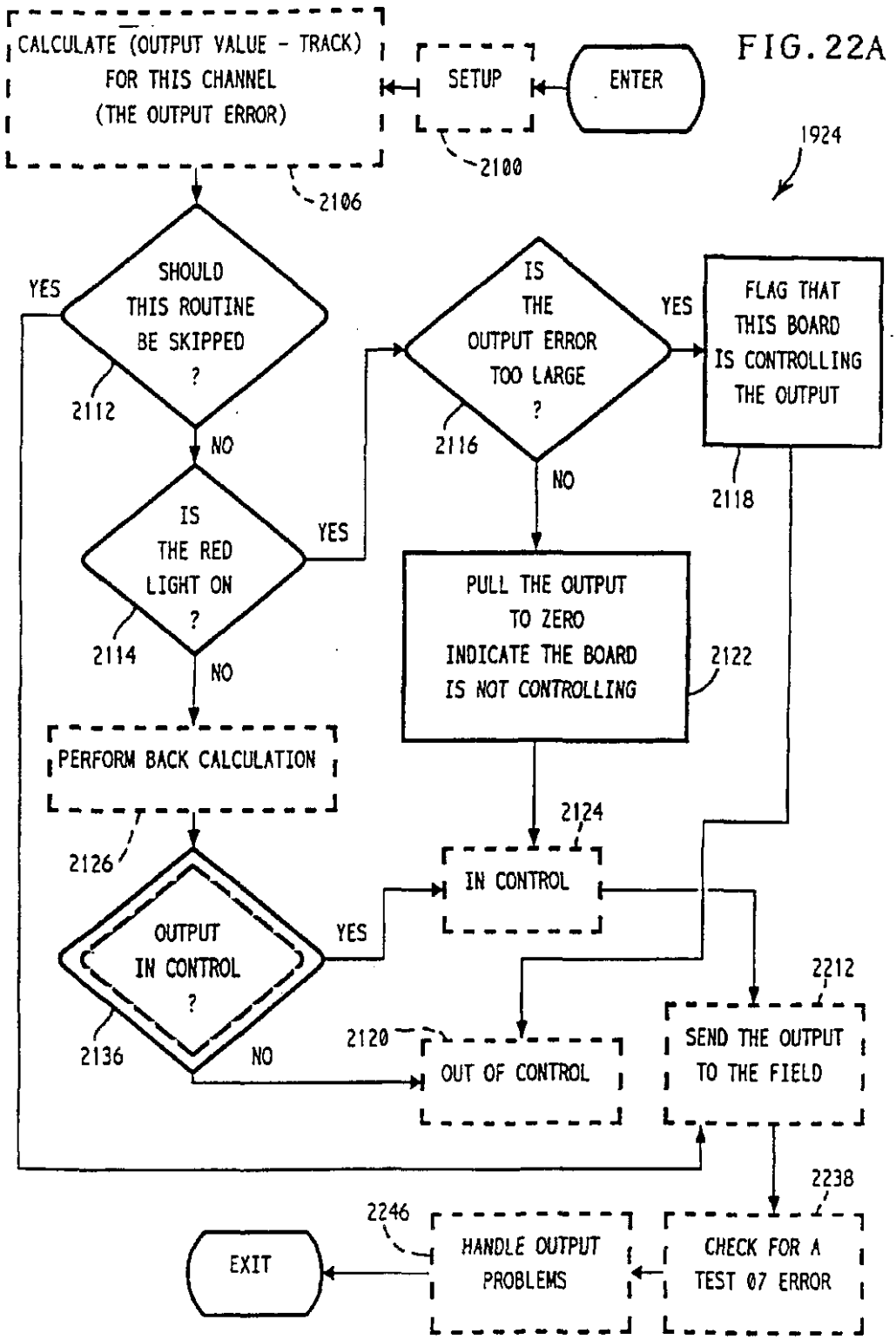

CALCULATE (OUTPUT VALUE - TRACK)
FOR THIS CHANNEL
(THE OUTPUT ERROR)
SETUP
ENTER
2100
1924
2106
SHOULD
THIS ROUTINE
BE SKIPPED
?
YES
2112
NO
IS
THE RED
LIGHT ON
?
YES
2114
NO
IS
THE
OUTPUT ERROR
TOO LARGE
?
YES
2116
NO
FLAG THAT
THIS BOARD
IS CONTROLLING
THE OUTPUT
2118
PULL THE OUTPUT
TO ZERO
INDICATE THE BOARD
IS NOT CONTROLLING
2122
PERFORM BACK CALCULATION
2126
2124
IN CONTROL
OUTPUT
IN CONTROL
?
YES
2136
NO
2120
OUT OF CONTROL
2212
SEND THE OUTPUT
TO THE FIELD
2238
2246
HANDLE OUTPUT
PROBLEMS
CHECK FOR A
TEST 07 ERROR
EXIT

FIG. 22A

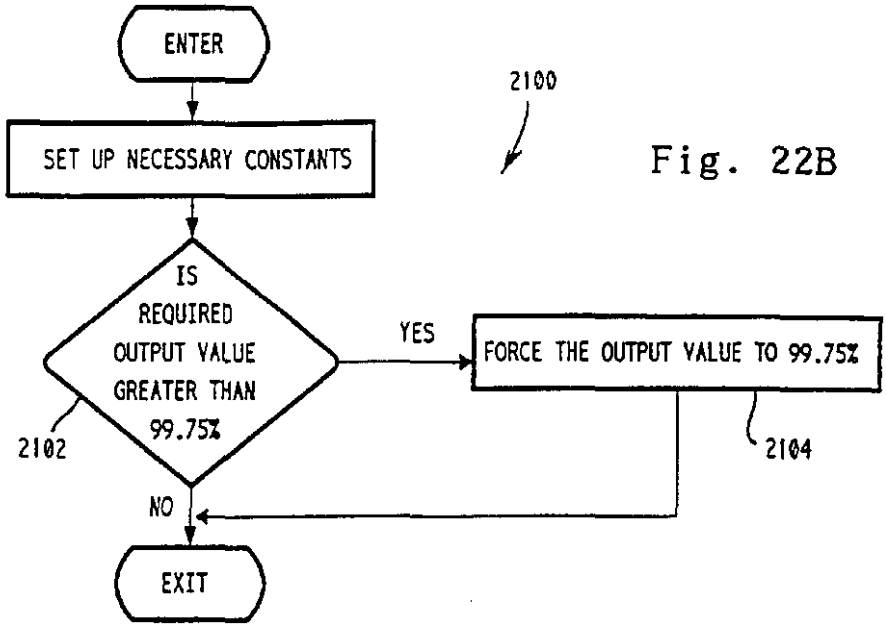


Fig. 22B

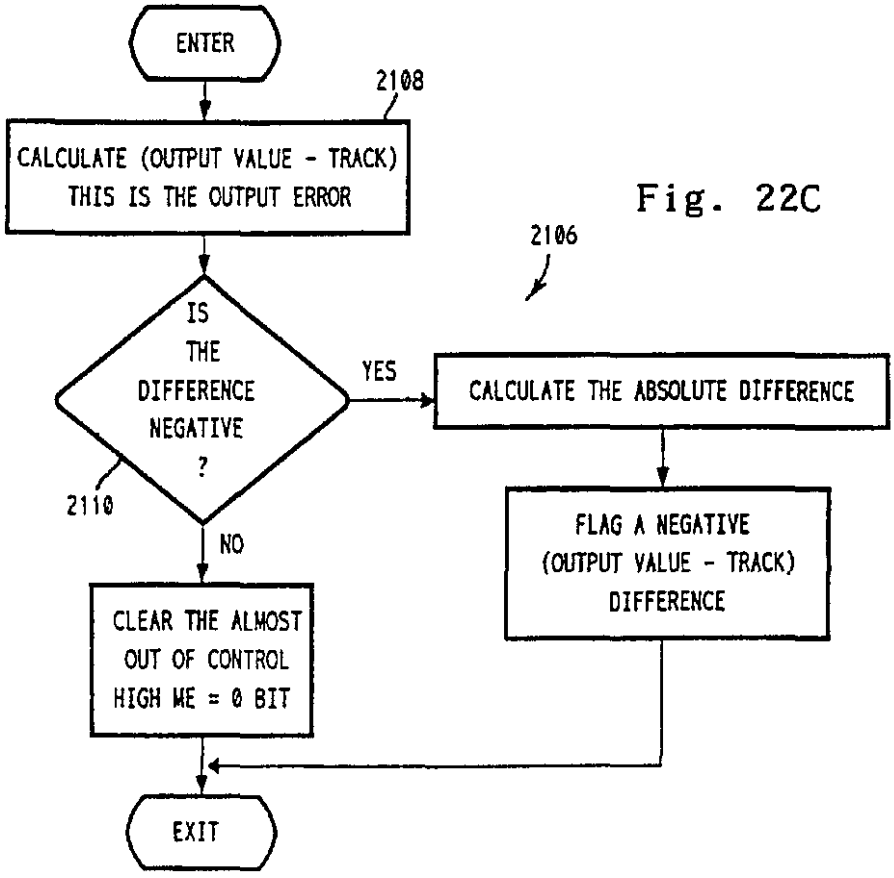


Fig. 22C

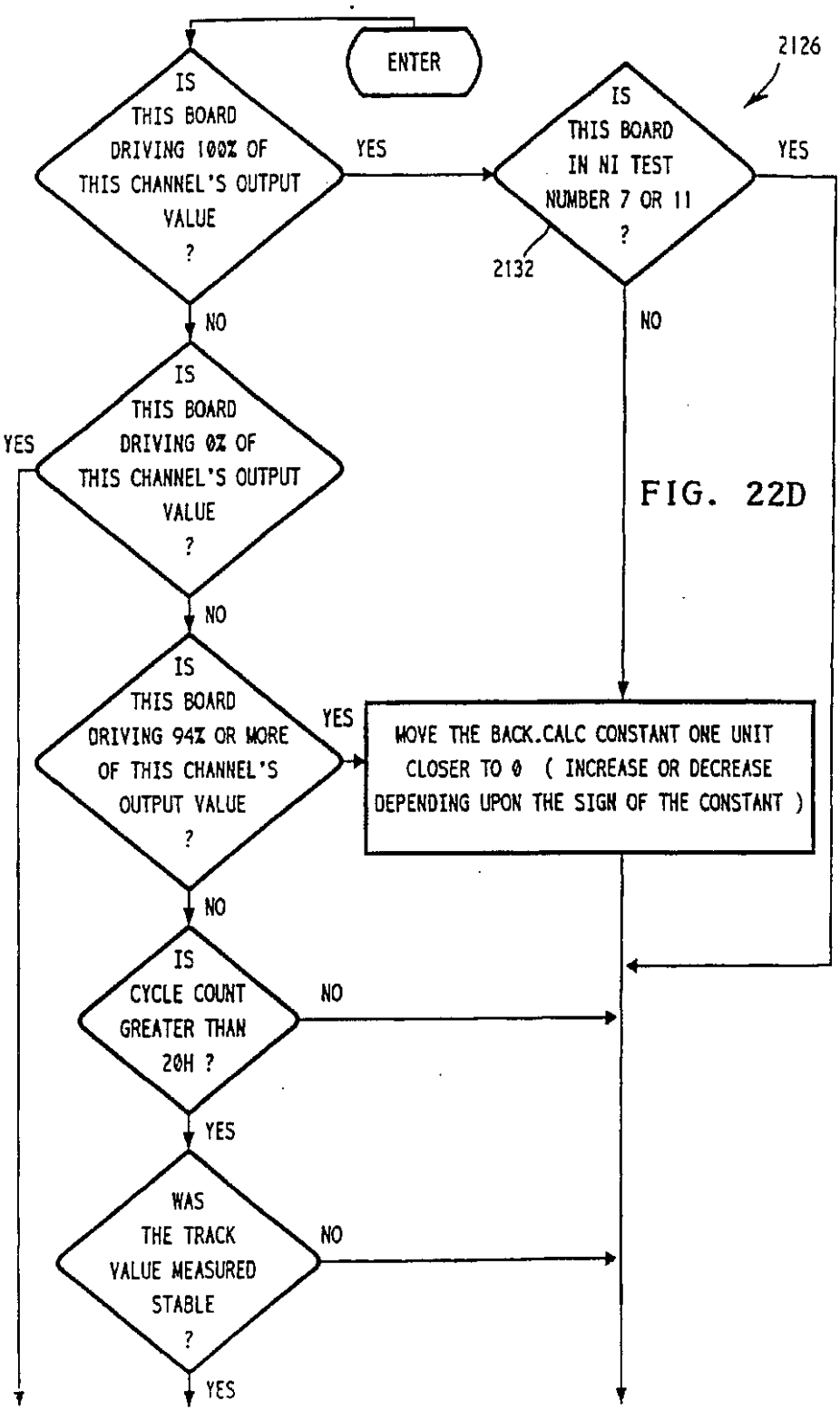

ENTER
IS THIS BOARD DRIVING 100% OF THIS CHANNEL'S OUTPUT VALUE ?
YES
IS THIS BOARD IN NI TEST NUMBER 7 OR 11 ?
2126
2132
YES
NO
NO
IS THIS BOARD DRIVING 0% OF THIS CHANNEL'S OUTPUT VALUE ?
YES
NO
IS THIS BOARD DRIVING 94% OR MORE OF THIS CHANNEL'S OUTPUT VALUE ?
YES
MOVE THE BACK.CALC CONSTANT ONE UNIT CLOSER TO 0 (INCREASE OR DECREASE DEPENDING UPON THE SIGN OF THE CONSTANT)
NO
IS CYCLE COUNT GREATER THAN 20H ?
NO
YES
WAS THE TRACK VALUE MEASURED STABLE ?
NO
YES

FIG. 22D

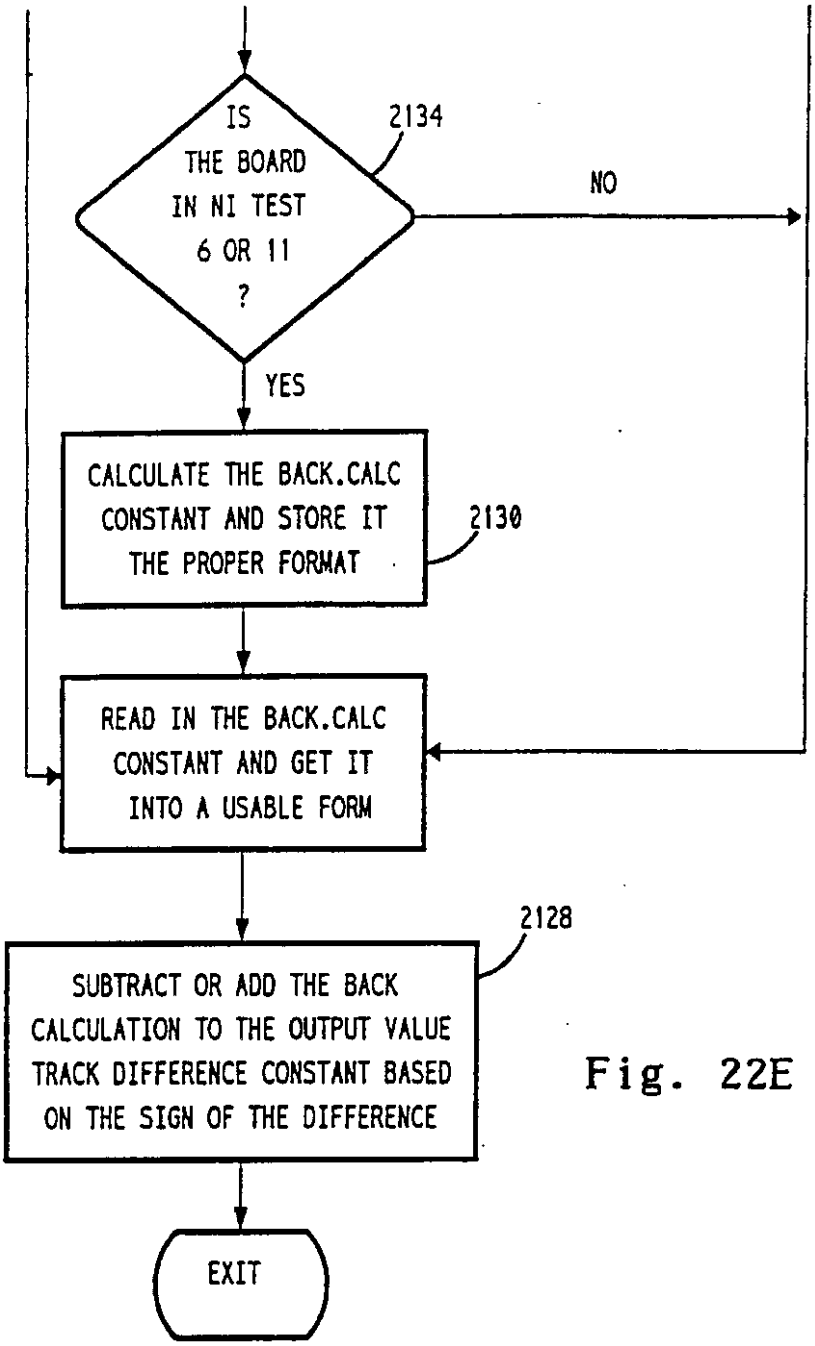

IS THE BOARD IN NI TEST 6 OR 11 ?
2134
NO
YES
CALCULATE THE BACK.CALC CONSTANT AND STORE IT THE PROPER FORMAT
2130
READ IN THE BACK.CALC CONSTANT AND GET IT INTO A USABLE FORM
2128
SUBTRACT OR ADD THE BACK CALCULATION TO THE OUTPUT VALUE TRACK DIFFERENCE CONSTANT BASED ON THE SIGN OF THE DIFFERENCE
EXIT

Fig. 22E

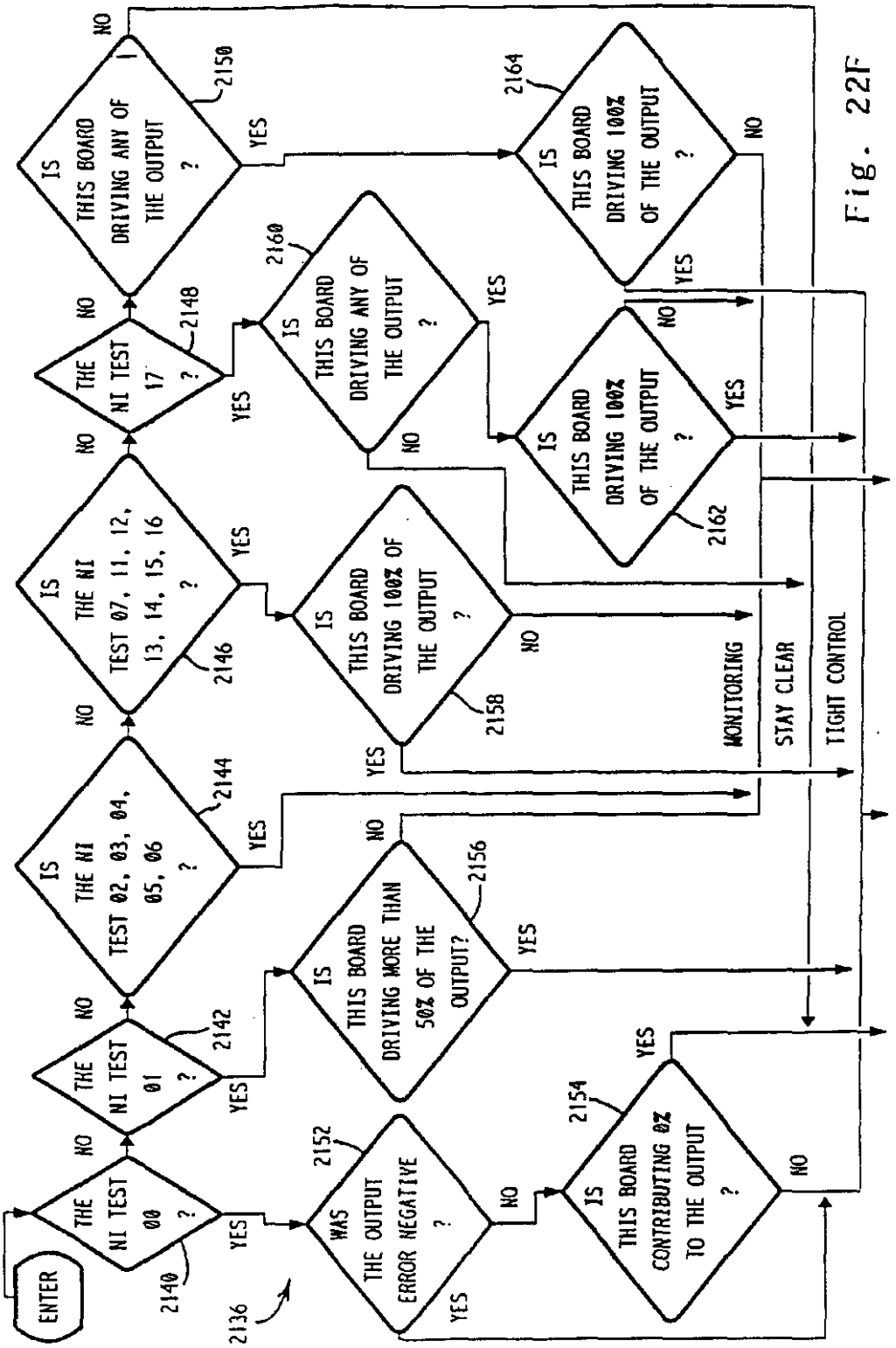

ENTER
2136
THE NI TEST 00 ?
2140
NO
YES
WAS THE OUTPUT ERROR NEGATIVE ?
2152
NO
YES
IS THIS BOARD CONTRIBUTING 0% TO THE OUTPUT ?
2154
NO
YES
THE NI TEST 01 ?
2142
NO
YES
IS THIS BOARD DRIVING MORE THAN 50% OF THE OUTPUT?
2156
NO
YES
IS THE NI TEST 02, 03, 04, 05, 06 ?
2144
NO
YES
IS THIS BOARD DRIVING 100% OF THE OUTPUT ?
2158
NO
YES
IS THE NI TEST 07, 11, 12, 13, 14, 15, 16 ?
2146
NO
YES
THE NI TEST 17 ?
2148
NO
YES
IS THIS BOARD DRIVING ANY OF THE OUTPUT ?
2160
NO
YES
IS THIS BOARD DRIVING 100% OF THE OUTPUT ?
2162
NO
YES
IS THIS BOARD DRIVING ANY OF THE OUTPUT ?
2150
NO
YES
IS THIS BOARD DRIVING 100% OF THE OUTPUT ?
2164
NO
YES
MONITORING
STAY CLEAR
TIGHT CONTROL

Fig. 22F

Fig. 22G

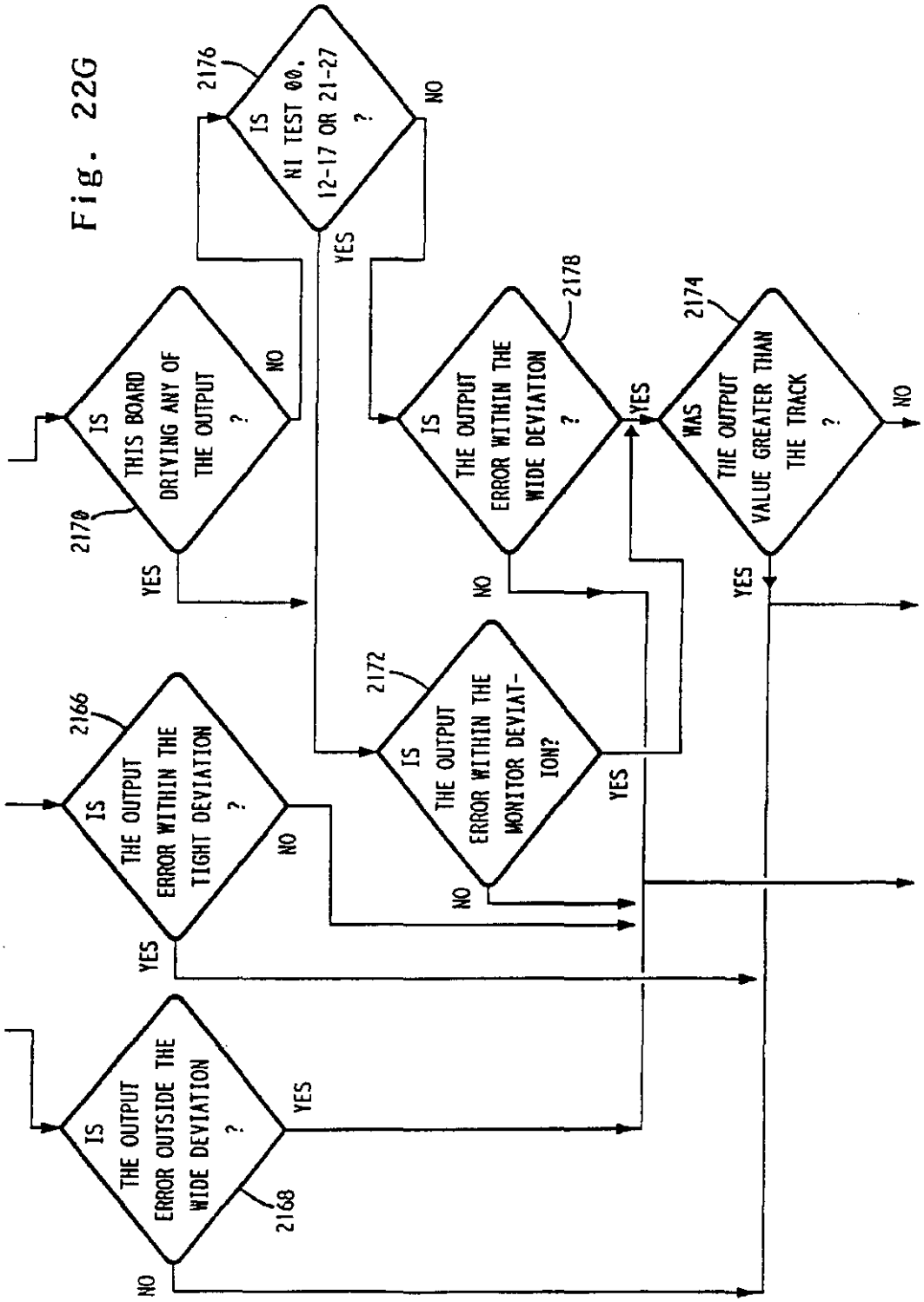

IS THE OUTPUT ERROR OUTSIDE THE WIDE DEVIATION ?
2168
YES
NO
IS THE OUTPUT ERROR WITHIN THE TIGHT DEVIATION ?
2166
NO
YES
IS THIS BOARD DRIVING ANY OF THE OUTPUT ?
2170
NO
YES
IS NI TEST 00, 12-17 OR 21-27 ?
2176
NO
YES
IS THE OUTPUT ERROR WITHIN THE MONITOR DEVIATION?
2172
NO
YES
IS THE OUTPUT ERROR WITHIN THE WIDE DEVIATION ?
2178
NO
YES
WAS THE OUTPUT VALUE GREATER THAN THE TRACK ?
2174
YES
NO

Fig. 22H

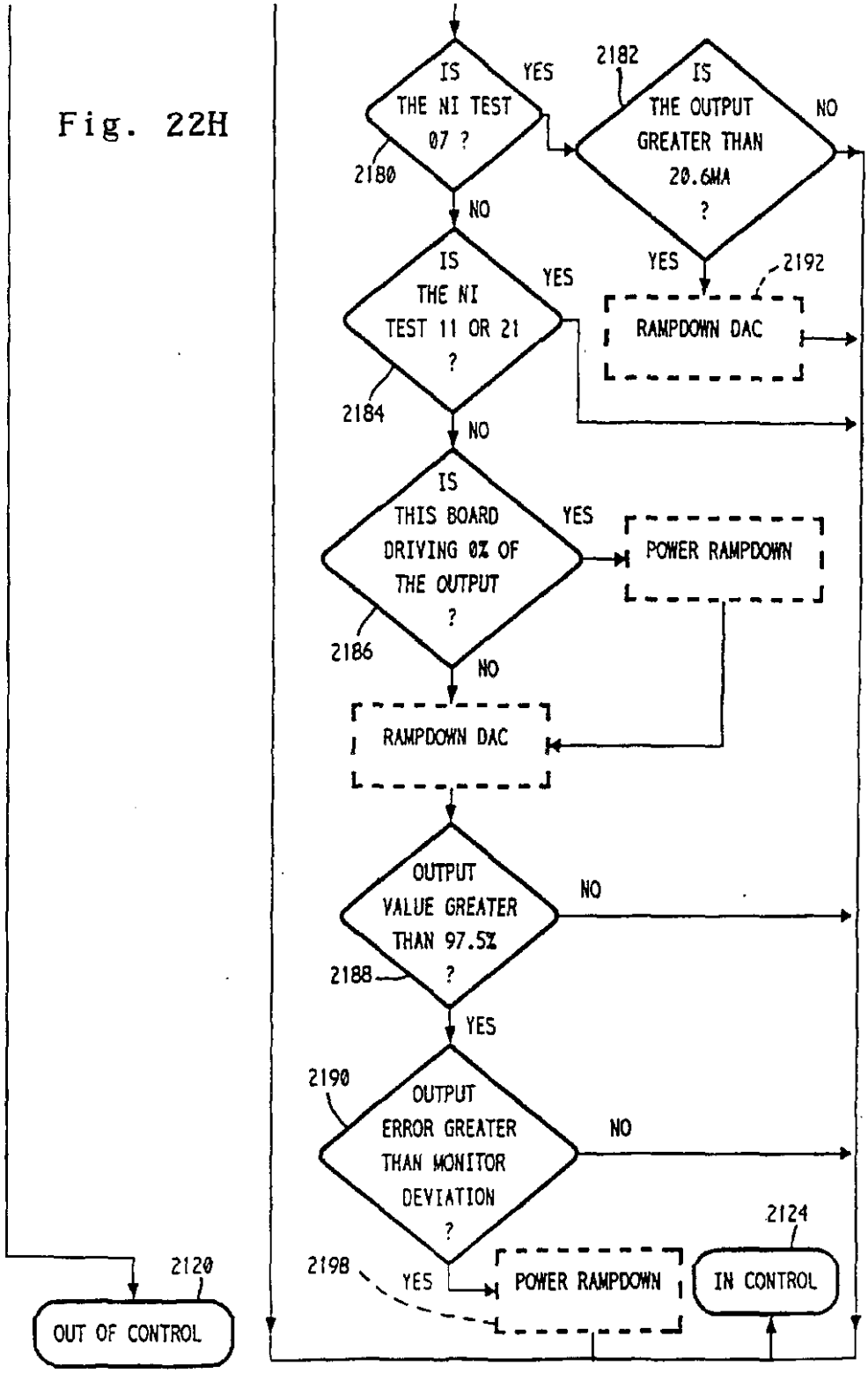

IS THE NI TEST 07 ?
YES
2180
NO
2182
IS THE OUTPUT GREATER THAN 20.6MA ?
NO
YES
2192
RAMPDOWN DAC
IS THE NI TEST 11 OR 21 ?
YES
2184
NO
IS THIS BOARD DRIVING 0% OF THE OUTPUT ?
YES
POWER RAMPDOWN
2186
NO
RAMPDOWN DAC
OUTPUT VALUE GREATER THAN 97.5% ?
NO
2188
YES
2190
OUTPUT ERROR GREATER THAN MONITOR DEVIATION ?
NO
2124
2198
YES
POWER RAMPDOWN
IN CONTROL
2120
OUT OF CONTROL

FIG 22I

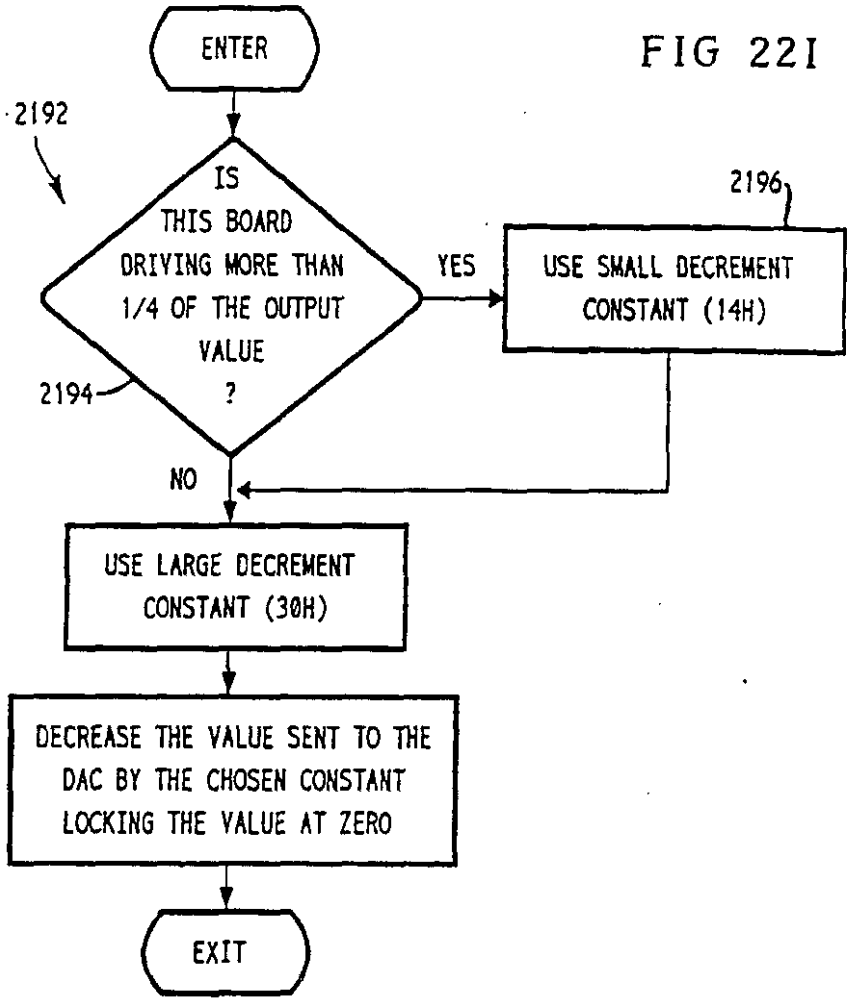
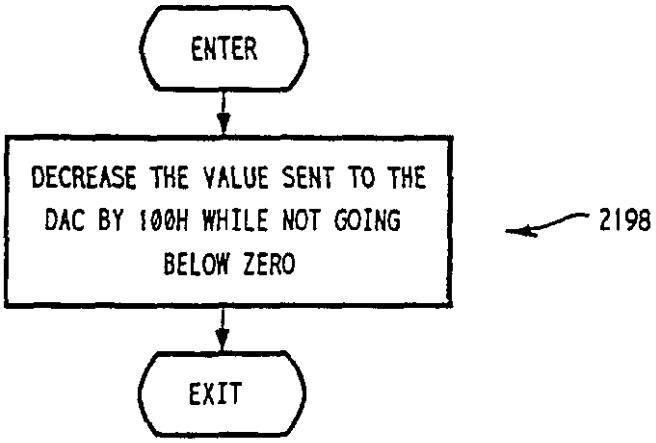


FIG 22J



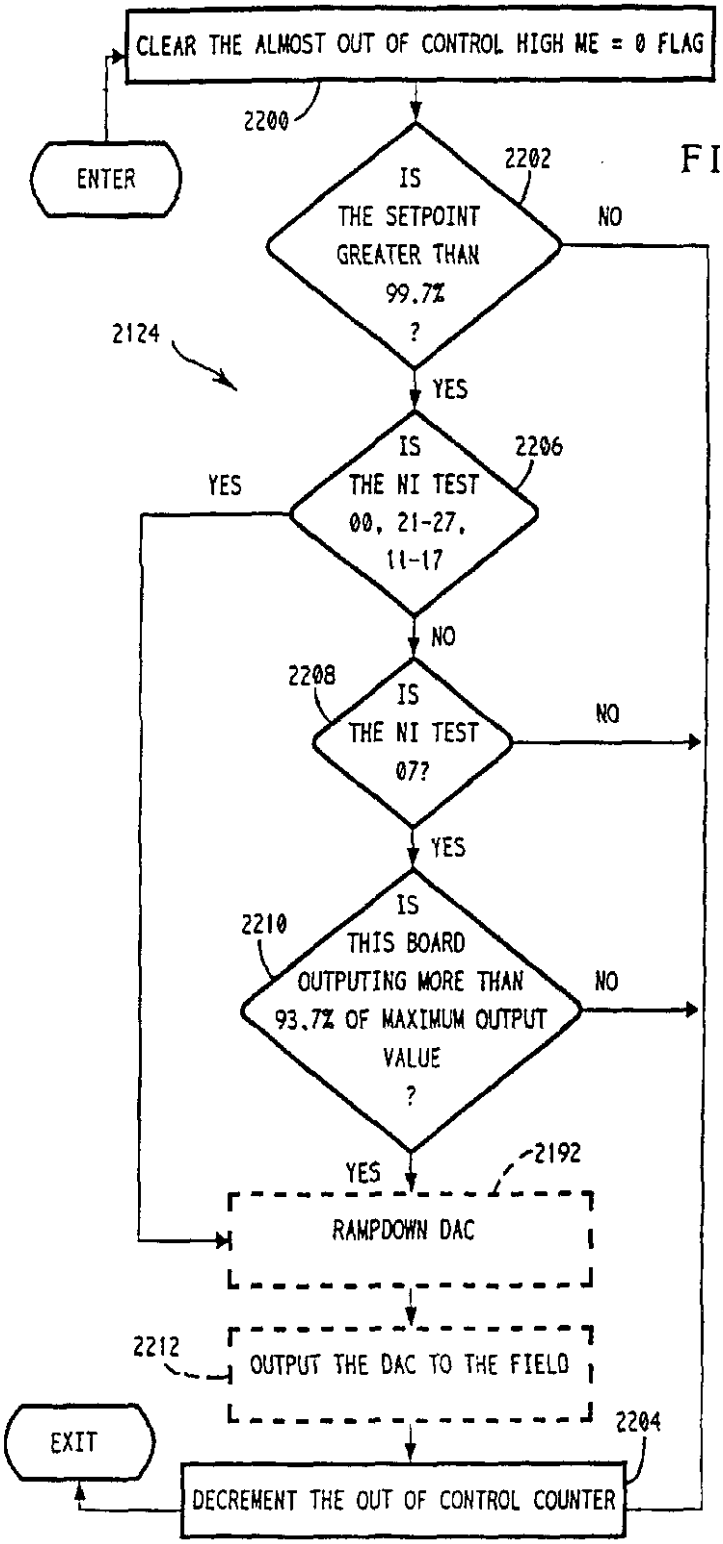

CLEAR THE ALMOST OUT OF CONTROL HIGH ME = 0 FLAG
2200
ENTER
2202
IS THE SETPOINT GREATER THAN 99.7% ?
NO
2124
YES
2206
IS THE NI TEST 00, 21-27, 11-17
YES
NO
2208
IS THE NI TEST 07?
NO
YES
2210
IS THIS BOARD OUTPUTING MORE THAN 93.7% OF MAXIMUM OUTPUT VALUE ?
NO
YES
2192
RAMPDOWN DAC
2212
OUTPUT THE DAC TO THE FIELD
EXIT
2204
DECREMENT THE OUT OF CONTROL COUNTER

FIG 22K

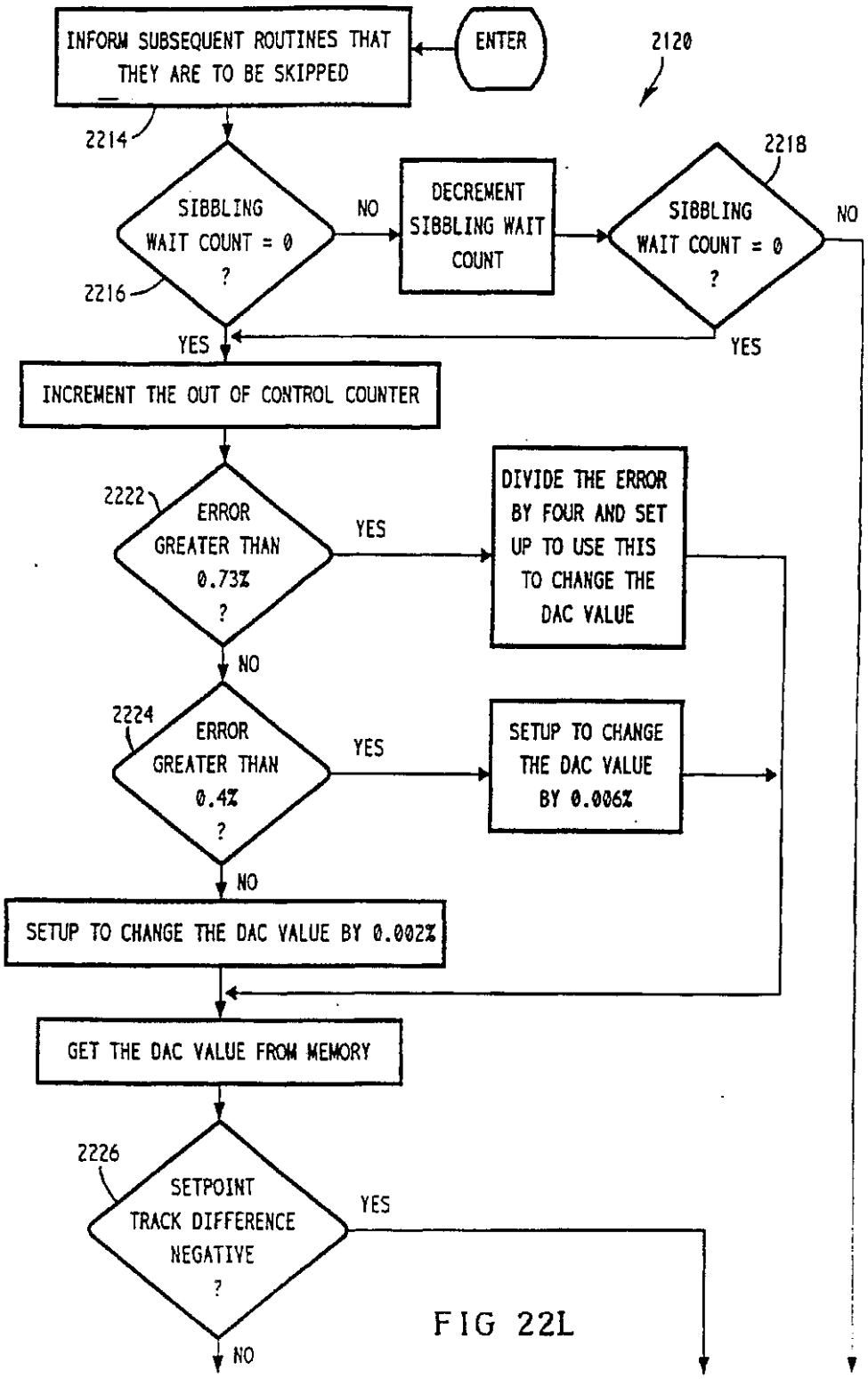
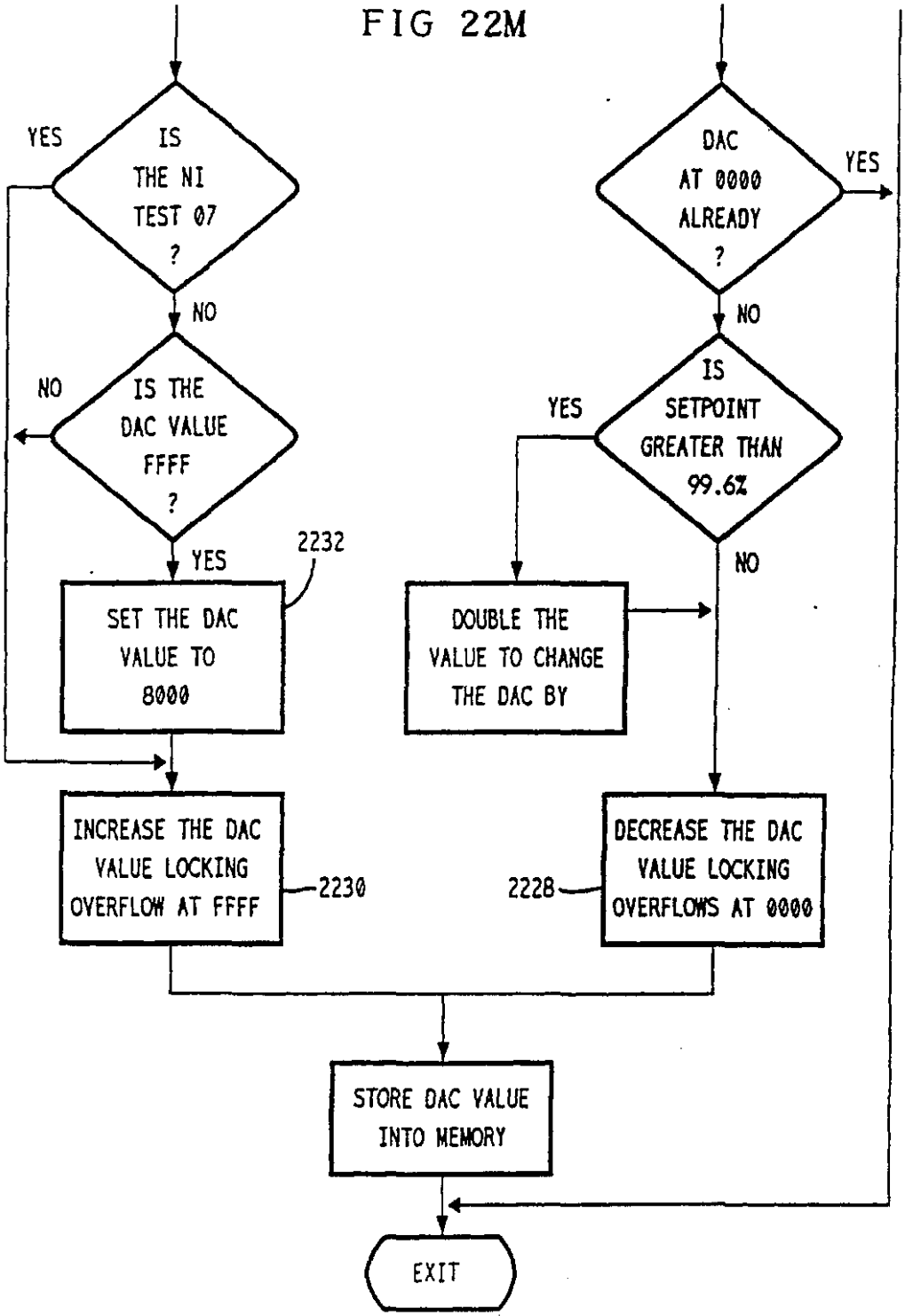

INFORM SUBSEQUENT ROUTINES THAT THEY ARE TO BE SKIPPED
ENTER
2120
2214
SIBBLING WAIT COUNT = 0 ?
2216
NO
DECREMENT SIBBLING WAIT COUNT
2218
SIBBLING WAIT COUNT = 0 ?
NO
YES
YES
INCREMENT THE OUT OF CONTROL COUNTER
2222
ERROR GREATER THAN 0.73% ?
YES
DIVIDE THE ERROR BY FOUR AND SET UP TO USE THIS TO CHANGE THE DAC VALUE
NO
2224
ERROR GREATER THAN 0.4% ?
YES
SETUP TO CHANGE THE DAC VALUE BY 0.006%
NO
SETUP TO CHANGE THE DAC VALUE BY 0.002%
GET THE DAC VALUE FROM MEMORY
2226
SETPOINT TRACK DIFFERENCE NEGATIVE ?
YES
NO

FIG 22L

FIG 22M


IS THE NI TEST 07 ?
YES
NO
IS THE DAC VALUE FFFF ?
NO
YES
2232
SET THE DAC VALUE TO 8000
INCREASE THE DAC VALUE LOCKING OVERFLOW AT FFFF
2230
DAC AT 0000 ALREADY ?
YES
NO
IS SETPOINT GREATER THAN 99.6%
YES
NO
DOUBLE THE VALUE TO CHANGE THE DAC BY
2228
DECREASE THE DAC VALUE LOCKING OVERFLOWS AT 0000
STORE DAC VALUE INTO MEMORY
EXIT

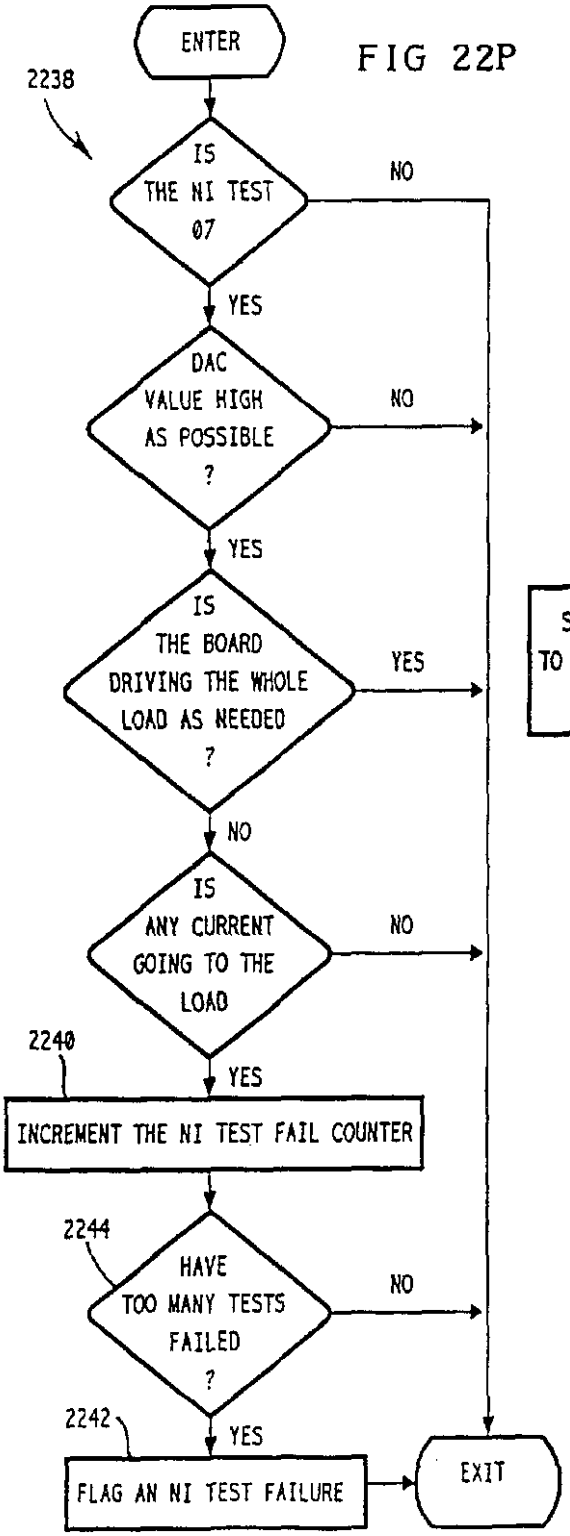


FIG 22P

FIG 22N

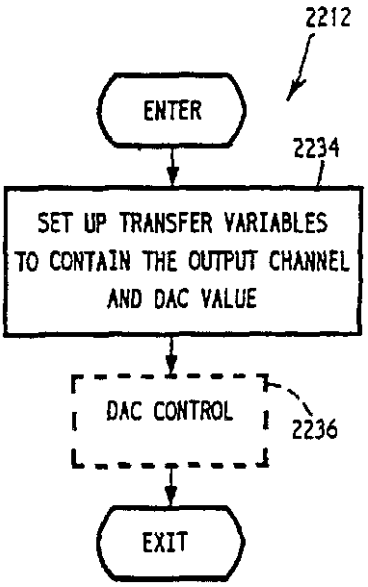
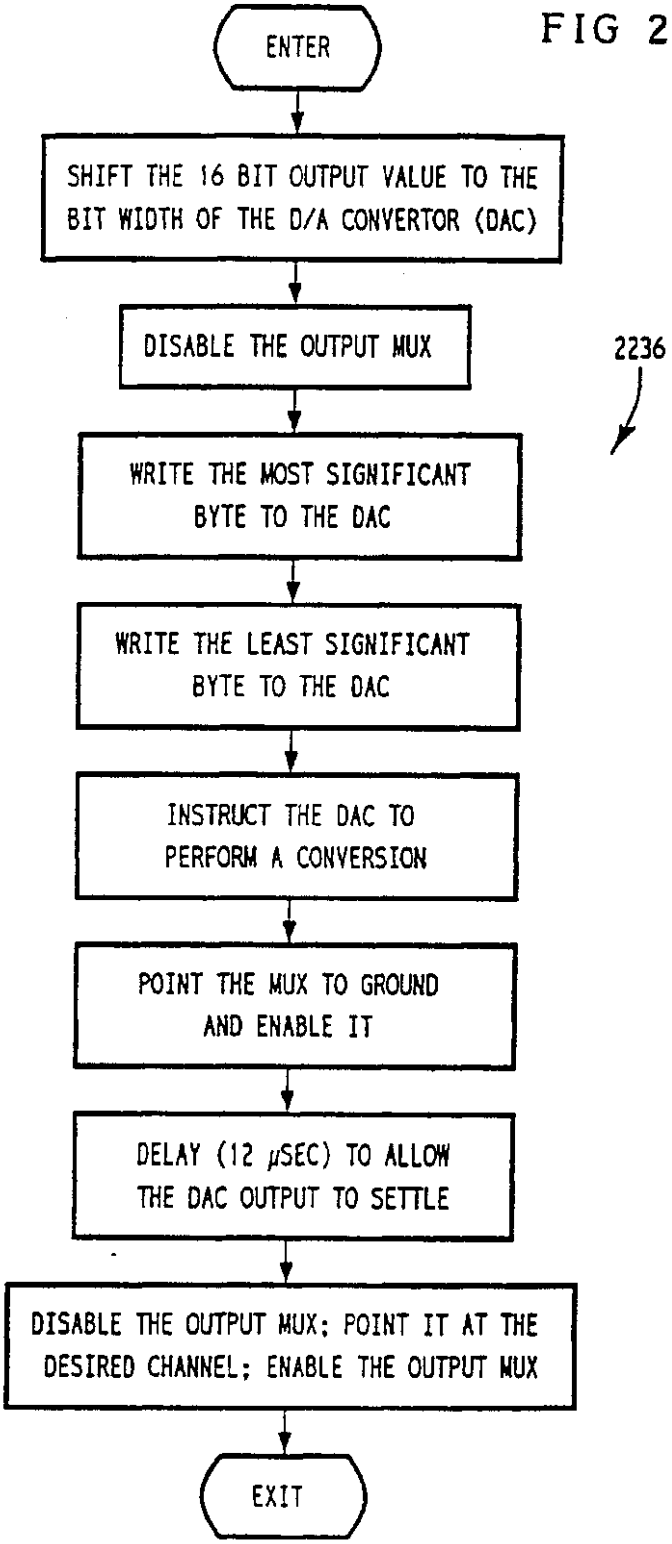


FIG 220


ENTER
SHIFT THE 16 BIT OUTPUT VALUE TO THE BIT WIDTH OF THE D/A CONVERTOR (DAC)
DISABLE THE OUTPUT MUX
2236
WRITE THE MOST SIGNIFICANT BYTE TO THE DAC
WRITE THE LEAST SIGNIFICANT BYTE TO THE DAC
INSTRUCT THE DAC TO PERFORM A CONVERSION
POINT THE MUX TO GROUND AND ENABLE IT
DELAY (12 μSEC) TO ALLOW THE DAC OUTPUT TO SETTLE
DISABLE THE OUTPUT MUX; POINT IT AT THE DESIRED CHANNEL; ENABLE THE OUTPUT MUX
EXIT

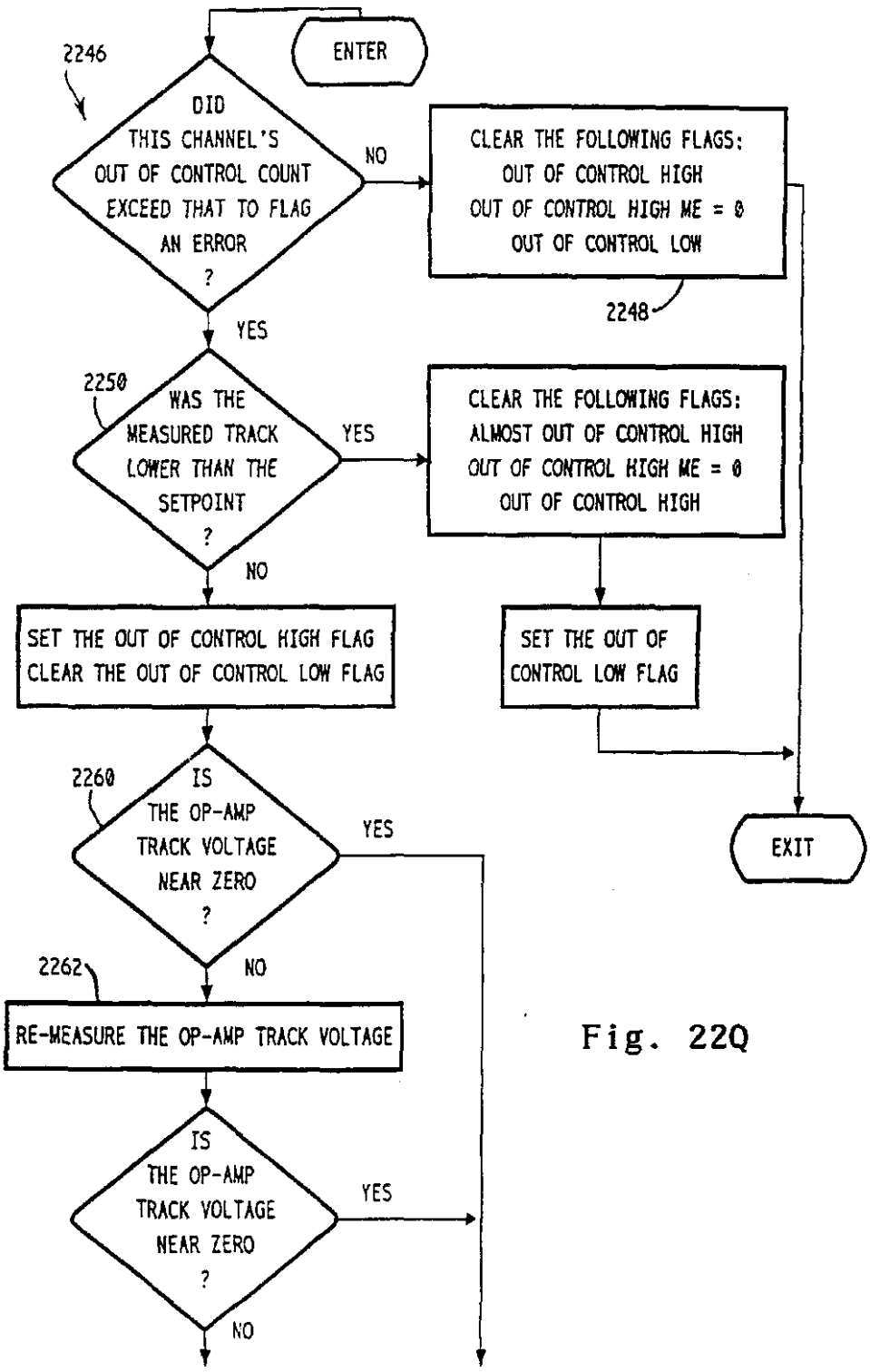

ENTER
2246
DID THIS CHANNEL'S OUT OF CONTROL COUNT EXCEED THAT TO FLAG AN ERROR ?
NO
CLEAR THE FOLLOWING FLAGS:
OUT OF CONTROL HIGH
OUT OF CONTROL HIGH ME = 0
OUT OF CONTROL LOW
2248
YES
2250
WAS THE MEASURED TRACK LOWER THAN THE SETPOINT ?
YES
CLEAR THE FOLLOWING FLAGS:
ALMOST OUT OF CONTROL HIGH
OUT OF CONTROL HIGH ME = 0
OUT OF CONTROL HIGH
NO
SET THE OUT OF CONTROL HIGH FLAG
CLEAR THE OUT OF CONTROL LOW FLAG
SET THE OUT OF CONTROL LOW FLAG
2260
IS THE OP-AMP TRACK VOLTAGE NEAR ZERO ?
YES
EXIT
NO
2262
RE-MEASURE THE OP-AMP TRACK VOLTAGE
IS THE OP-AMP TRACK VOLTAGE NEAR ZERO ?
YES
NO

Fig. 22Q

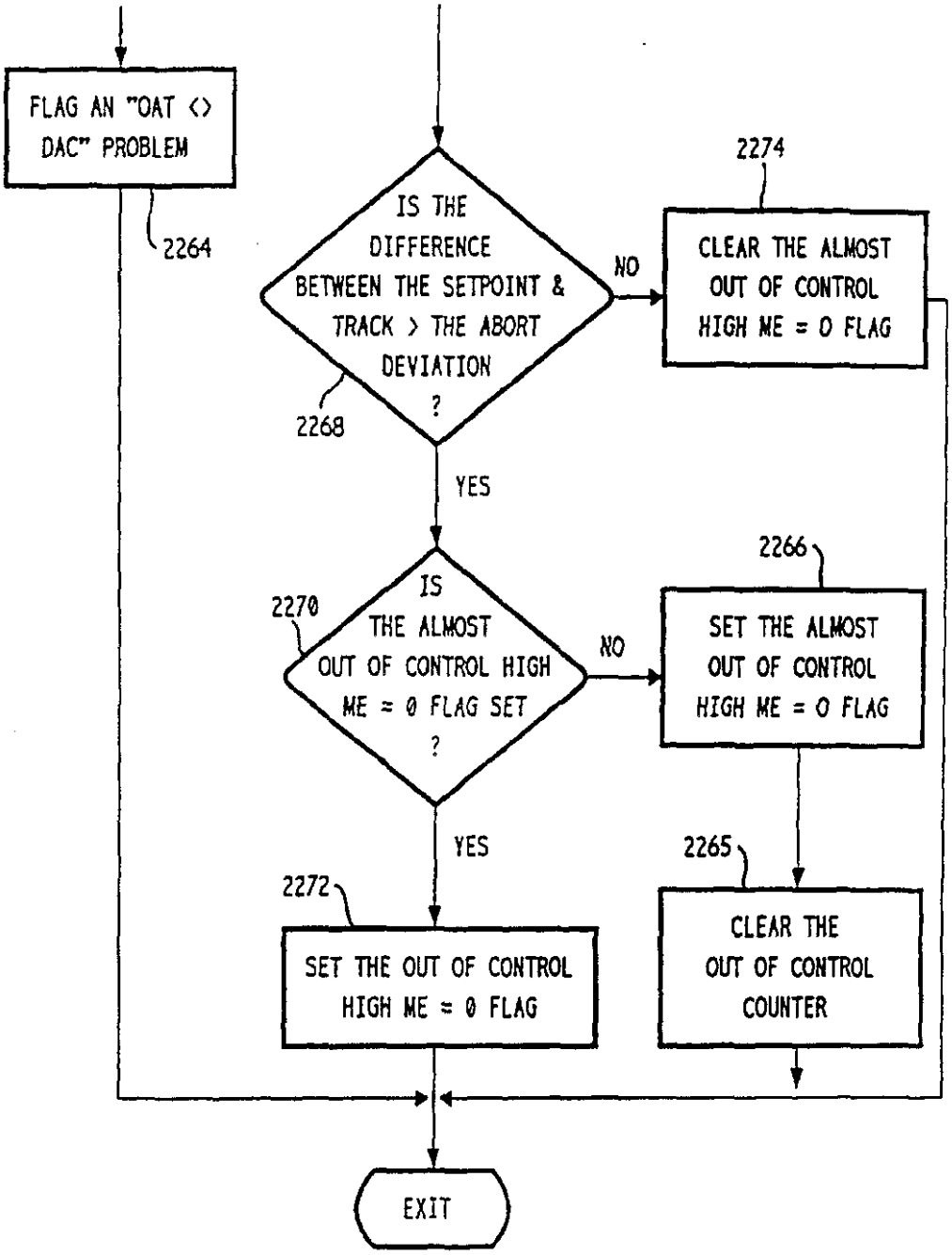
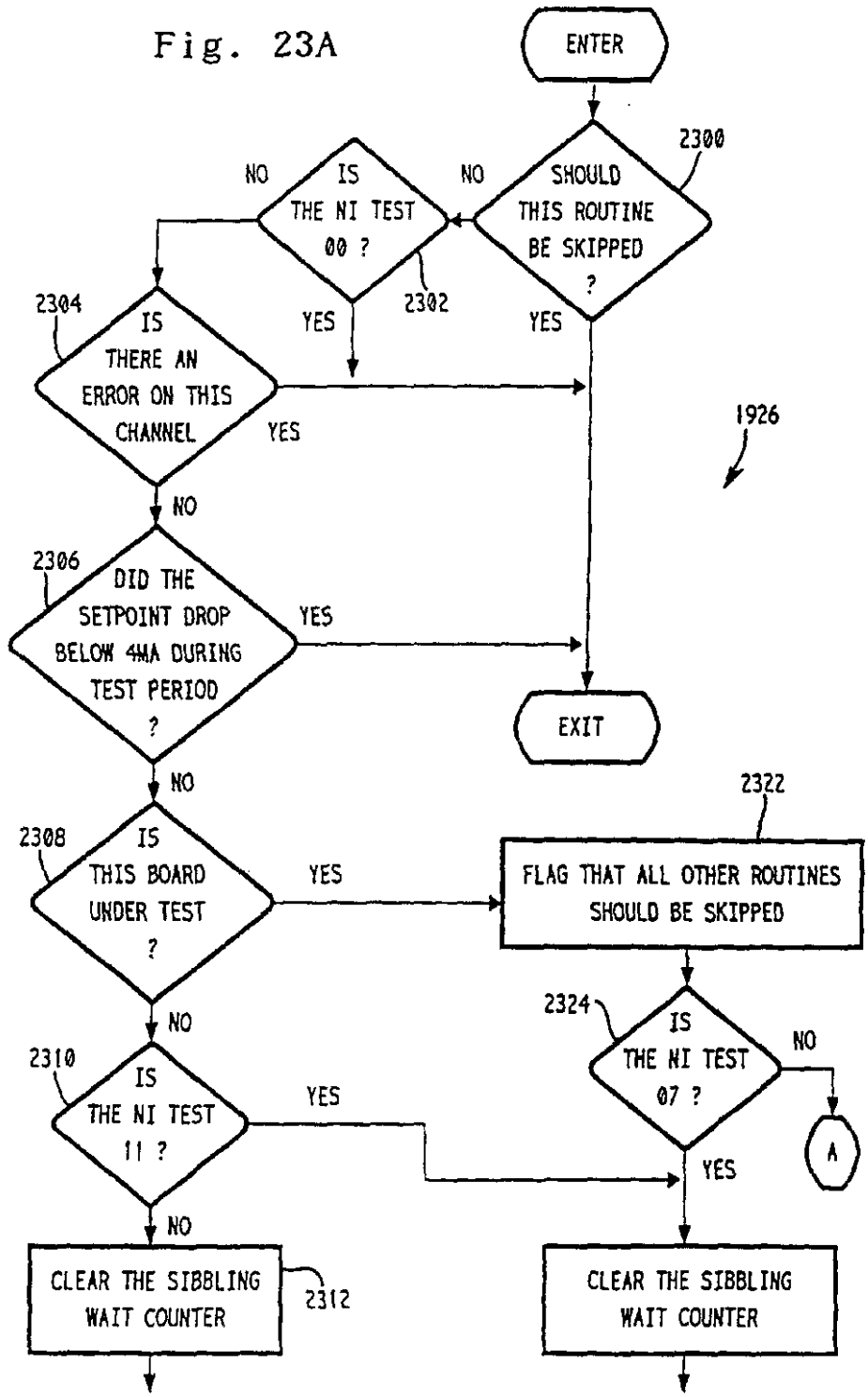

FLAG AN "OAT <> DAC" PROBLEM
2264
IS THE DIFFERENCE BETWEEN THE SETPOINT & TRACK > THE ABORT DEVIATION ?
2268
NO
2274
CLEAR THE ALMOST OUT OF CONTROL HIGH ME = 0 FLAG
YES
IS THE ALMOST OUT OF CONTROL HIGH ME = 0 FLAG SET ?
2270
NO
2266
SET THE ALMOST OUT OF CONTROL HIGH ME = 0 FLAG
YES
2272
SET THE OUT OF CONTROL HIGH ME = 0 FLAG
2265
CLEAR THE OUT OF CONTROL COUNTER
EXIT

Fig. 22R

Fig. 23A


ENTER
2300
SHOULD THIS ROUTINE BE SKIPPED ?
NO
IS THE NI TEST 00 ?
2302
NO
YES
YES
2304
IS THERE AN ERROR ON THIS CHANNEL
YES
1926
NO
2306
DID THE SETPOINT DROP BELOW 4MA DURING TEST PERIOD ?
YES
EXIT
NO
2322
2308
IS THIS BOARD UNDER TEST ?
YES
FLAG THAT ALL OTHER ROUTINES SHOULD BE SKIPPED
NO
2324
IS THE NI TEST 07 ?
NO
2310
IS THE NI TEST 11 ?
YES
A
YES
NO
CLEAR THE SIBBLING WAIT COUNTER
2312
CLEAR THE SIBBLING WAIT COUNTER

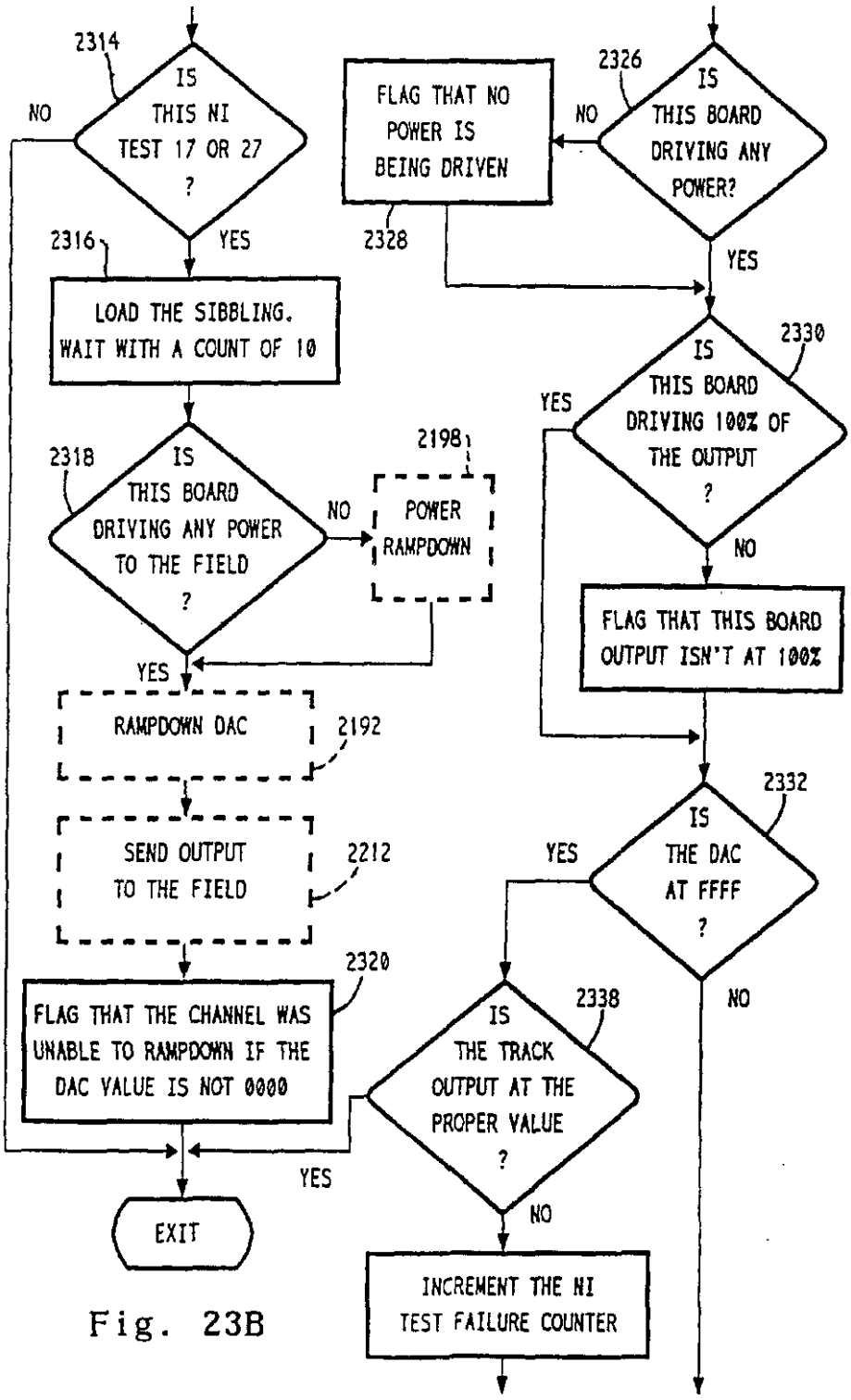

2314
IS THIS NI TEST 17 OR 27 ?
NO
YES
2316
LOAD THE SIBBLING. WAIT WITH A COUNT OF 10
2318
IS THIS BOARD DRIVING ANY POWER TO THE FIELD ?
NO
2198
POWER RAMPDOWN
YES
RAMPDOWN DAC
2192
SEND OUTPUT TO THE FIELD
2212
2320
FLAG THAT THE CHANNEL WAS UNABLE TO RAMPDOWN IF THE DAC VALUE IS NOT 0000
YES
EXIT
FLAG THAT NO POWER IS BEING DRIVEN
2328
2326
IS THIS BOARD DRIVING ANY POWER?
NO
YES
2330
IS THIS BOARD DRIVING 100% OF THE OUTPUT ?
YES
NO
FLAG THAT THIS BOARD OUTPUT ISN'T AT 100%
2332
IS THE DAC AT FFFF ?
YES
NO
2338
IS THE TRACK OUTPUT AT THE PROPER VALUE ?
NO
INCREMENT THE NI TEST FAILURE COUNTER

Fig. 23B

Fig. 23C

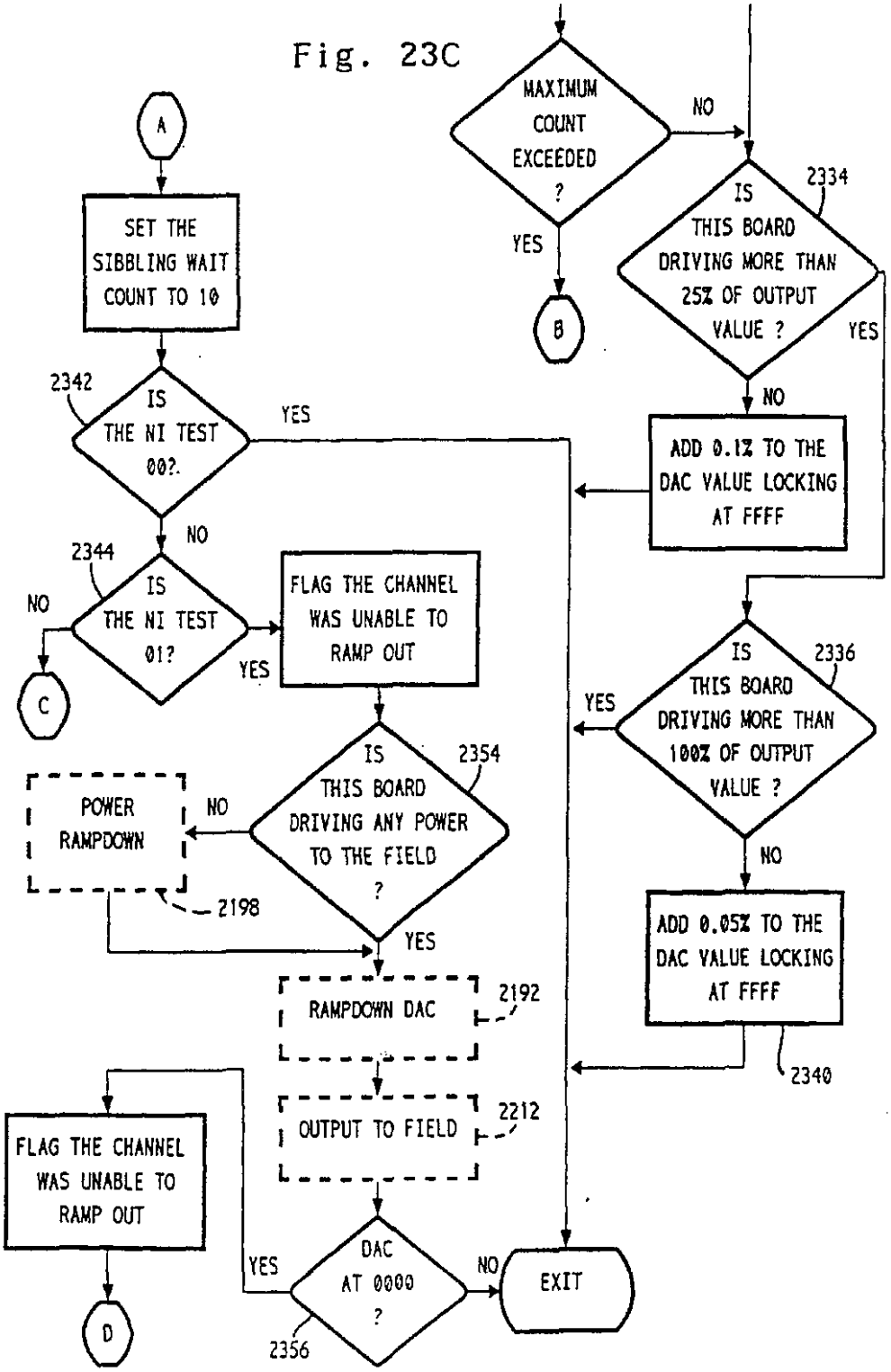
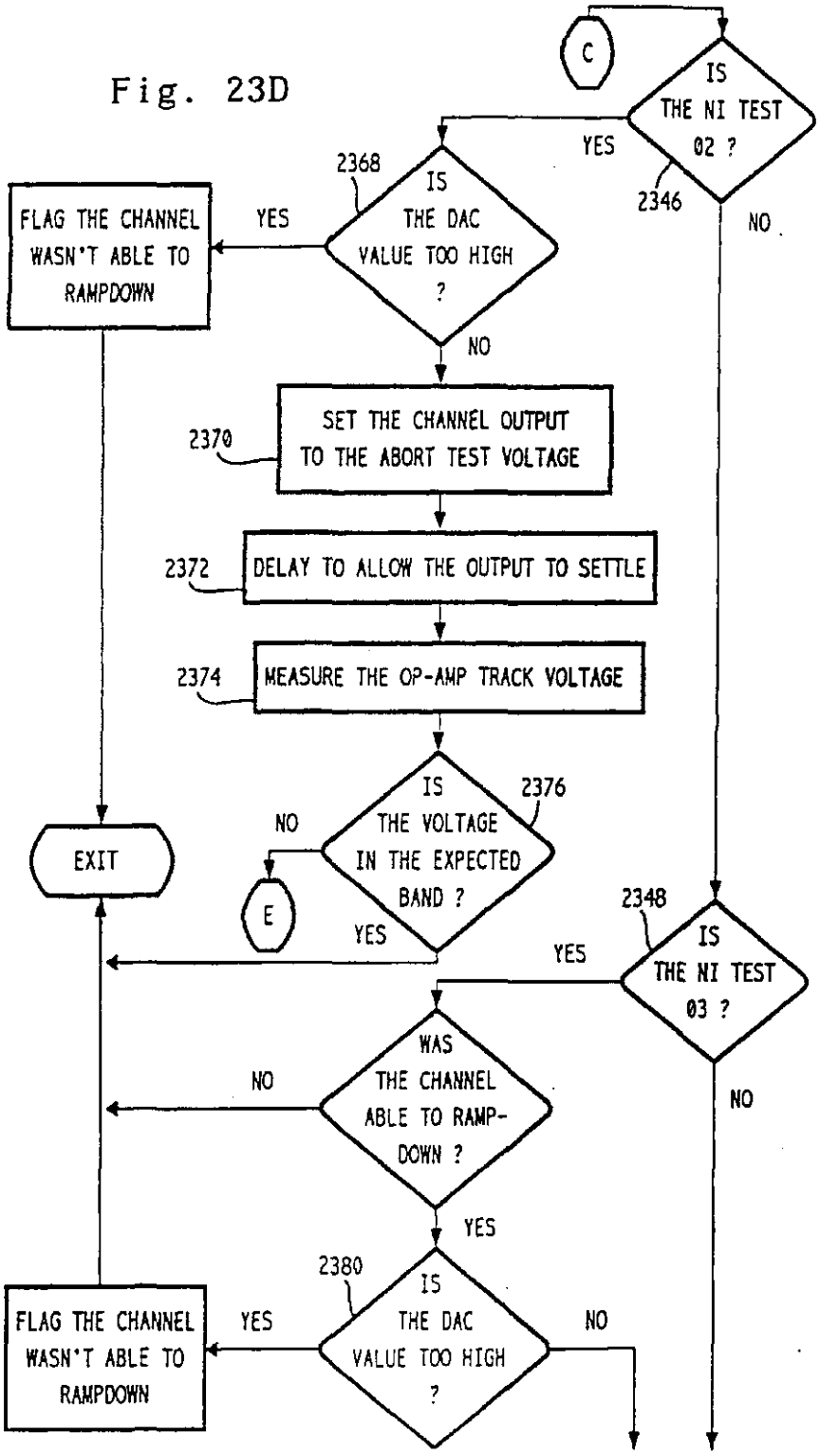

A
SET THE SIBBLING WAIT COUNT TO 10
2342
IS THE NI TEST 00?
YES
NO
2344
IS THE NI TEST 01?
NO
C
YES
FLAG THE CHANNEL WAS UNABLE TO RAMP OUT
IS THIS BOARD DRIVING ANY POWER TO THE FIELD ?
2354
NO
POWER RAMPDOWN
2198
YES
RAMPDOWN DAC
2192
OUTPUT TO FIELD
2212
FLAG THE CHANNEL WAS UNABLE TO RAMP OUT
D
YES
DAC AT 0000 ?
2356
NO
EXIT
MAXIMUM COUNT EXCEEDED ?
NO
YES
B
IS THIS BOARD DRIVING MORE THAN 25% OF OUTPUT VALUE ?
2334
YES
NO
ADD 0.1% TO THE DAC VALUE LOCKING AT FFFF
IS THIS BOARD DRIVING MORE THAN 100% OF OUTPUT VALUE ?
2336
YES
NO
ADD 0.05% TO THE DAC VALUE LOCKING AT FFFF
2340

Fig. 23D


C
IS THE NI TEST 02 ?
YES
NO
2346
2368
IS THE DAC VALUE TOO HIGH ?
YES
FLAG THE CHANNEL WASN'T ABLE TO RAMPDOWN
NO
2370
SET THE CHANNEL OUTPUT TO THE ABORT TEST VOLTAGE
2372
DELAY TO ALLOW THE OUTPUT TO SETTLE
2374
MEASURE THE OP-AMP TRACK VOLTAGE
2376
IS THE VOLTAGE IN THE EXPECTED BAND ?
NO
YES
E
EXIT
2348
IS THE NI TEST 03 ?
YES
NO
WAS THE CHANNEL ABLE TO RAMP-DOWN ?
NO
YES
2380
IS THE DAC VALUE TOO HIGH ?
YES
NO
FLAG THE CHANNEL WASN'T ABLE TO RAMPDOWN

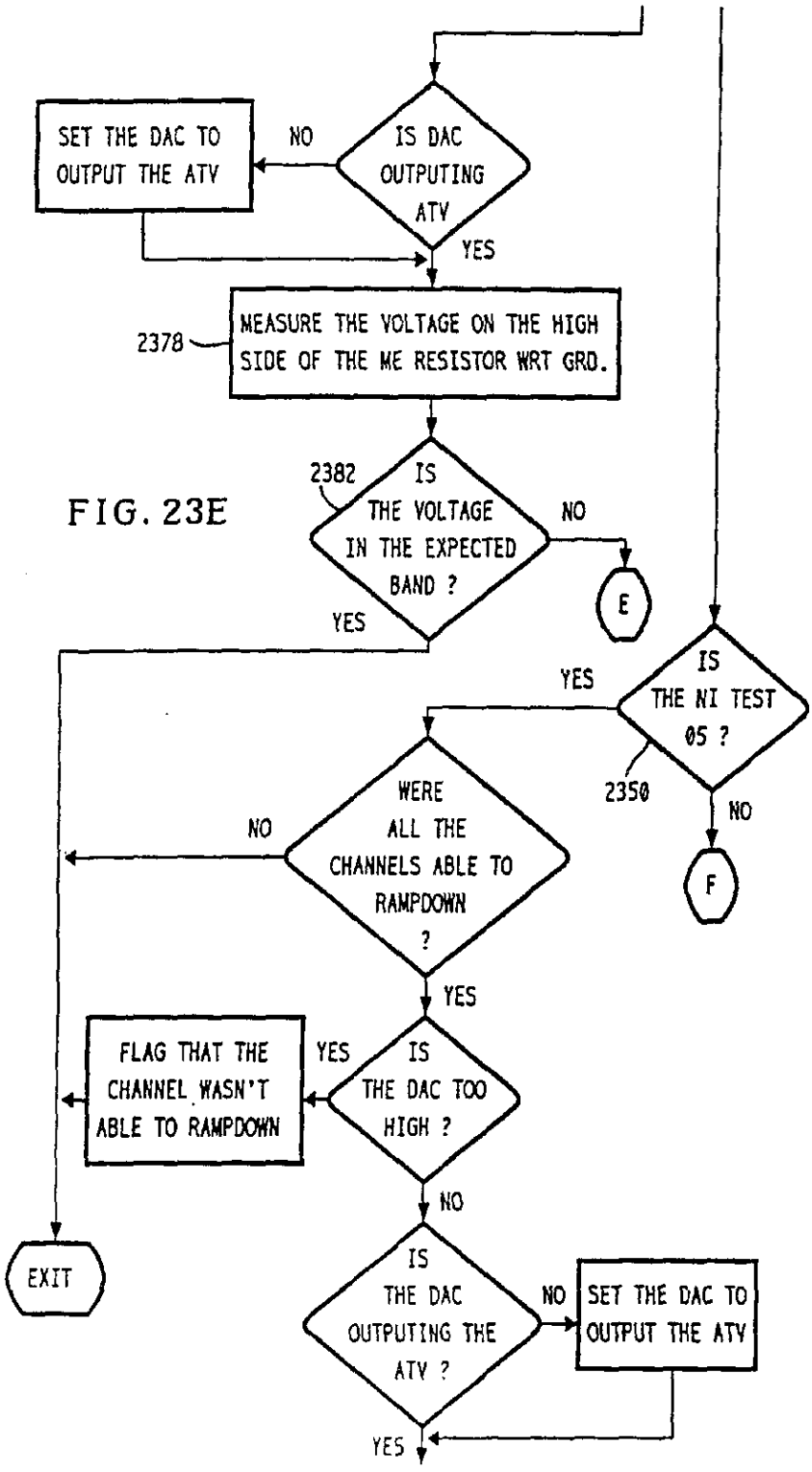

SET THE DAC TO OUTPUT THE ATV
NO
IS DAC OUTPUTING ATV
YES
2378
MEASURE THE VOLTAGE ON THE HIGH SIDE OF THE ME RESISTOR WRT GRD.
2382
IS THE VOLTAGE IN THE EXPECTED BAND ?
NO
E
YES
IS THE NI TEST 05 ?
YES
2350
NO
F
WERE ALL THE CHANNELS ABLE TO RAMPDOWN ?
NO
YES
FLAG THAT THE CHANNEL WASN'T ABLE TO RAMPDOWN
YES
IS THE DAC TOO HIGH ?
NO
EXIT
IS THE DAC OUTPUTING THE ATV ?
NO
SET THE DAC TO OUTPUT THE ATV
YES

FIG. 23E

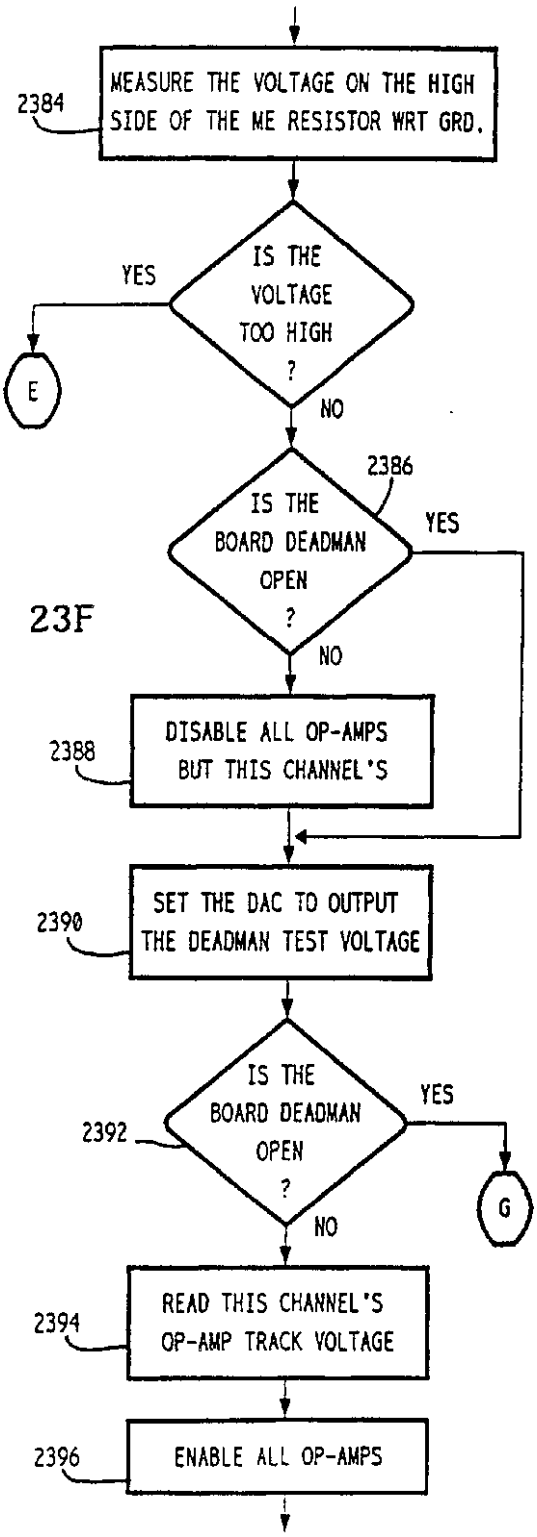

MEASURE THE VOLTAGE ON THE HIGH SIDE OF THE ME RESISTOR WRT GRD.
2384
IS THE VOLTAGE TOO HIGH ?
YES
NO
E
IS THE BOARD DEADMAN OPEN ?
2386
YES
NO
DISABLE ALL OP-AMPS BUT THIS CHANNEL'S
2388
SET THE DAC TO OUTPUT THE DEADMAN TEST VOLTAGE
2390
IS THE BOARD DEADMAN OPEN ?
2392
YES
NO
G
READ THIS CHANNEL'S OP-AMP TRACK VOLTAGE
2394
ENABLE ALL OP-AMPS
2396

FIG. 23F

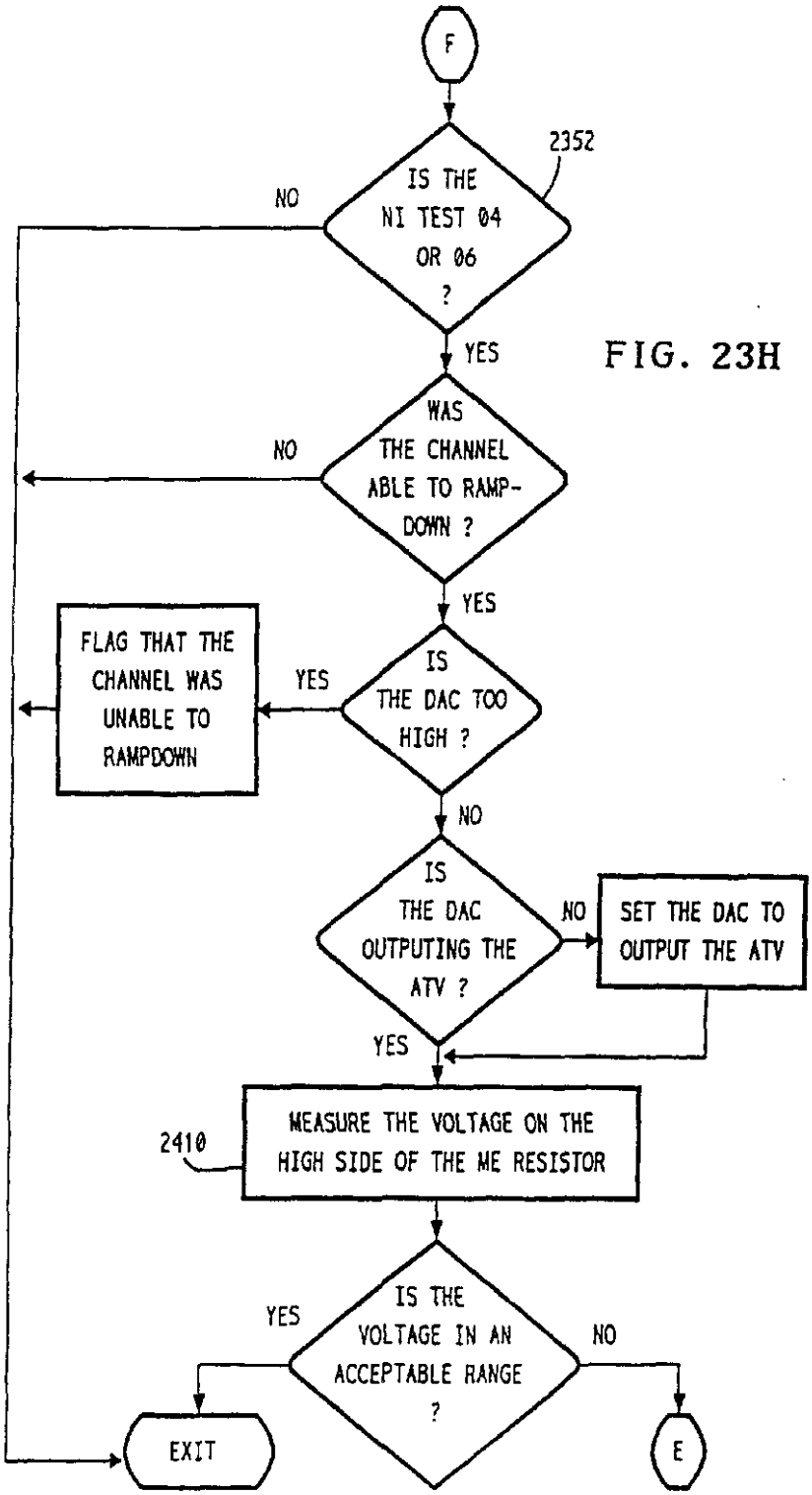

F
2352
IS THE NI TEST 04 OR 06 ?
NO
YES
WAS THE CHANNEL ABLE TO RAMP-DOWN ?
NO
YES
IS THE DAC TOO HIGH ?
YES
FLAG THAT THE CHANNEL WAS UNABLE TO RAMPDOWN
NO
IS THE DAC OUTPUTING THE ATV ?
NO
SET THE DAC TO OUTPUT THE ATV
YES
2410
MEASURE THE VOLTAGE ON THE HIGH SIDE OF THE ME RESISTOR
IS THE VOLTAGE IN AN ACCEPTABLE RANGE ?
YES
NO
EXIT
E

FIG. 23H

FIG. 23G

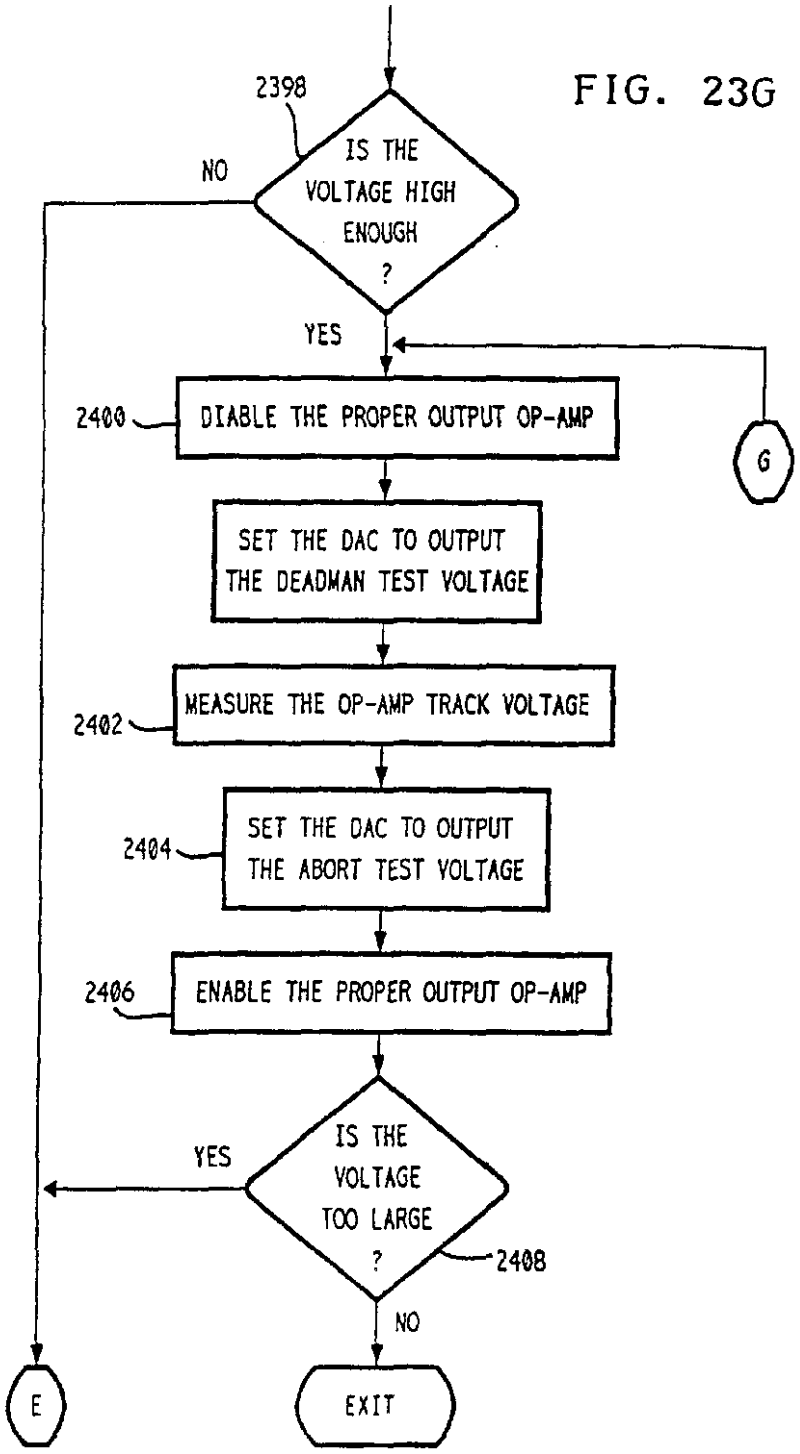
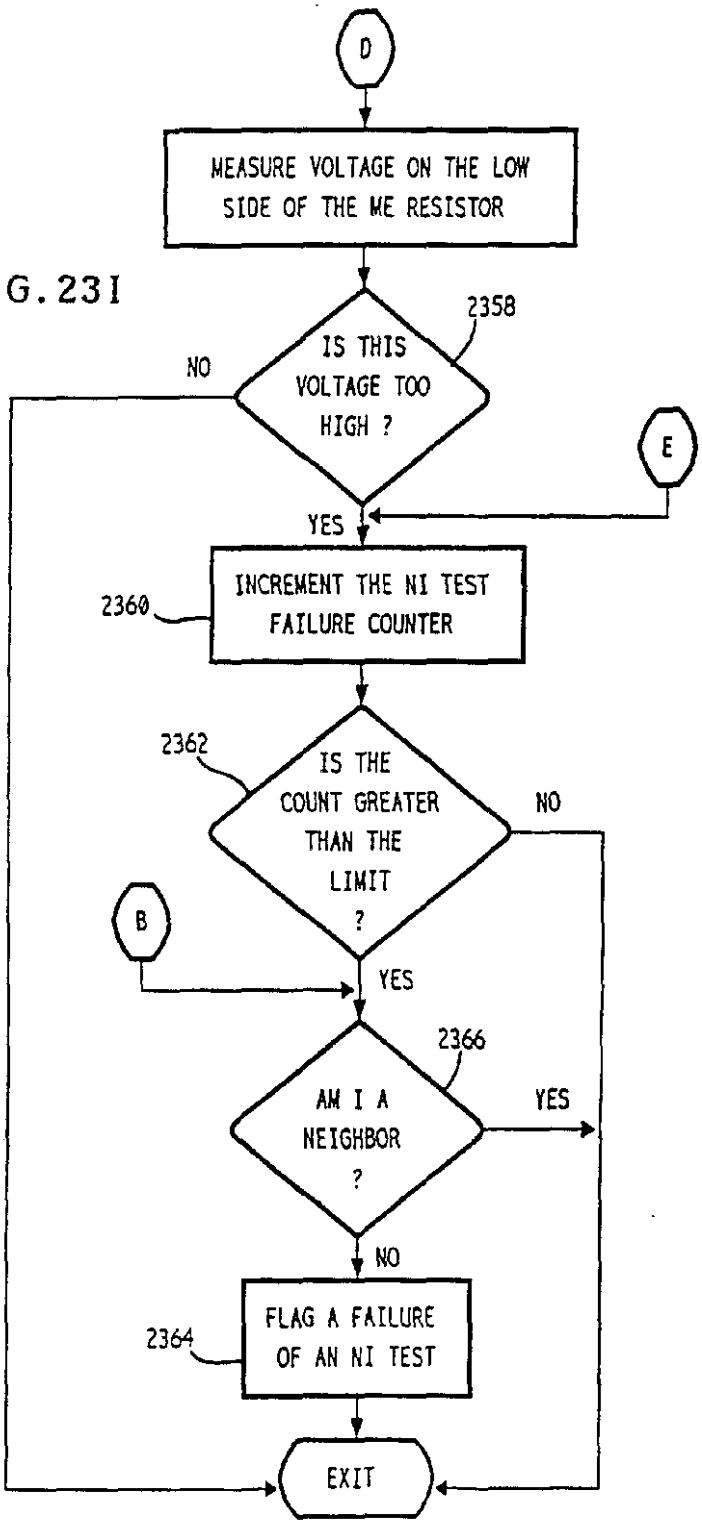

2398
IS THE VOLTAGE HIGH ENOUGH ?
NO
YES
2400
DIABLE THE PROPER OUTPUT OP-AMP
G
SET THE DAC TO OUTPUT THE DEADMAN TEST VOLTAGE
2402
MEASURE THE OP-AMP TRACK VOLTAGE
2404
SET THE DAC TO OUTPUT THE ABORT TEST VOLTAGE
2406
ENABLE THE PROPER OUTPUT OP-AMP
YES
IS THE VOLTAGE TOO LARGE ?
2408
NO
E
EXIT

FIG. 23I


D
MEASURE VOLTAGE ON THE LOW SIDE OF THE ME RESISTOR
2358
IS THIS VOLTAGE TOO HIGH ?
NO
E
YES
2360
INCREMENT THE NI TEST FAILURE COUNTER
2362
IS THE COUNT GREATER THAN THE LIMIT ?
NO
B
YES
2366
AM I A NEIGHBOR ?
YES
NO
2364
FLAG A FAILURE OF AN NI TEST
EXIT

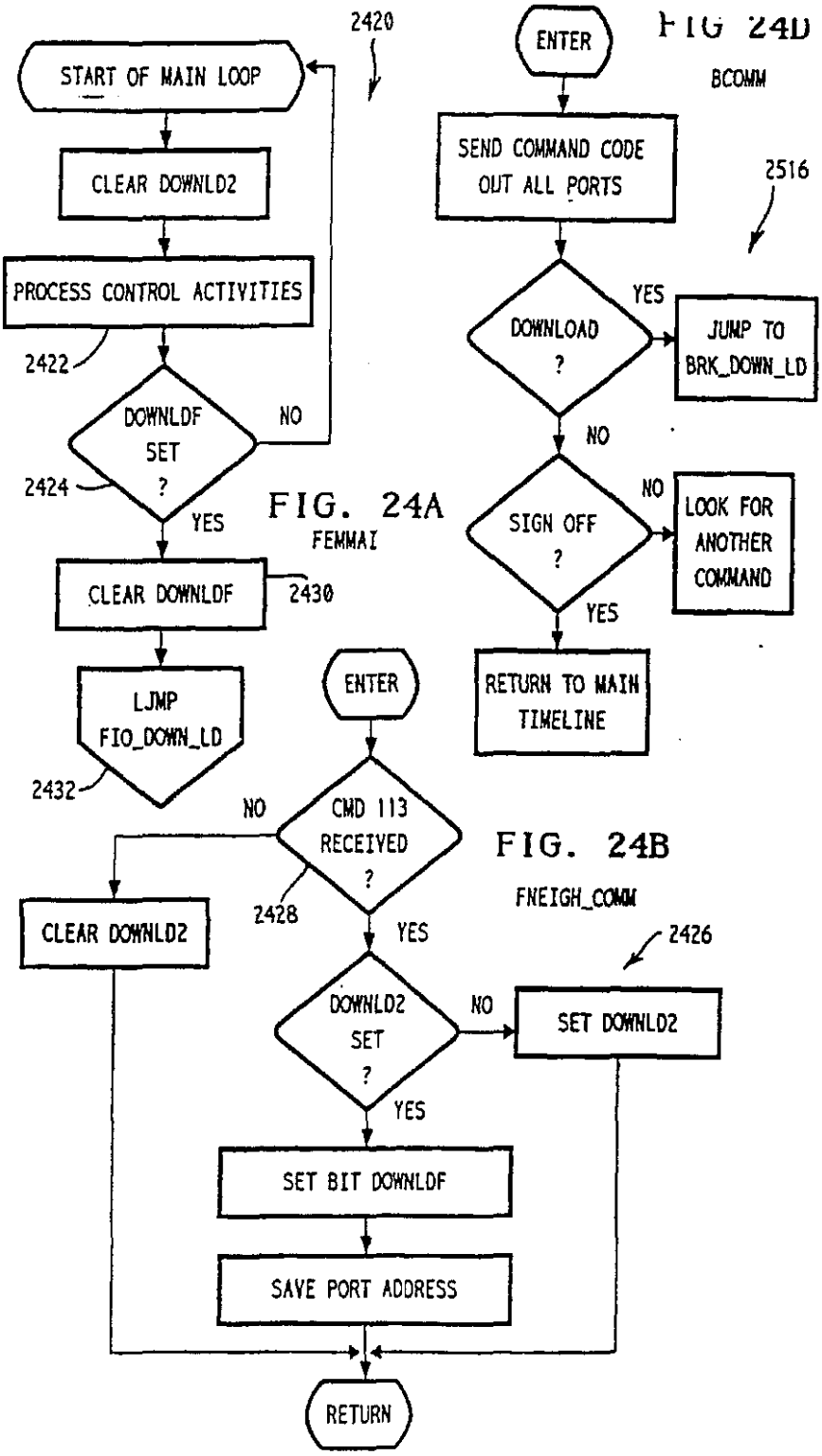
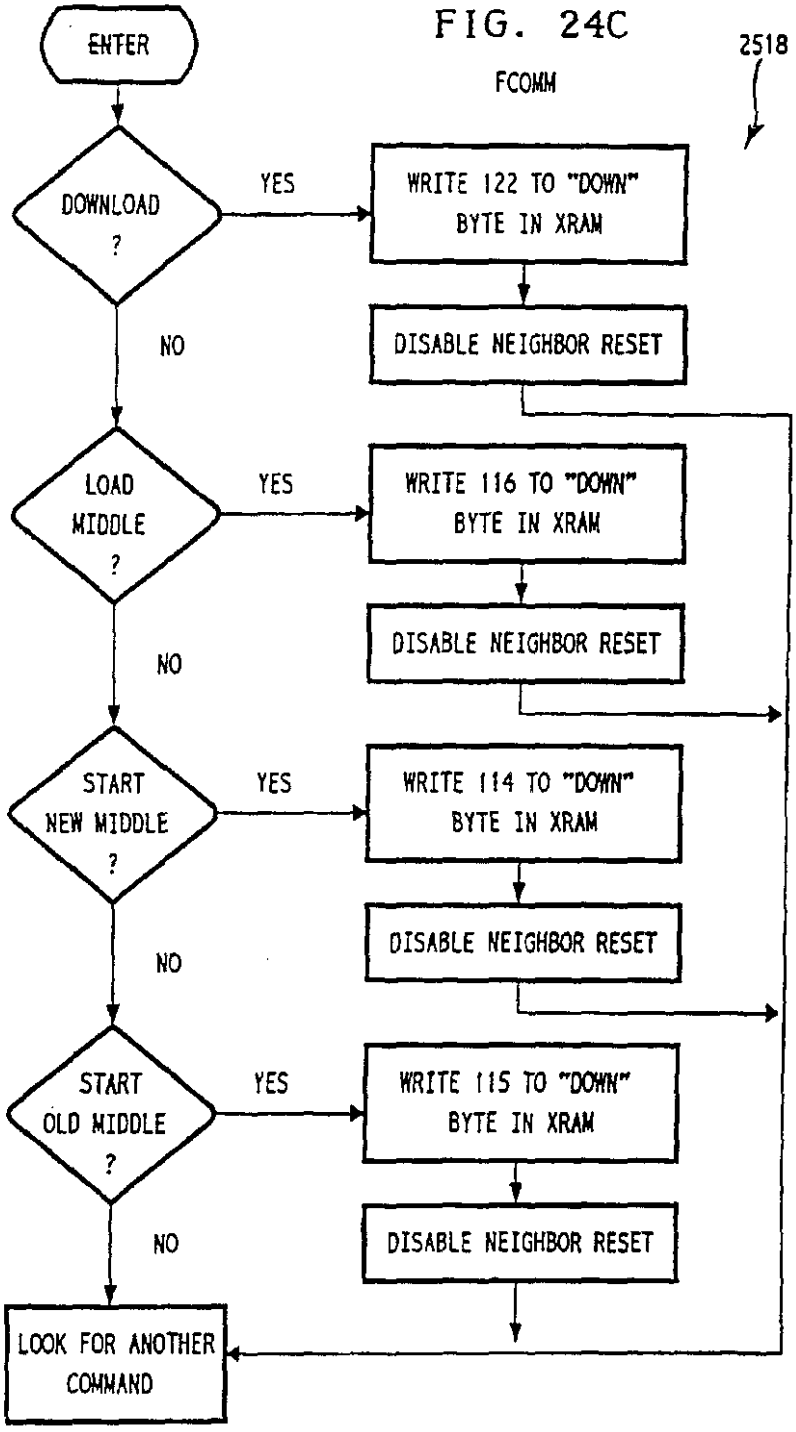


FIG. 24A
FEMMAI

FIG. 24B
FNEIGH_COMM

FIG. 24D
BCOMM

FIG. 24C


ENTER
FCOMM
2518
DOWNLOAD ?
YES
WRITE 122 TO "DOWN" BYTE IN XRAM
DISABLE NEIGHBOR RESET
NO
LOAD MIDDLE ?
YES
WRITE 116 TO "DOWN" BYTE IN XRAM
DISABLE NEIGHBOR RESET
NO
START NEW MIDDLE ?
YES
WRITE 114 TO "DOWN" BYTE IN XRAM
DISABLE NEIGHBOR RESET
NO
START OLD MIDDLE ?
YES
WRITE 115 TO "DOWN" BYTE IN XRAM
DISABLE NEIGHBOR RESET
NO
LOOK FOR ANOTHER COMMAND

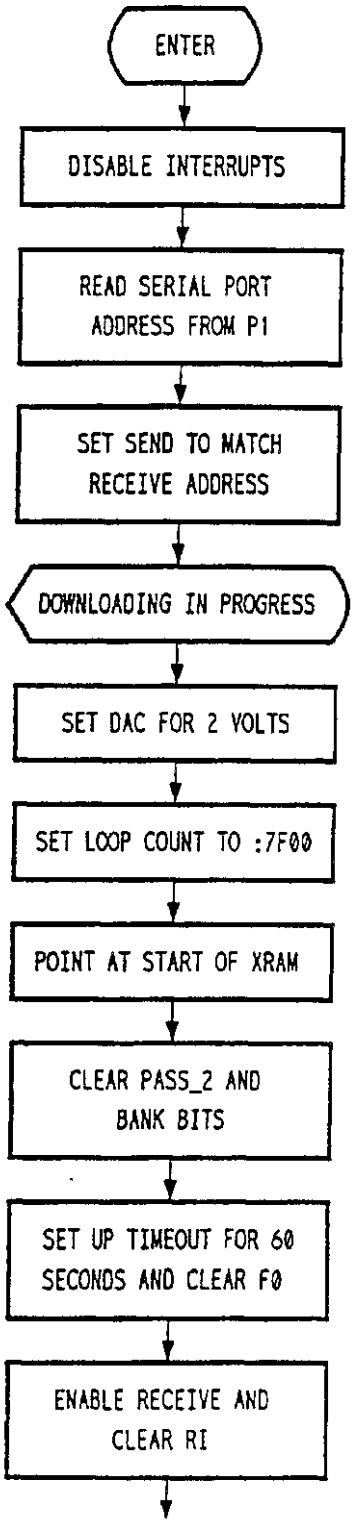


FIG. 24E

FIO_DOWN_LD

2524

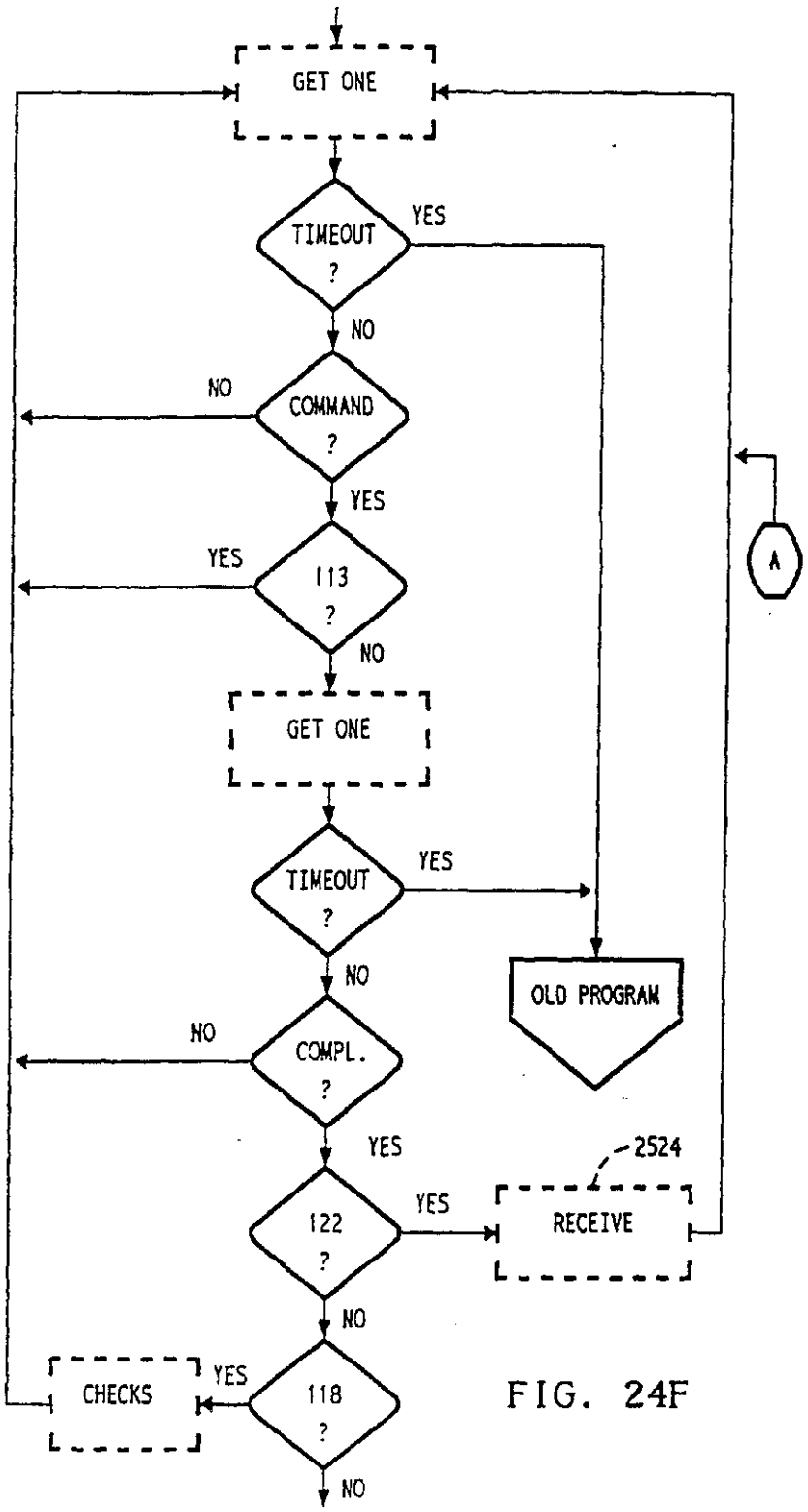

GET ONE
TIMEOUT ?
YES
NO
COMMAND ?
NO
YES
113 ?
YES
NO
GET ONE
TIMEOUT ?
YES
NO
OLD PROGRAM
COMPL. ?
NO
YES
2524
122 ?
YES
RECEIVE
NO
CHECKS
YES
118 ?
NO
A

FIG. 24F

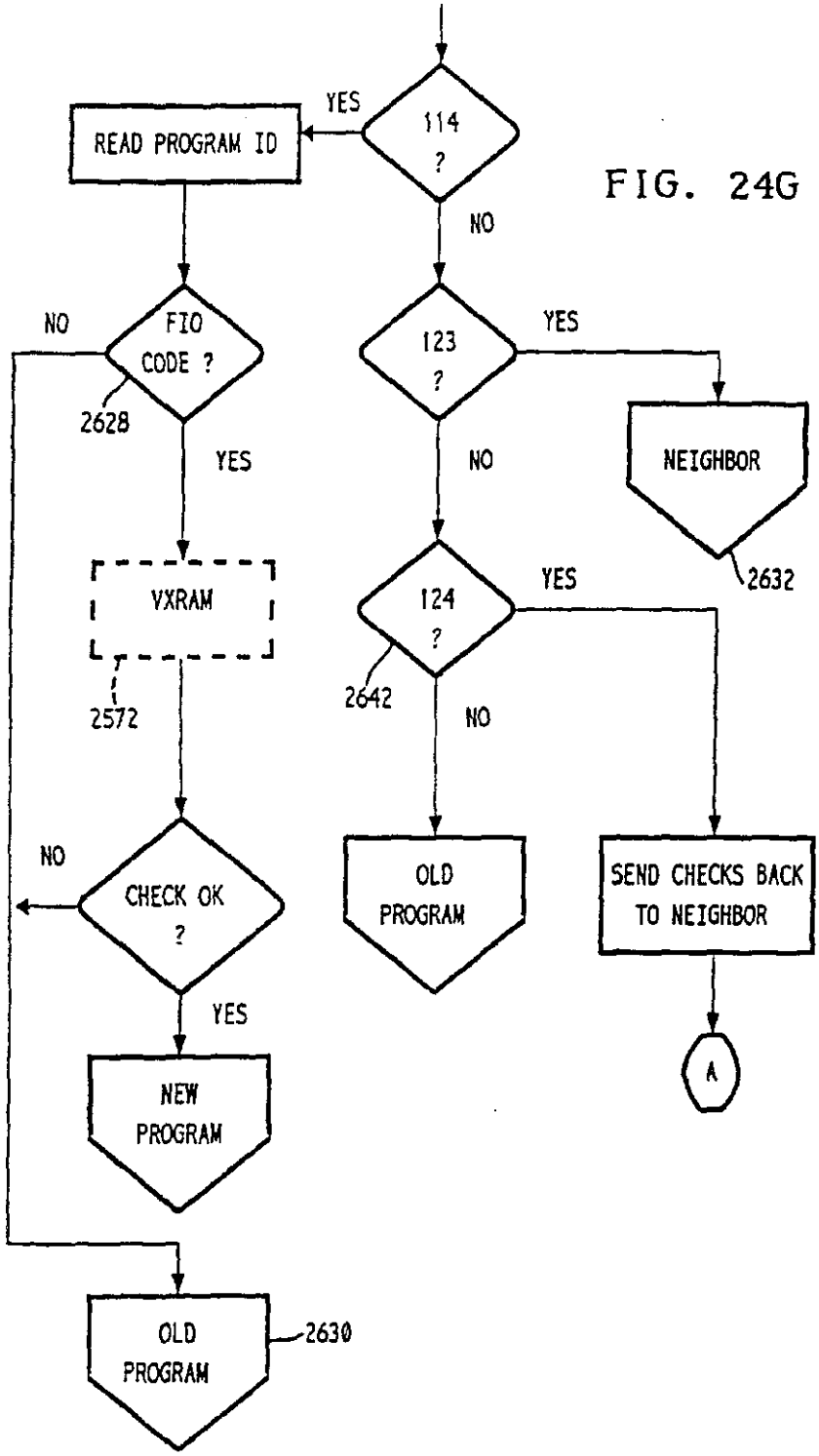

114 ?
YES
READ PROGRAM ID
NO
FIO CODE ?
NO
2628
YES
123 ?
YES
NEIGHBOR
2632
NO
VXRAM
2572
124 ?
YES
2642
NO
CHECK OK ?
NO
YES
OLD PROGRAM
SEND CHECKS BACK TO NEIGHBOR
A
NEW PROGRAM
OLD PROGRAM
2630

FIG. 24G

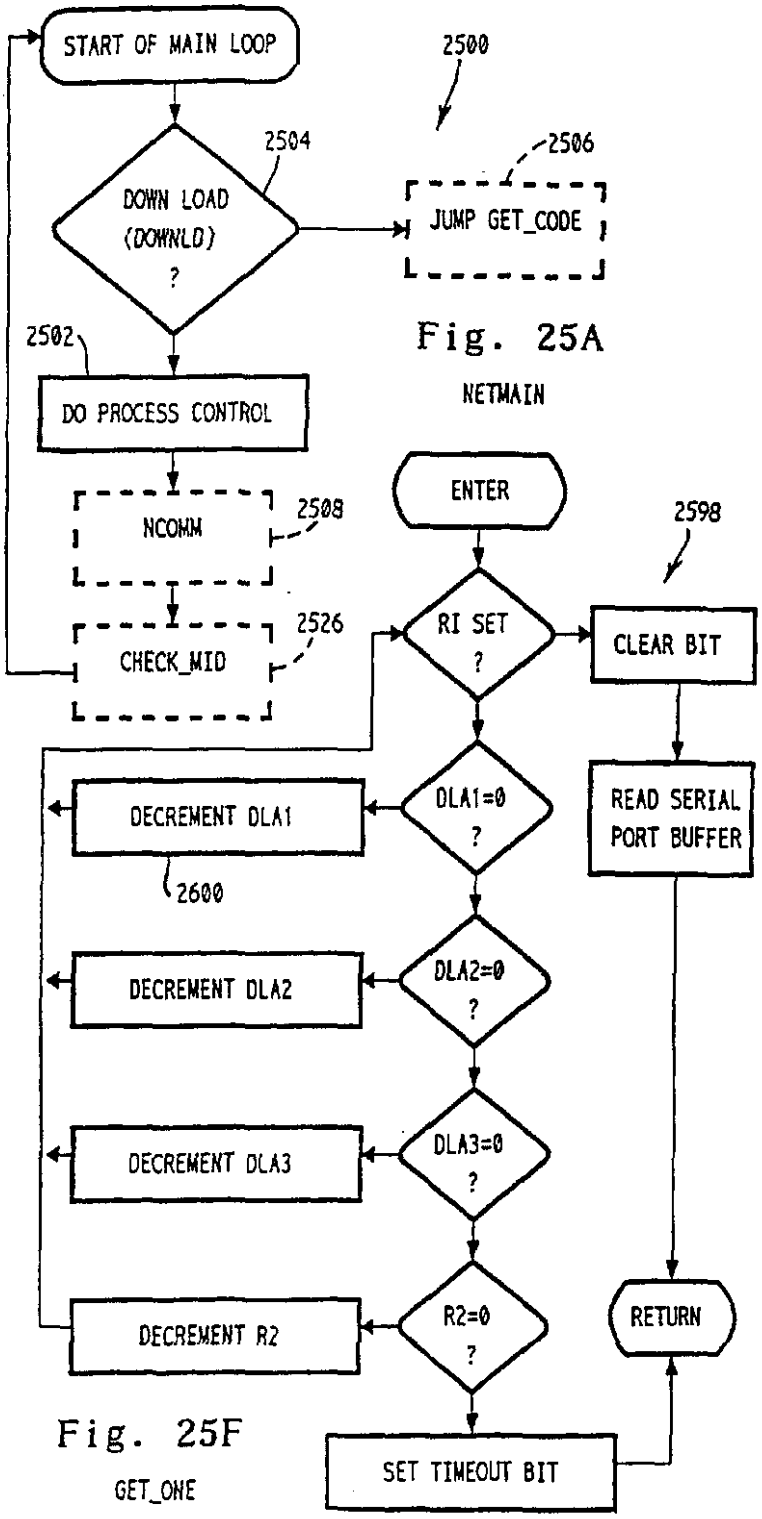


Fig. 25A

NETMAIN

Fig. 25F

GET_ONE

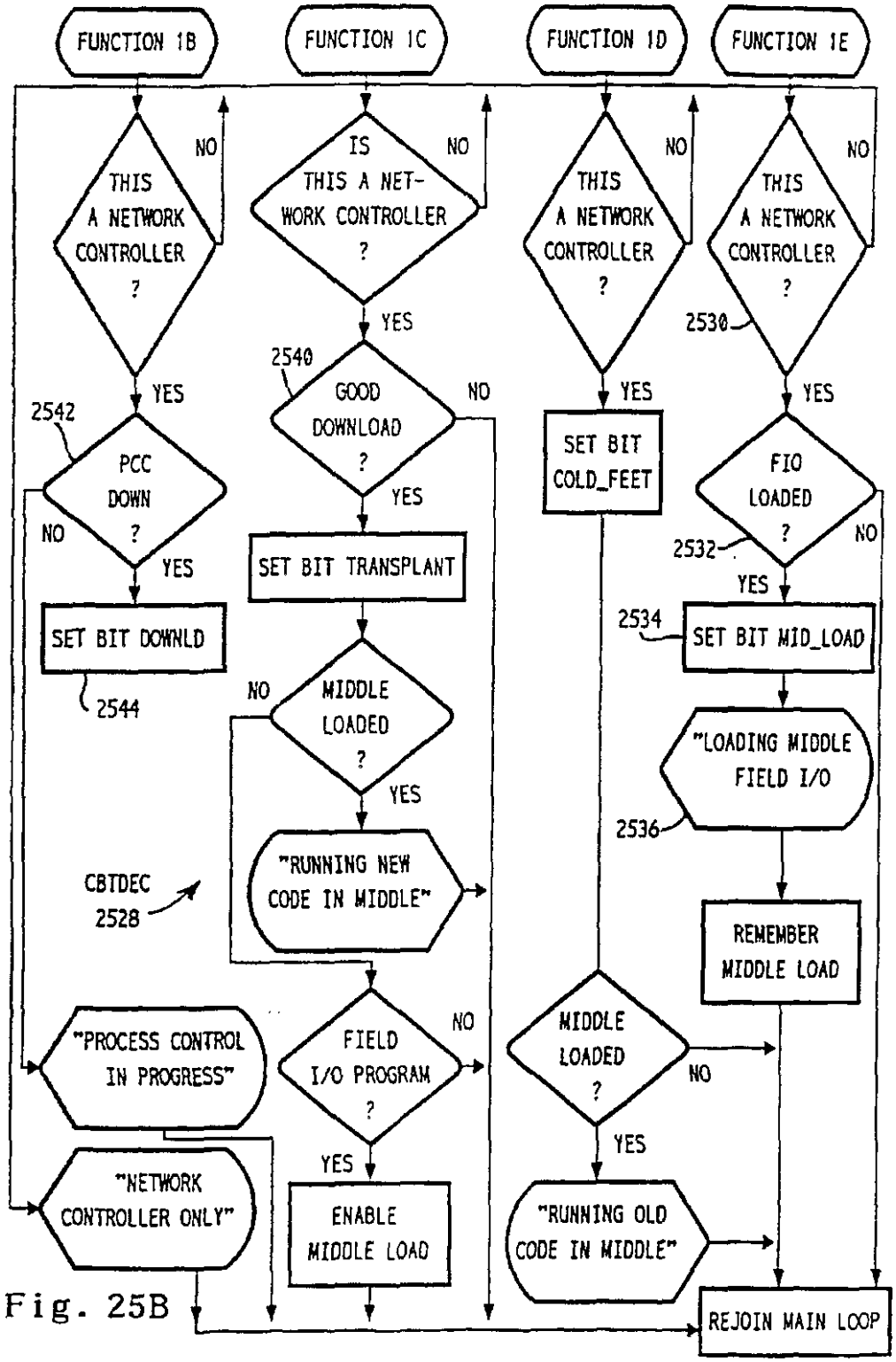

FUNCTION 1B
FUNCTION 1C
FUNCTION 1D
FUNCTION 1E
THIS A NETWORK CONTROLLER ?
NO
YES
IS THIS A NET-WORK CONTROLLER ?
NO
YES
THIS A NETWORK CONTROLLER ?
NO
YES
2530
THIS A NETWORK CONTROLLER ?
NO
YES
2542
PCC DOWN ?
NO
YES
SET BIT DOWNLD
2544
2540
GOOD DOWNLOAD ?
NO
YES
SET BIT TRANSPLANT
SET BIT COLD_FEET
FIO LOADED ?
2532
NO
YES
2534
SET BIT MID_LOAD
NO
MIDDLE LOADED ?
YES
"LOADING MIDDLE FIELD I/O
2536
CBTDEC
2528
"RUNNING NEW CODE IN MIDDLE"
REMEMBER MIDDLE LOAD
"PROCESS CONTROL IN PROGRESS"
FIELD I/O PROGRAM ?
NO
YES
MIDDLE LOADED ?
NO
YES
"NETWORK CONTROLLER ONLY"
ENABLE MIDDLE LOAD
"RUNNING OLD CODE IN MIDDLE"
REJOIN MAIN LOOP

Fig. 25B

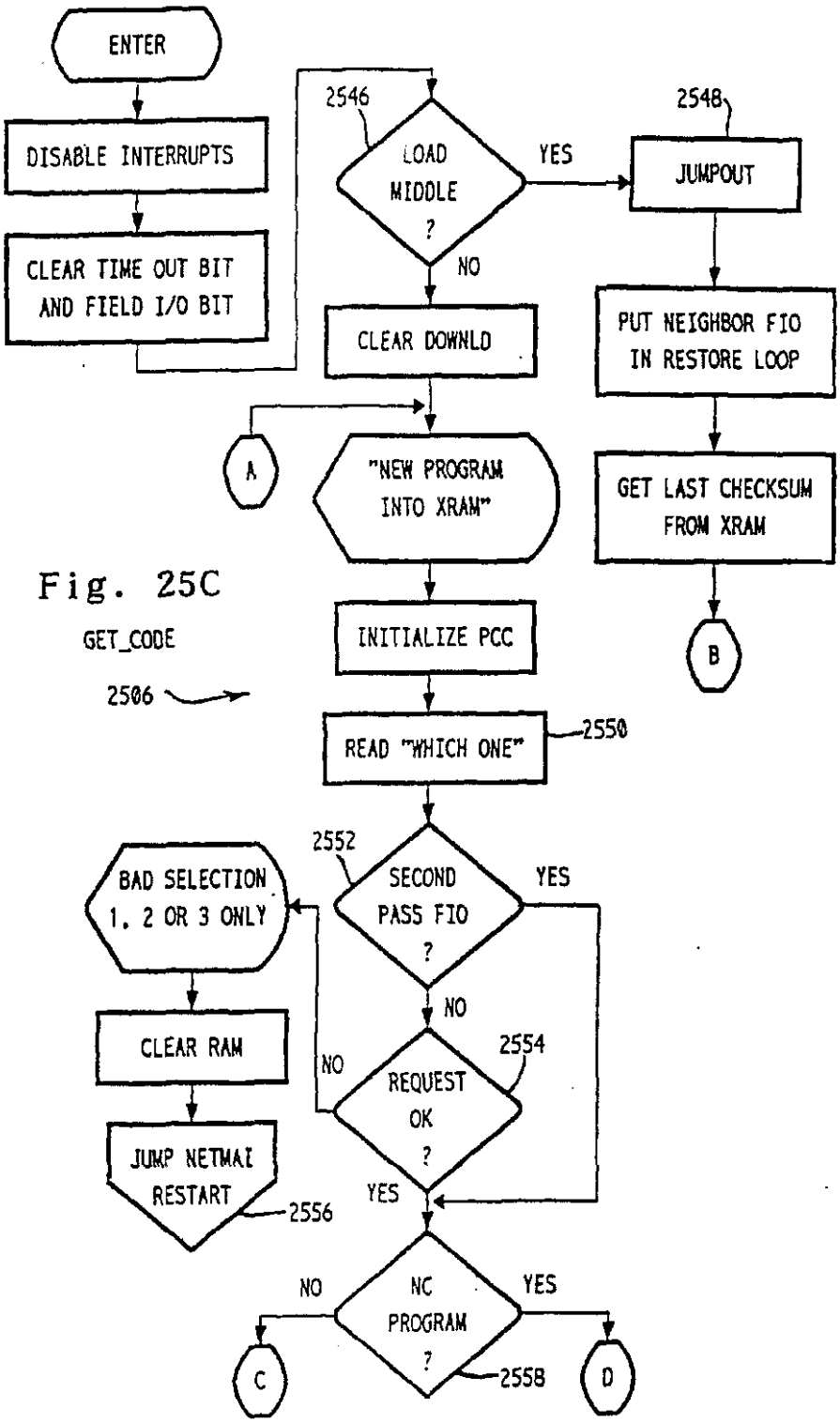


Fig. 25C

GET_CODE

2506

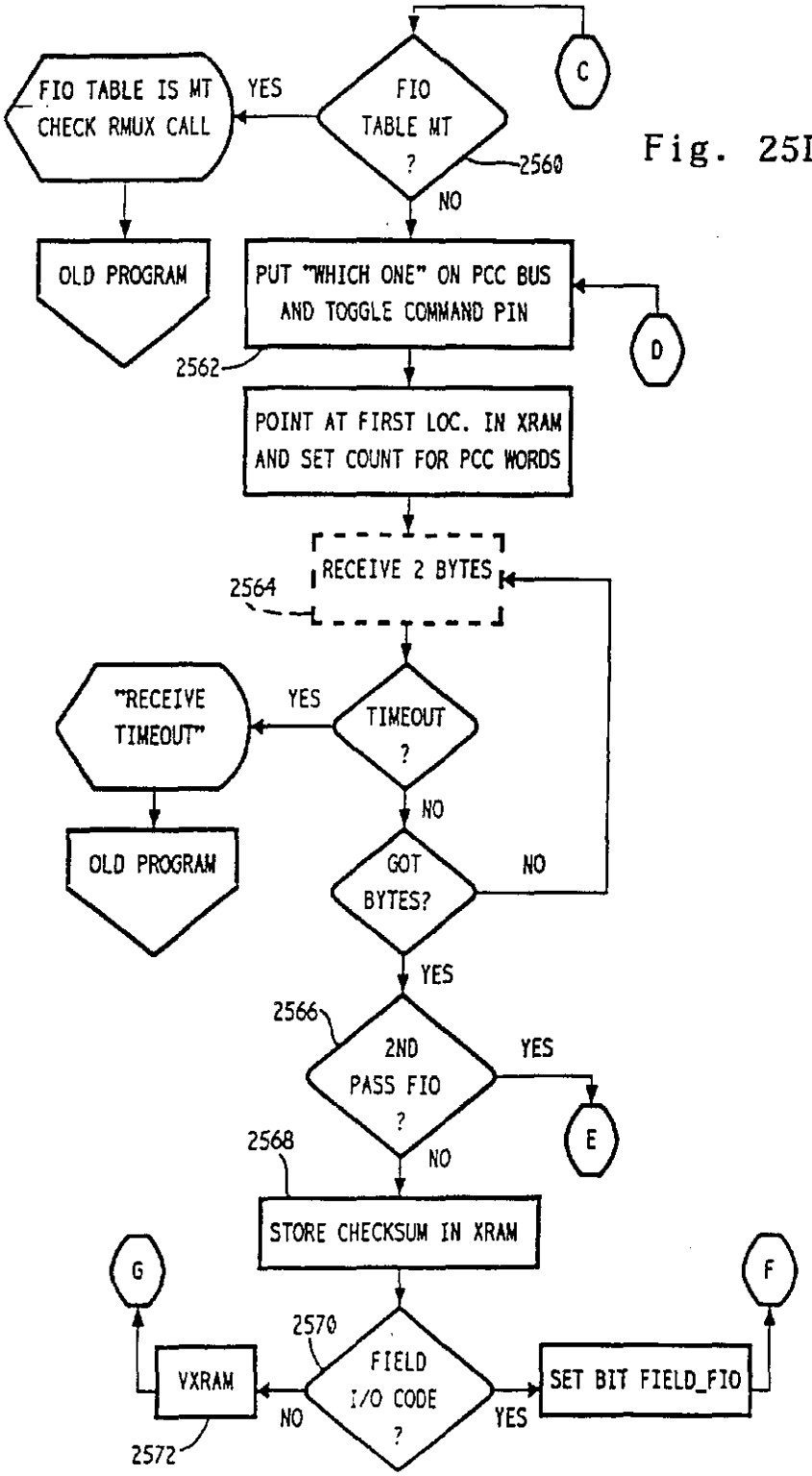

FIO TABLE IS MT
CHECK RMUX CALL
YES
FIO
TABLE MT
?
2560
C
NO
OLD PROGRAM
PUT "WHICH ONE" ON PCC BUS
AND TOGGLE COMMAND PIN
D
2562
POINT AT FIRST LOC. IN XRAM
AND SET COUNT FOR PCC WORDS
RECEIVE 2 BYTES
2564
"RECEIVE
TIMEOUT"
YES
TIMEOUT
?
NO
OLD PROGRAM
GOT
BYTES?
NO
YES
2566
2ND
PASS FIO
?
YES
E
NO
2568
STORE CHECKSUM IN XRAM
G
F
2570
FIELD
I/O CODE
?
VXRAM
NO
YES
SET BIT FIELD_FIO
2572

Fig. 25D

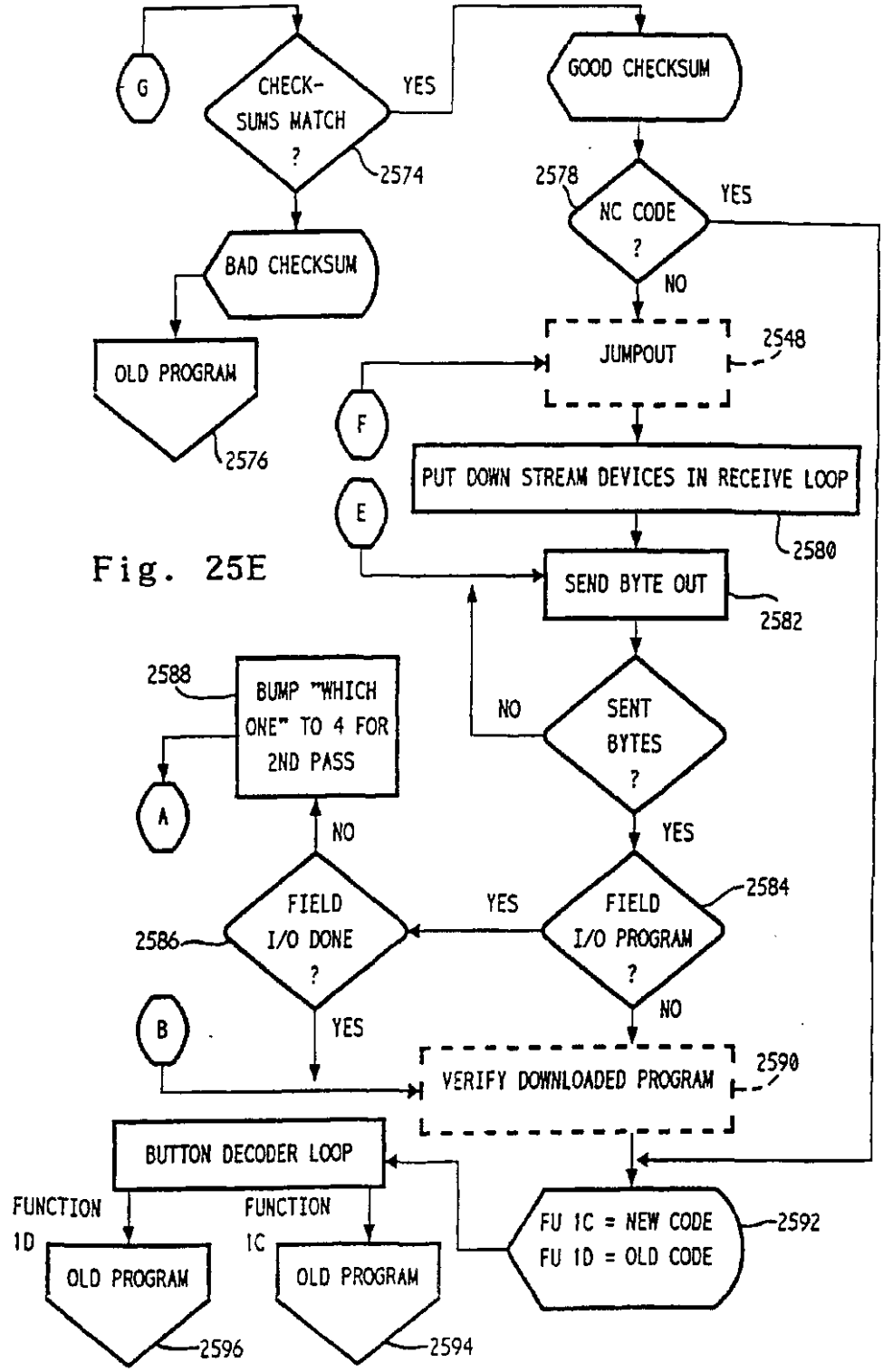

G
CHECK-SUMS MATCH ?
YES
2574
GOOD CHECKSUM
2578
NC CODE ?
YES
NO
BAD CHECKSUM
OLD PROGRAM
2576
JUMPOUT
2548
F
E
PUT DOWN STREAM DEVICES IN RECEIVE LOOP
2580
SEND BYTE OUT
2582
2588
BUMP "WHICH ONE" TO 4 FOR 2ND PASS
A
NO
SENT BYTES ?
YES
NO
FIELD I/O DONE ?
YES
FIELD I/O PROGRAM ?
2584
2586
B
YES
NO
VERIFY DOWNLOADED PROGRAM
2590
BUTTON DECODER LOOP
FUNCTION 1D
FUNCTION 1C
FU 1C = NEW CODE
FU 1D = OLD CODE
2592
OLD PROGRAM
2596
OLD PROGRAM
2594

Fig. 25E

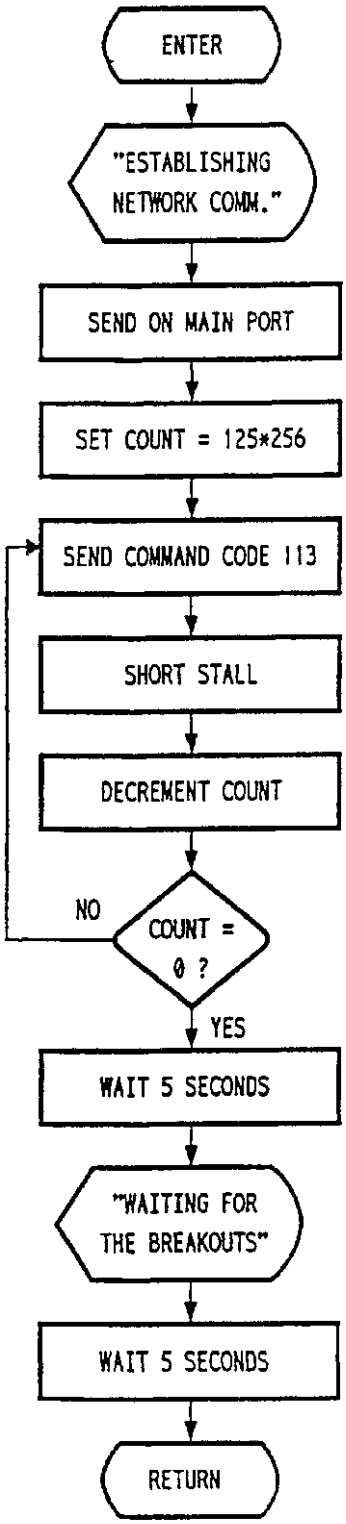


Fig. 25H

GET_CODE/JUMPOUT

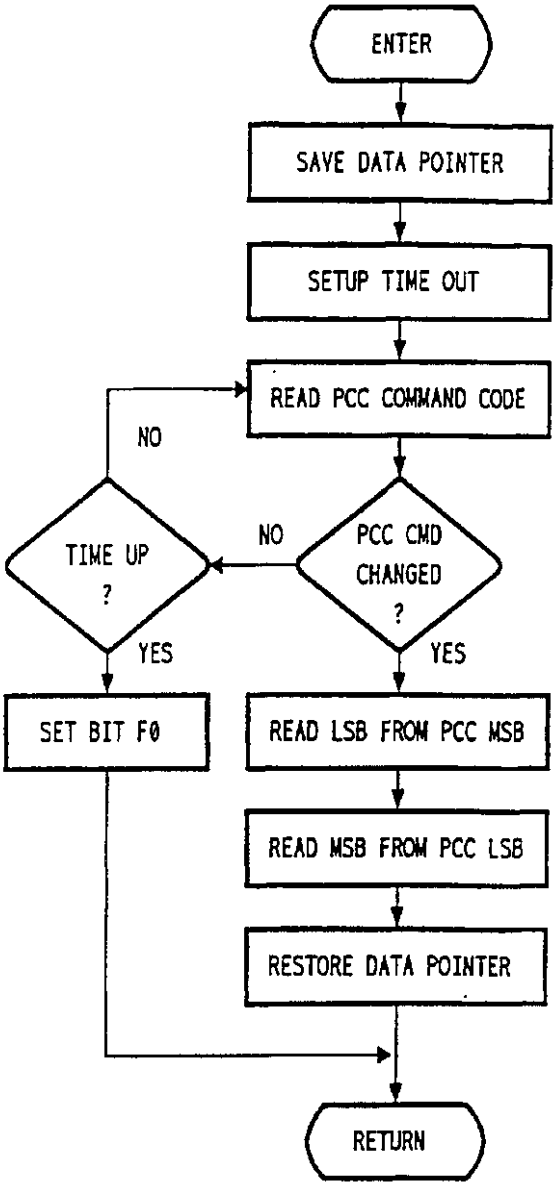


Fig. 25G

GET_CODE/RECEIVE2BYTES

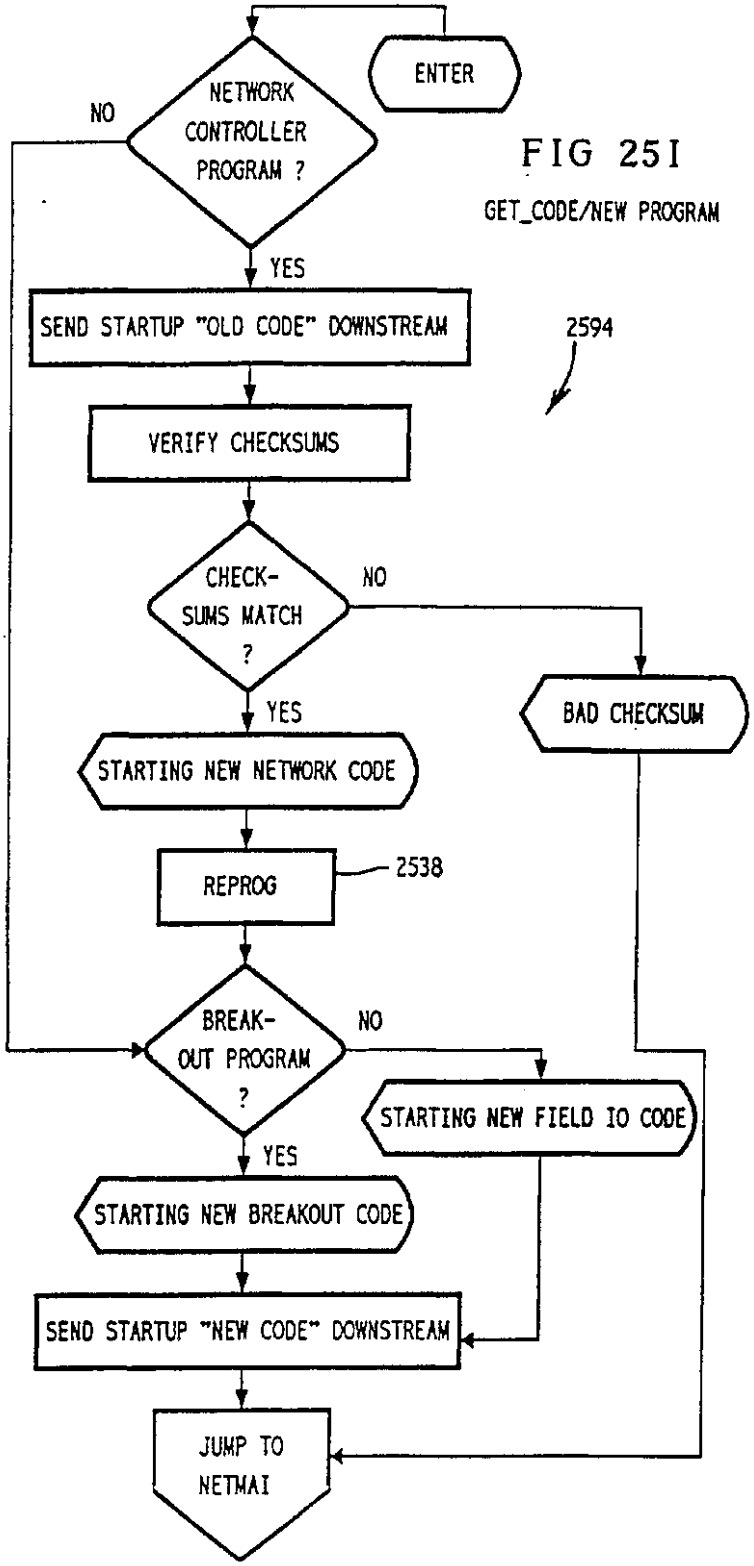


FIG 25I

GET_CODE/NEW PROGRAM

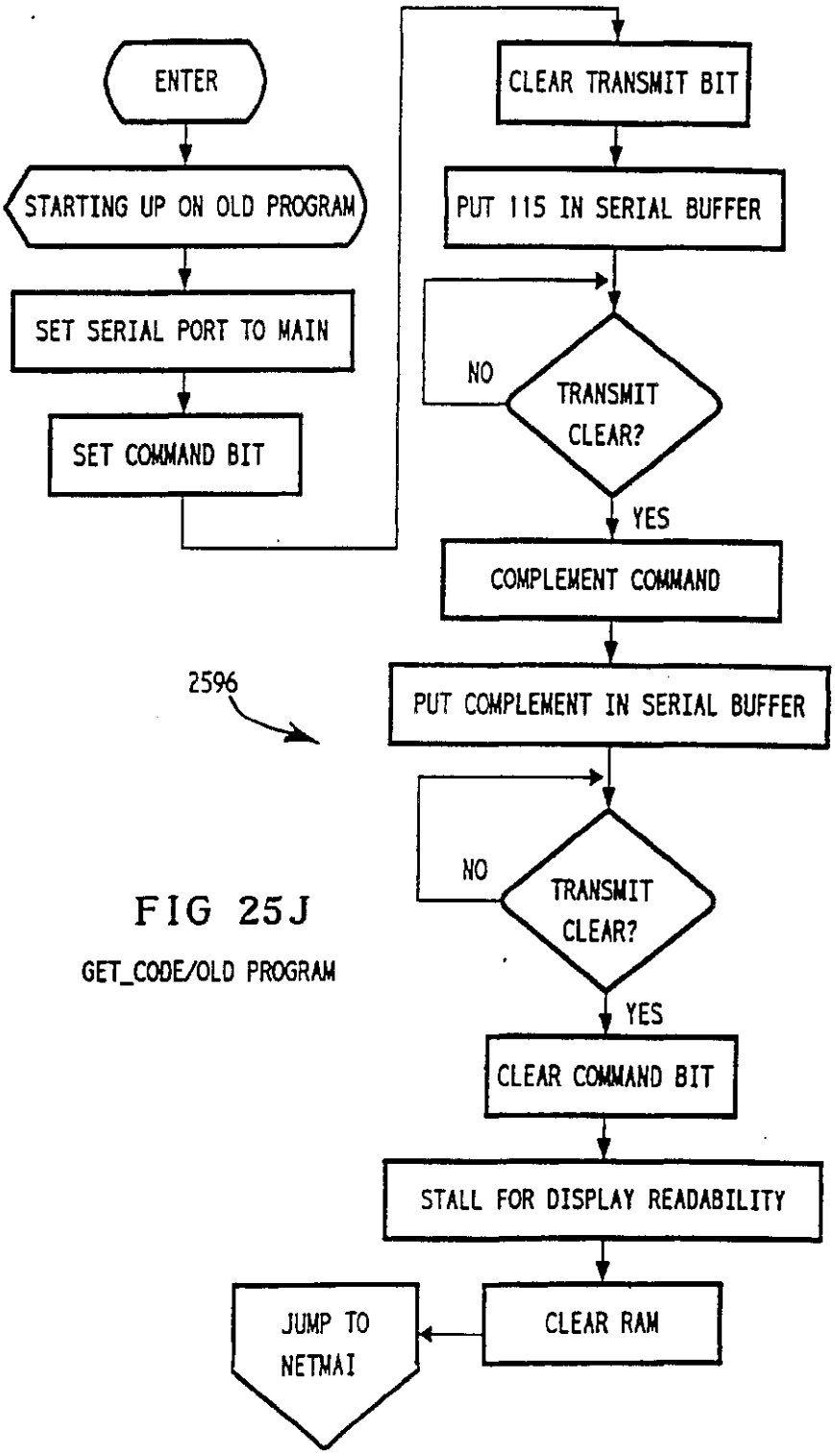


FIG 25J

GET_CODE/OLD PROGRAM

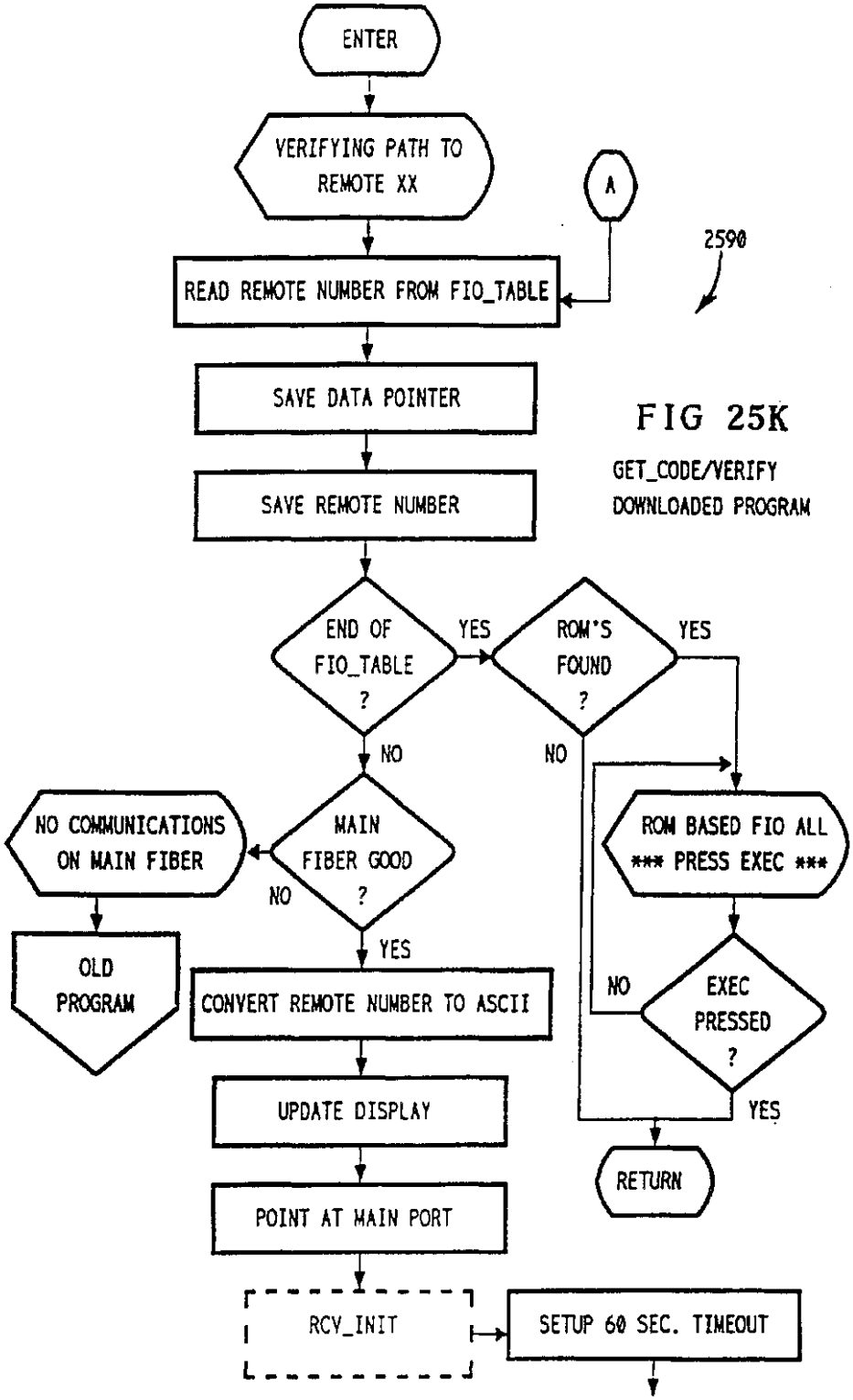


FIG 25K

GET_CODE/VERIFY DOWNLOADED PROGRAM

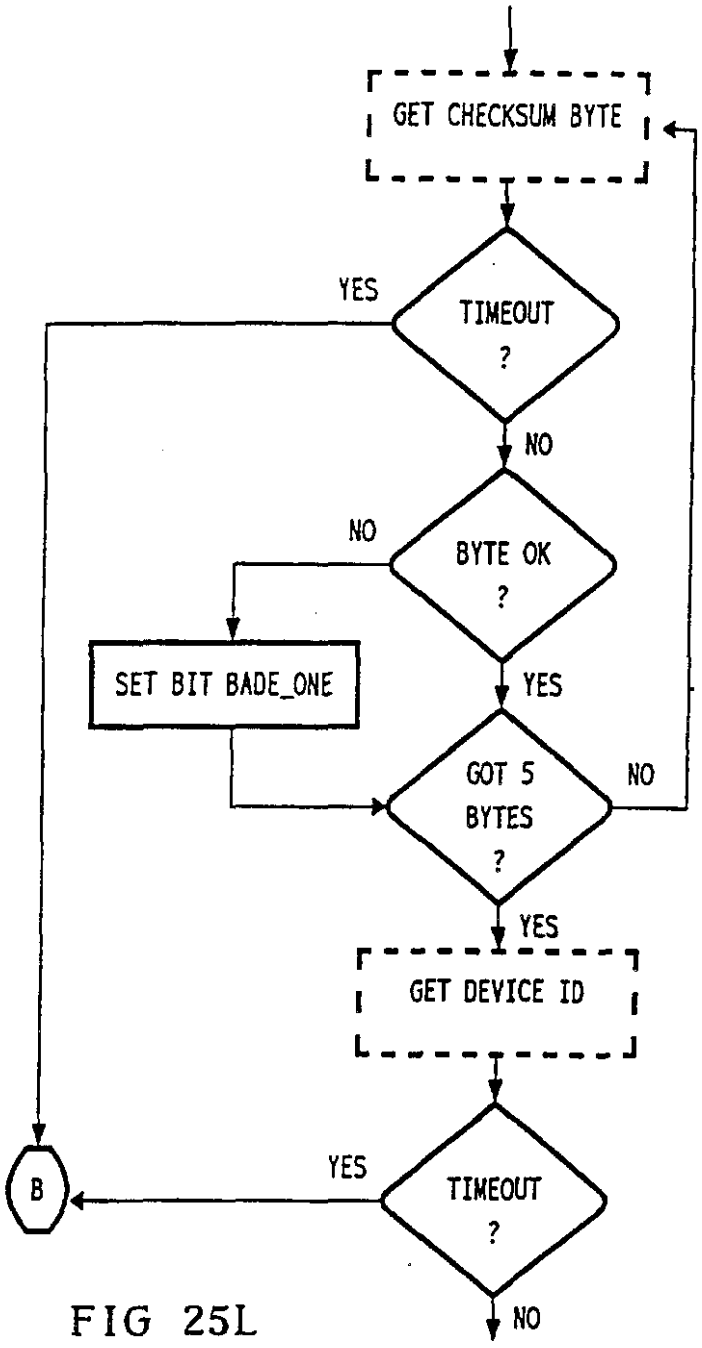

GET CHECKSUM BYTE
TIMEOUT ?
YES
NO
BYTE OK ?
NO
SET BIT BADE_ONE
YES
GOT 5 BYTES ?
NO
YES
GET DEVICE ID
TIMEOUT ?
YES
B
NO

FIG 25L

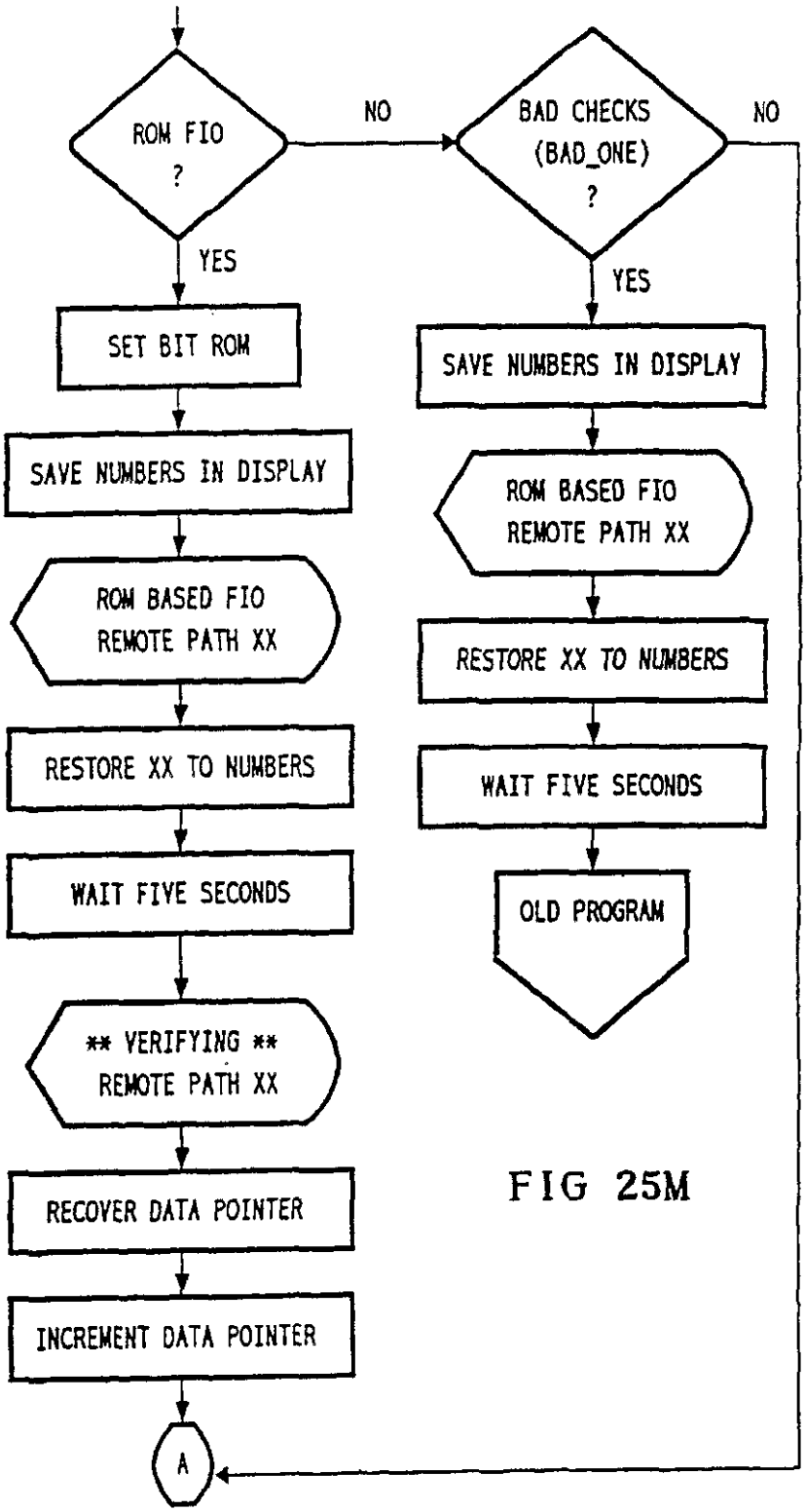

ROM FIO ?
NO
BAD CHECKS (BAD_ONE) ?
NO
YES
YES
SET BIT ROM
SAVE NUMBERS IN DISPLAY
SAVE NUMBERS IN DISPLAY
ROM BASED FIO REMOTE PATH XX
ROM BASED FIO REMOTE PATH XX
RESTORE XX TO NUMBERS
RESTORE XX TO NUMBERS
WAIT FIVE SECONDS
WAIT FIVE SECONDS
OLD PROGRAM
** VERIFYING ** REMOTE PATH XX
RECOVER DATA POINTER
INCREMENT DATA POINTER
A

FIG 25M

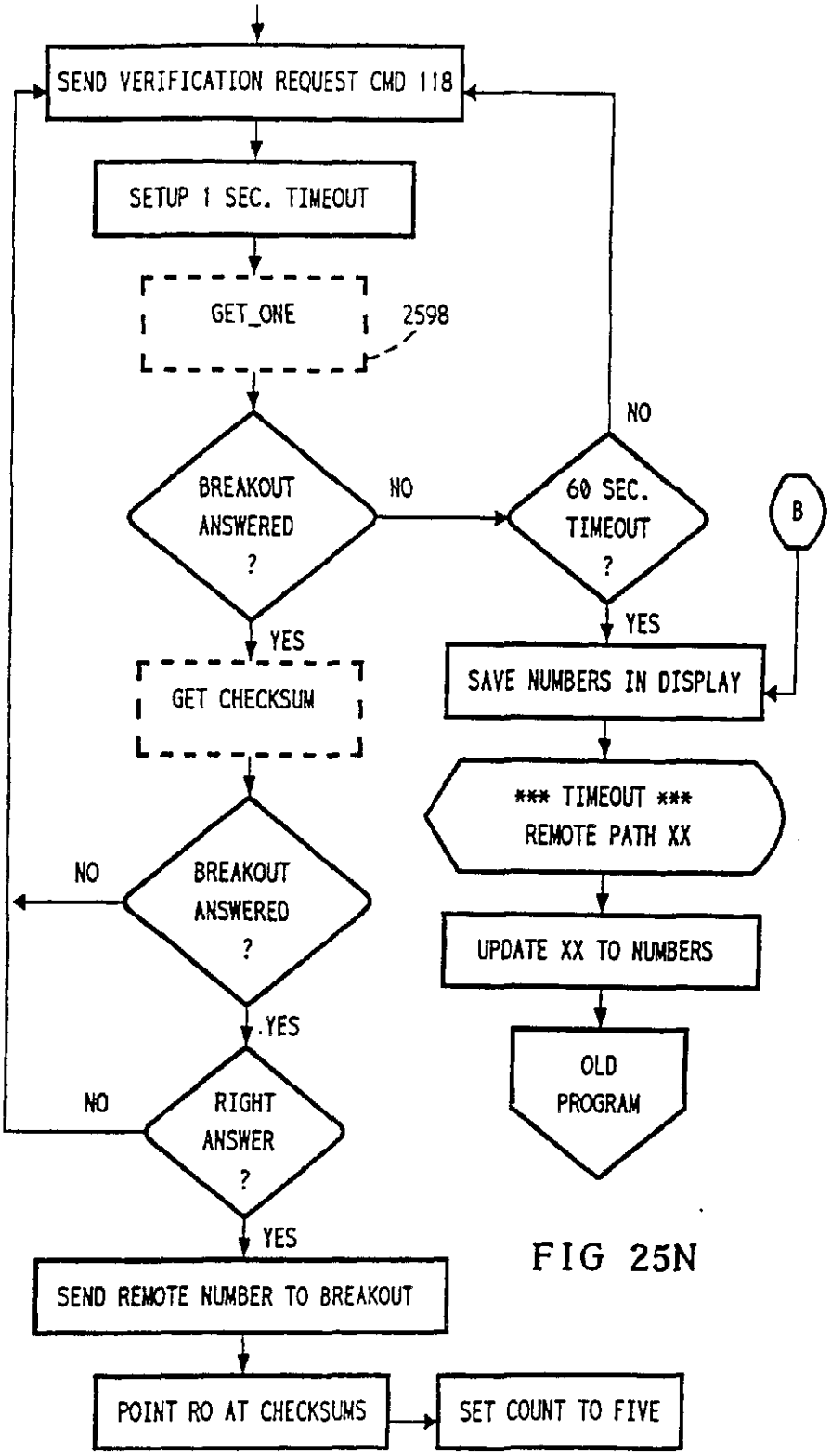

SEND VERIFICATION REQUEST CMD 118
SETUP 1 SEC. TIMEOUT
GET_ONE
2598
BREAKOUT ANSWERED ?
NO
60 SEC. TIMEOUT ?
NO
YES
B
YES
GET CHECKSUM
SAVE NUMBERS IN DISPLAY
*** TIMEOUT *** REMOTE PATH XX
NO
BREAKOUT ANSWERED ?
UPDATE XX TO NUMBERS
YES
OLD PROGRAM
NO
RIGHT ANSWER ?
YES
SEND REMOTE NUMBER TO BREAKOUT
POINT RO AT CHECKSUMS
SET COUNT TO FIVE

FIG 25N

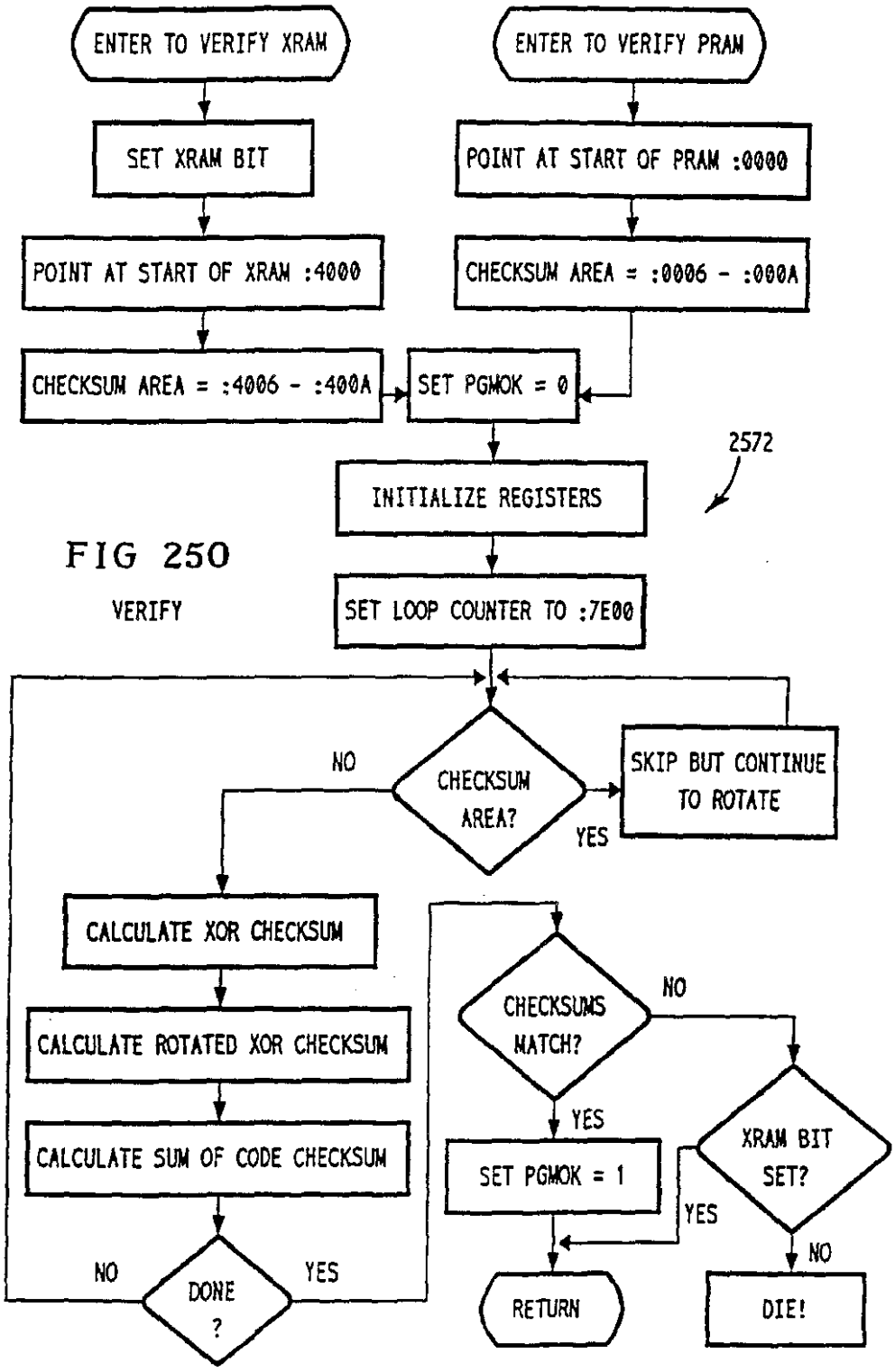
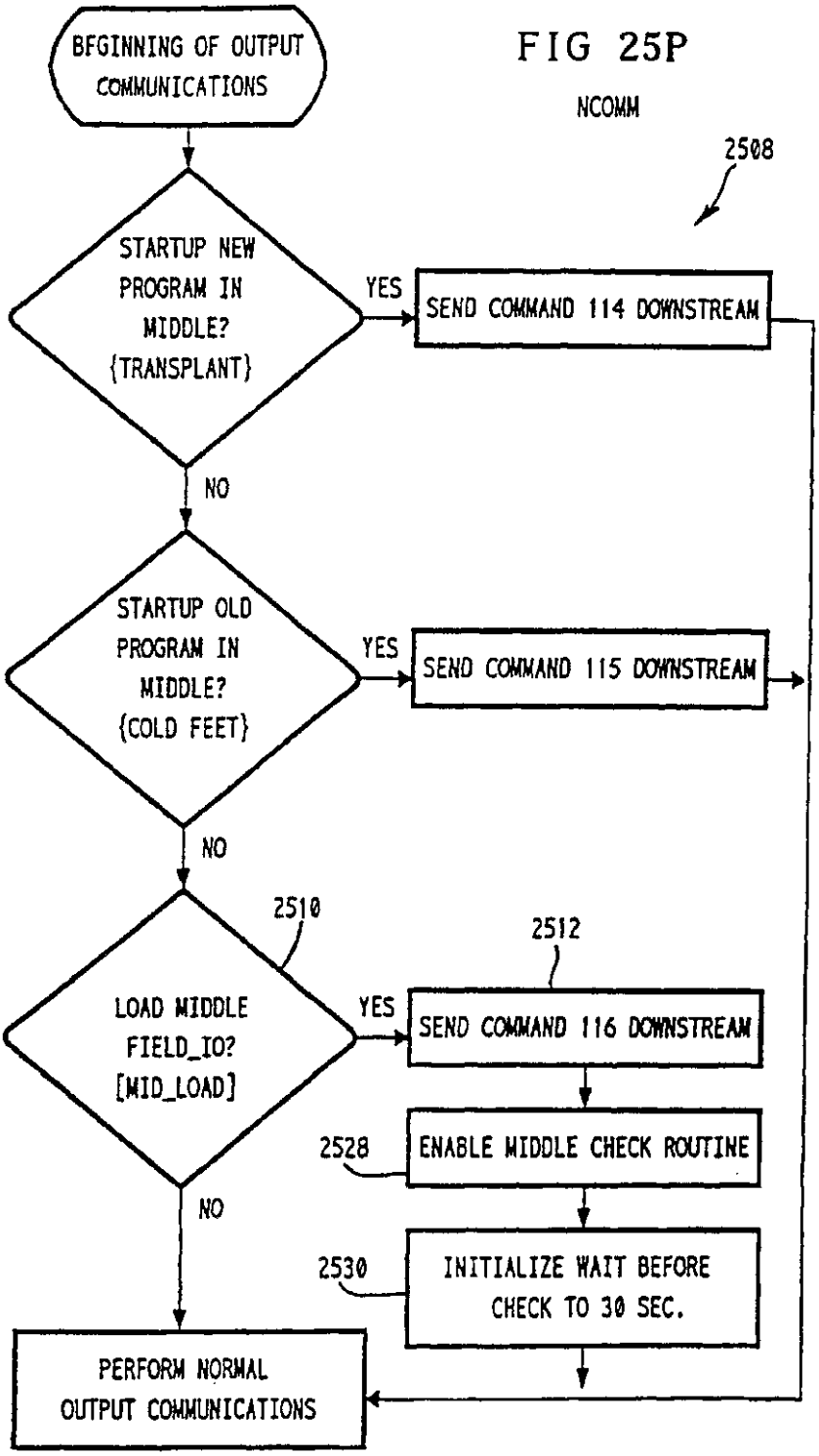


FIG 250

VERIFY

FIG 25P


BEGINNING OF OUTPUT COMMUNICATIONS
NCOMM
2508
STARTUP NEW PROGRAM IN MIDDLE? {TRANSPLANT}
YES
SEND COMMAND 114 DOWNSTREAM
NO
STARTUP OLD PROGRAM IN MIDDLE? {COLD FEET}
YES
SEND COMMAND 115 DOWNSTREAM
NO
2510
2512
LOAD MIDDLE FIELD_IO? [MID_LOAD]
YES
SEND COMMAND 116 DOWNSTREAM
2528
ENABLE MIDDLE CHECK ROUTINE
NO
2530
INITIALIZE WAIT BEFORE CHECK TO 30 SEC.
PERFORM NORMAL OUTPUT COMMUNICATIONS

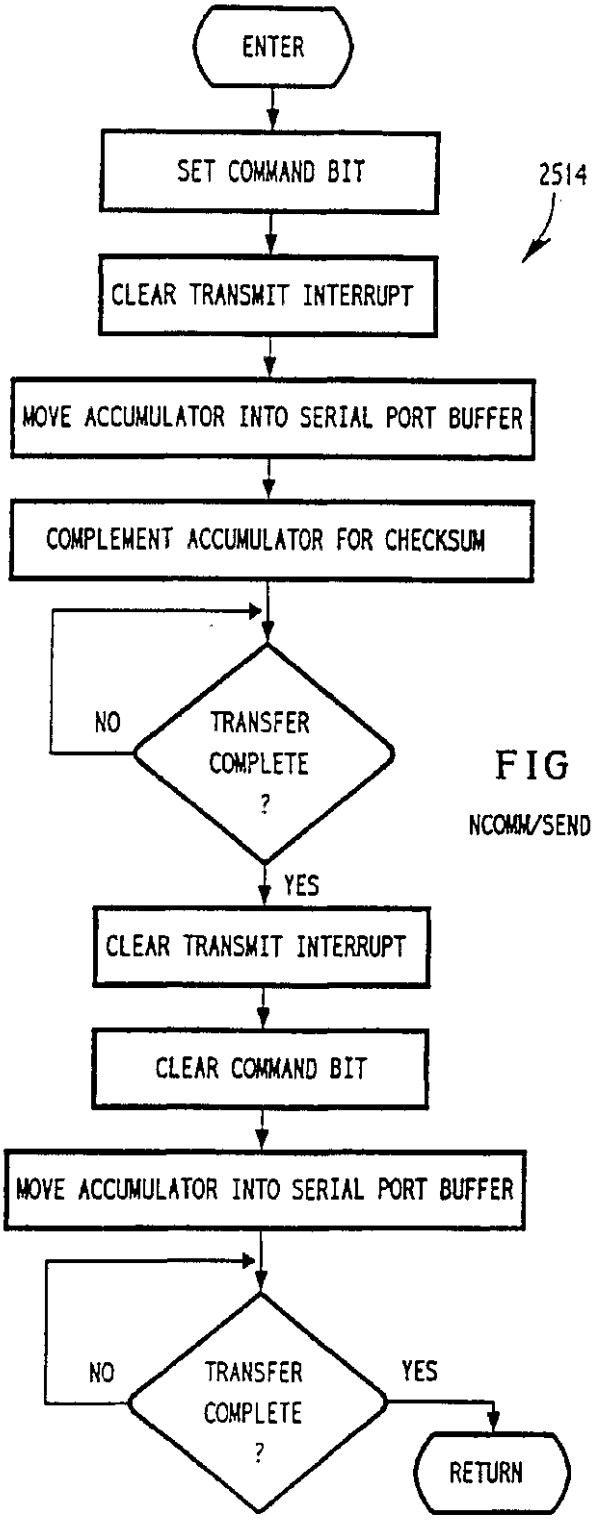
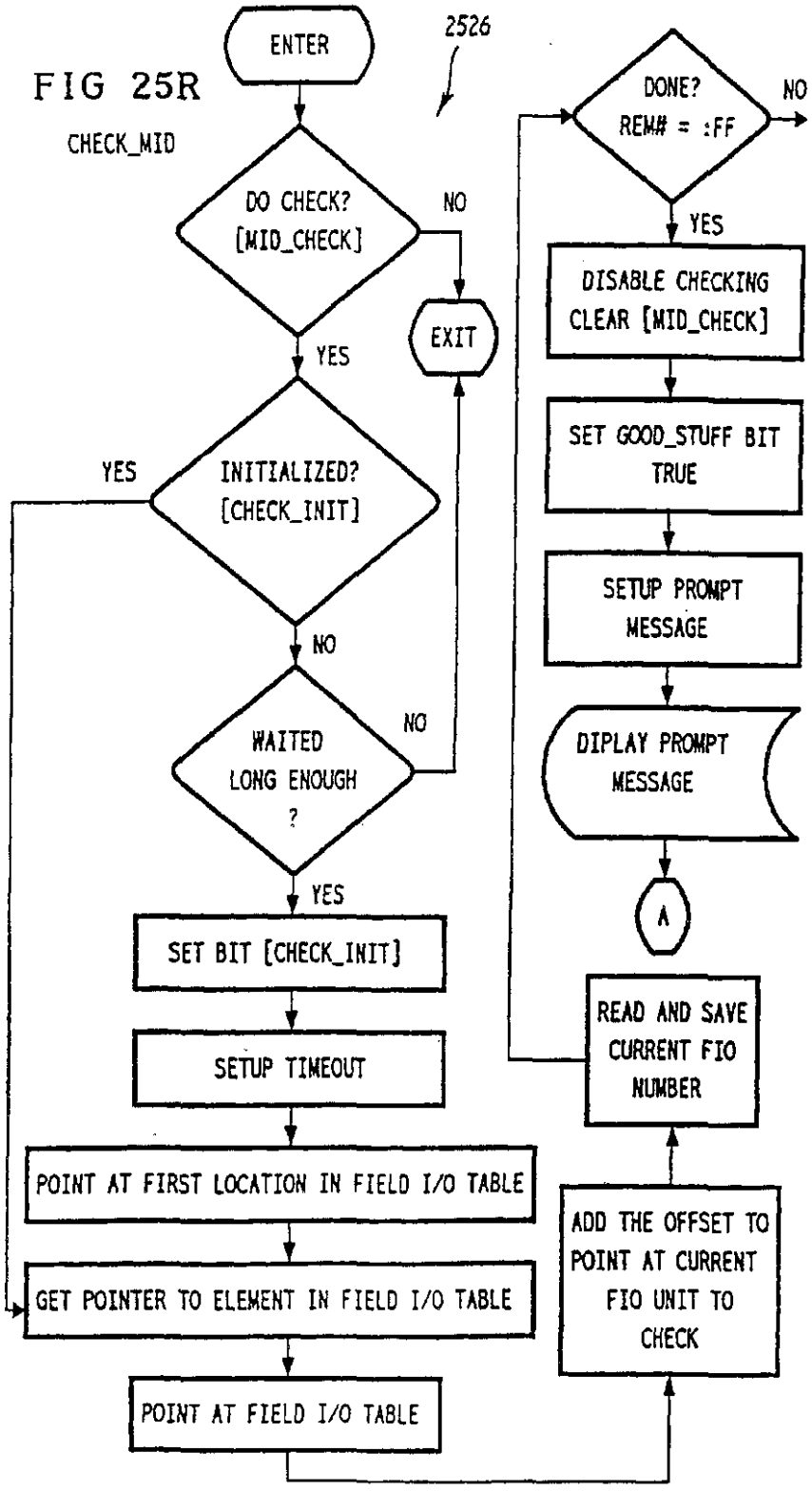


FIG 25Q

NCOMM/SEND COMMAND

FIG 25R

CHECK_MID



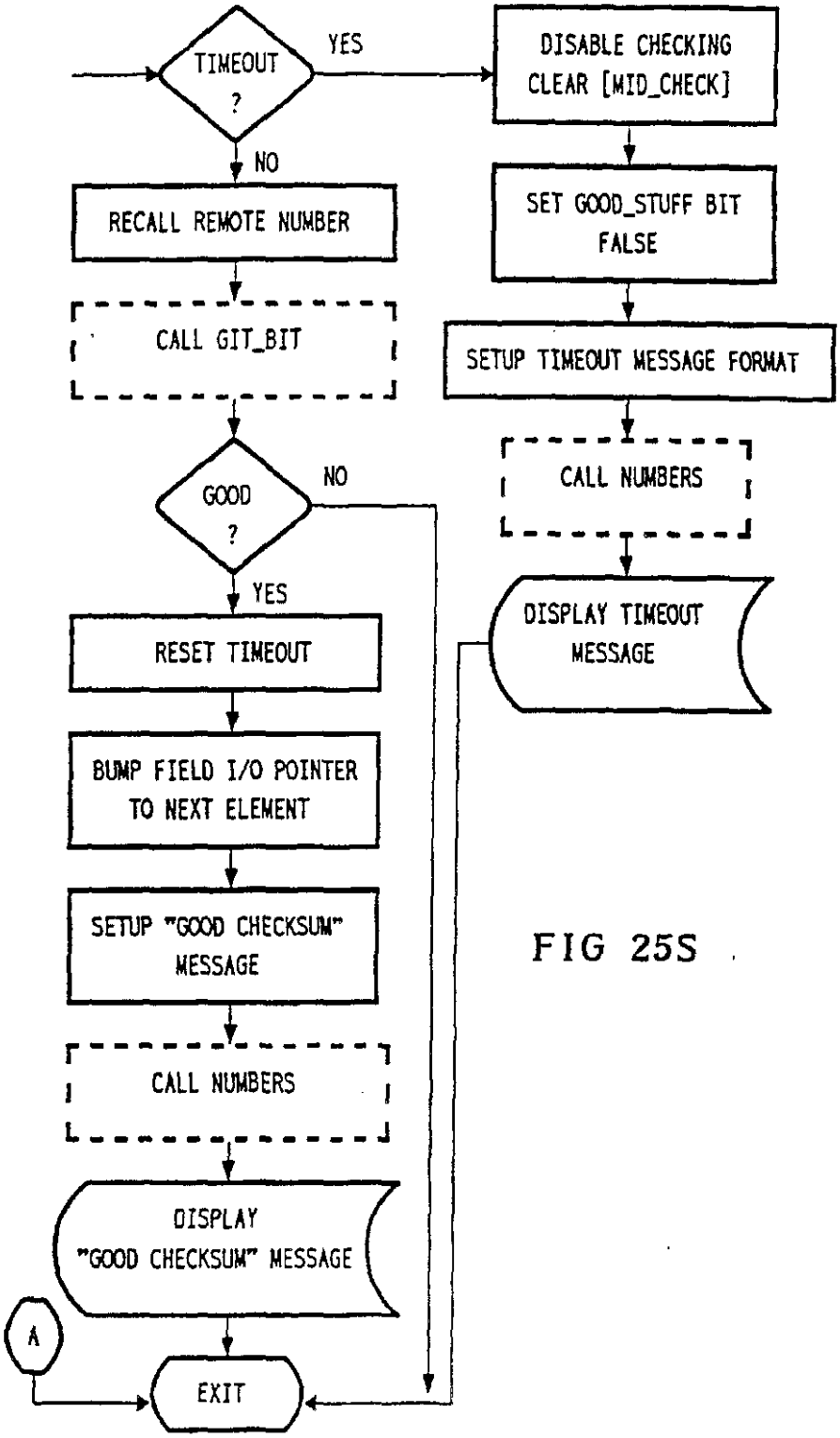

TIMEOUT ?
YES
NO
DISABLE CHECKING
CLEAR [MID_CHECK]
SET GOOD_STUFF BIT
FALSE
SETUP TIMEOUT MESSAGE FORMAT
CALL NUMBERS
DISPLAY TIMEOUT
MESSAGE
RECALL REMOTE NUMBER
CALL GIT_BIT
GOOD ?
NO
YES
RESET TIMEOUT
BUMP FIELD I/O POINTER
TO NEXT ELEMENT
SETUP "GOOD CHECKSUM"
MESSAGE
CALL NUMBERS
DISPLAY
"GOOD CHECKSUM" MESSAGE
A
EXIT

FIG 25S

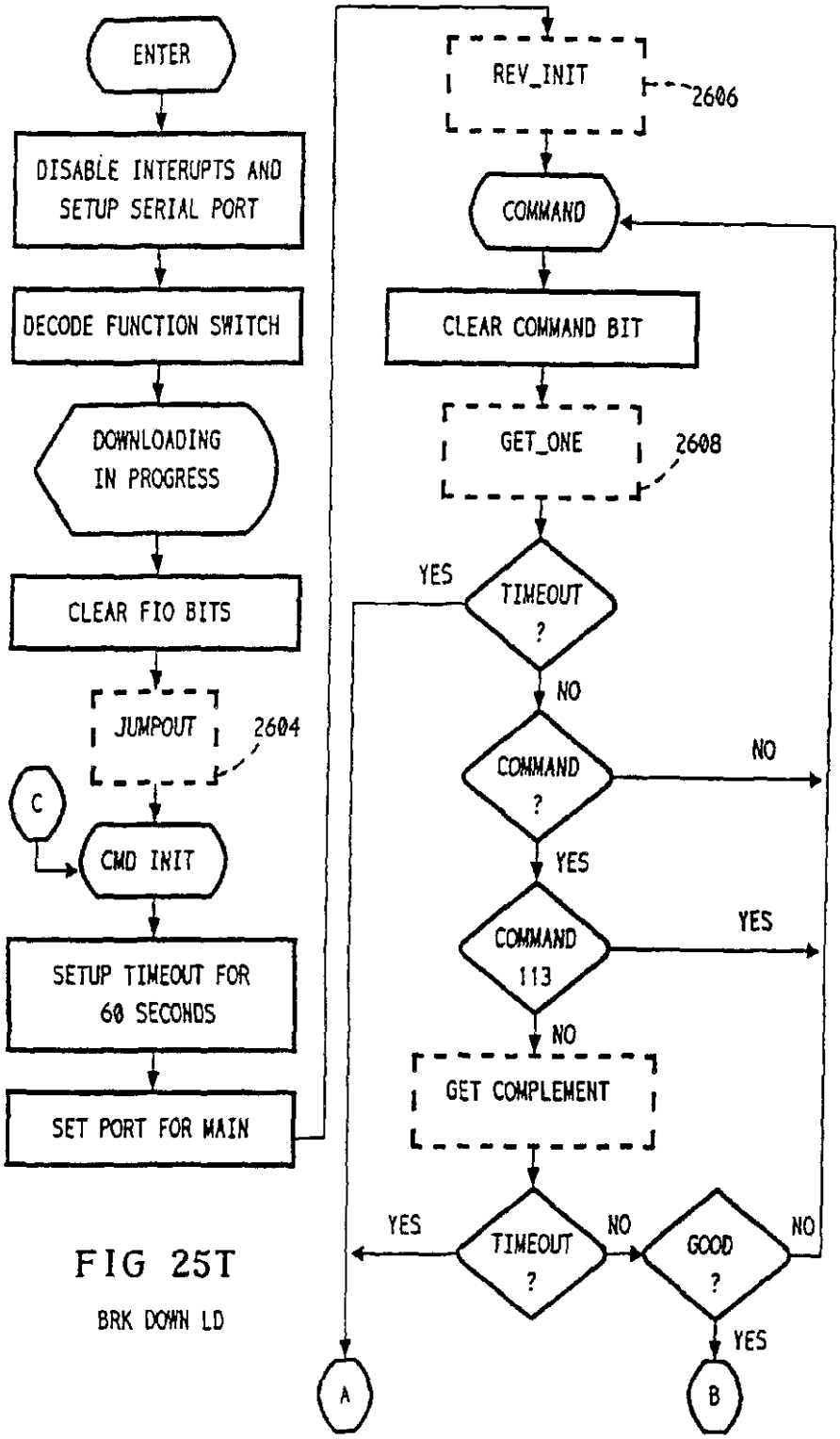


FIG 25T

BRK DOWN LD

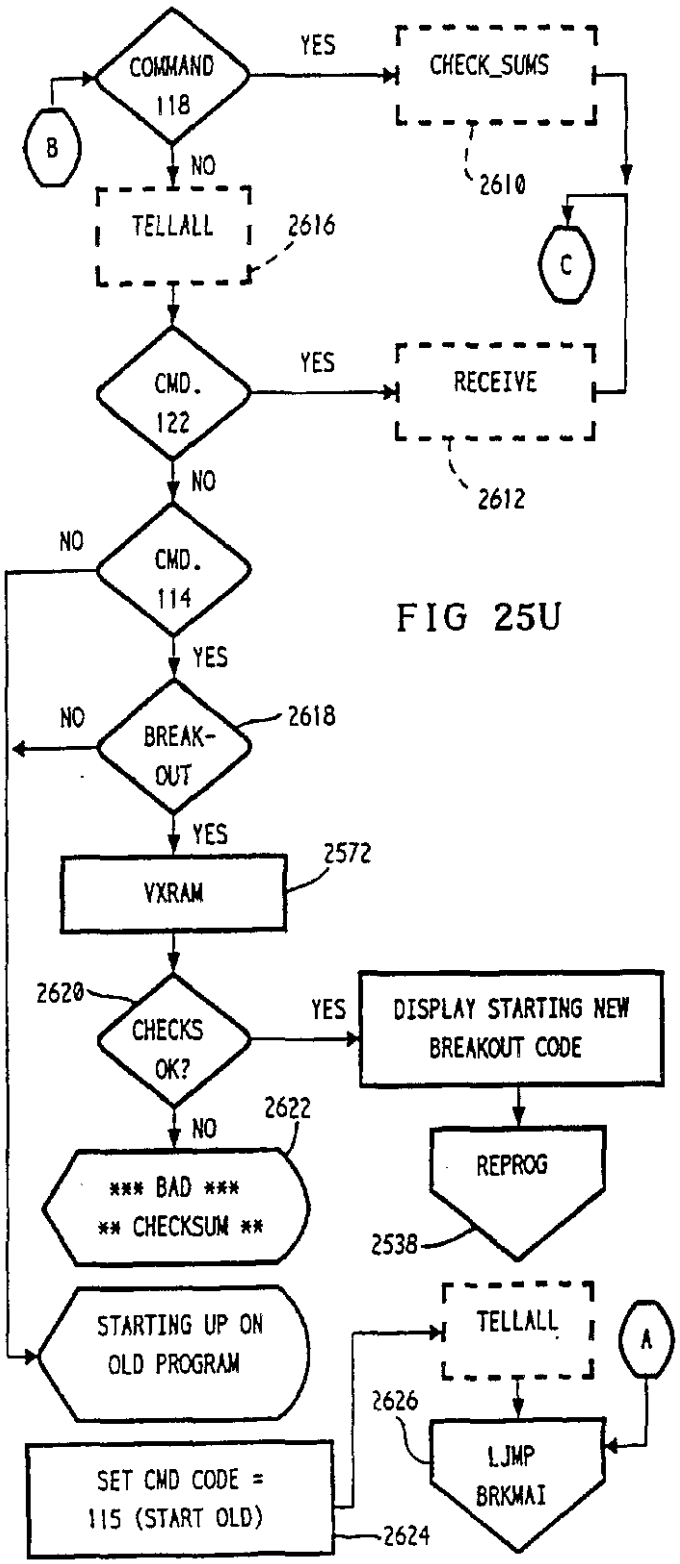

COMMAND 118
YES
CHECK_SUMS
2610
B
NO
TELLALL
2616
C
CMD. 122
YES
RECEIVE
2612
NO
CMD. 114
NO
YES
BREAK-OUT
2618
NO
YES
VXRAM
2572
2620
CHECKS OK?
YES
DISPLAY STARTING NEW BREAKOUT CODE
NO
2622
*** BAD *** ** CHECKSUM **
REPROG
2538
STARTING UP ON OLD PROGRAM
TELLALL
A
2626
LJMP BRKMAI
SET CMD CODE = 115 (START OLD)
2624

FIG 25U

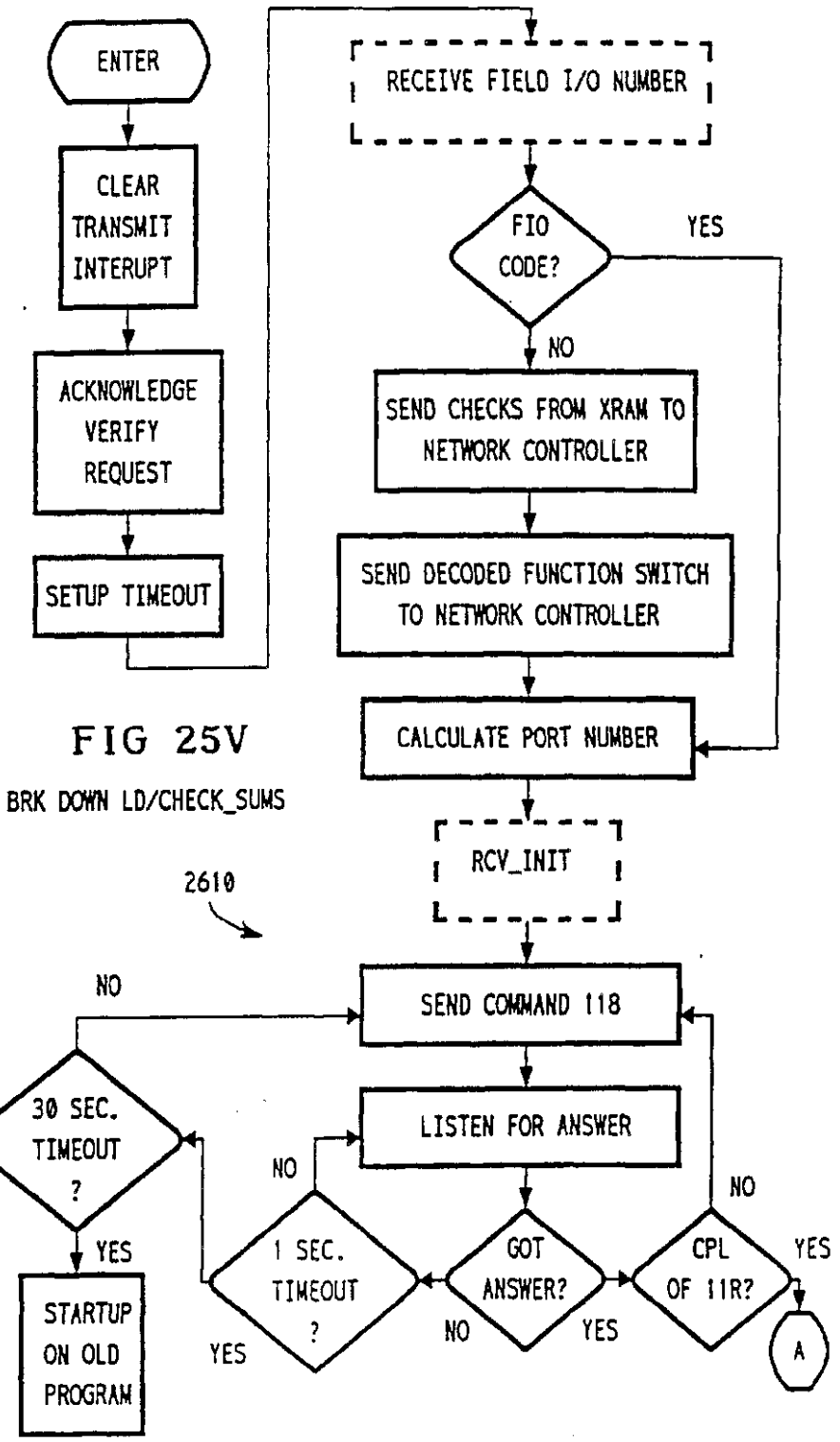


FIG 25V

BRK DOWN LD/CHECK_SUMS

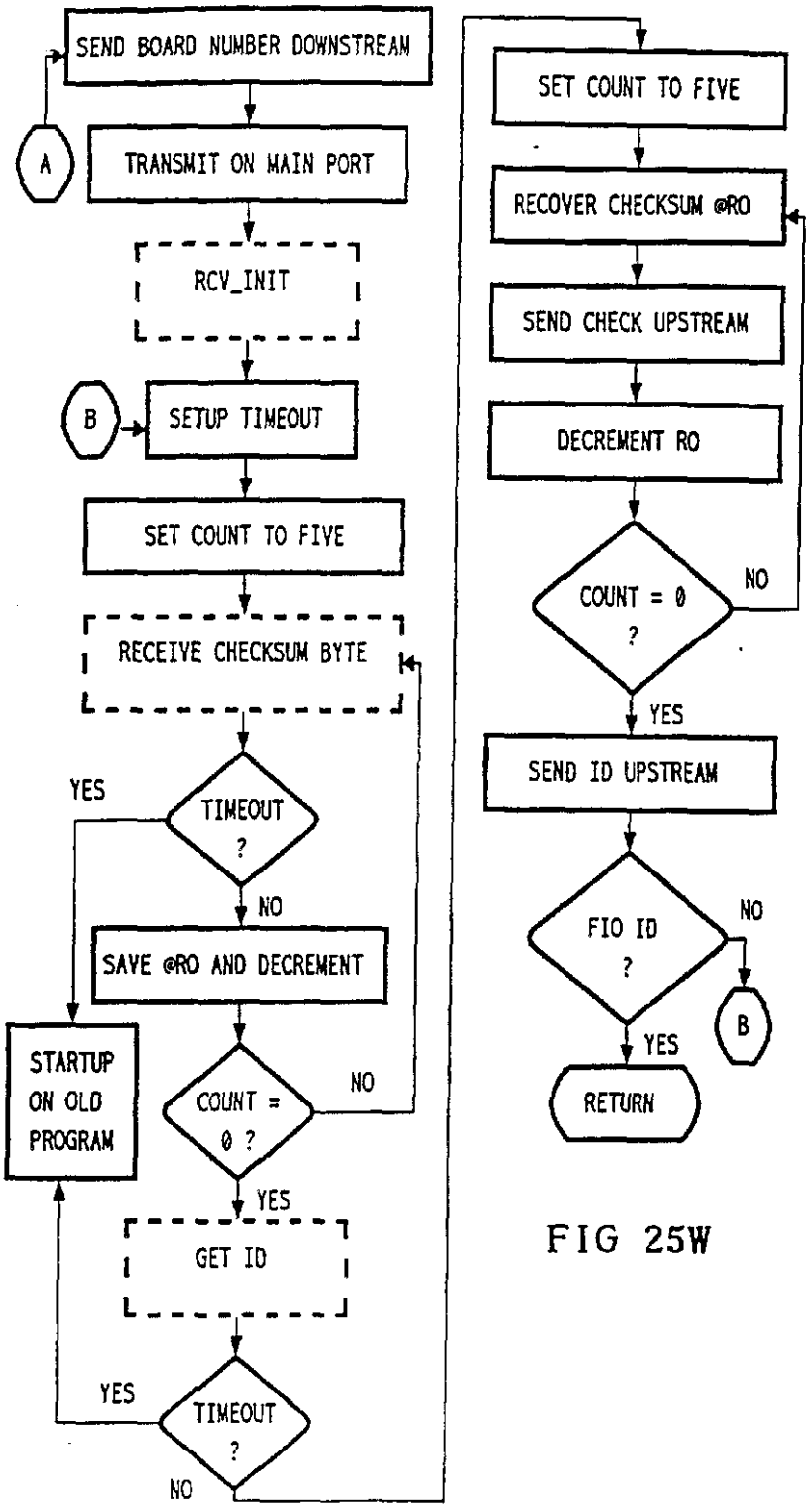

SEND BOARD NUMBER DOWNSTREAM
A
TRANSMIT ON MAIN PORT
RCV_INIT
B
SETUP TIMEOUT
SET COUNT TO FIVE
RECEIVE CHECKSUM BYTE
YES
TIMEOUT ?
NO
SAVE @RO AND DECREMENT
STARTUP ON OLD PROGRAM
COUNT = 0 ?
NO
YES
GET ID
YES
TIMEOUT ?
NO
SET COUNT TO FIVE
RECOVER CHECKSUM @RO
SEND CHECK UPSTREAM
DECREMENT RO
COUNT = 0 ?
NO
YES
SEND ID UPSTREAM
FIO ID ?
NO
B
YES
RETURN

FIG 25W

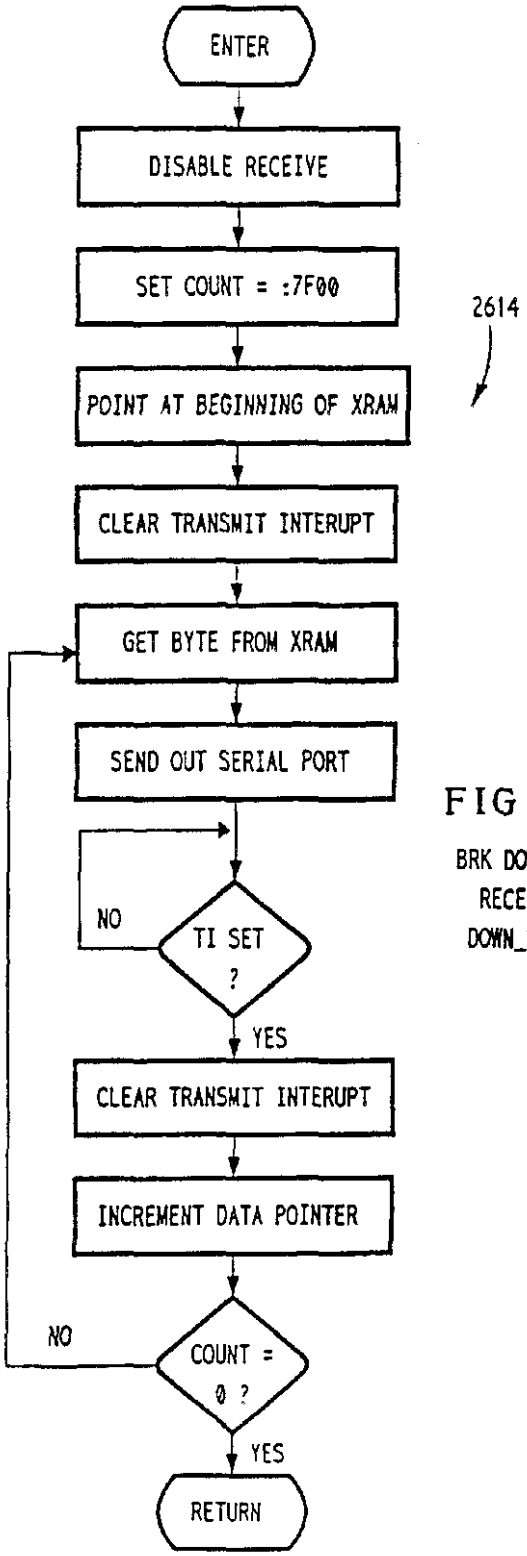
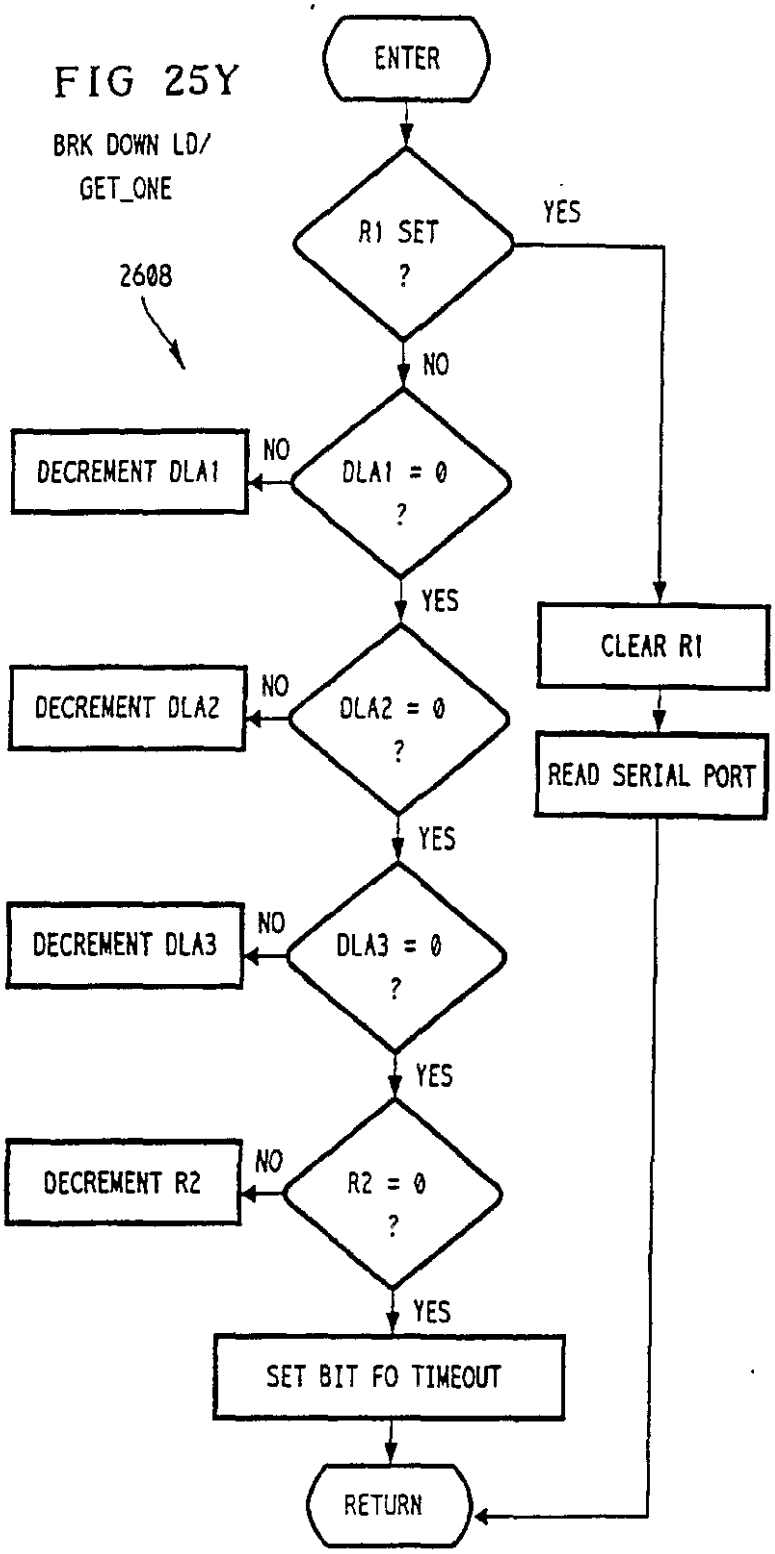


FIG 25X

BRK DOWN LD/
RECEIVE/
DOWN_LOAD

FIG 25Y

BRK DOWN LD/
GET_ONE



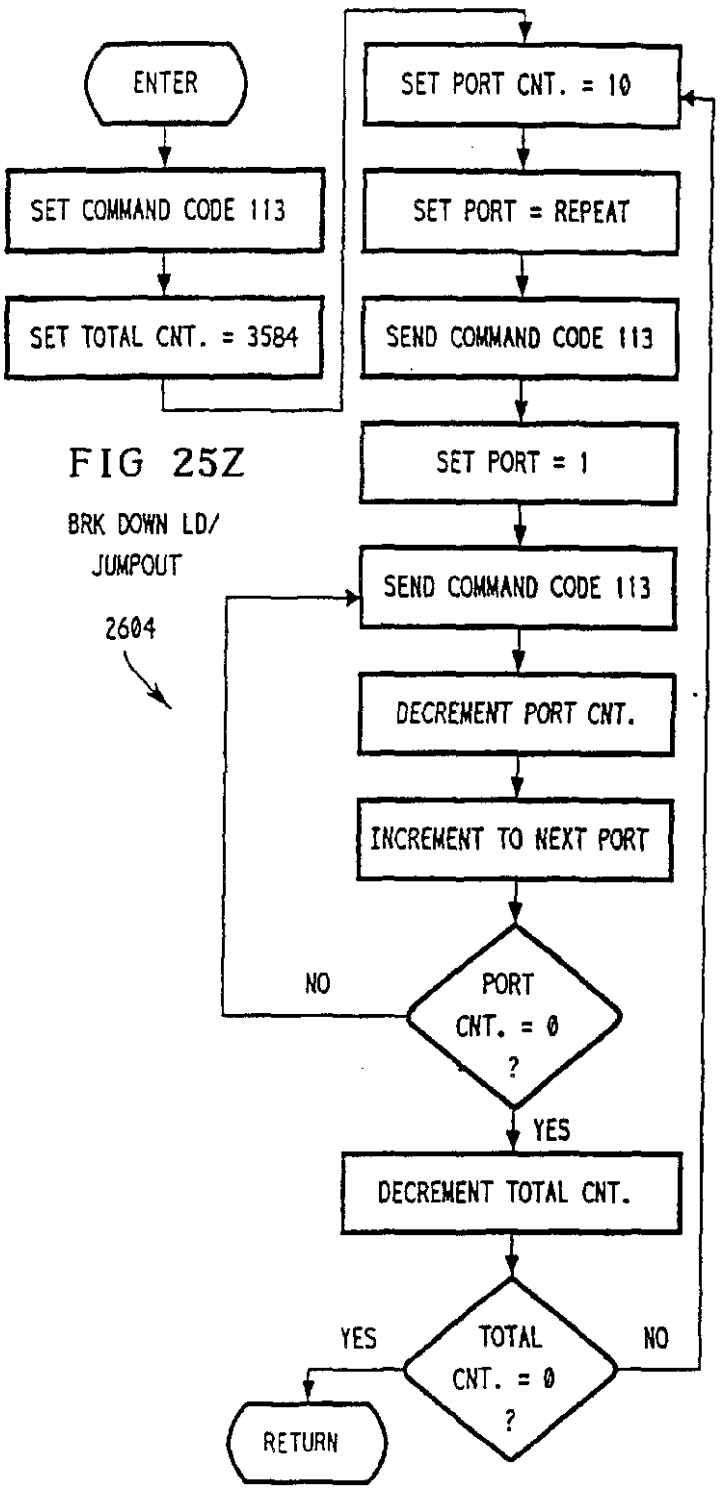

ENTER
SET COMMAND CODE 113
SET TOTAL CNT. = 3584
SET PORT CNT. = 10
SET PORT = REPEAT
SEND COMMAND CODE 113
SET PORT = 1
SEND COMMAND CODE 113
DECREMENT PORT CNT.
INCREMENT TO NEXT PORT
PORT CNT. = 0 ?
NO
YES
DECREMENT TOTAL CNT.
TOTAL CNT. = 0 ?
YES
NO
RETURN
BRK DOWN LD/ JUMPOUT
2604

FIG 25Z

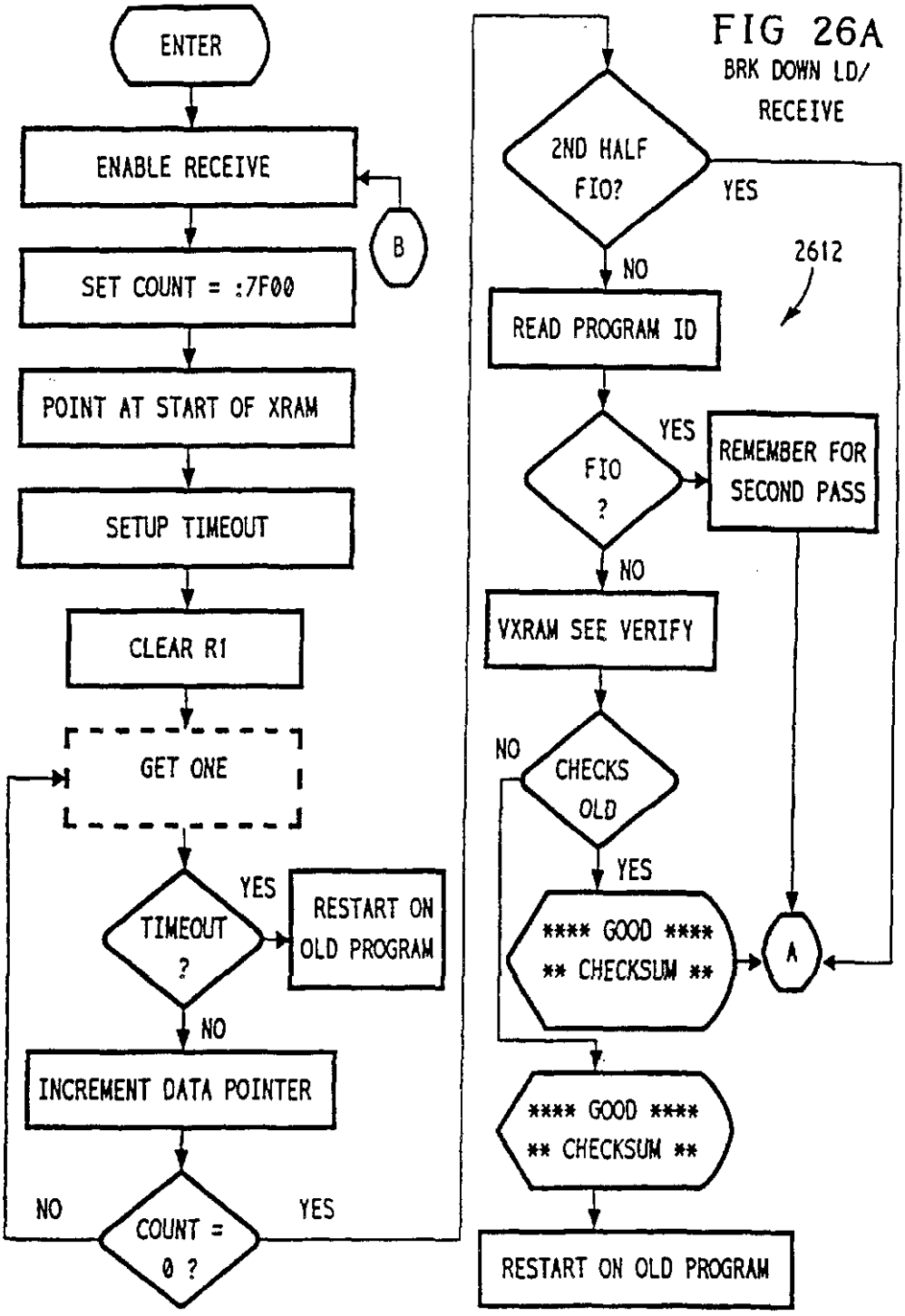

ENTER
ENABLE RECEIVE
B
SET COUNT = :7F00
POINT AT START OF XRAM
SETUP TIMEOUT
CLEAR R1
GET ONE
TIMEOUT ?
YES
RESTART ON OLD PROGRAM
NO
INCREMENT DATA POINTER
COUNT = 0 ?
NO
YES
2ND HALF FIO?
YES
NO
2612
READ PROGRAM ID
FIO ?
YES
REMEMBER FOR SECOND PASS
NO
VXRAM SEE VERIFY
CHECKS OLD
NO
YES
**** GOOD ****
** CHECKSUM **
A
**** GOOD ****
** CHECKSUM **
RESTART ON OLD PROGRAM
BRK DOWN LD/
RECEIVE

FIG 26A

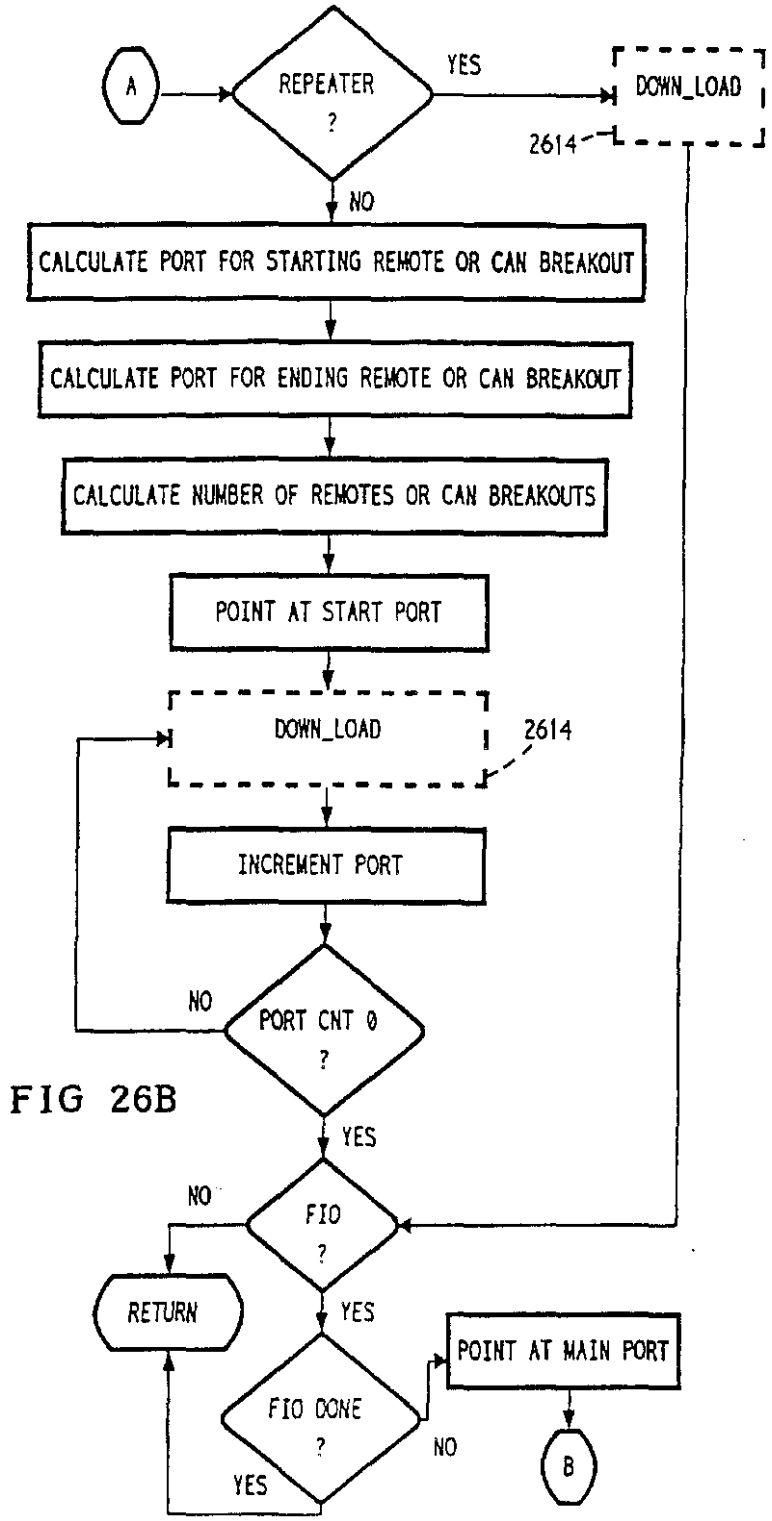

A
REPEATER ?
YES
DOWN_LOAD
2614
NO
CALCULATE PORT FOR STARTING REMOTE OR CAN BREAKOUT
CALCULATE PORT FOR ENDING REMOTE OR CAN BREAKOUT
CALCULATE NUMBER OF REMOTES OR CAN BREAKOUTS
POINT AT START PORT
DOWN_LOAD
2614
INCREMENT PORT
NO
PORT CNT 0 ?
YES
NO
FIO ?
RETURN
YES
FIO DONE ?
NO
YES
POINT AT MAIN PORT
B

FIG 26B

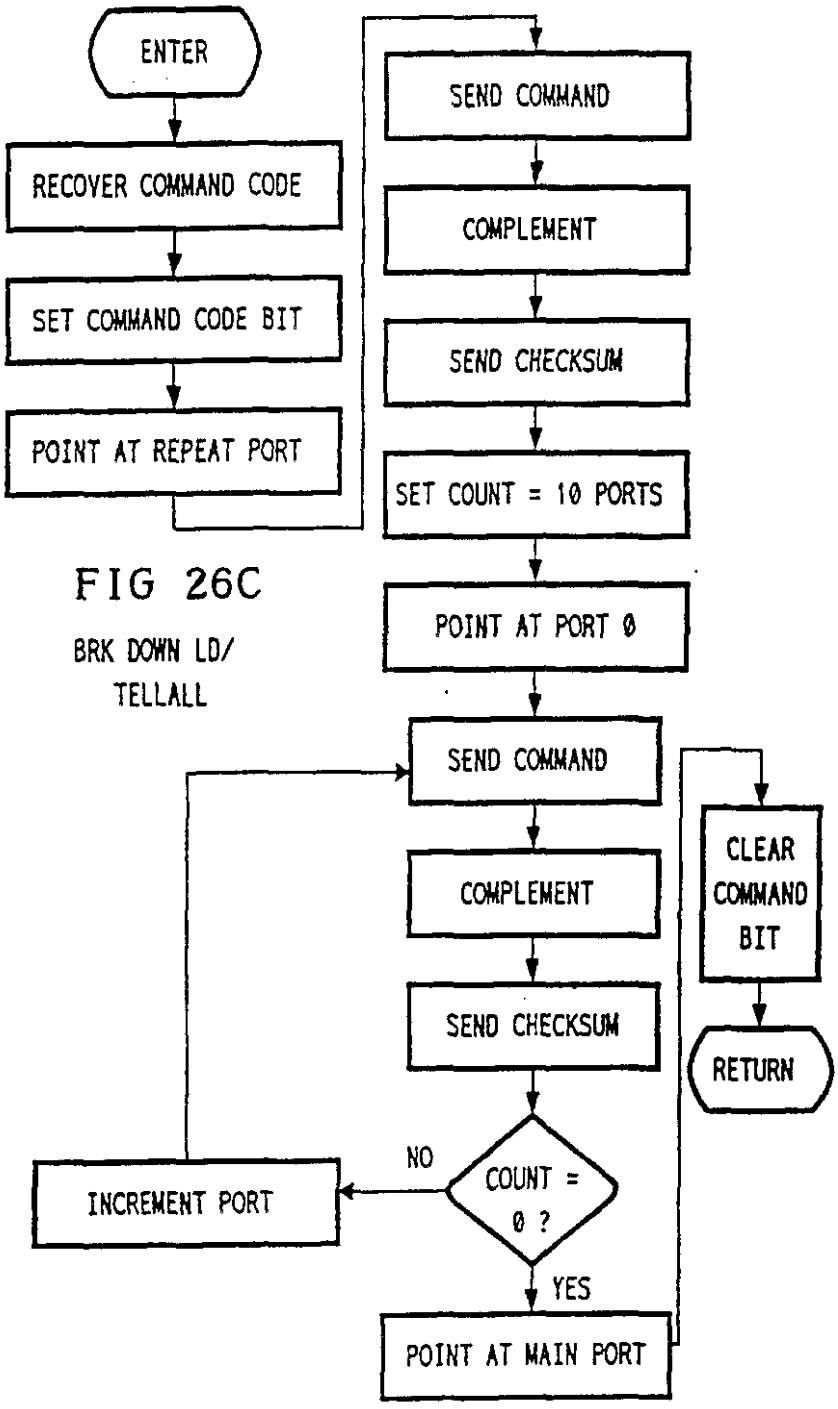


FIG 26C

BRK DOWN LD/
TELLALL

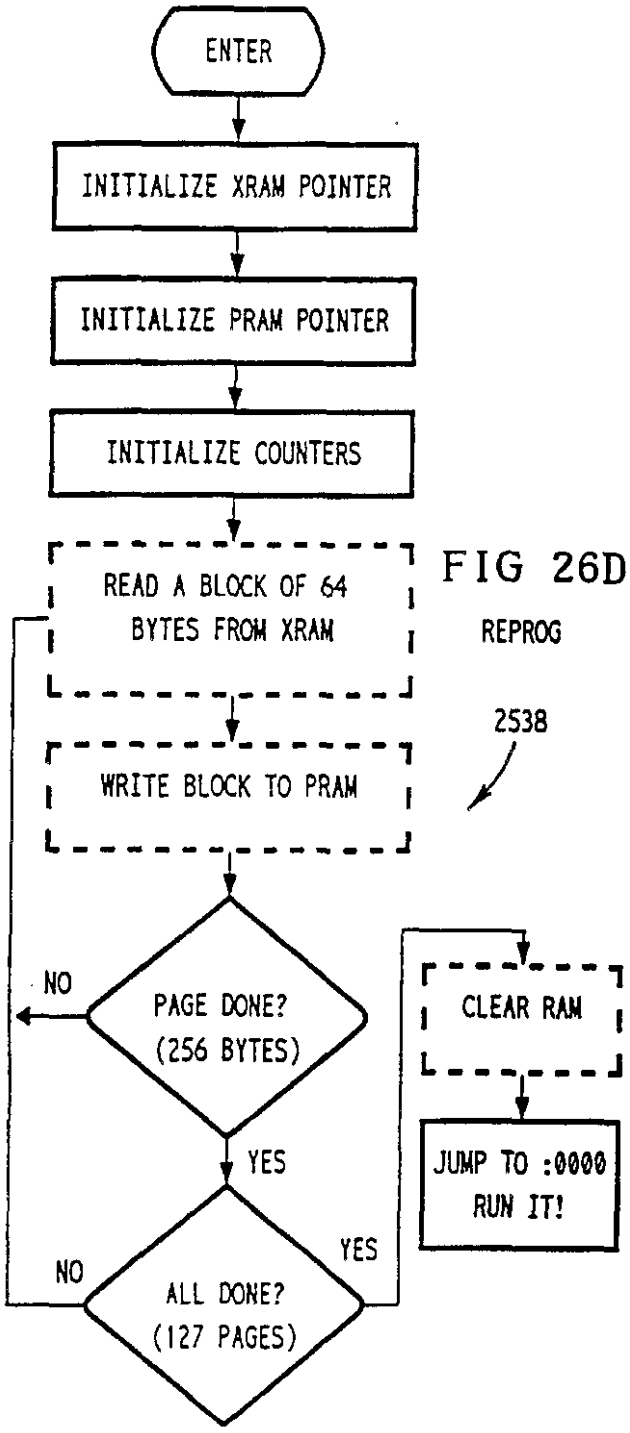


FIG 26D

REPROG

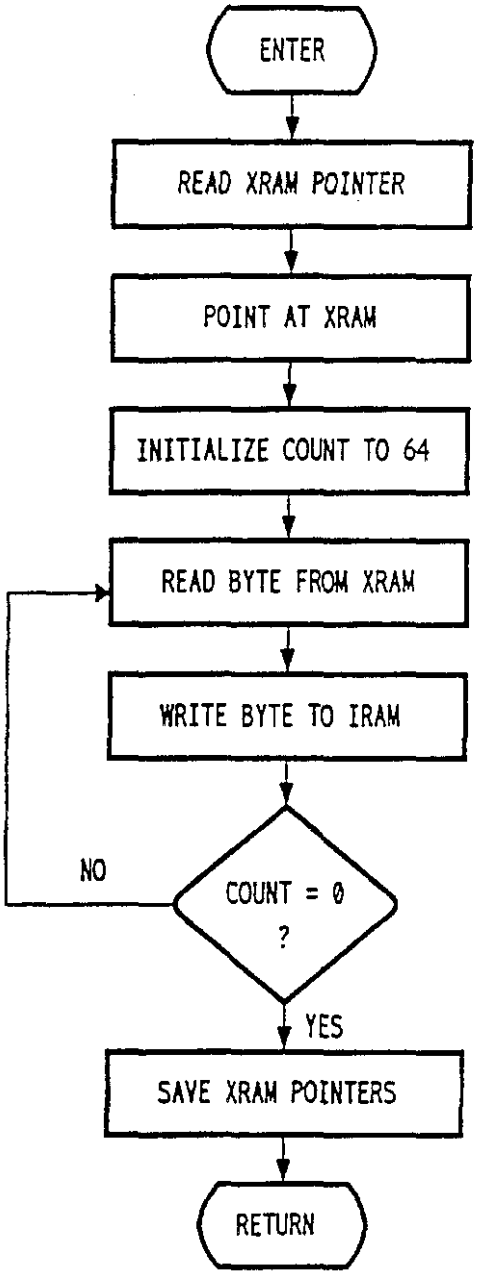


FIG 26E

REPROG/READ A BLOCK

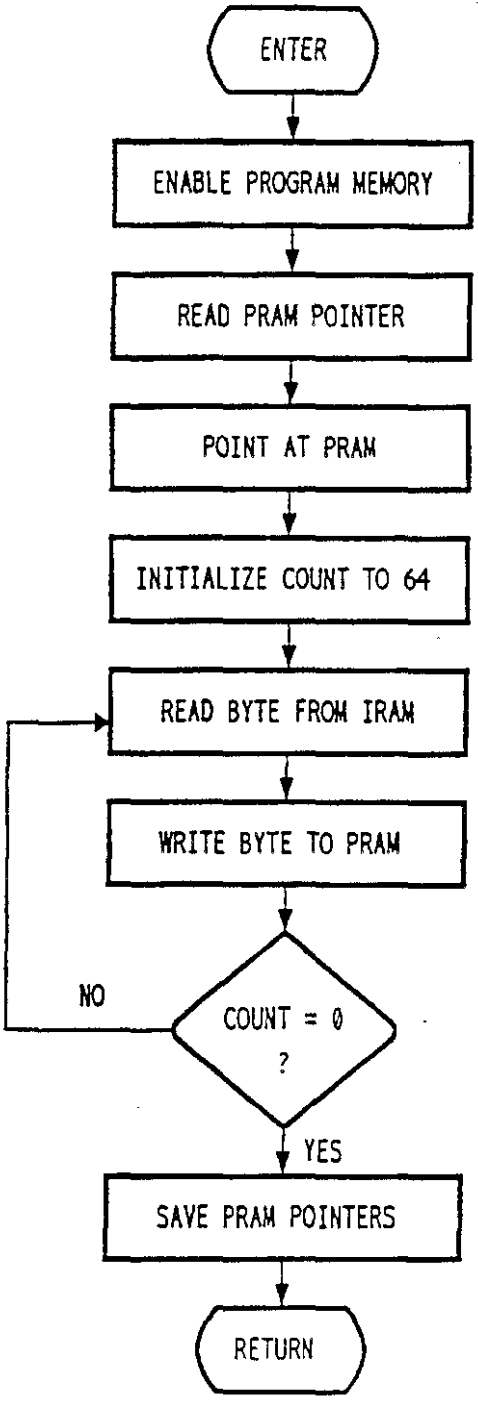


FIG 26F

REPROG/WRITE A BLOCK

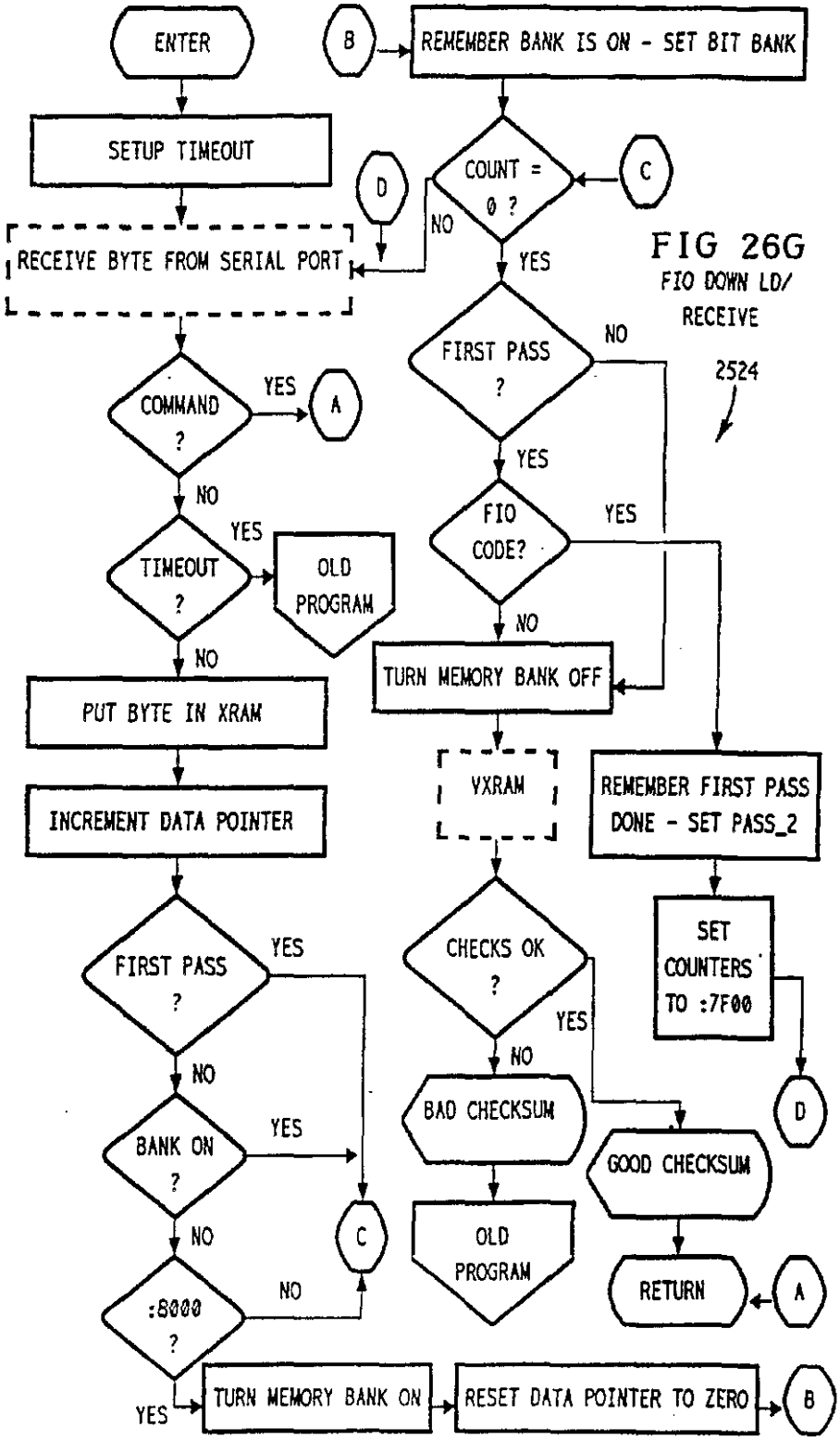
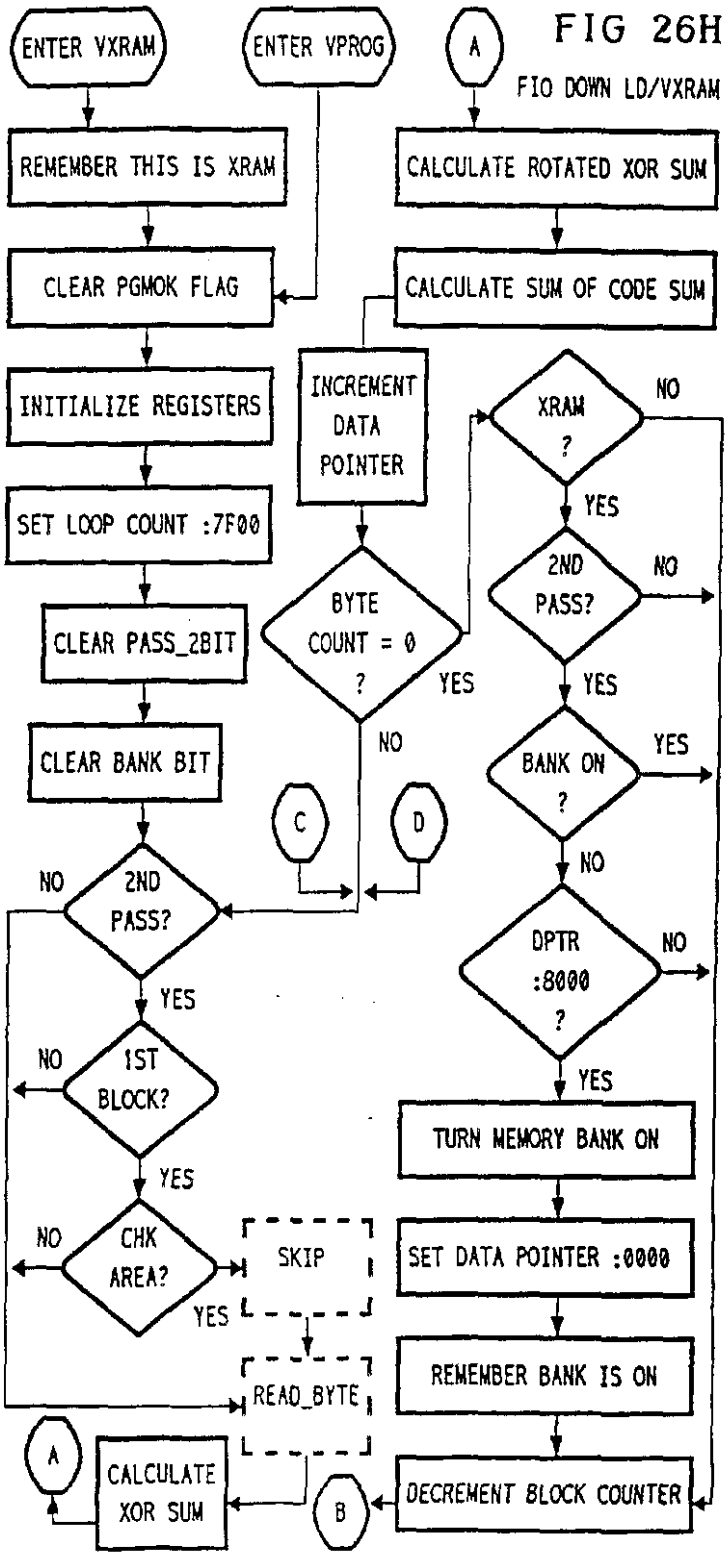

ENTER
B
REMEMBER BANK IS ON - SET BIT BANK
SETUP TIMEOUT
COUNT = 0 ?
C
D
NO
YES
RECEIVE BYTE FROM SERIAL PORT
FIG 26G
FIO DOWN LD/ RECEIVE
2524
FIRST PASS ?
NO
COMMAND ?
YES
A
YES
NO
FIO CODE?
YES
TIMEOUT ?
YES
OLD PROGRAM
NO
NO
TURN MEMORY BANK OFF
PUT BYTE IN XRAM
VXRAM
REMEMBER FIRST PASS DONE - SET PASS_2
INCREMENT DATA POINTER
FIRST PASS ?
YES
CHECKS OK ?
SET COUNTERS TO :7F00
NO
YES
NO
BAD CHECKSUM
D
BANK ON ?
YES
GOOD CHECKSUM
NO
C
OLD PROGRAM
:8000 ?
NO
RETURN
A
YES
TURN MEMORY BANK ON
RESET DATA POINTER TO ZERO
B

FIG 26H


ENTER VXRAM
ENTER VPROG
A
FIO DOWN LD/VXRAM
REMEMBER THIS IS XRAM
CALCULATE ROTATED XOR SUM
CLEAR PGMOK FLAG
CALCULATE SUM OF CODE SUM
INITIALIZE REGISTERS
INCREMENT DATA POINTER
XRAM ?
NO
YES
SET LOOP COUNT :7F00
BYTE COUNT = 0 ?
YES
2ND PASS?
NO
CLEAR PASS_2BIT
NO
YES
CLEAR BANK BIT
C
D
BANK ON ?
YES
NO
NO
2ND PASS?
DPTR :8000 ?
NO
YES
YES
NO
1ST BLOCK?
TURN MEMORY BANK ON
YES
NO
CHK AREA?
SKIP
SET DATA POINTER :0000
YES
READ_BYTE
REMEMBER BANK IS ON
A
CALCULATE XOR SUM
B
DECREMENT BLOCK COUNTER

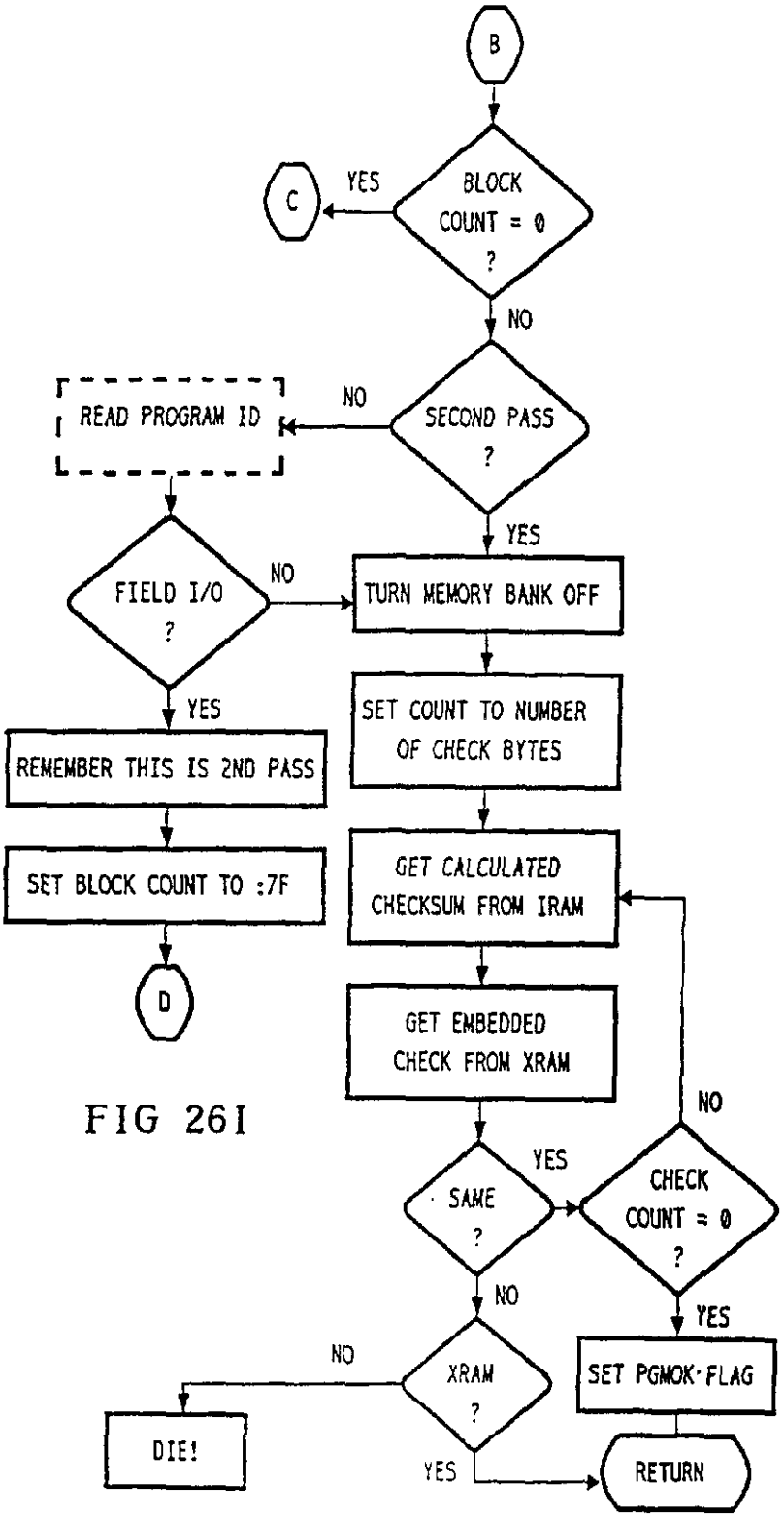

B
BLOCK COUNT = 0 ?
YES
C
NO
SECOND PASS ?
NO
READ PROGRAM ID
YES
FIELD I/O ?
NO
TURN MEMORY BANK OFF
YES
REMEMBER THIS IS 2ND PASS
SET COUNT TO NUMBER OF CHECK BYTES
SET BLOCK COUNT TO :7F
GET CALCULATED CHECKSUM FROM IRAM
D
GET EMBEDDED CHECK FROM XRAM
SAME ?
YES
NO
CHECK COUNT = 0 ?
NO
YES
XRAM ?
NO
YES
SET PGMOK FLAG
DIE!
RETURN

FIG 261

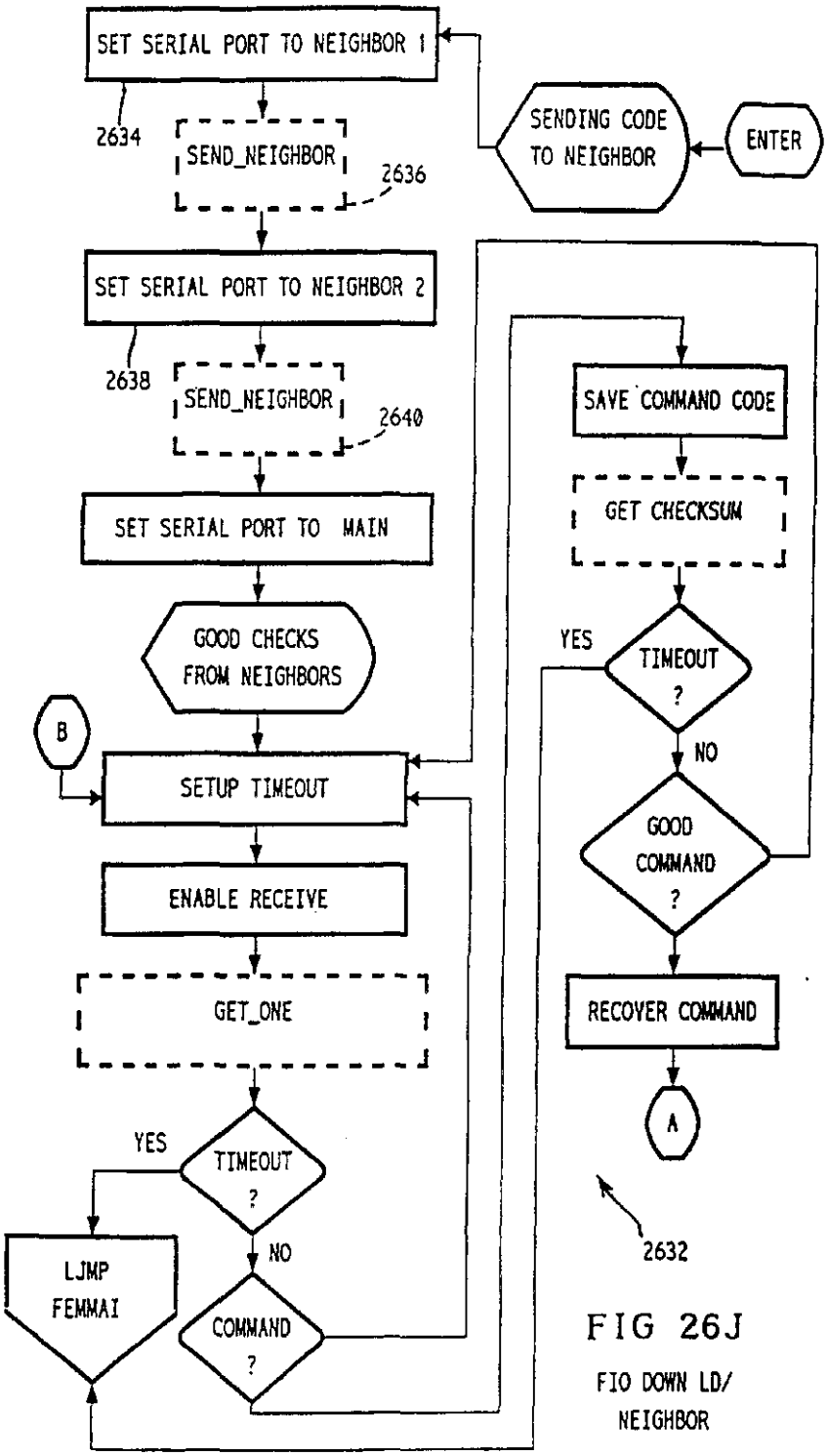
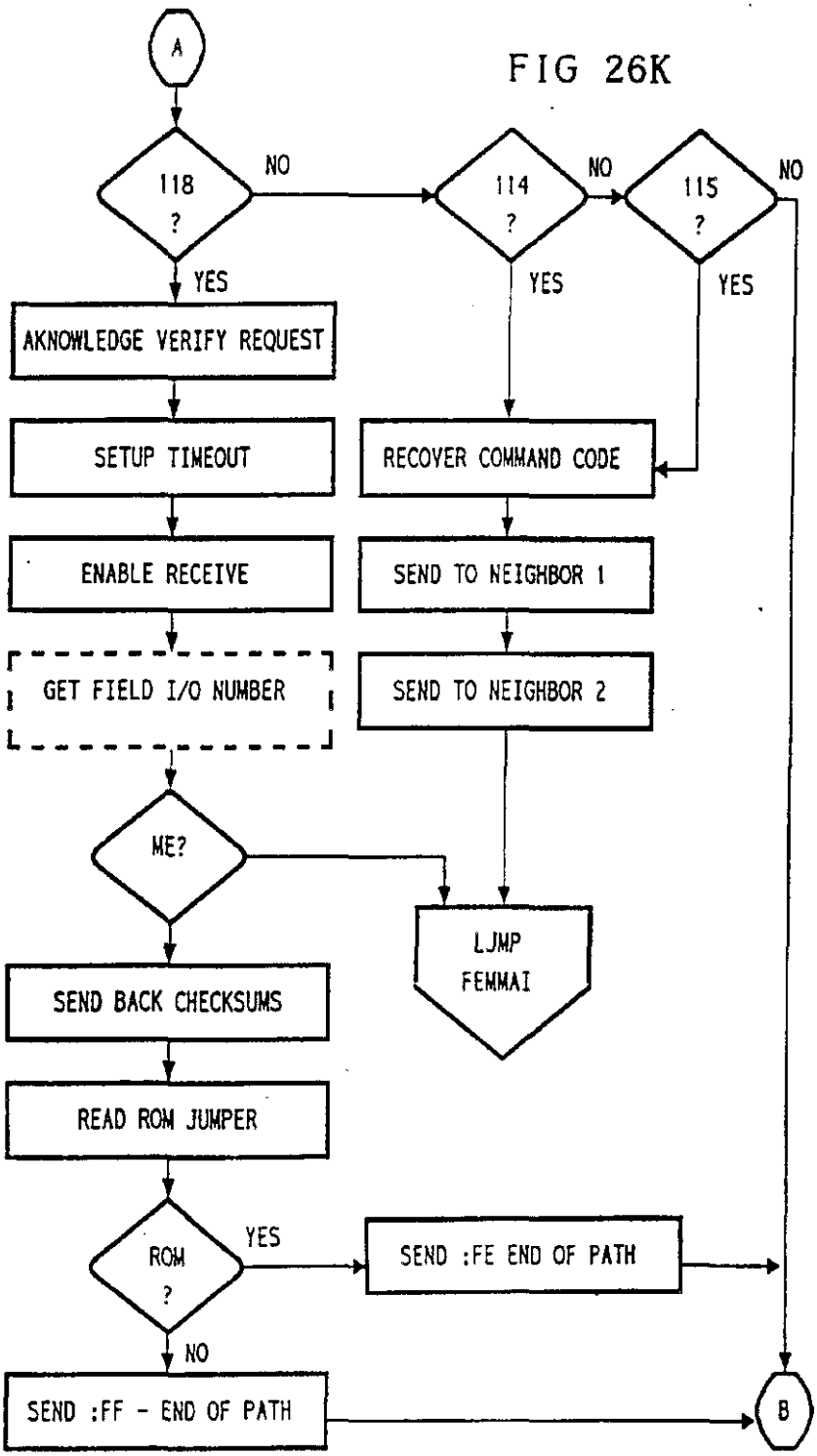


FIG 26J

FIO DOWN LD/ NEIGHBOR

FIG 26K


A
118 ?
NO
114 ?
NO
115 ?
NO
YES
YES
YES
AKNOWLEDGE VERIFY REQUEST
SETUP TIMEOUT
RECOVER COMMAND CODE
ENABLE RECEIVE
SEND TO NEIGHBOR 1
GET FIELD I/O NUMBER
SEND TO NEIGHBOR 2
ME?
LJMP FEMMAI
SEND BACK CHECKSUMS
READ ROM JUMPER
ROM ?
YES
SEND :FE END OF PATH
NO
SEND :FF - END OF PATH
B

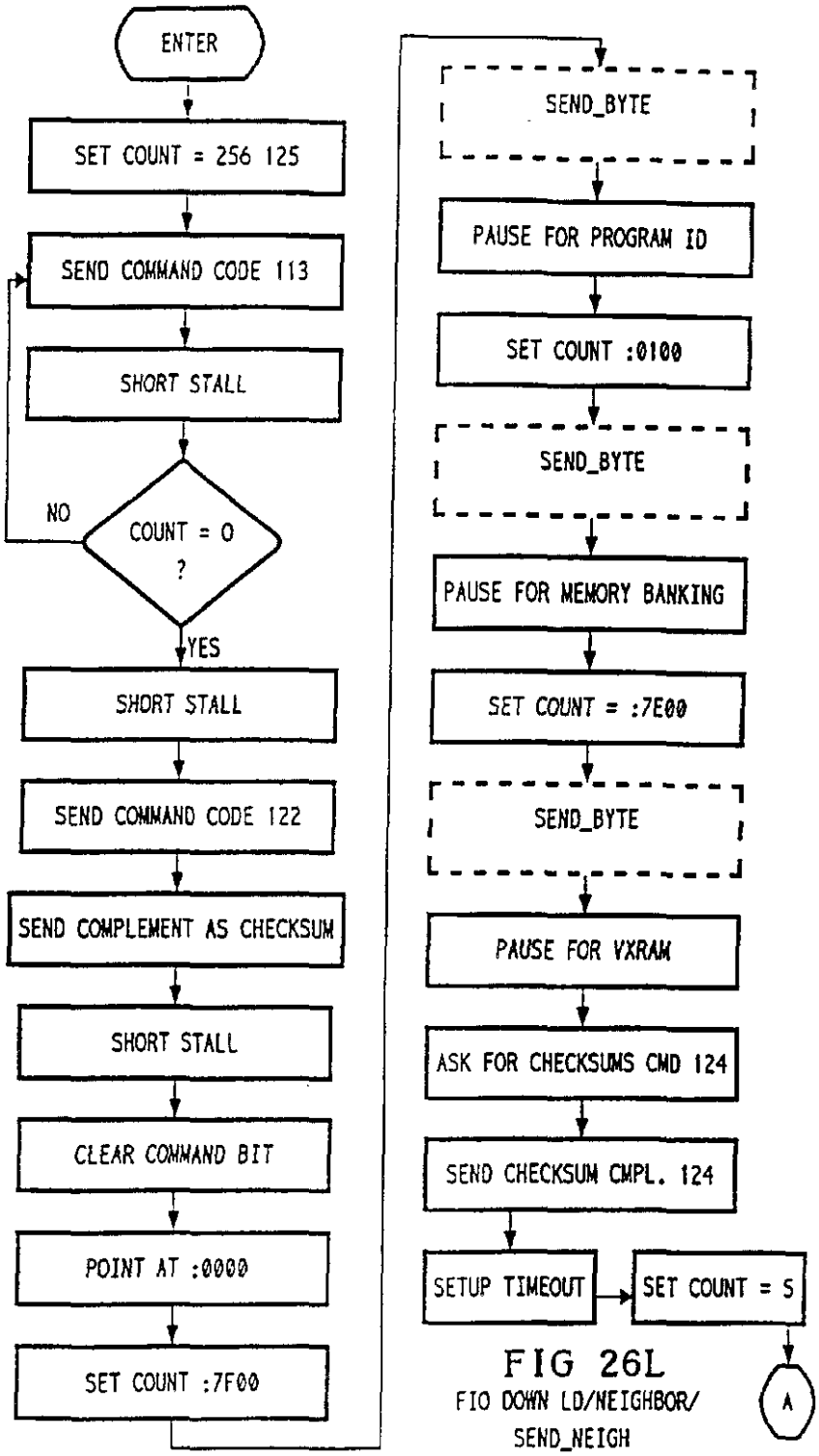


FIG 26L

FIO DOWN LD/NEIGHBOR/ SEND_NEIGH

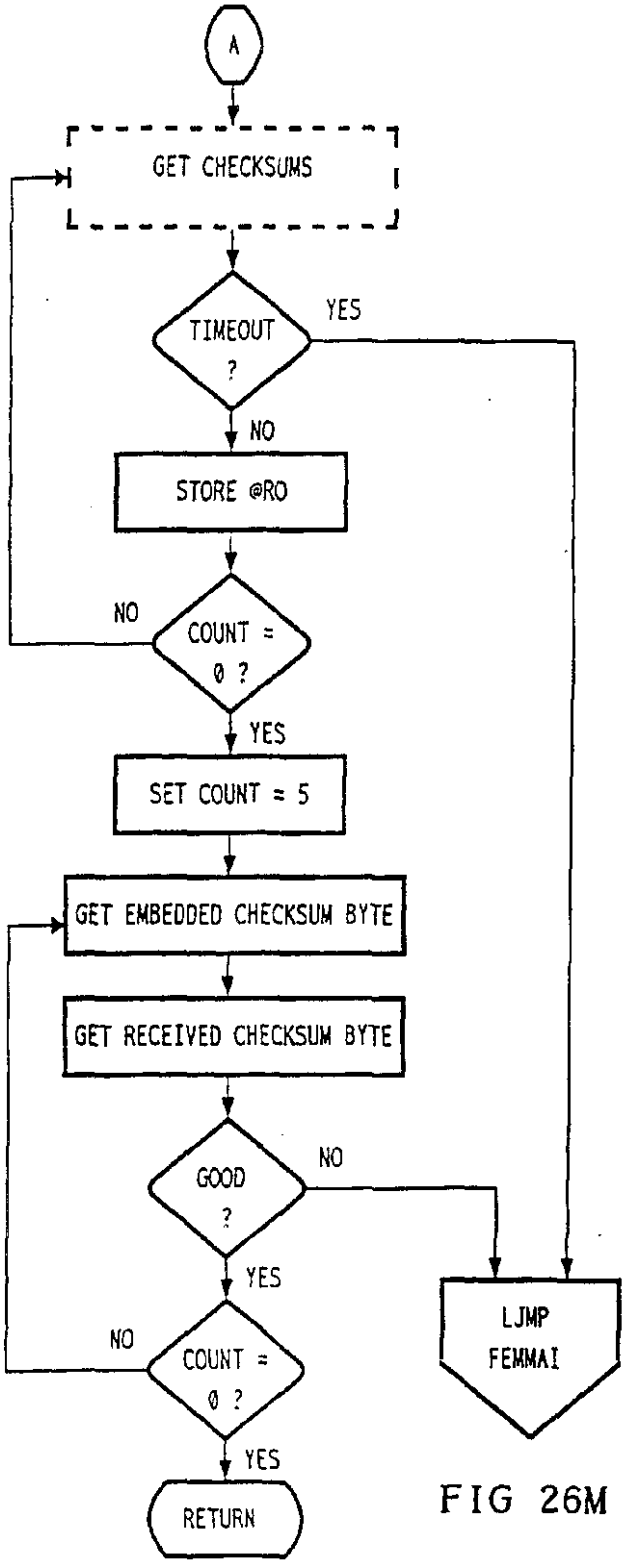

A
GET CHECKSUMS
TIMEOUT ?
YES
NO
STORE @RO
NO
COUNT = 0 ?
YES
SET COUNT = 5
GET EMBEDDED CHECKSUM BYTE
GET RECEIVED CHECKSUM BYTE
GOOD ?
NO
YES
NO
COUNT = 0 ?
YES
RETURN
LJMP FEMMAI

FIG 26M

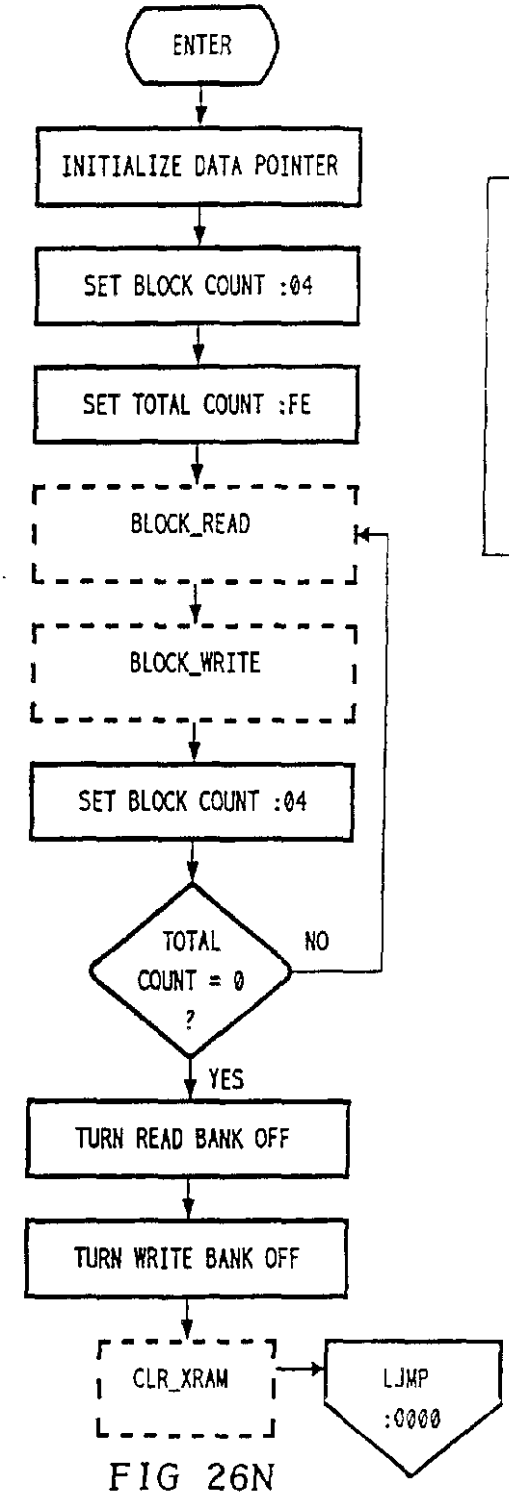


FIG 26N

FIO DOWN LD/NEW PROGRAM/LJMP REPROG

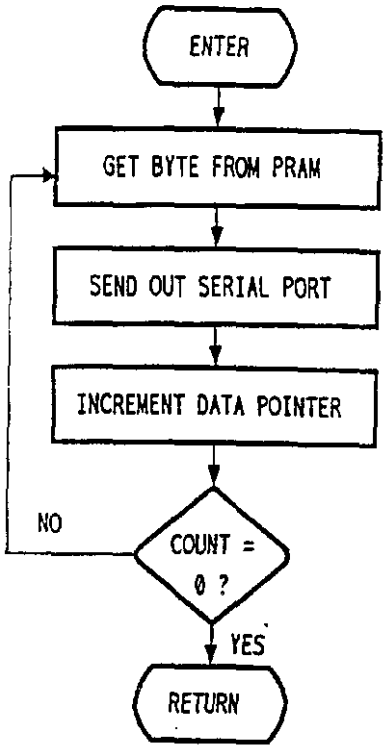


FIG 27L

FIO DOWN LD/NEIGHBOR/ SEND_NEIGH/SEND_BYTE

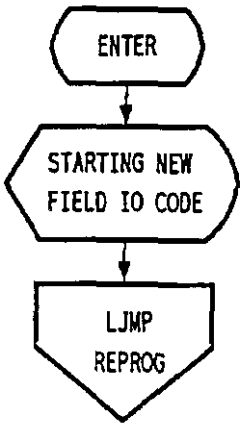


FIG 27M

FIO DOWN LD/ NEW PROGRAM

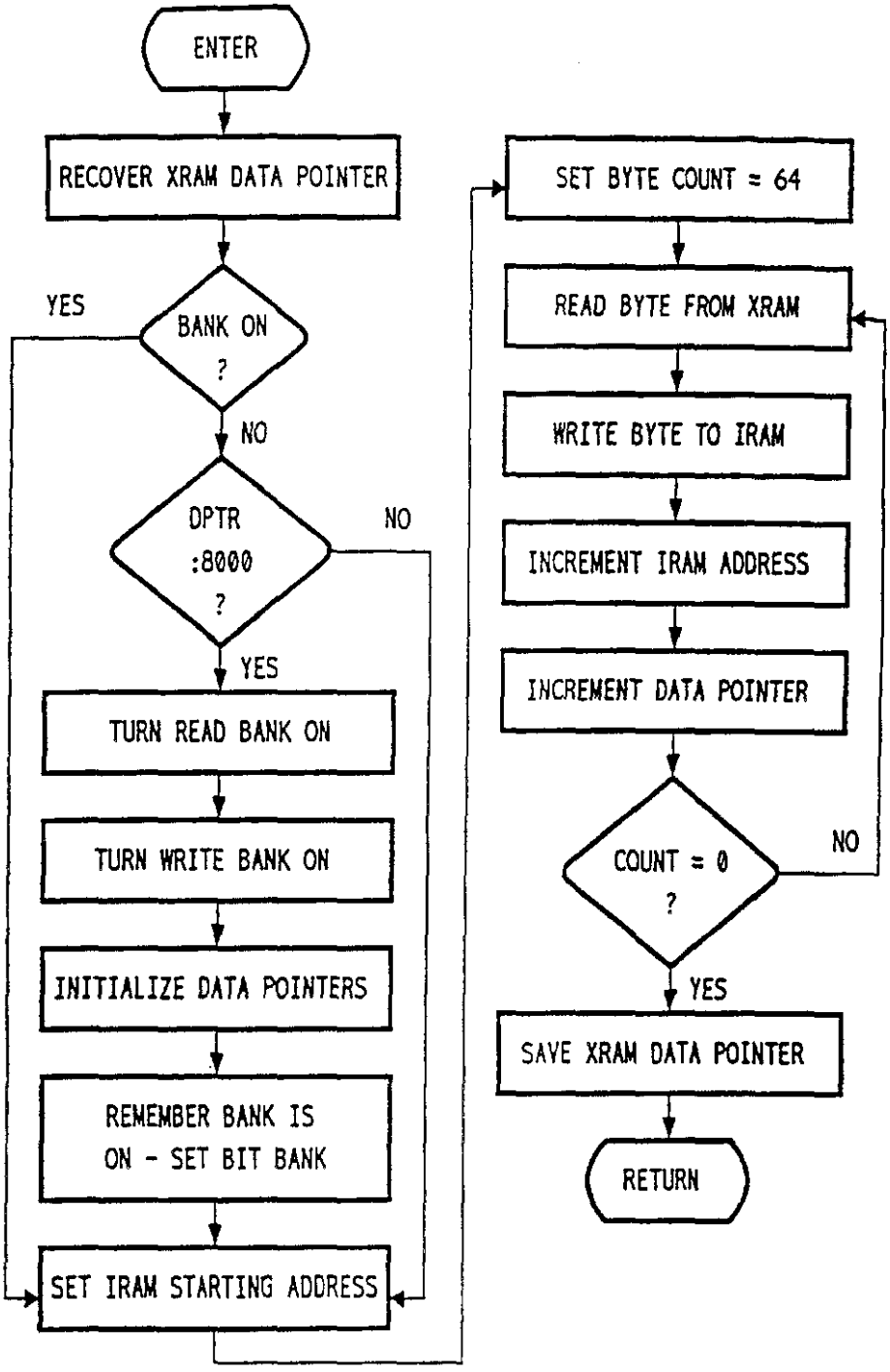


FIG 260

FIO DOWN LD/NEW PROGRAM/LJMP REPROG/BLOCK_READ

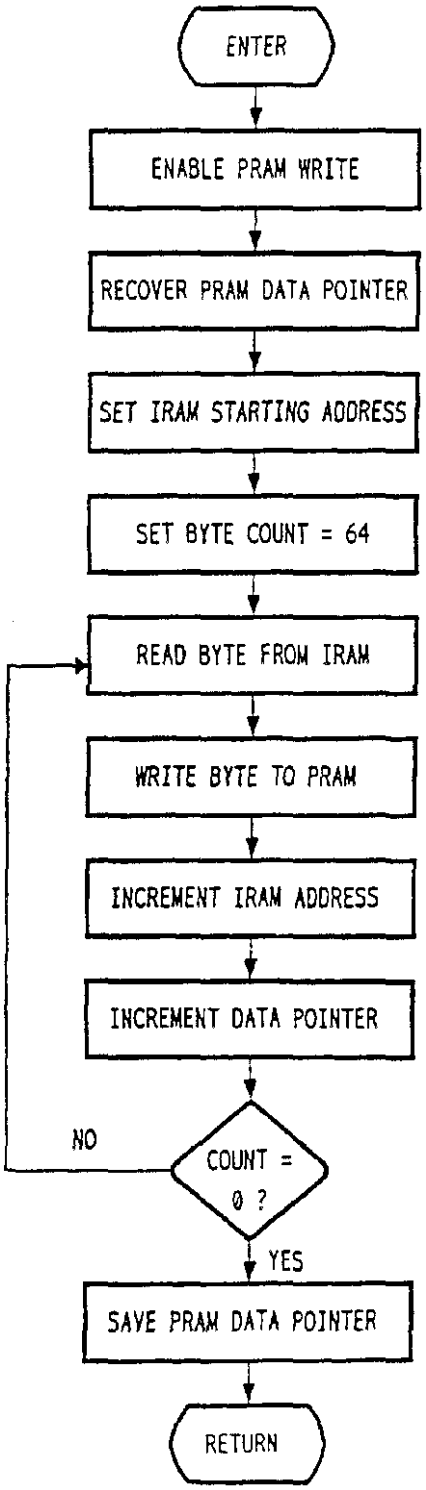


FIG 26P

FIO DOWN LD/
NEW PROGRAM/
LJMP REPROG/
BLOCK WRITE

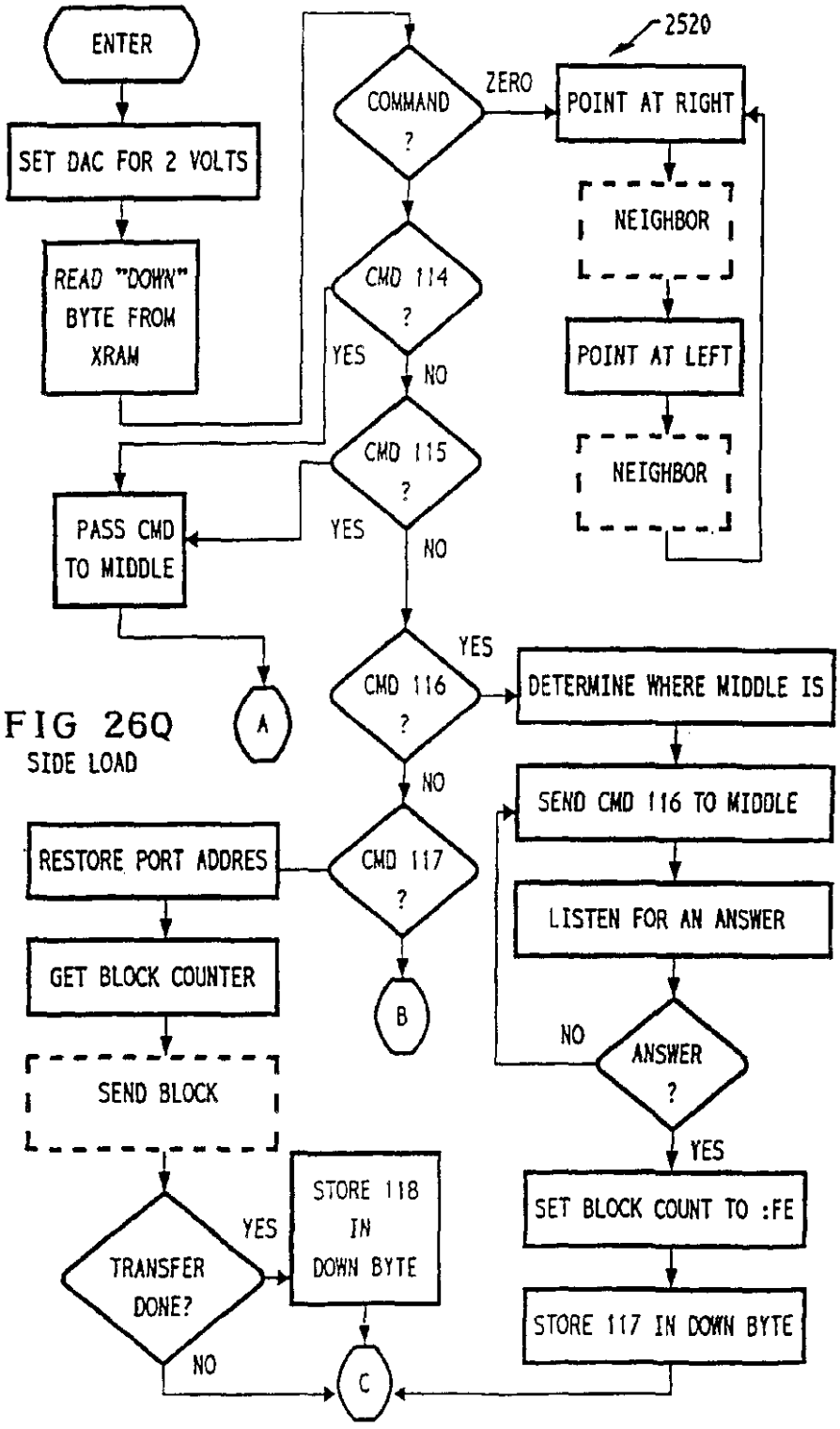
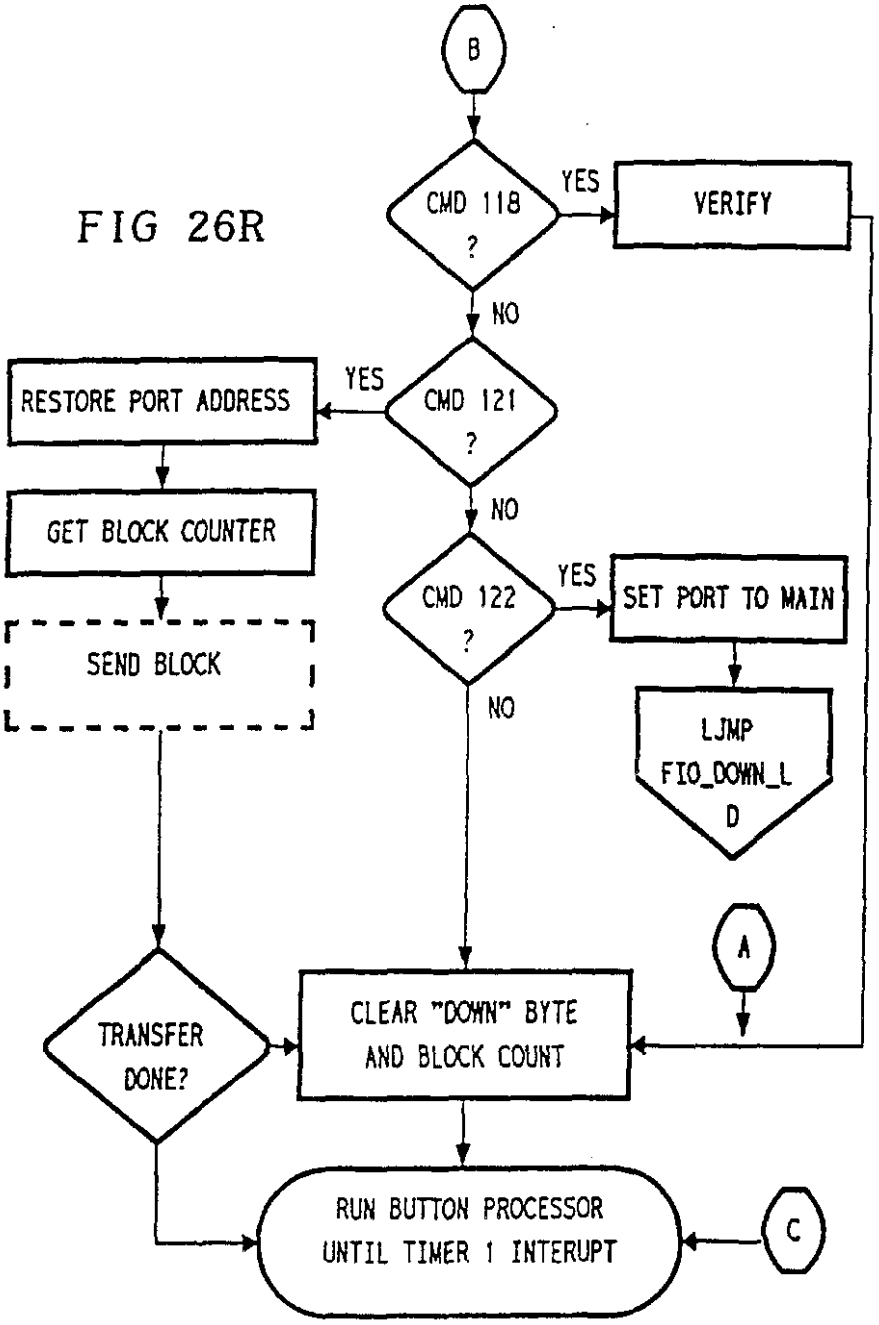


FIG 26Q
SIDE LOAD

FIG 26R


B
CMD 118 ?
YES
VERIFY
NO
RESTORE PORT ADDRESS
YES
CMD 121 ?
NO
GET BLOCK COUNTER
SEND BLOCK
CMD 122 ?
YES
SET PORT TO MAIN
NO
LJMP FIO_DOWN_LD
A
TRANSFER DONE?
CLEAR "DOWN" BYTE AND BLOCK COUNT
RUN BUTTON PROCESSOR UNTIL TIMER 1 INTERUPT
C

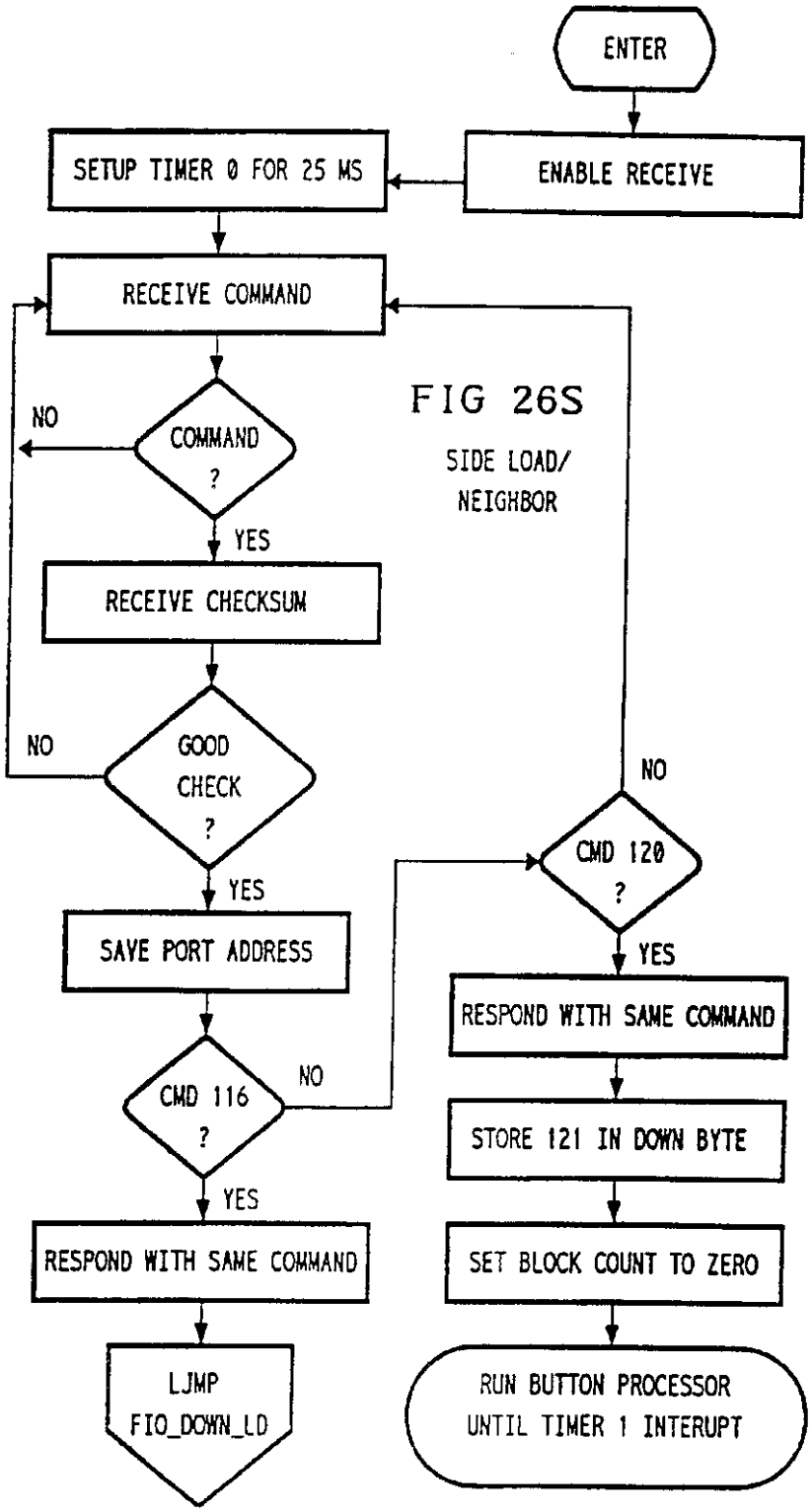
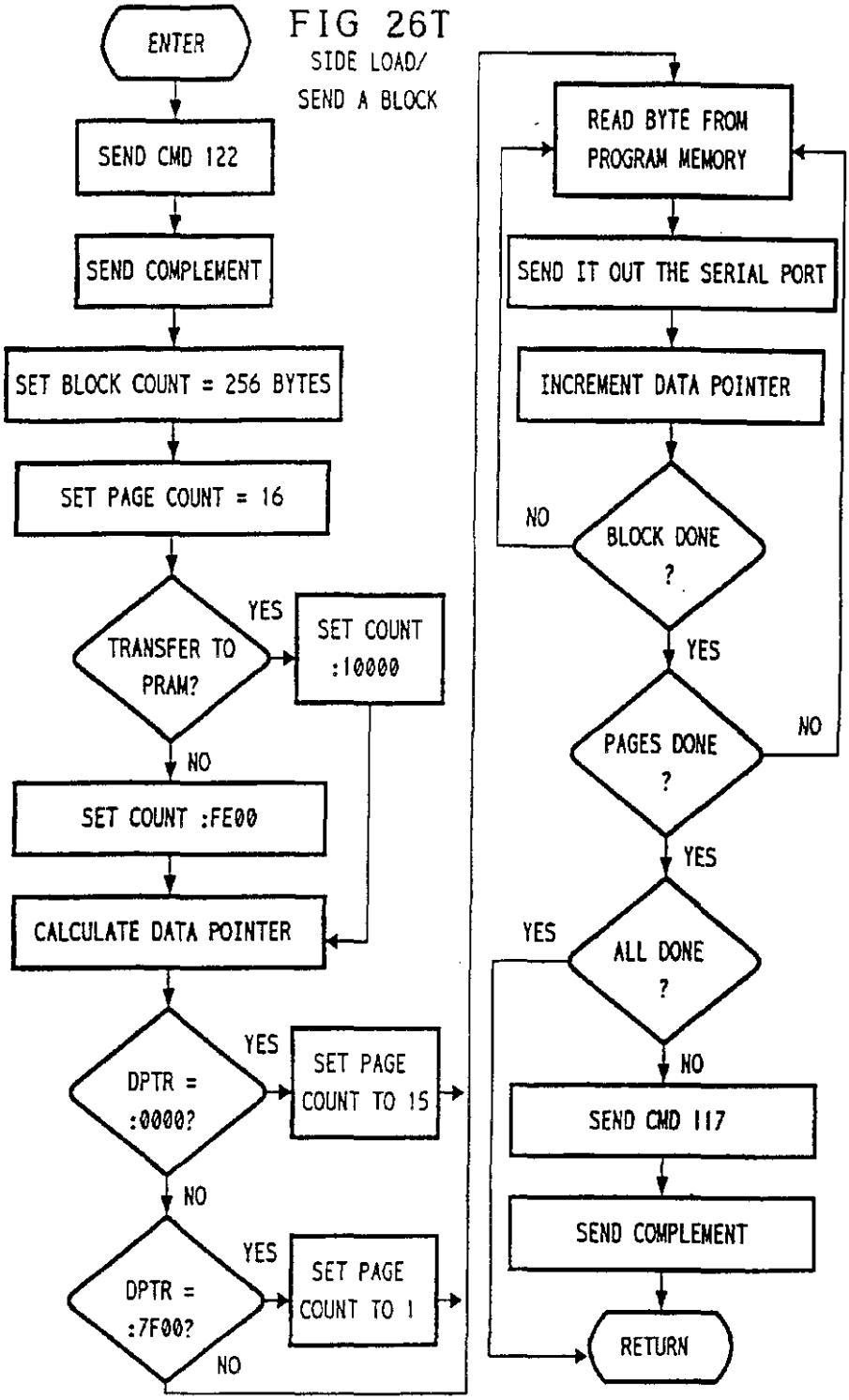


FIG 26S

SIDE LOAD/ NEIGHBOR

FIG 26T
SIDE LOAD/
SEND A BLOCK



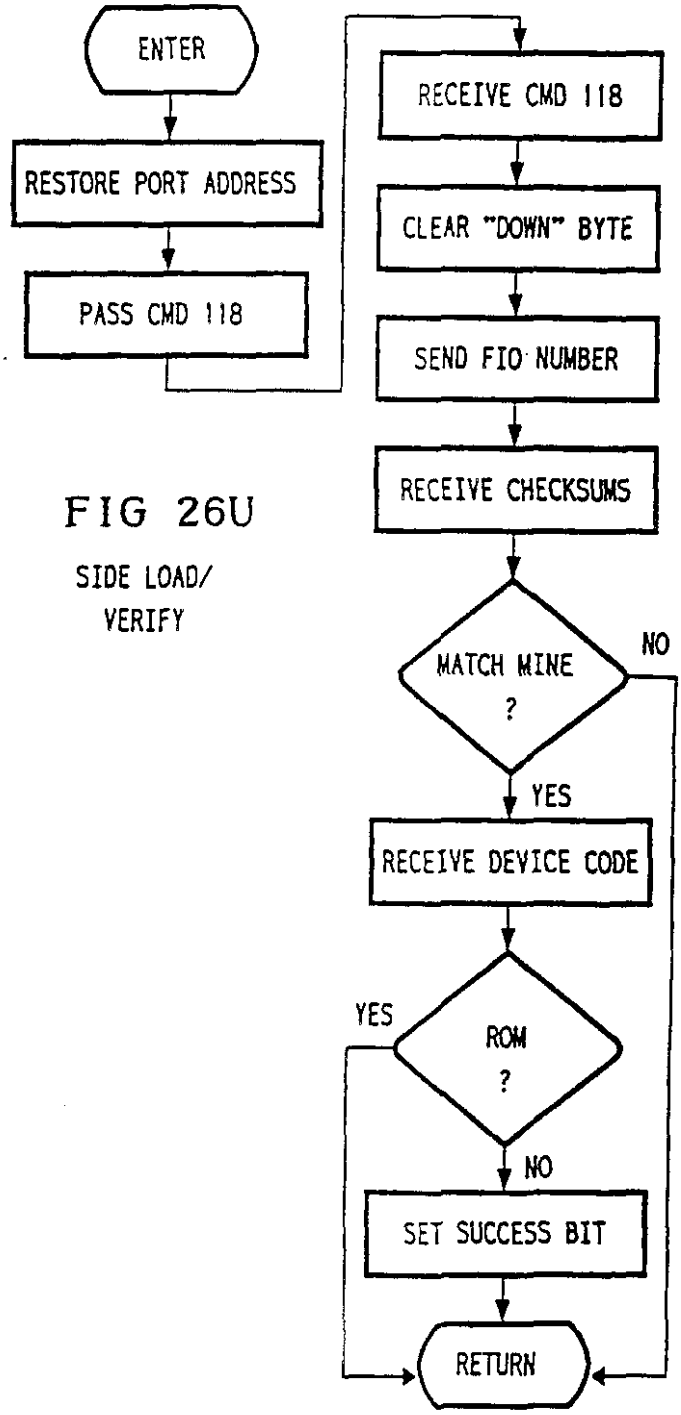


FIG 26U

SIDE LOAD/
VERIFY

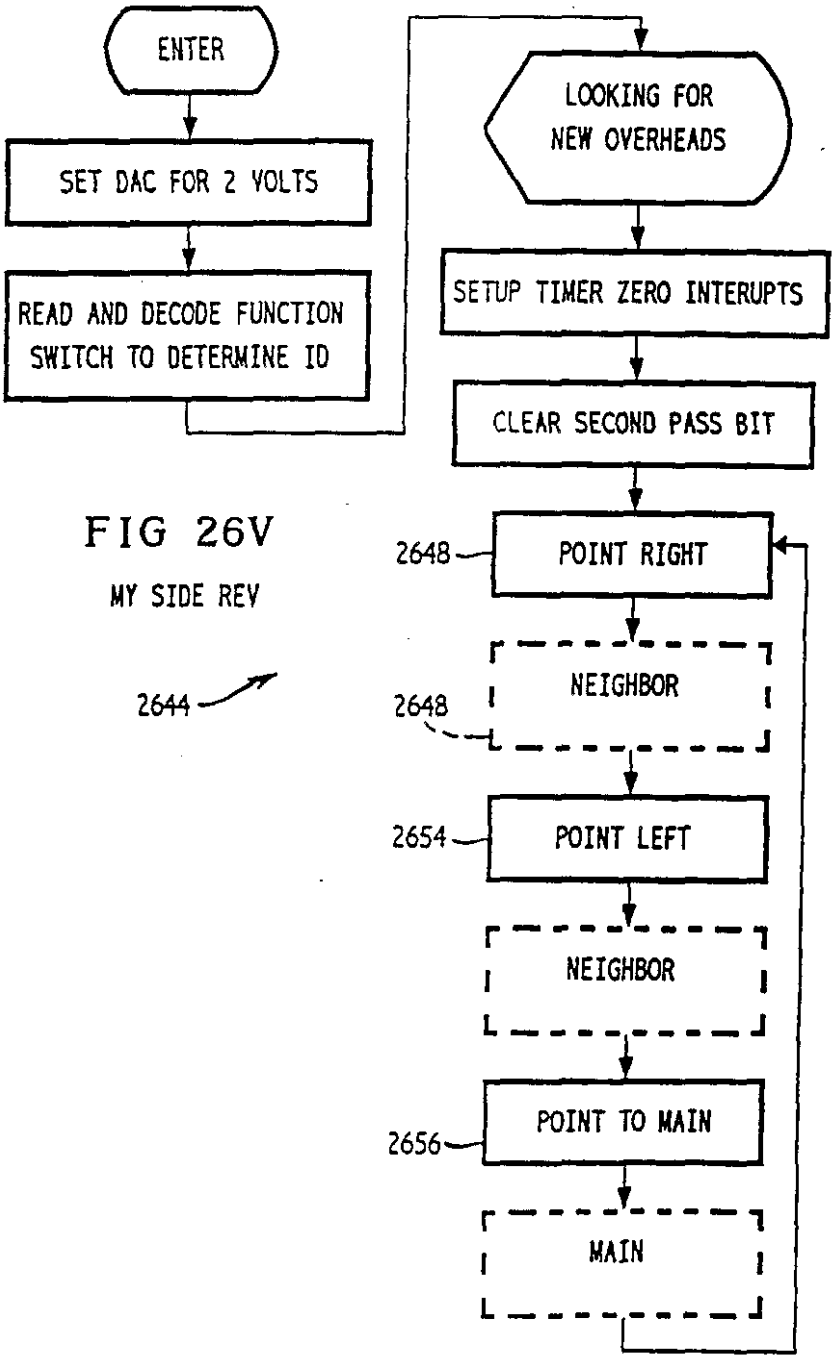


FIG 26V

MY SIDE REV

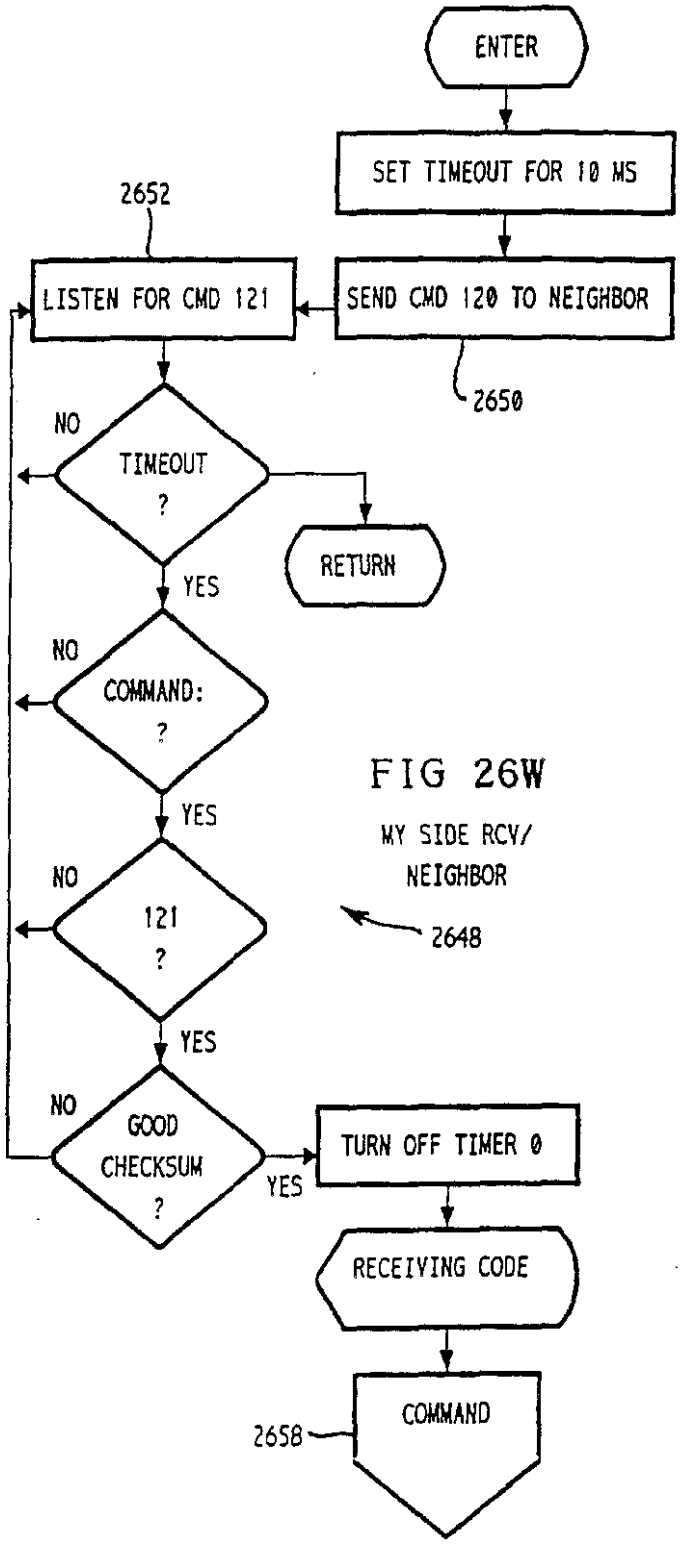


FIG 26W

MY SIDE RCV/
NEIGHBOR

2648

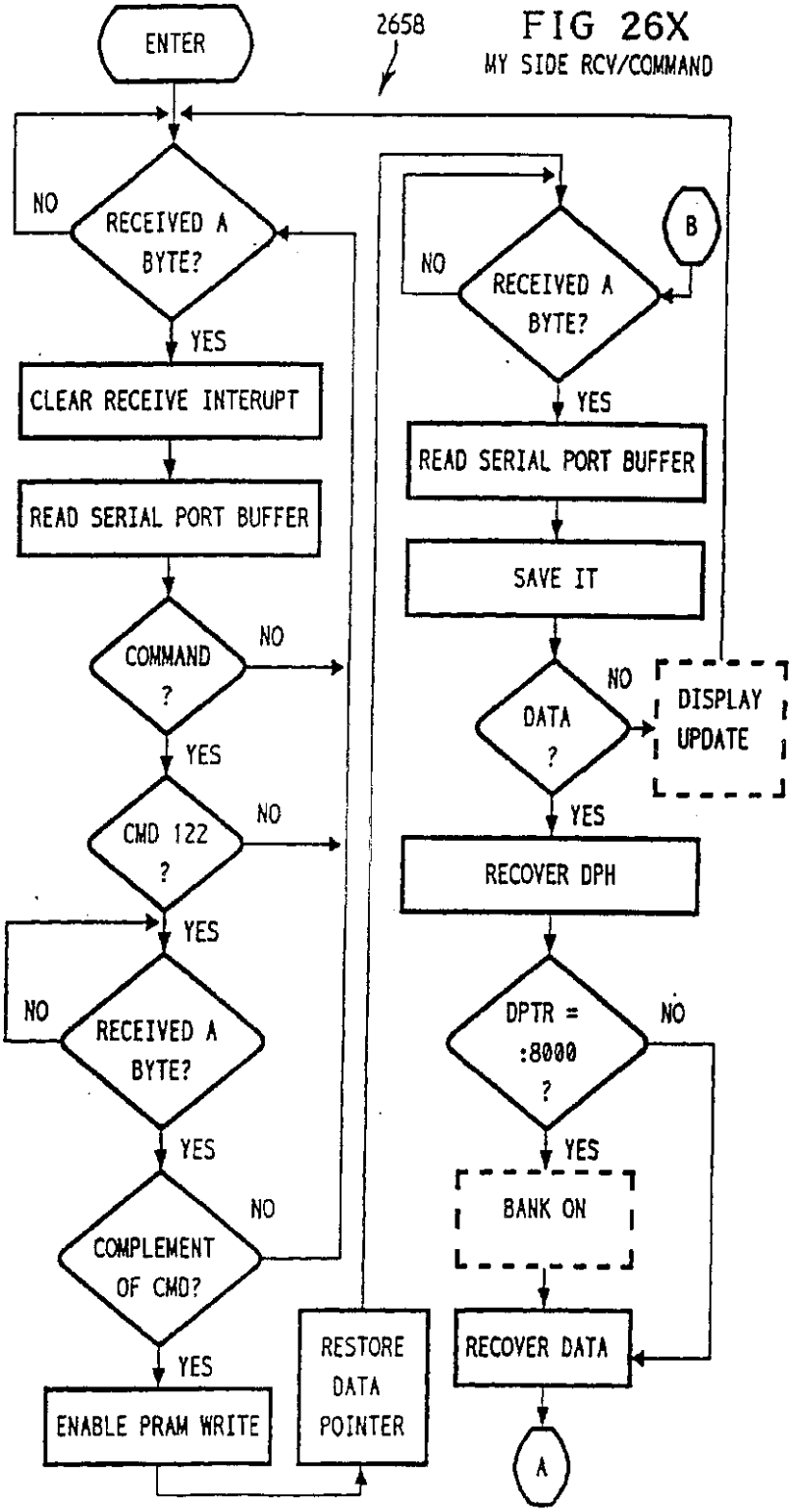
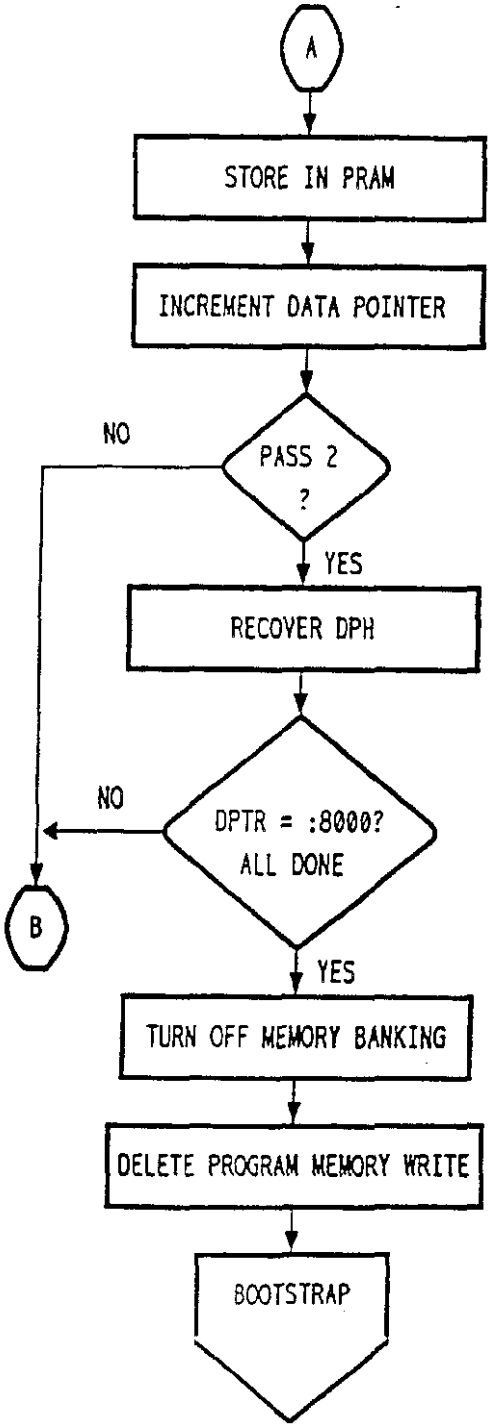

ENTER
2658
FIG 26X
MY SIDE RCV/COMMAND
NO
RECEIVED A BYTE?
YES
CLEAR RECEIVE INTERUPT
READ SERIAL PORT BUFFER
COMMAND ?
NO
YES
CMD 122 ?
NO
YES
NO
RECEIVED A BYTE?
YES
COMPLEMENT OF CMD?
NO
YES
ENABLE PRAM WRITE
RESTORE DATA POINTER
B
NO
RECEIVED A BYTE?
YES
READ SERIAL PORT BUFFER
SAVE IT
DATA ?
NO
DISPLAY UPDATE
YES
RECOVER DPH
DPTR = :8000 ?
NO
YES
BANK ON
RECOVER DATA
A

FIG 26Y


A
STORE IN PRAM
INCREMENT DATA POINTER
NO
PASS 2
?
YES
RECOVER DPH
NO
DPTR = :8000?
ALL DONE
B
YES
TURN OFF MEMORY BANKING
DELETE PROGRAM MEMORY WRITE
BOOTSTRAP

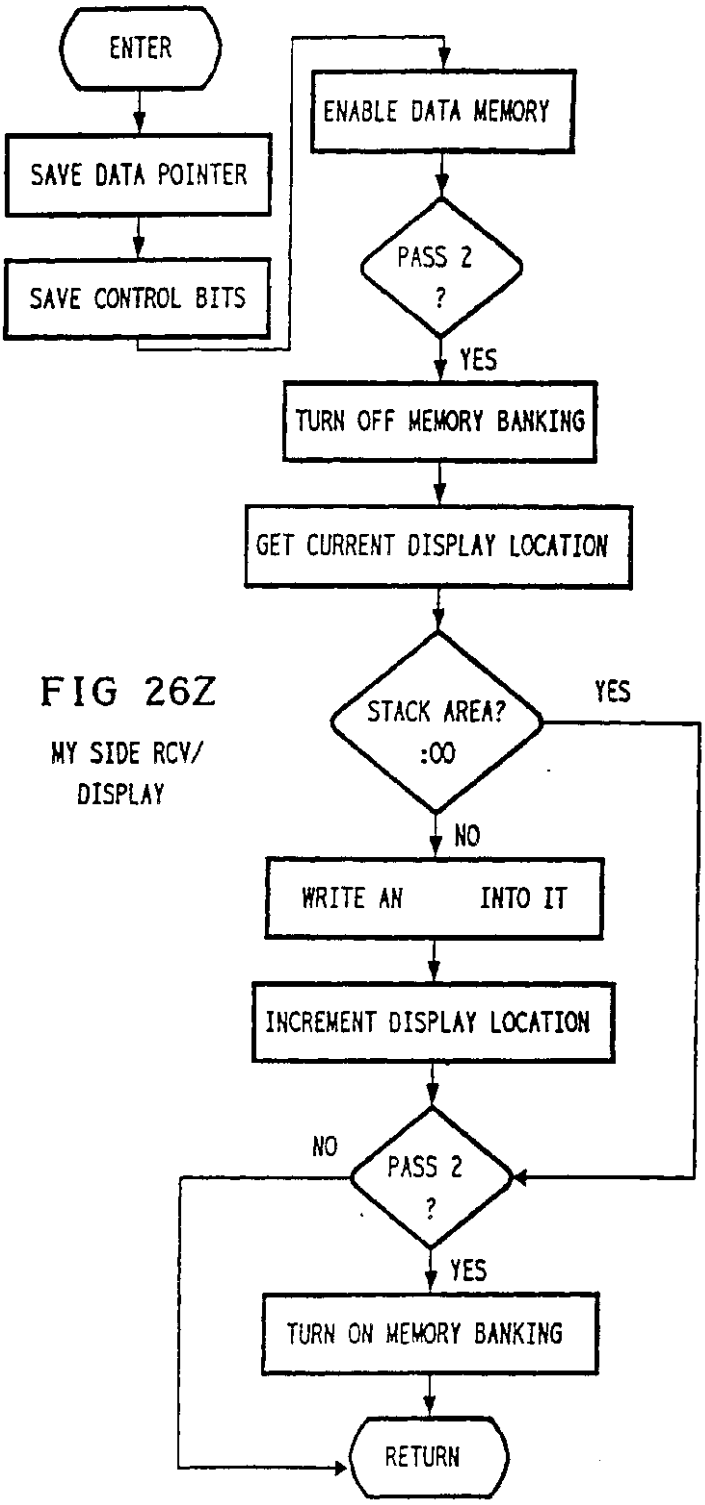


FIG 26Z

MY SIDE RCV/ DISPLAY

FIG 27A
MY SIDE RCV/MAIN

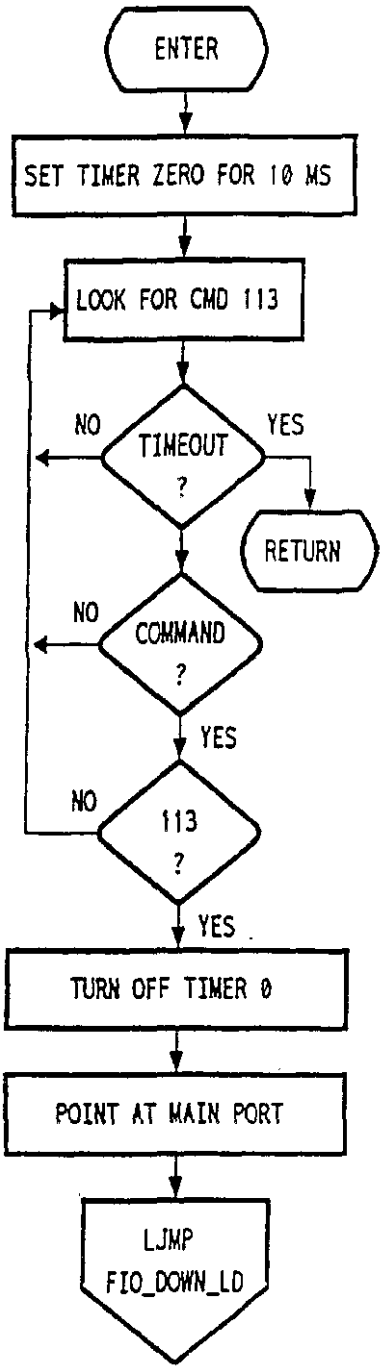


FIG 27B
MY SIDE RCV/BANK ON

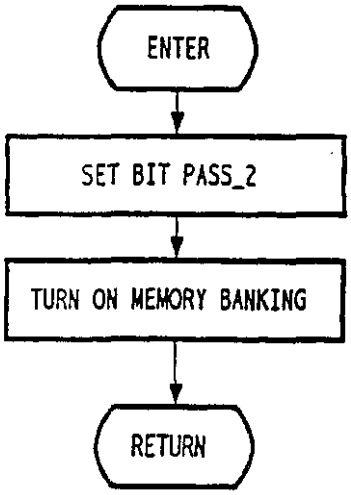


FIG 27C
GET_CODE/CLEAR RAM

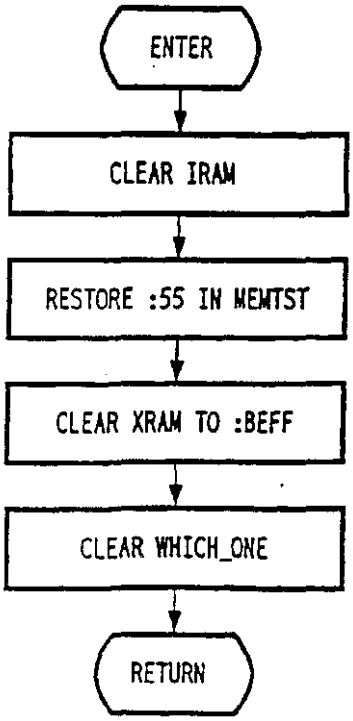


FIG 27D

GET CODE/
VERIFY DOWNLOADED PROGRAM/
RCV_INIT

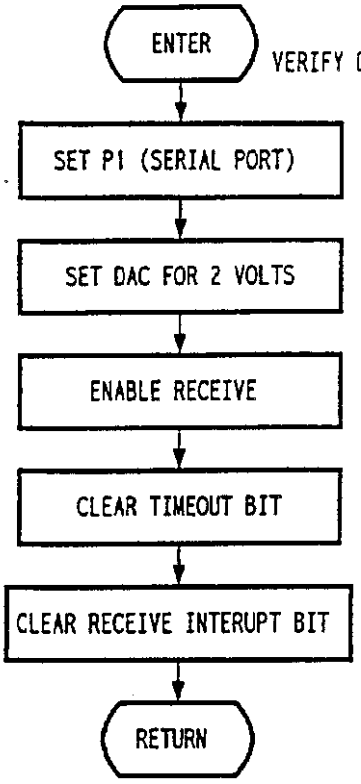


FIG 27E

CHECK_MID/CALL GIT_BIT

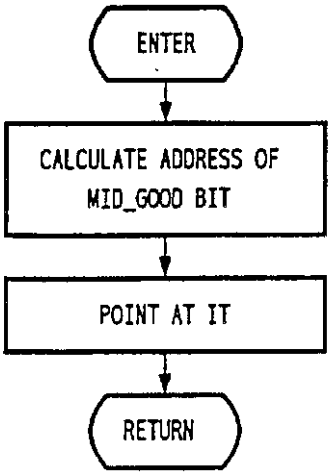
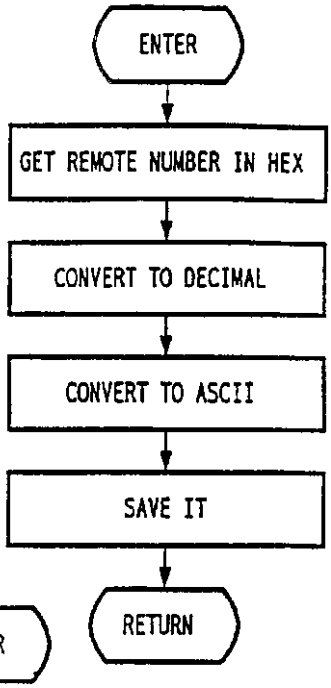


FIG 27F

CHECK_MID/CALL NUMBERS



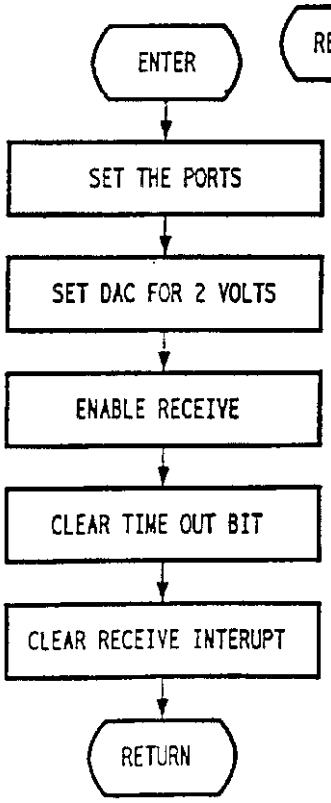


FIG 27G

BRK DOWN LD/
RCV_INIT

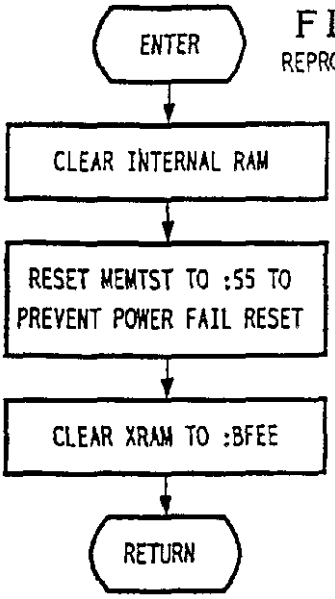


FIG 27H
REPROG/CLEAR RAM

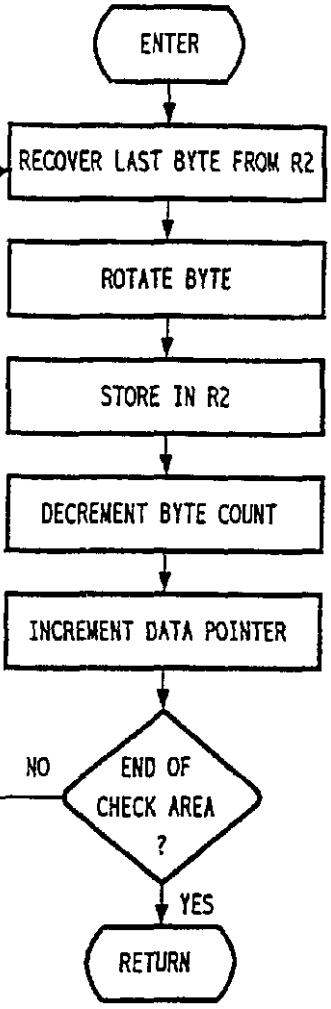


FIG 27J
FIO DOWN LD/VXRAM/SKIP

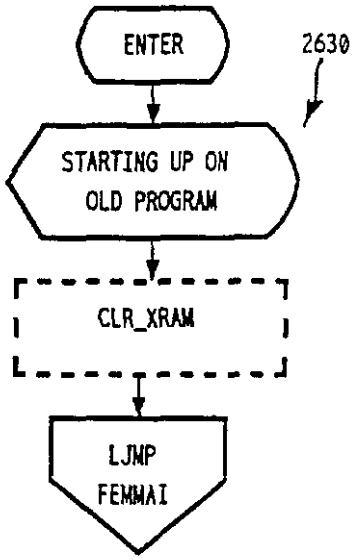


FIG 27I
FIO DOWN LD/OLD PROGRAM

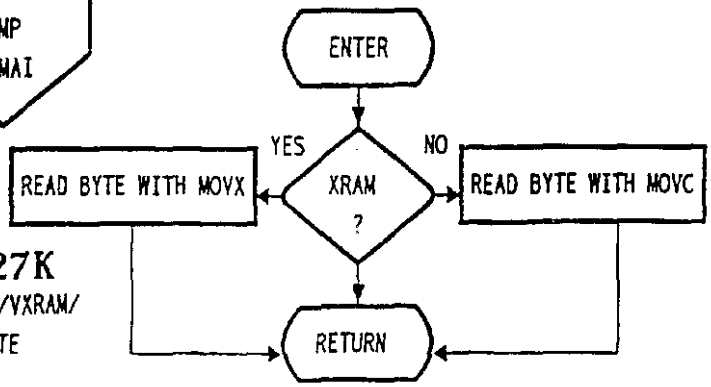


FIG 27K
FIO DOWN LD/VXRAM/
READ_BYTE

具有三個多余的遙控區單元的程序控制交互界面系統

撮錄

一种程序控制交互界面系統（ 10 ）具有分配的三重多余輸入/輸出程序區段計算机單元（ 12 ）。該系統包括許多自身含有的遙控配置三重多余程序區段計算机單元（ 12 ），它經過一個雙方向的通訊网絡聯接到行使多余程序控制計算机（ 14 ）判斷上，該通訊网絡具有至少兩個同向活躍的通訊頻道（ 46，48 ）。每一個程序區段計算机單元包括一組至少三個多余區段計算机（ 92，94，96 ）用以判优輸入和輸出信號。該區段計算机單元還包括單個的中斷電路（ 510，606 ）使得每一個輸出信號都可以北傳送到影響物理程序操作的裝置（ 84，96 ）。這些中斷電路有效地使得輸出值信號，經過使用表決程序，分別判斷經過每一個三個多余區段計算机。