PERIPHERAL INPUT/OUTPUT CONTROL
SOFTWARE SUBSYSTEM DESCRIPTION
NO. 3 ELECTRONIC SWITCHING SYSTEM

CONTENTS PAGE

Figures

NOTICE

Not for use or disclosure outside the
Bell System except under written agreement

Printed in U.S.A.

Tables

1. GENERAL

INTRODUCTION

1.01 The peripheral order subsystem of the No. 3 Electronic Switching System (ESS) is the primary interface between the call processing programs and peripheral devices in an office. Through it, all network controller orders, all distribute orders, and many scan orders are issued.

1.02 ➧This section is being reissued to include 3E3 generic information and order formats that are used in controlling the scanner controller, peripheral pulse distributor, and network controller. Revision arrows are used to emphasize significant changes.➧

1.03 The Peripheral Input-Output Control System consists primarily of a table-driven interpreter which is executed by base level call processing programs and by an interrupt level monitor.

1.04 In order to assist in understanding the I/O control system, a brief discussion of the peripheral devices is included in this section. Related call processing methods are also reviewed.

REFERENCE INFORMATION

1.05 Table A is a list of abbreviations and acronyms which are applicable to this section. The following documents contain information relevant to this section.

SECTION	TITLE
233-120-100	Switching Network, Description and Theory of Operation, No. 3 ESS
233-121-100	Scanner, Description and Theory of Operation, No. 3 ESS
233-121-105	Peripheral Pulse Distributor and Peripheral Decoder, Description and Theory of Operation, No. 3 ESS
233-121-110	Frame Input/Output Controller, Description and Theory of Operation, No. 3 ESS
233-151-105	Call Processing, Software Subsystem Description, No. 3 ESS
233-151-140	Network Path Hunt, Software Subsystem Description, No. 3 ESS

PERIPHERAL I/O CONTROL PROGRAMS

1.06 The following programs implement the major portion of the No. 3 ESS Peripheral Input/Output Control System.

- Peripheral Order Interpreter (POINT)—PR-3H168: the interpreter portion of the table-driven peripheral order subsystem.

- Catalog of Peripheral Control Sequences (PCAT)—PR-3H167: Contains sequences of indices which are interpreted by POINT and generate the desired peripheral action.

- Peripheral Work Subroutines (PSUBS)—PR-3H170: contain subroutines to support various macros and other miscellaneous tasks.
- Peripheral Operation Subroutines (POPS)—PR-3H169: contain subroutines which perform high level peripheral operations, peripheral error handling, and auxiliary operations.
- Network Queue and Timing Hopper Monitor (QTMON)—PR-3H171: administers the network queues and issues all network orders needed for peripheral processing, performs timing hopper scanning, and contains associated subroutines.

2. NO. 3 ESS PERIPHERAL DEVICES

2.01 The major peripheral devices in a No. 3 ESS office are the:

(a) Scanner

(b) Peripheral Pulse Distributor

(c) Network Controller.

These devices are connected to the control unit through the Frame Input/Output Controller (FIOC) as shown in Fig. 1. A description is presented for each with regard to order formats and software considerations. (See paragraph 2.04, 2.08, and 2.13.)

2.02 ▶The 3A Central Control (3A CC) controls the entire No. 3 ESS periphery. The FIOC provides the interface for data transmission between the 3A CC and the peripheral controllers (scanner controller, peripheral pulse distributor, and network controller). Transmitted bits are received one at a time by peripheral controllers and collected in a register. The 3A CC processor transmits orders over channel 0 or 1, subchannel 16 S (send), 13 S (send), 11 S (send), or 17 S (send). Four coaxial cables (subchannels A through D) which are used to send data to the FIOC correspond to each 3A CC subchannel (Fig. 1). Orders on subchannel B are directed to the peripheral pulse distributor (PPD), and orders on subchannel C are directed to the network controller (NWC). The scanner controller is accessed by two subchannels (A and D). The replies for all orders arrive at the 3A CC on subchannel 13 R (receive) via a fifth coaxial cable.◀

2.03 ▶Data is transmitted in 6.67 megabit serial bipolar pulses between the 3A CC and the FIOC. Sixteen serial data bits are sent from the 3A CC in a peripheral order (Fig. 2). Data is transmitted in parallel form between the FIOC and the peripheral controllers. The peripheral controller enable lead is activated when a peripheral order is transmitted over a particular subchannel (A through D). The data is shifted into a register and is gated out in parallel over several leads (FDD00 through FD015). These leads are shared by all the peripheral controllers, but only the enabled controller actually performs the order. Approximately 15 microseconds after the 3A CC sends a peripheral order, the FIOC transmits a 16-bit reply from the device to the 3A CC. The peripheral controllers send replies in parallel (via reply leads) to the FIOC. The FIOC converts the parallel data to serial data, and then transmits the reply to the 3A CC (via subchannel E). The peripheral controller identifies itself by activating a corresponding reply-enable lead. The 3A CC knows the source of a reply because the processor waits for a reply before sending out another peripheral order. Replies are then interrogated by software to determine the results of the order.◀

SCANNER

2.04 The scanner provides a means of interrogating the states of the various ferrods (also known as scan points) in an office. There may be up to 16 logical scanners in an office with each consisting of 32 rows of scan points having 16 scan points in each row (Fig. 3). Four logical scanners compose a physical scanner (2048 scan points). ▶There are four possible physical scanners which consist of logical scanners 0-3, 4-7, 8-11, and 12-15. Each physical scanner contains a maximum of 128 rows. The scanner serves as a constant monitor of subscriber lines. Upon request, the scanner transmits system status to the 3A CC. The scanner controller receives scan orders (Fig. 4) containing address information from the 3A CC via the FIOC. The scanner controller then generates interrogate pulses for a row of ferrods and detects the responses. The address information causes one of 128 rows in a physical scanner to be selected. Interrogation results from the 16 ferrods in the accessed row are detected and returned to the 3A CC via the

FIOC (Fig. 5).◀ An order to a physical scanner contains the row (5 bits), the logical scanner (2 bits), and control information (9 bits). The reply to this order is the state of the selected row of ferrods with a one indicating an off-hook or scan point saturated state.

2.05 In software, a scan point is identified by a 13-bit scan point number (SPN). This number contains 4 bits identifying the point, 5 bits identifying the row, and 4 bits identifying the scanner. A layout of the SPN is shown in part of Fig. 6.

2.06 Scanner 0 is the master scanner which contains all fixed scan point assignments. Supervisory scan points for all service circuits are located in the first 16 rows of this scanner thus allowing an abbreviated form of the SPN to be used for service circuits. The service circuit number (SVC) consists of 8 bits which indicate the row and point of the supervisory scan point for the service circuits and serve as a unique identifier for each service circuit in an office.

2.07 Logical scanners 1 through 15 are the line scanners and are associated with network frames 1 through 15 respectively. The first 24 rows of each scanner contain the network terminal scan points of the associated frame. Rows 24 through 27 are the scan points for the 32 junctors on each frame while rows 28 through 31 contain trunk and miscellaneous scan points.

PERIPHERAL PULSE DISTRIBUTOR

2.08 The peripheral pulse distributor (PPD) is used to control the state relays of the various circuits in a No. 3 ESS office. One or two PPDs may be required in an office depending on its size. Each PPD can access up to 256 peripheral decoders (PD) with each PD capable of controlling four groups of relays with three relays contained in each group. (See Fig. 7.) Each group of three relays is referred to as a triplet.

2.09 ▶The PD controls the peripheral circuits relays (service, trunk, line, test, junctors, tone plant, etc) as directed by the 3A CC via the PPD through the FIOC. The 3A CC transmits 6.67 megabit serial bipolar pulses to the FIOC via subchannel B. The FIOC then converts the serial data to parallel. The parallel data is transmitted to the PPD via a data bus. The PPD receives the 16 data bits (Fig. 8) from the FIOC. The data is now used to select 1 of 256 PDs. Having received the address and control information, the PPD generates a 6.7 volt pulse lasting .45 msec. The PPD transmits these low-speed serial bipolar pulses to the PD. Each order from the FIOC results in a single pulse. Seven input pulses are needed to set the state of the relays in a triplet.◀

2.10 For software consideration, a triplet is identified by its distribute triplet address (DTA). The DTA (Fig. 3) is an 11-bit number which indicates the PPD (1 bit), the PD (8 bits), and the triplet (2 bits).

2.11 Each logical order to the PPD controls a single triplet and can place the relays of that triplet into any desired state. Interrogation of the PD is not possible and any record of relay states must be maintained by software in temporary store. For this reason, only one circuit is assigned to each triplet even though it may have less than three relays. An exception is coin line circuits which are single relay devices and a software record of their states is maintained in order to avoid poor utilization of PD points. Circuits requiring more than three relays are assigned consecutive triplets.

2.12 A logical order to a PPD is composed of seven physical orders. Each physical order contains seven bits of control information, eight bits to denote the PD to be accessed, and one information bit. The PPD sends the information bit into a shift register contained within the PD. Thus, for each logical order, a seven-bit serial string is transmitted to the desired PD. The first and last bits of the string are start and stop bits and are always ones. Two bits are used to denote the triplet to be controlled while the remaining three bits denote the states for the three relays.

NETWORK

2.13 No. 3 ESS uses a three-stage (or five-stage folded) network as depicted in Fig. 9. An office may contain up to 15 network frames which provide the first two stages of switching. Network controller 0 controls frames 1 through 7 and controller 1 controls frames 8 through 15. The network frames are interconnected by the third-stage switches which are controlled by both network controllers.

2.14 Each network frame (or concentrator group) consists of 48 input (or first-stage) switches and 16 output (or second-stage switches) providing a 6-to-1 concentration (Fig. 9). Each switch has 8 input and 8 output levels. The first-stage input levels are also connected to a line cutoff which is used to connect a supervisory ferrod to a network terminal. The first stage output levels and the second stage input levels are connected by the A-links.

2.15 Each network frame also contains 32 junctors and 32 wire B-links which serve as links to the third stage. One side of each junctor is connected to an even-numbered second stage output level and the other side is connected to a third-stage input level. This connection is referred to as a circuit B-link. The odd-numbered second stage output levels are connected to the third-stage output levels and are referred to as wired B-links.

2.16 Associated with each network frame are the circuit and wire test verticals (CTV and WTV) which are connected to the circuit and wire B-links by the test vertical switches. The test verticals are used for no-test operator access and for performing various tests on the network fabric. ➧A network order (two 21-bit words) is transmitted by the 3A CC to the FIOC. The FIOC sends nineteen bits of each word over a parallel data bus to the network controller. The network controller returns 17-bit responses to the 3A CC via the FIOC. The network order and response formats are shown in Fig. 10, 11, 12, 13, and 14. The network controller directs current pulses to open and close crosspoints, perform checks, etc.➧
Each logical order to a network controller consists of three physical orders. The first two are issued immediately and provide the controller with all the information necessary to execute the order. After these are issued, a 10-millisecond break is required so that the controller may react. The third order is then issued which recycles the controller and interrogates it. The reply to this order indicates the success or failure of the controller action.

2.17 A network terminal is identified in software by a unique 13-bit terminal equipment number (TEN). This number (whose format is shown in Fig. 6) utilities 3 bits to specify the input level of the first stage switch, 6 bits to specify the first stage switch, and 4 bits to specify the concentrator group. For software convenience this information is arranged so that the TEN and SPN of a line are identical.

2.18 A path between two network terminals is defined by two TENs, a junctor switch number (JSWN), and the path direction from the RVRS (reverse) bit in the transient call record. The JSWN specifies the second stage switch (3 bits) and level (2 bits) of the junctor in use. Only two bits are required to specify the level since junctors are connected to even levels only.

2.19 The software used for network control must take into consideration two aspects of the network fabric. First, crosspoints must never be operated while there is current flow through the switch so as to avoid switch damage. Also, if a crosspoint is closed more than once without an intervening release (this is defined as scrubbing), the crosspoint can become stuck closed. While it is believed that one scrub is acceptable, the network order sequences are designed to operate on a no-scrub basis.

3. RELATED CALL PROCESSING CONSIDERATIONS

BASE LEVEL AND INTERRUPT LEVEL

3.01 The base level loop is a major software loop of nontime critical programs which include all functions not performed during interrupt level. Figure 15 depicts the base level loop, which includes most call processing programs and maintenance tasks which can be deferred.

3.02 ➧For the 3E3 generic, the major components of the base level loop have been changed. The 3E3 base level loop (Fig. 16) is arranged by nondeferrable and deferrable work. Nondeferrable work consists of normal call processing tasks such as the base level transient call record scanning routine (TCRSCN), the input monitor program (INMON), the base level scanning program (SCANS), and the application portion of the base level monitor program (BLMMA). Deferrable jobs are defined in the JOBADDR table in the maintenance monitor applications program (MMONA). Subroutine DSPTCHER (dispatcher) in CMMON (Common Maintenance Monitor) gives control to 1 of 16 possible waste-time modules. At present only 3 waste-time modules have been defined, which are multiscan functions controller, call processing audits, and main store audits. When jobs fall behind the performance schedule, a priority or force bit is set. The base level loop programs run sequentially and are repeated every 100 milliseconds. The time remaining before the start of a new base level loop is used to execute deferrable work which begins after the CMMON program checks for forced jobs.◆

3.03 There are two types of interrupts in the base level loop. The one of interest in this discussion is referred to as a timed interrupt (Fig. 17). The timed interrupts are hardwire-initiated occurring every 10 milliseconds and lasting for the period of time necessary to perform frequently required functions such as sending and receiving tasks, immediate start, operator trunk scanning, and most peripheral control functions. After an interrupt of the base level loop has completed its functions, control is returned to the base level loop which continues from the point of interruption.

TRANSIENT CALL RECORD

3.04 A transient call record (TCR) is a 16-word block of writable main store which contains all pertinent information concerning a call in progress. The format of the first 5 words of this block of information is fixed. The remaining 11 words are defined by the current usage of the TCR and the phase of transient processing in which it is involved. Figure 18 depicts the TCR layout as it is seen by the peripheral order programs.

3.05 Items of interest relative to the peripheral input-output system are:

(a) The peripheral progress mark (PPM) used by program POINT to keep track of its location in the peripheral catalog (PCAT) (further defined in Part 5 of this section).

(b) The interrupt level progress mark (INTPM) used to indicate that the TCR is performing peripheral actions. (The interrupt timing hopper is also used to time digit sending and the INTPM is used to denote which function the TCR is performing.)

(c) The peripheral error return bit (PERTN). In normal operation when an error is detected during the execution of a peripheral sequence, standard recovery procedures will be taken which will result in the TCR being passed to the failure program (FALTCR). However, certain errors occurring in special call processing situations may require special handling. The PERTN bit is used to mark this condition. In the event of an error, control will be returned to the program which issued the peripheral action.

(d) The base level progress mark (BASEPM), invoked at completion of the peripheral action, determines the next phase of processing for the TCR.

(e) The base level action bit (BACTION) which, when set, will cause the base level TCR scanning program (TCRSN) to invoke the BASEPM of the transient call record.

(f) The base level TIMER used to time long duration intervals. Its value is specified in units of 10 milliseconds. At the beginning of every peripheral action, POINT stores a value (the maximum time for the peripheral action to execute) into this field. The TCRSCN program decrements this timer once every base level loop by the computed duration of the current loop. If the timer passes through zero (times out) prior to the peripheral action being completed, an error condition is recognized and the call in progress will be failed.

(g) The auxiliary peripheral progress mark (AUXPPM), a work area used by POINT. This field contains auxiliary information used by the peripheral programs.

(h) The active path (ACTP) - from one to three network paths which may be defined in words 5 through 9 of the TCR. The interpreter uses this field to remember which path it is operating upon.

(i) The PARTY bit indicates which terminal of the active path is currently being operated upon.

(j) The return address (RAD) used to reduce the number of base level progress marks required. By storing an address in this field, a single base level progress mark can be used to load this address and branch to it. Progress mark routines which issue peripheral actions normally have control returned in this way.

3.06 Words 5 through 9 contain path information used by POINT. Up to three separate paths may be defined in this area:

(a) The A path, which is defined by the APARTY TEN, the AJCTR (junctor), and the ASVC (A party service circuit). The most typical use of this path is as a receiver connection, either from a line to a customer dial pulse receiver (CDPR) or from a trunk to a multifrequency (MF) receiver.

(b) The B path, which is defined by the BPARTY TEN, BJCTR, and the BSVC. This path is most often used for a connection from a ringing circuit to a line or from a transmitter (MF or dial pulse) to an outgoing trunk. (Note that for the A and B paths the service circuit number is used rather than the TEN. This conserves space in the TCR since the TEN is available via a simple translation.)

(c) The talk (T) path which is defined by the APARTY TEN, the TJCTR, and the BPARTY TEN. This path is most often used for talking connections. The contents of the remaining six words are variable and depend on the individual peripheral action being executed. A typical use for this area would be storage of trunk or service circuit DTAs used to operate the state relays of those circuits.

TIMING HOPPER

3.07 In order to avoid having the processor in a waiting situation, real time breaks may be required in the peripheral order sequence. An example of this is relay operations that require settling times. The timing hopper provides a means for requesting real time breaks in the range from 10 to 630 milliseconds. Bits 0 through 7 of each timing hopper slot specify the associated transient call record (TCR). Bits 8 through 15 specify the length of the real time break and are decremented at every 10-millisecond interrupt.

NETWORK CONTROLLER QUEUES

3.08 The network controller queues (Fig. 19) are used to control the sending of peripheral orders to the network controllers and ensure a high degree of occupancy for each controller. The queues are located in writable main store. The block of words is divided into left and right halves. The right half (bits 0-7) of each word forms the queues for controller 0 and the left half (bits 8-15) forms the queues for controller 1. The first two words of the queue are used as control blocks with the remaining slots containing the TCR numbers. One order is sent to each controller every 10-millisecond interrupt, if needed, and each order is verified during the next interrupt.

RELATED TERMS

3.09 In discussing the operation of the peripheral input-output control, certain terms are often used which require clarification or definition.

3.10 The term PACT stands for peripheral action and is the sum total of the events which result when the interpreter (POINT) executes the catalog (PCAT). A PACT is initiated when a call processing program issues a PACT macro. It is terminated when it reaches its point of logical termination or an error occurs which forces a premature termination. In further references, the terms "PACT" and "PACT sequence" will be used interchangeably.

3.11 Figure 20 shows the standard network path connections. The first and second party refer to the first and second terminals of a network connection. For the A path, the APARTY is the first party and the ASVC is the second party. For the B path the BPARTY and the BSVC are the first and second party, respectively. For the talk path the APARTY is the first and the BPARTY is the second.

3.12 The XPARTY and YPARTY refer to the physical arrangement of the terminals involved in a network path connection. Network path connections are described as being either normal or reversed. Figure 21a illustrates a normal path connection where the first party is connected through the A-link to the second stage switch. The second stage switch is then connected via a circuit B-link (junctor) to the third stage switch; then a wire B-link returns to a stage two switch to an A-link to the second party. In a reverse path (Fig. 21b) the positions of the parties are reversed. Each path has (in the TCR) a reverse bit (RVRSA, RVRSB, and RVRST) which indicates whether the path is normal or reversed. The XPARTY of any path is the party closer to the junctor and the YPARTY is the one farther from it (ie, the party separated from the junctor by the third-stage switch). For a normal path, the XPARTY is the first party and the YPARTY is the second party; in a reverse path the opposite is true.

4. PERIPHERAL ACTION EXAMPLE

4.01 The organization of the peripheral order subsystem is shown in Fig. 22. A description of this subsystem is best begun with a peripheral action example. This example will describe a typical PACT in order to detail some of the workings of the programs involved.

BASE LEVEL OPERATIONS

4.02 The description of the PACT sequence example begins with the base level TCR scan (TCRSCN), which scans the TCRs once every base level loop, decrementing the timers and checking the BACTION bits. Whenever a timer passes through zero or its BACTION bit is set, the TCR scan invokes the base level progress mark BASEPM of the TCR via a branch table. The base level progress mark routine then performs whatever actions are necessary at this stage of the call. For this example let the necessary action be a peripheral sequence designated CONNECT, which will connect the TALK path defined in the TCR.

4.03 The progress mark routine issues a PACT macro which expands into a call to POINTD, which is an entry into the interpreter program POINT. POINT proceeds to initialize the TCR as follows:

- The peripheral progress mark (PPM) is stored along with the interrupt progress mark (INTPM) which indicates that the TCR is performing peripheral actions.
- The BACTION bit is zeroed and a BASEPM (PWAIT) is stored in the TCR.
- A return address pointing to code following the PACT macro is stored. (When the sequence is completed, the BASEPM will be invoked which will use the return address to return control to the progress mark routine which issued the PACT. When the initialization is complete, execution of the peripheral sequence begins.

4.04 The example catalog sequence CONNECT is shown in Table B and is used to connect a TALK path (APARTY to BPARTY through TJCTR). POINT fetches each catalog word, decodes it via branch tables, and performs the necessary actions for its execution. When a real time break is required, whether for timing purposes or for issuing network orders, the sequence is interrupted. The sequence is resumed, however, when POINT is called by the interrupt timing hopper monitor (TOMN), which is included in QTMON.

4.05 The first instruction of the sample PACT is PATH T. This command sets the active path indicators in the TCR (ACTP) to indicate that the path being operated upon is the talk path. This will remain in effect for the entire sequence or until another PATH command (or one containing a PATH function) is executed.

4.06 In operating on a network path, a crosspoint must not be operated or released while current is flowing. To ensure no current flow, the DJT OPEN command (distribute to junctor and time) is used to issue a distribute order to open the junctor in the active path (in this case the TJCTR). The timing is required since the junctor relay requires a reaction and settling time (nominally 30 milliseconds).

4.07 Up until now POINT has been executing at base level, having been called by the progress mark routine which was invoked by the base level TCR scan. At this point the catalog execution must be interrupted so that the junctor relay may react and a real time break must be taken.

4.08 Breaks of this nature are effected by the timing hopper monitor (TMON) which is a part of the QTMON program. POINT stores the peripheral progress mark (which points to the next catalog word to be executed) in the TCR and places the 8-bit TCR number along with an 8-bit timer into an idle slot in the timing hopper (TMHOPR in ATSD). Then control is relinquished to the base level TCR scan and the progress mark routine which issued the PACT is bypassed. When the sequence is complete, the PWAIT progress mark, which was stored in the TCR during initialization, will be invoked. Using the return address stored in the TCR, it will transfer control to the location in the base level progress mark routine immediately following the PACT macro. At this point all base level peripheral action is complete and the remaining work of the base level loop continues unhindered. From this point on in this peripheral action example, all action will take place at interrupt level.

INTERRUPT LEVEL OPERATIONS

4.09 The timing hopper monitor (TMON) is invoked once every interrupt to service the timing hopper. The structure of the interrupt program is shown in Fig. 23. It scans through the hopper, decrementing the timers of active slots and invoking POINT for those slots that time out. The slot in use by the example sequence will remain in the hopper for three interrupts (30 ms for junctor relay react time). After that, it will time out and the TCR will be passed to POINTI for continued catalog execution.

4.10 After checking the TCR to ensure that it contains valid data, POINT resumes execution of the catalog using the PPM to determine the location from which the next word is to be fetched. In this example, the next word is NET Z, meaning "issue a network order to close the third stage crosspoint of the active path."

4.11 A brief description of the syntex of the NET command is in order at this point. "X" and "Y" are used to refer to the first and second stage crosspoints, respectively. "C" and "T" refer to the cutoff and test vertical switches while "Z" refers to the third stage crosspoint. The appearance of one of these letters (in conjunction with the NET command) generates an order causing the named crosspoint to be closed. Prefacing a particular letter with an "0" will open the named crosspoint.

4.12 In order to execute the command to close the third stage crosspoint, POINT first determines which network controller is to receive the order. Next, it places the TCR number at the bottom of the queue (NET Q in ATSD) for the selected controller. Then, after storing the PPM, POINT returns control to TMON which continues processing the remainder of the hopper.

4.13 The network queue monitor (QMON) is invoked once every interrupt. It removes the top entry from each queue (there is one for each controller in the office) and services it. This service involves issuing an order to the controller associated with the queue from which the entry was removed. In short, QMON selects the queue for controller 0 and (assuming that it is not empty) removes the top entry (referred to as the head) which is the 8-bit TCR number for which the order will be issued. In this case, it is the TCR of the example which is attempting to close the third stage crosspoint of its talk path. Using the PPM to reference the catalog word which contains the order information and the path data in the TCR, QMON constructs the first two words of the order which are then sent to the controller. The TCR number is then placed in a special slot of the queue designated as the VERIFY slot. QMON then repeats the process for the queue of controller 1 (provided it is equipped and not empty) and then passes control to the next portion of the interrupt program.

4.14 Ten milliseconds later, during the next interrupt and prior to processing the top entry in the queue, QMON retrieves the TCR number stored in the verify slot and issues a third order which recycles the network controller and interrogates it as to the results of the action generated by the previous two words. If everything operates correctly, QMON then places the TCR number into a temporary word of writable store and proceeds to service the next entry in the network queue. This TCR number remains in tempory store until TMON (which is invoked in a later portion of the interrupt) retrieves this number and calls POINT. Now the execution of catalog functions continues.

4.15 The next two words of the sample catalog sequence are also network orders. The command operand OCXYOT can be read "open the cutoff (OC), close the first stage (X), close the second stage (Y), and open the test vertical (OT)." This order is performed twice, once for each side of the network path. The second operands, 1PARTY and 2PARTY are used to indicate which side of the path is being operated on.

4.16 After all crosspoints have been closed, it is necessary to take a short real time break to ensure that all crosspoint chatter has subsided before the final connection is made since current through a chattering crosspoint could weld it. The T (or time) command is used for this purpose. Its operand (a one in this example) indicates that the duration of the break is one period of 10 milliseconds. The TCR now goes back into the timing hopper and returns 10 milliseconds later. The DJT TALK command is now used to place the junctor into the correct "talking" state. This state is determined by examination of the line/trunk bit for each party in the TCR. This state is set so that a line will be supervised by the junctor scan point. Another 30-millisecond break is now required to ensure that the junctor relays have settled.

4.17 Finally, using the BASE command, the peripheral action sequence example is concluded. BASE cleans up the TCR and sets the BACTION bit so that the TCRs BASEPM will be invoked during the next base level loop. Having done that, POINT returns control to TMON and the sample catalog sequence is completed.

4.18 During the next base level TCR scan, TCRSCN calls the PWAIT progress mark routine. This routine loads the return address which POINT stored during the PACT initialization and branches to it. The progress mark routine which issued the PACT has control once again and can proceed with the next processing step.

5. PERIPHERAL I/O CONTROL PROGRAM DESCRIPTIONS

PACT MACROS

5.01 Macros are used in call processing to obtain access to other routines or subroutines. The PACT (peripheral action) macro is the primary interface between call processing programs and the peripheral order subsystem. Its use initiates a peripheral action by transferring control to POINT for interpretation and execution of one of the sequences in the peripheral catalog (PCAT).

PERIPHERAL ORDER INTERPRETER

5.02 The peripheral order interpreter (POINT) is organized into six main sections:

(a) Entry points for the PACT macro and interrupt monitors

(b) Peripheral catalog access mechanism

(c) Execution routines for catalog indices

(d) Execution routines for AUX work indices

(e) Subroutines

(f) PEER, the peripheral error recovery progress mark.

A flow diagram of the interpreter program (POINT) is shown in Fig. 24.

5.03 Four entries are provided in POINT: POINTD, POINTR, POINTF, and POINTI. POINTD and POINTR are used by the PACT macros. POINTD is used when the PACT parameters are contained in variable field data (VFD) words following the macro-generated branch and save address indirect (BSAI) instruction. POINTR is used when the information is contained in registers. It should be noted that all POINTD does is load the data into registers, update the return address in the hold-get stack to bypass the in-line VFDs, and then transfer control to POINTR.

5.04 POINT is designed to check its inputs and POINTR is no exception. It first determines if the TCR address being passed to it is an active TCR. If not, it simply ignores the erroneous call and returns. It then scans the network queues and timing hopper to see if there is any active entry for this TCR. If one is found, the action taken depends on the PPM in the TCR. If it is zero, the queue or hopper entry is assumed to be in error and is ignored. If the PPM is nonzero, then the queue or hopper entry indicates that the TCR is already involved in a peripheral action. In this case, the TCR will be failed with an error code indicating that a PACT was issued with one already in progress. In either case, the offending queue or hopper entry will be destroyed.

5.05 Further initialization involves storing the BASEPM (either passed from the PACT macro, or if not, the default of PWAIT), setting the INTPM to one (indicating that peripheral action is in progress), storing the time-out limit in the TCR timer, and saving the return address if requested. Finally, the hold-get stack is set to the level indicated by the value of SAVEHG and catalog execution is initiated.

5.06 POINTI is the entry point used by the timing monitor (TMON) to service TCRs performing peripheral actions or digit sending. This entry expects a TCR number rather than an address and if the TCR indicated is not active (verified by checking the ACTIU bit) and performing peripheral actions or digit sending (INTPM=1 or 2), the call is ignored. Otherwise, if the INTPM is two, control is transferred to the SENDER entry in the DIGPRO program. (As stated previously, the timing hopper is also used for digit sending.)

If the INTPM equals one, then catalog execution is initiated.

5.07 The POINTF entry is provided solely for the network queue monitor, but the associated code is used by all routines in the interpreter as a common location for peripheral failure processing. It expects an error code which it stores in bits 12 through 8 of TCR word 3. It then sets the BASEPM to PEER (unless error return was requested by the user) and sends the TCR back to base level by setting the BACTION bit.

5.08 The catalog execution operation involves using the peripheral progress mark to locate the catalog word to be executed, loading it and transferring control to the correct routine for execution. Certain checks are made, however, when the PPM to be used is loaded from the TCR or developed in some way from TCR data. This check is actually a range check of the PPM to ensure that it points to a location in the catalog. However, PPMs developed simply by adding one to the current value or branch locations taken from catalog words are not checked.

5.09 Layouts of the general formats of the catalog words are shown in Fig. 25. The low five bits (0-4) contain the catalog index which POINT uses to interpret the word. For indices 0 through 15, bit 5 indicates the presence or absence of an auxiliary (AUXWORK) index. If the AUXWORK index is present it occupies bits 6 through 10. For words with no AUXWORK index, bits 6 through 15 are available as variable data; with an AUXWORK index present, only bits 11 through 15 may be used as variable data. For indices 16 through 31, bits 5 through 15 are used as variable data.

5.10 The auxiliary work index, if present, is executed before the main catalog index. AUXWORK functions are limited to simple functions which do not require any variable data in the catalog word. Many of these are testing functions (such as FCG or power cross) which will abort the sequence if they fail. A list of the indices is shown in TABLE C.

5.11 Main catalog indices 0 through 15 fall into four categories: enqueuing for network orders, issuing distribute orders, taking real time breaks via the timing hopper, and completing the sequence by marking the BACTION bit in the TCR so that its BASEPM will be invoked during the next base level TCR scan.

5.12 For network order enqueuing, the correct network queue is selected and the TCR placed in it. Normally, selection is made using the network of the first party (for stages 1 and 2 orders) or the network of the X party (for stage 3 orders). Since this involves a certain amount of work, and since many catalog sequences issue several network orders in a row, a special index is provided which specifies that the TCR be enqueued for the same controller for which it was previously queued. POINT remembers the source of a TCR while it is processing it and can determine from which network queue a TCR was removed, provided that no real time breaks are taken between one network order and the next.

5.13 Distribute orders vary only in the source of the distribute triplet address. It can be stored in a word of the TCR or it can be the active path junctor. In either case, the variable data of the catalog word contains a bit, which if set, specifies that a 30-millisecond real time break should be taken after the order. In most cases a real time break is needed to provide for relay reaction time.

5.14 Indices 16 through 31 furnish logical functions. They provide various conditional and unconditional branches, provisions for calling and returning from subroutines, and modification of areas of the TCR and other areas in call store.

5.15 Conditional branching can be made on the setting of bits in the TCR, on the value of data in certain words of the TCR, or on a variety of miscellaneous tests such as the availability of a test vertical, or the results of a power cross scan. A list of these miscellaneous tests is included in TABLE C. Conditional branching is forward only and limited in range to 9 or 65 words, depending on the format of the particular word. Unconditional branches and subroutine calls have unlimited range in both directions.

5.16 Two indices are provided for changing data in the TCR and other locations in call store. One stores variable data into the active path and AUXPPM area of the TCR. This is used, for example, to set the active path and return offset and data bits prior to execution of a subroutine. The other contains a 3-bit index which allows seven different functions. Present capabilities provide for setting a bit in the TCR, marking certain bits in the terminal memory record (TMR) and the test vertical status block.

5.17 Only two subroutines, each with several entry points, exist in POINT. One issues distribute orders and the other scans. The various entry points are provided for various types of scanning and distributing required during catalog execution and simply build the needed SPN or DTA from data in the TCR.

5.18 PEER, the error recovery progress mark, is not executed directly as part of the peripheral order interpreter but is included in the program for logical reasons only. When POINT fails a TCR, if the caller has not requested that control be returned (PERTN=0), then the BASEPM is changed to PEER and an error code is stored in bits 8 through 12 of TCR word 3.

5.19 When PEER receives control, it first increments traffic counters associated with peripheral failures and then examines the error code to determine the correct action to be taken. In most cases, this simply involves passing the TCR to the failure program using the correct invocation of the FAIL macro. Certain special failures will generate a TTY message. In the case of power cross failures, the failure program is told to leave the offending party high and dry.

5.20 Continuity failures require special processing by PEER. When this type of failure is encountered and the party being scanned was a line, the path involved must be disconnected and the line restored to its line scan point. The line must then be scanned to determine if there was an actual continuity failure (line still off-hook) or if the party simply hung up while the connection was being made (line on-hook). For true continuity failures, error analysis will be informed and the TCR failed. Otherwise, the TCR will be idled and the involved line will simply reoriginate.

PERIPHERAL CATALOG

5.21 The peripheral catalog is the "brain" of the peripheral order subsystem. It contains the sequences which, when interpreted and executed by POINT, result in the peripheral actions necessary for call processing.

5.22 The catalog is divided into three main sections. The first contains the general purpose sequences, known as GPSEQ. These are common sequences used by other special purpose catalog sequences and fall into two categories. The true subroutine sequences which are invoked by a CALL and use a direct return, and larger more complex sequences which are entered via unconditional branches and use the indirect return mechanism. These sequences may also be executed independently.

5.23 The second section consists of special purpose sequences associated with call processing and are known as CPSEQ. These are sequences which are used in specific call processing situations. For example, in a simple intraoffice call, four PACTS will be used. CDPRORG connects the originating line to a dial pulse or TOUCH-TONE® receiver. RING disconnects the receiver connection, connects the calling party to the talk junctor, and connects the called party to a ringer. TALK disconnects the ringing path and cuts through the talking connection. Finally, when the parties hang up, DISC is used to tear down the talk path.

5.24 The third section contains maintenance sequences referred to as MTSEQ. This section contains sequences required to perform the network fabric exercises and make connections using the test vertical.

PERIPHERAL SUBROUTINES

5.25 Peripheral Work Subroutines (PSUBS) are a collection of miscellaneous subroutines used by both the peripheral order programs and other call processing programs. PSUBS provide support for the SETJCTR and DISTRIBUTE macros which are used by call processing programs to issue distribute orders directly. PSUBS contain the SCAN subroutine through which most directed scanning is performed.

PERIPHERAL OPERATION SUBROUTINES

5.26 The Peripheral Operation Subroutines (POPS) contain several "pseudo-subroutines" which perform high level operations. The pseudo operations are named such since they take real time breaks. Several real subroutines are provided for user handling of peripheral errors. In addition, a subroutine is provided for loading the error analysis buffer for any type of peripheral error.

NETWORK QUEUE AND TIMING HOPPER MONITOR

5.27 The network queue and timing hopper monitor is comprised of three sections:

- Network controller queue monitor (QMON)
- Timing hopper monitor (TMON)
- Associated subroutines.

5.28 The network controller queue monitor is given control once every interrupt, immediately after receiver scanning. Its function is to administer the network queues and issue all network orders needed for peripheral processing. Its action is described in the peripheral action example in 4.13 and 4.14.

5.29 The timing hopper monitor (TMON) provides timed real time breaks which are necessary for peripheral order execution and also for digit timing. The mechanics of TMON are common to both functions.

5.30 The TMON is also invoked once every 10-millisecond interrupt to service entries in the timing hopper and invoke POINT for those which have timed out. It also passes to POINT the TCRs which successfully completed network orders earlier in the interrupt. TMON also controls the length of the interrupt by deferring a portion of its workload until the next interrupt if the length of the current one exceeds a particular threshold.

5.31 Each slot in the hopper contains an eight bit TCR number and an eight bit timer number. The slot timers have a resolution of 2.5 milliseconds. This is accomplished in the following manner: When a TCR is placed into the timing hopper by POINT, the high eight bits of the timer contain the duration of the desired real time break in 10 millisecond increments. The value for the low two bits is obtained from the hardware TImer (TI) register and is the number of 2.5 millisecond units which have elapsed since the start of the most recent 10-millisecond interrupt.

5.32 At the beginning of each hopper scan, TMON obtains a similar value which indicates the amount of time which has elapsed since the beginning of the current interrupt and updates the value stored previously. It then computes the value to be used in decrementing the slot timers by subtracting the value stored during the previous scan from the current value plus ten.

5.33 The reason for this is that in the 3A CC the timed interrupt is precisely triggered from the TI register. However, base level blockage by an uninterruptible task or the occurrence of a higher priority interrupt can delay the 10 millisecond slightly. Also, there is a certain variance in the amount of interrupt work performed prior to TMDN invocation. In other words, a TCR being in the timing hopper for three interrupts does not assure that it has been there for 30 milliseconds. It may have entered the hopper toward the end of a lengthy interrupt and timed out toward the beginning of a short interrupt. The value obtained from the TI is not the time past the point when the interrupt program began execution, but rather, it is the time past the point when the TI triggered the 10-millisecond interrupt. Thus, all delays are accounted for when the decrement is computed and the timing provided is assured of not being less than that duration requested.

5.34 Various subroutines are included which construct network orders, get the time past start of an interrupt, and related operations required to administer the timing hopper functions.

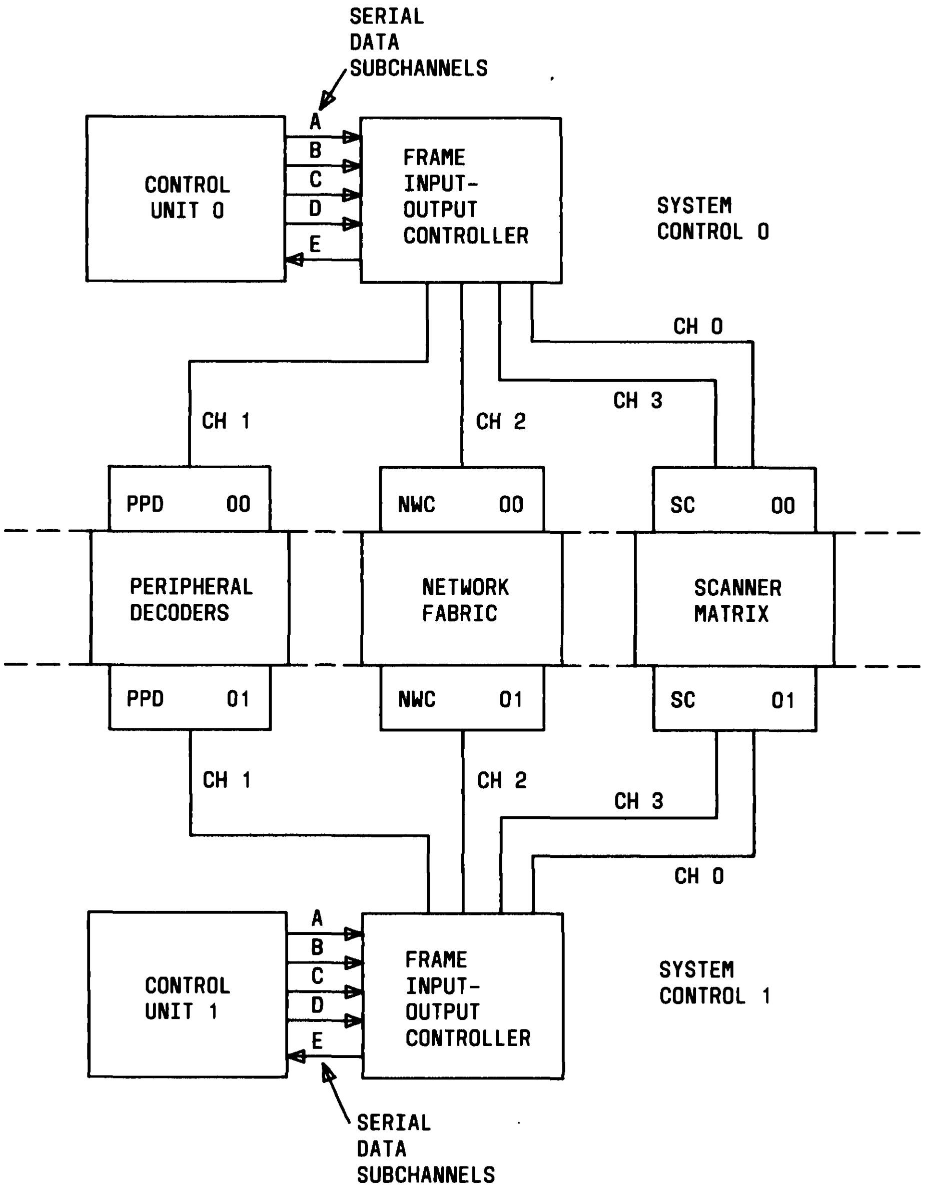
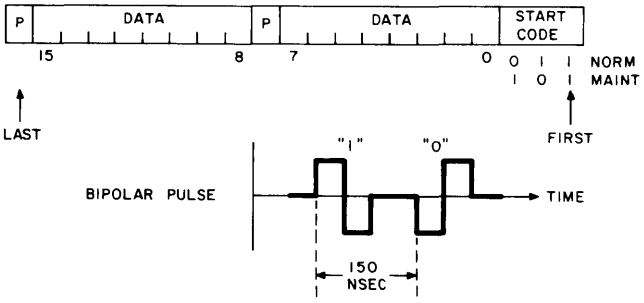
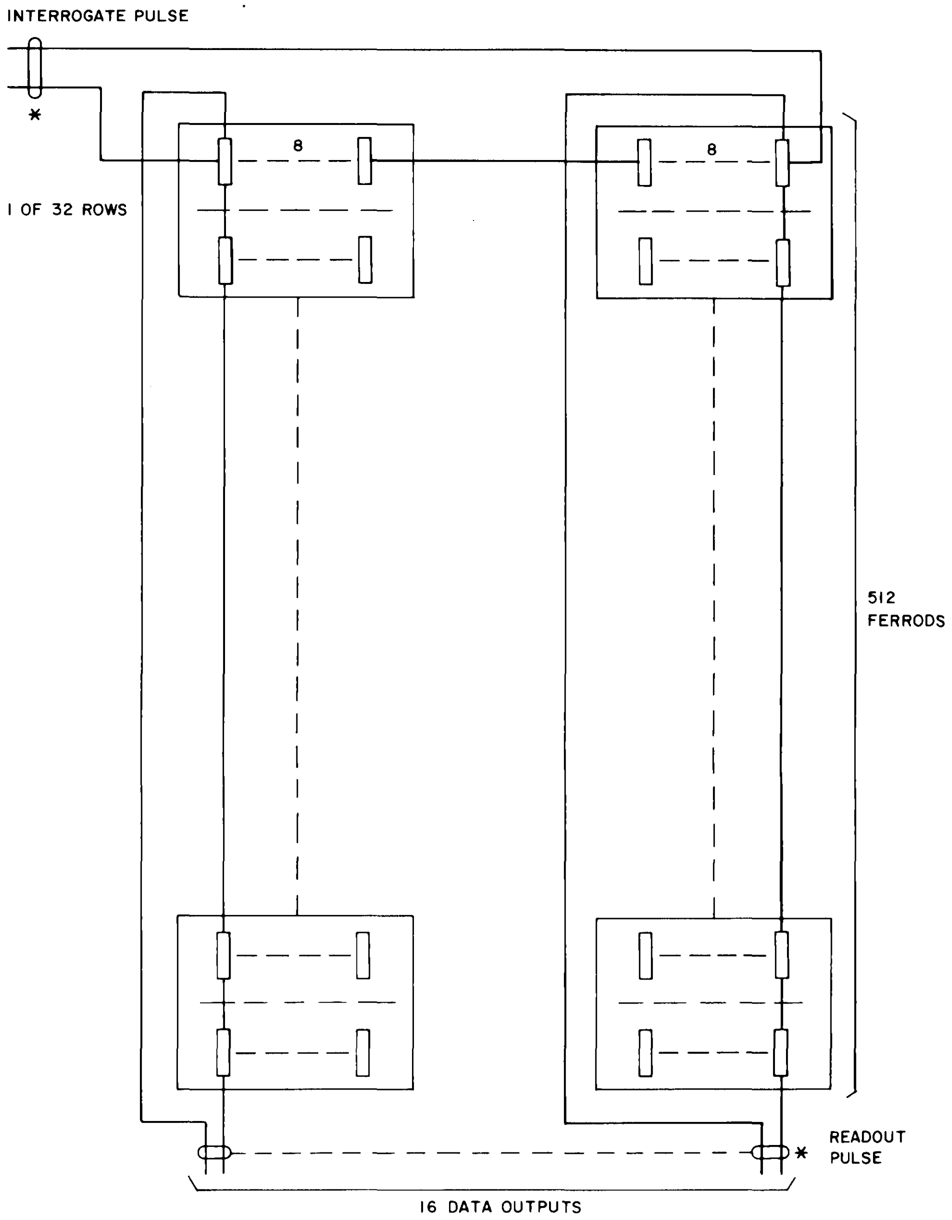


Fig. 1—▶Single Control Frame Office Configuration◀



- CC SENDS MESSAGE FOLLOWED BY A STREAM OF ZEROS UNTIL A REPLY IS RECEIVED
- PERIPHERAL CIRCUIT USES THE STREAM OF ZEROS TO CLOCK THE REPLY BACK
- REPLY FORMAT IS IDENTICAL TO MESSAGE FORMAT

Fig. 2—No. 3 ESS Data I/O Message Format



* MULTIPLE TO SYC-0 AND SYC-1

Fig. 3—Ferrod Array Interfaces

ORDER	REPLY FORMAT	ORDER FORMAT																ORDER TYPE
			ORDER				LS		ROW									
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
12	A	0	1	1	0	0	V	V	V	H	H	H	H	A	A	A	A	NORMAL SCAN
0	A	0	0	0	0	0	V	V	0	X	X	X	X	A	A	A	A	ALL ZEROS
0	A	0	0	0	0	0	V	V	1	X	X	X	X	A	A	A	A	ALL ONES
3	B	0	0	0	1	1	V	V	V	H	H	H	H	A	A	A	A	ASW STATUS
3	B	0	0	0	1	1	V	V	V	H	H	H	H	B	B	B	B	<1 HORIZ
3	B	1	0	0	1	1	V	V	V	H	H	H	H	A	A	A	A	<1 VERT
10	B	0	1	0	1	0	V	V	V	H	H	H	H	A	A	A	A	>1 HORIZ (HHHH ≠ 0000)
5	B	0	0	1	0	1	V	V	V	H	H	H	H	A	A	A	A	>1 VERT (VVV ≠ 000)
6	C	X	0	1	1	0	X	X	X	X	X	X	X	X	X			LOAD (BIT 1 ⟶ A1-FF, BIT 0 ⟶ B1-FF)
9	C	X	1	0	0	1	X	X	X	X	X	X	X	X	X	X	X	TRANSFER

AAAA = ANY COMBINATION OF BITS HAVING EVEN PARITY.

BBBB = ANY COMBINATION OF BITS HAVING ODD PARITY.

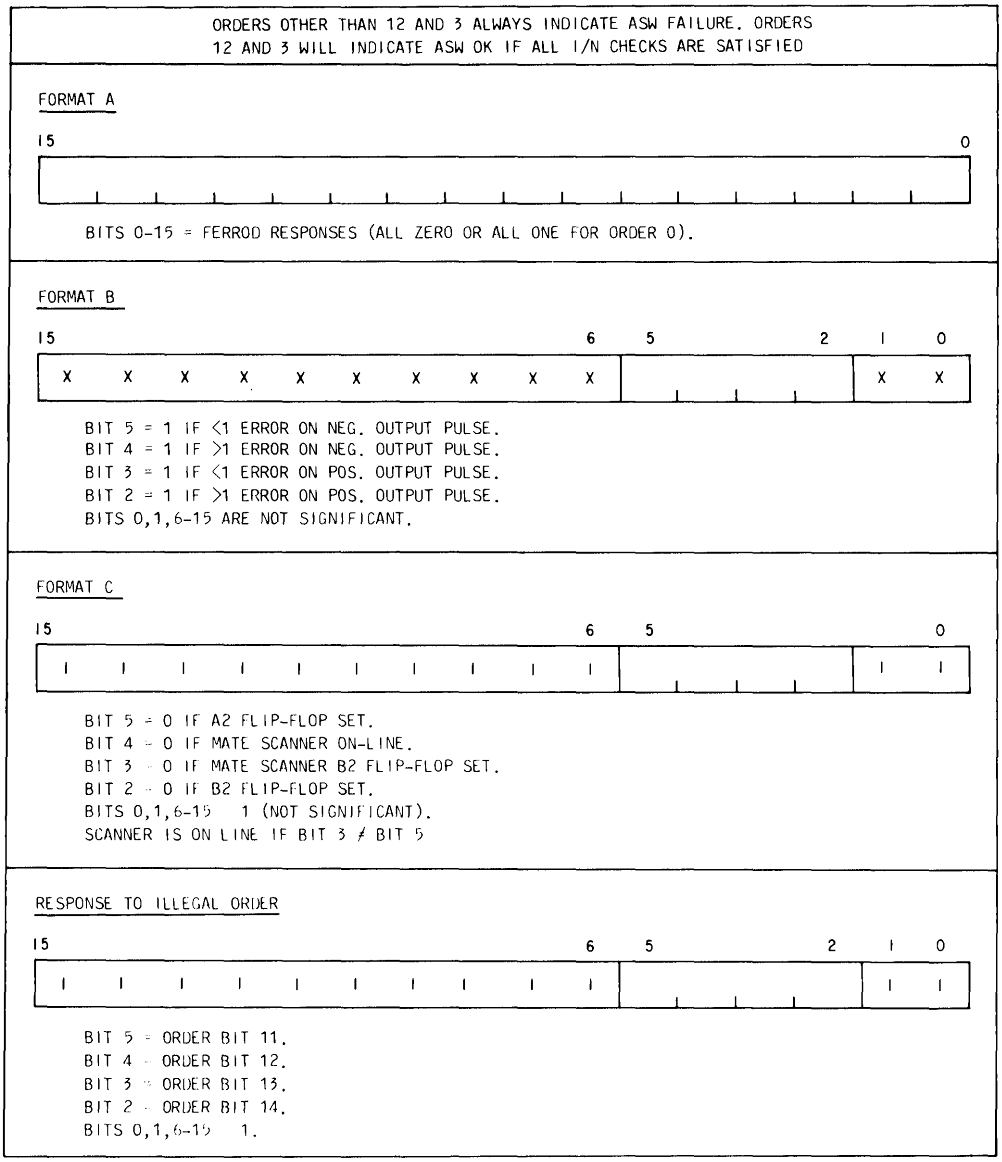
X = NOT SIGNIFICANT

H = INPUT TO 1/16 HORIZONTAL (DR0-15) DECODER.

V = INPUT TO 1/8 VERTICAL (SL0-7) DECODER.

ASW = ALL SEEMS WELL

Fig. 4—Scan Order Formats



ORDERS OTHER THAN 12 AND 3 ALWAYS INDICATE ASW FAILURE. ORDERS 12 AND 3 WILL INDICATE ASW OK IF ALL I/N CHECKS ARE SATISFIED

FORMAT A

BITS 0-15 = FERROD RESPONSES (ALL ZERO OR ALL ONE FOR ORDER 0).

FORMAT B

BIT 5 = 1 IF <1 ERROR ON NEG. OUTPUT PULSE.
BIT 4 = 1 IF >1 ERROR ON NEG. OUTPUT PULSE.
BIT 3 = 1 IF <1 ERROR ON POS. OUTPUT PULSE.
BIT 2 = 1 IF >1 ERROR ON POS. OUTPUT PULSE.
BITS 0,1,6-15 ARE NOT SIGNIFICANT.

FORMAT C

BIT 5 = 0 IF A2 FLIP-FLOP SET.
BIT 4 = 0 IF MATE SCANNER ON-LINE.
BIT 3 = 0 IF MATE SCANNER B2 FLIP-FLOP SET.
BIT 2 = 0 IF B2 FLIP-FLOP SET.
BITS 0,1,6-15 = 1 (NOT SIGNIFICANT).
SCANNER IS ON LINE IF BIT 3 ≠ BIT 5

RESPONSE TO ILLEGAL ORDER

BIT 5 = ORDER BIT 11.
BIT 4 = ORDER BIT 12.
BIT 3 = ORDER BIT 13.
BIT 2 = ORDER BIT 14.
BITS 0,1,6-15 = 1.

Fig. 5—▶Scan Reply Formats◀

			SCANNER				ROW					POINT			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

SPN

			CONCENTRATOR GROUP				INPUT SW GROUP		CONC	INPUT SW			INPUT LEVEL		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TEN

					TRIPLET		PPD	PERIPHERAL DECODER							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

DTA

Fig. 6—SPN/TEN/DTA Formats

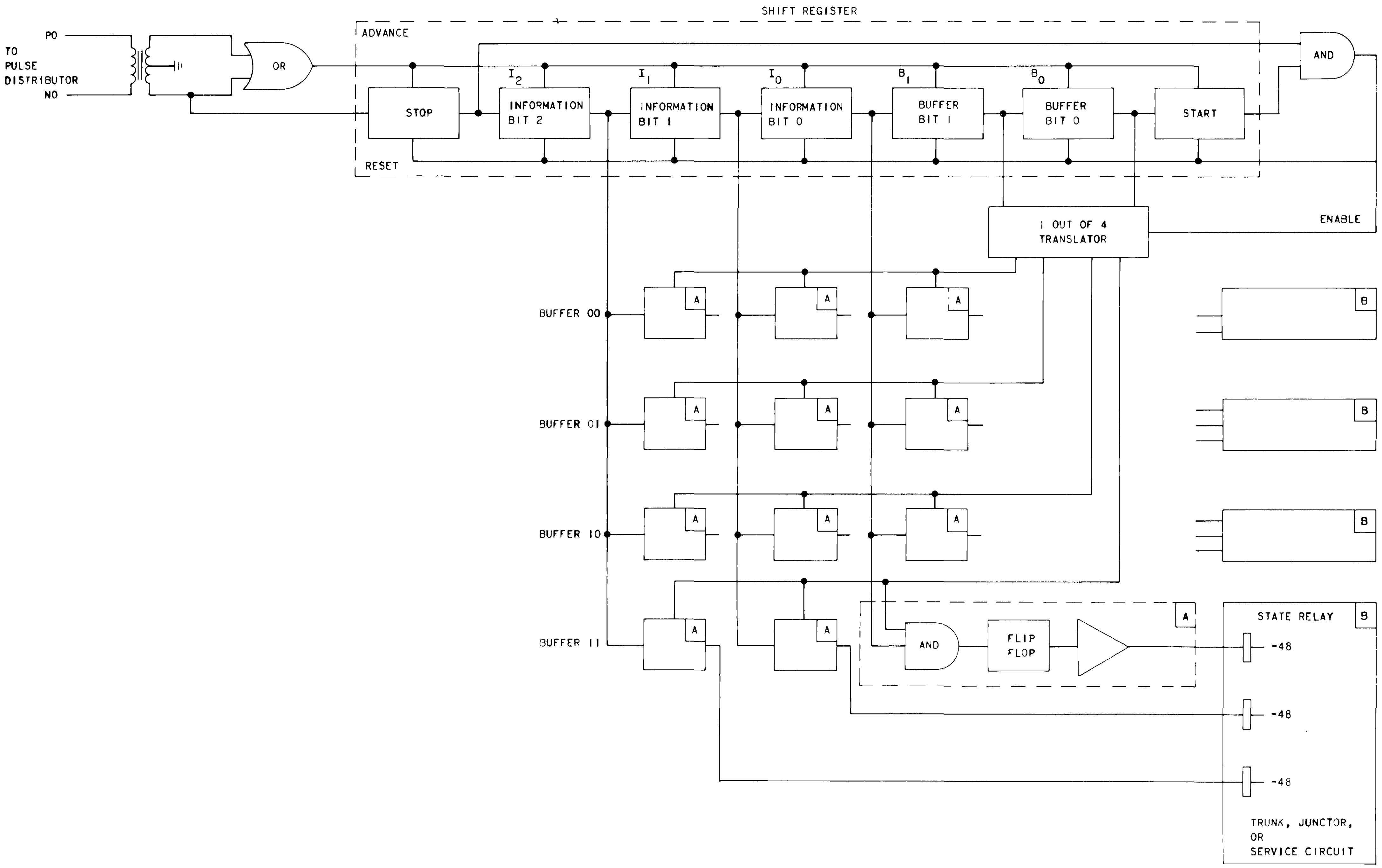
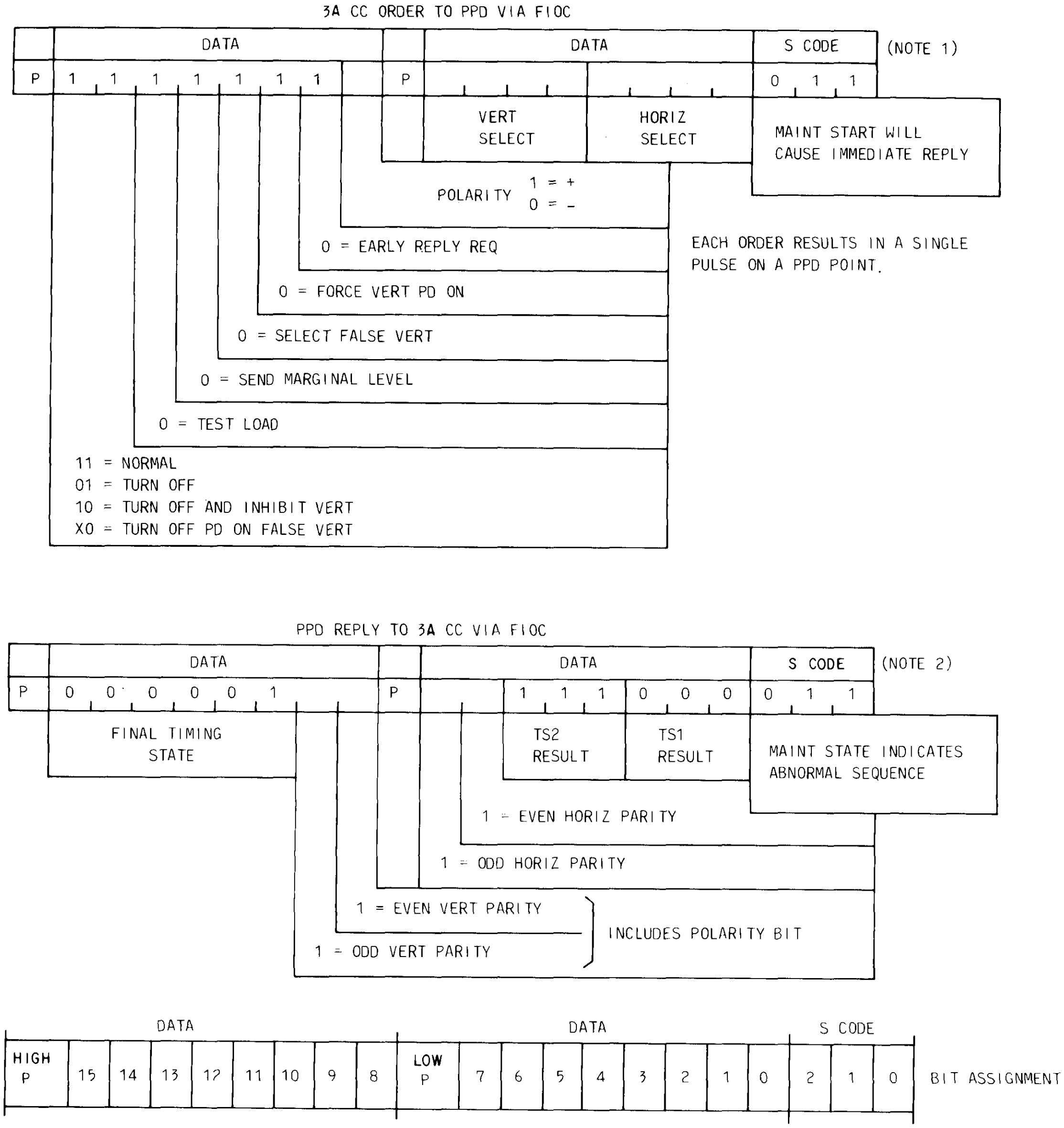


Fig. 7—Peripheral Decoder Functional Diagram



NOTES:
1. THE START CODE IS USED IN THE FIOC TO GENERATE THE DST2 OUTPUT TO THE PPD.
2. THE START CODE IS GENERATED IN THE FIOC USING THE STCT INPUT FROM THE PPD.

Fig. 8—▶Peripheral Pulse Distributor Data Format◀

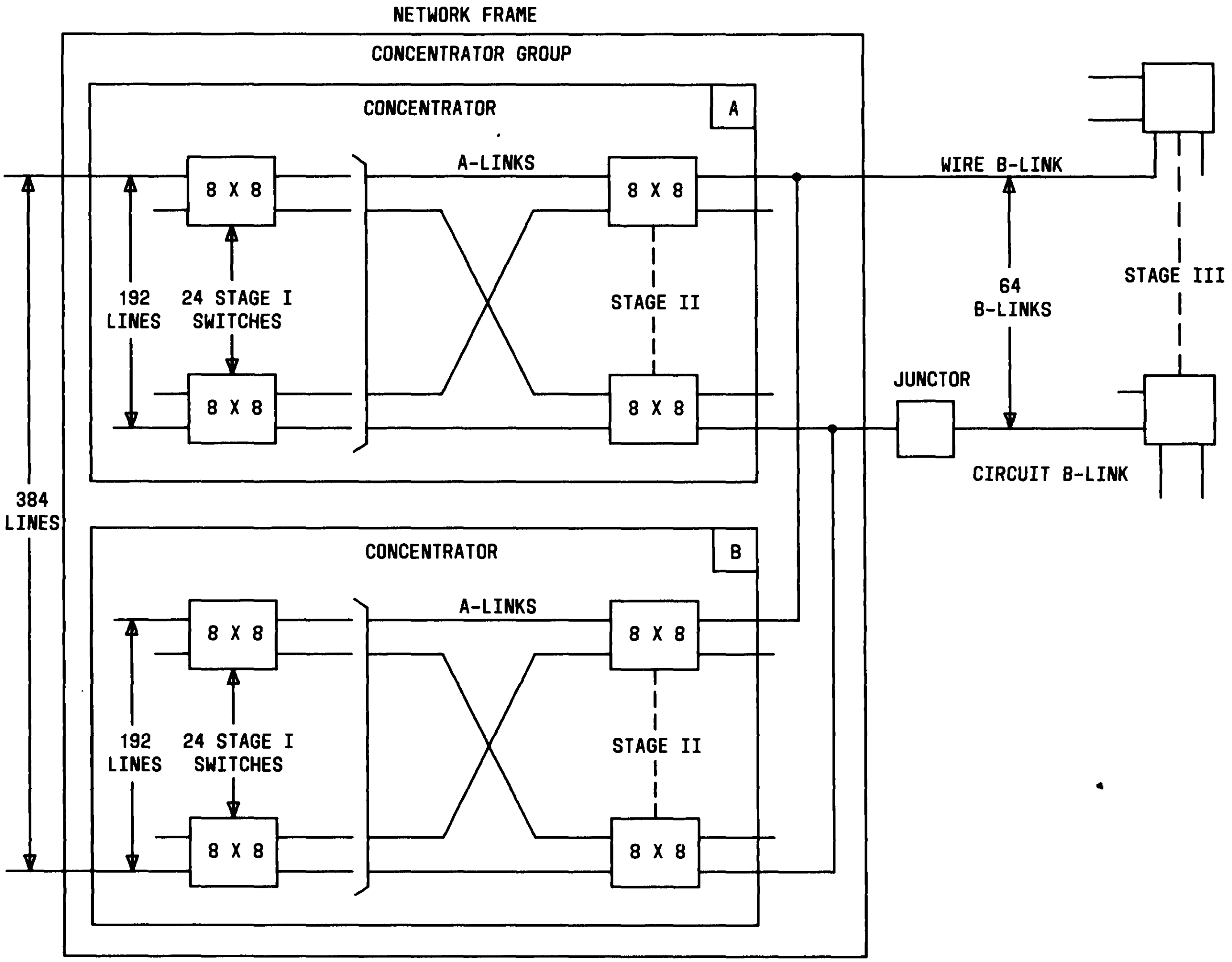
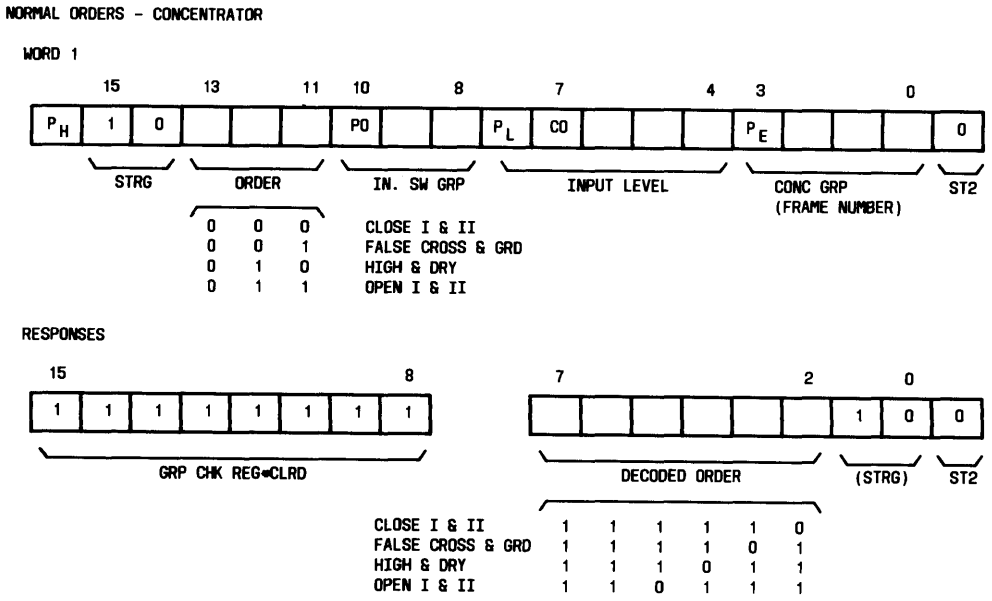


Fig. 9—No. 3 ESS Folded Network



ORDER			
0	0	0	CLOSE I & II
0	0	1	FALSE CROSS & GRD
0	1	0	HIGH & DRY
0	1	1	OPEN I & II

	DECODED ORDER					
CLOSE I & II	1	1	1	1	1	0
FALSE CROSS & GRD	1	1	1	1	0	1
HIGH & DRY	1	1	1	0	1	1
OPEN I & II	1	1	0	1	1	1

Fig. 10—♦Normal Orders—Concentrator (Word 1)♦

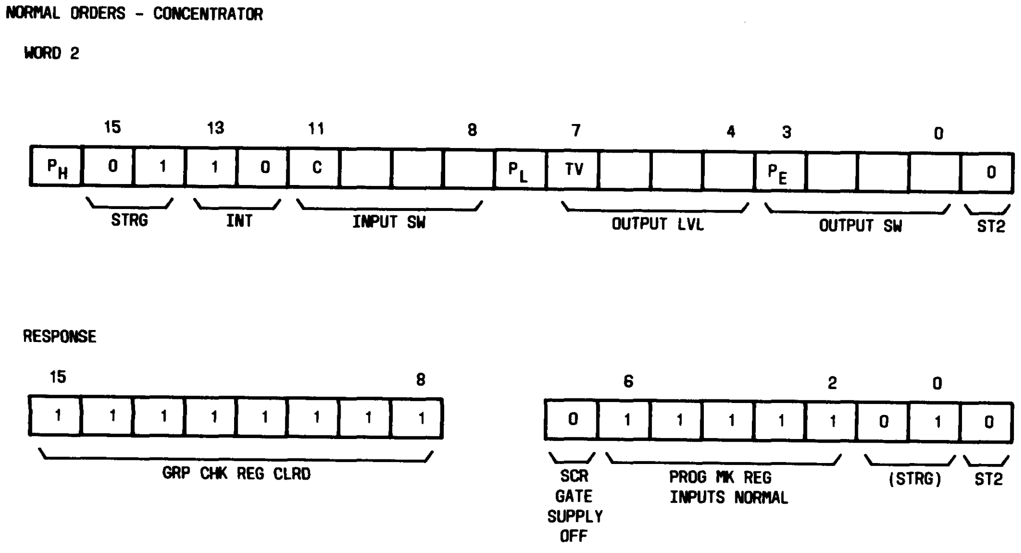


Fig. 11—♦Normal Orders—Concentrator (Word 2)♦

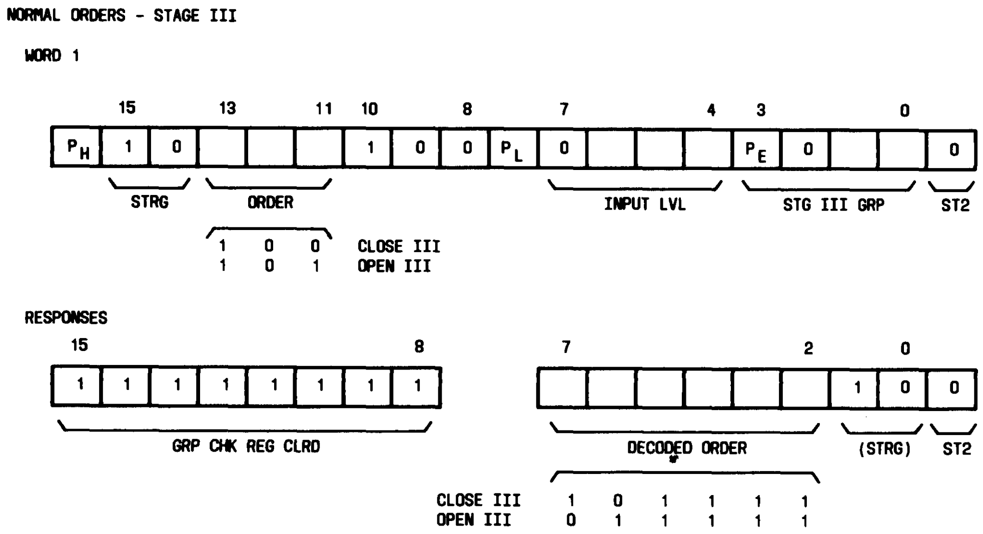


Fig. 12—♦Normal Orders—Stage III (Word 1)♦

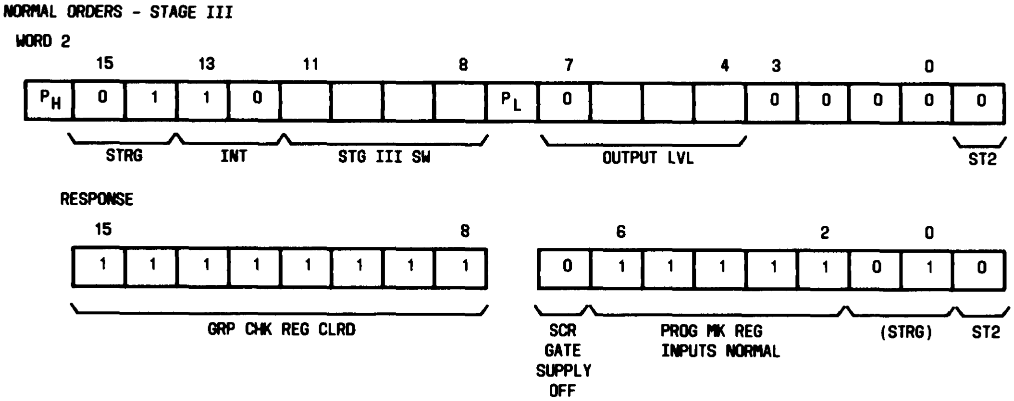


Fig. 13—♦Normal Orders—Stage III (Word 2)♦

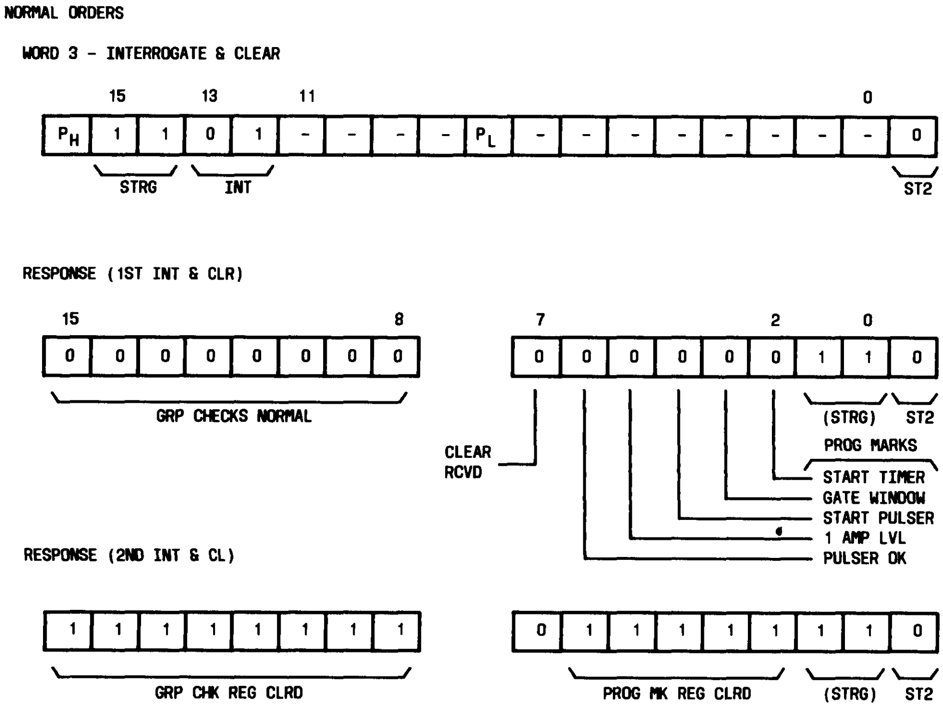


Fig. 14—Normal Orders—(Word 3)

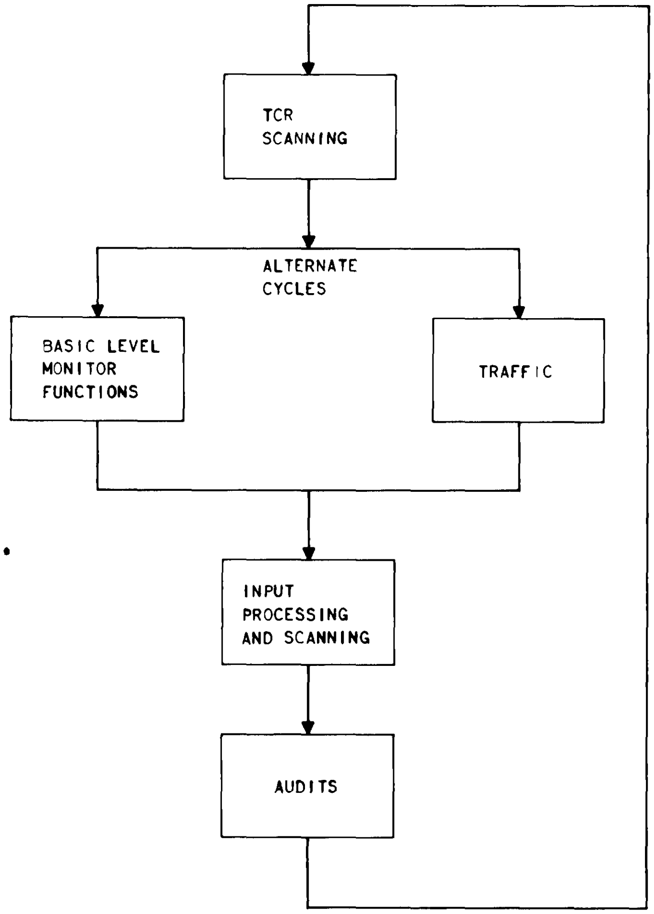


Fig. 15—Base Level Loop

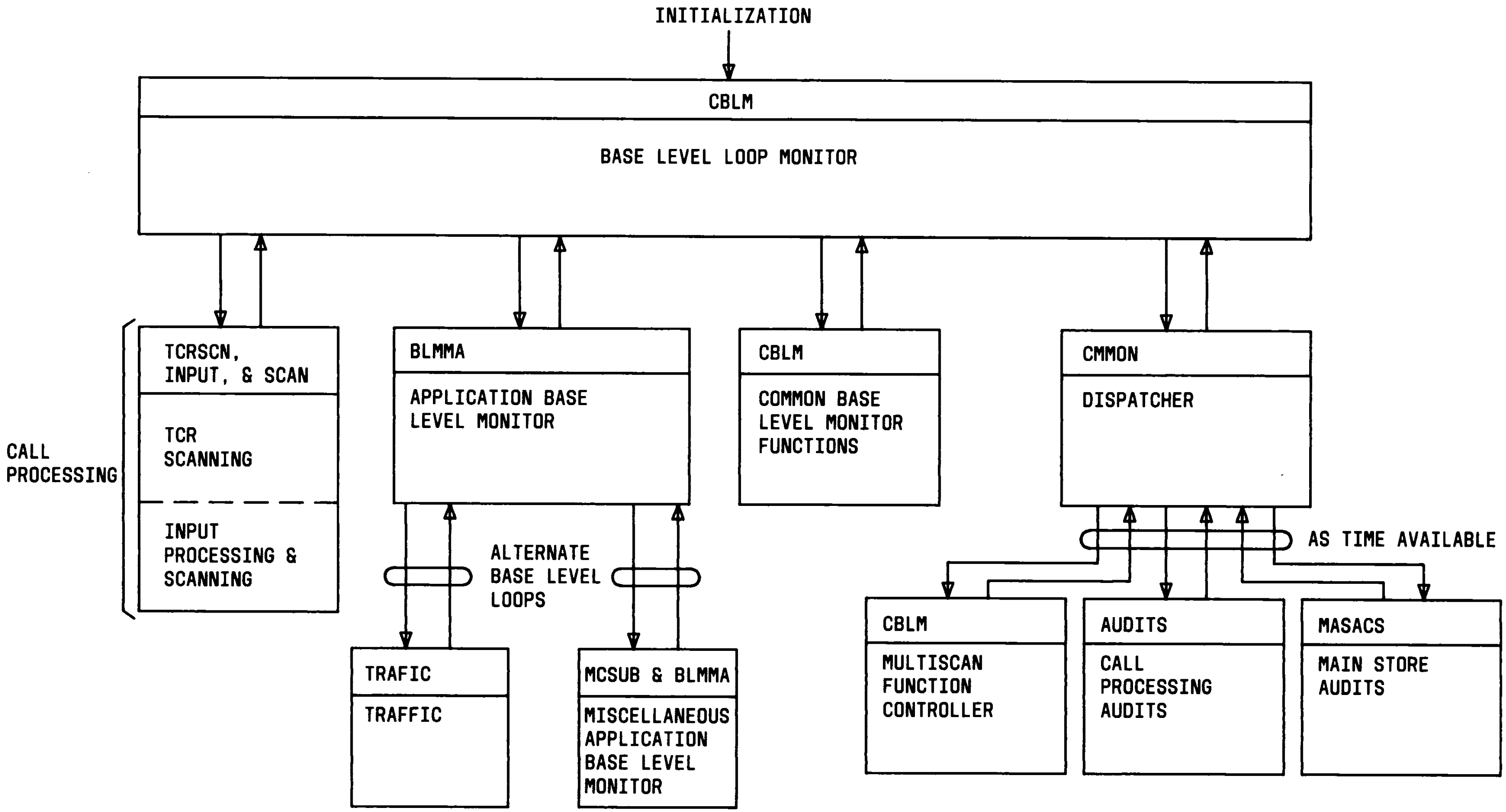


Fig. 16—◗The 3E3 Base Level Loop◖

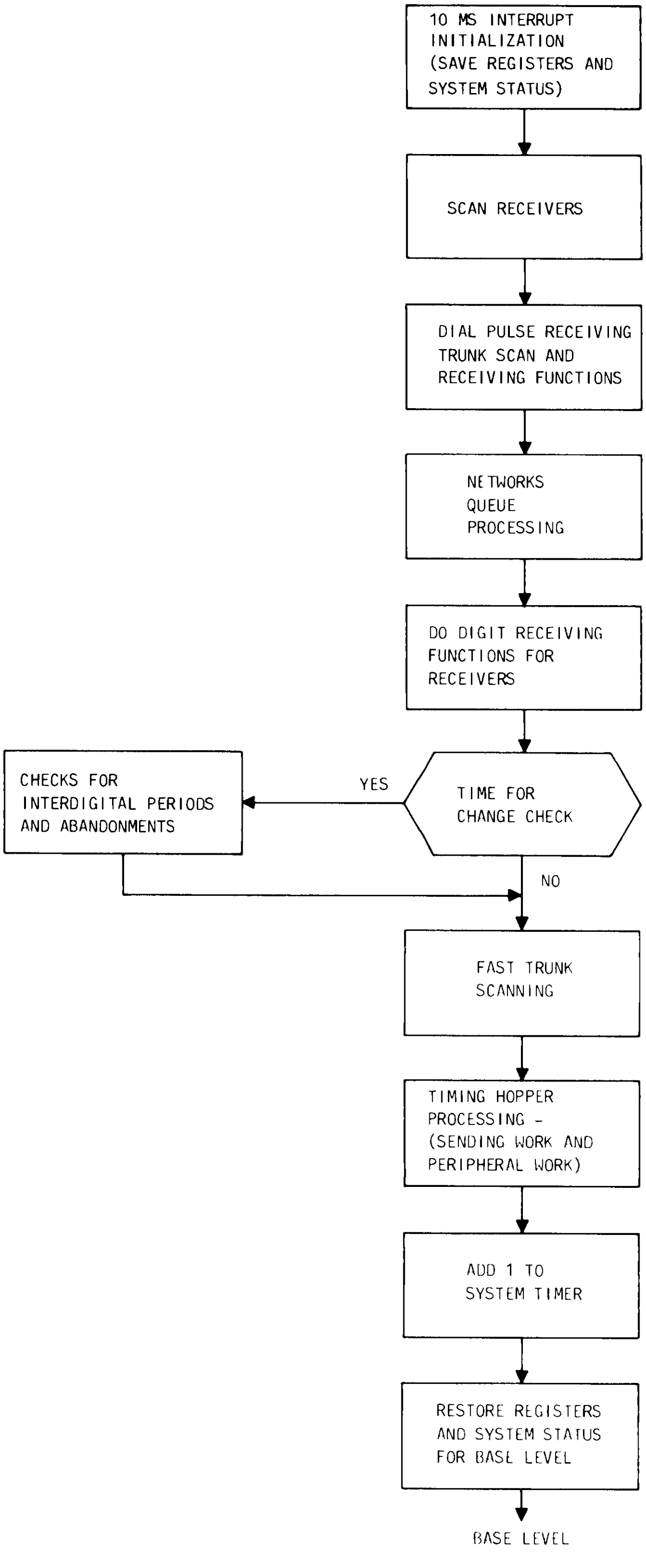

10 MS INTERRUPT INITIALIZATION (SAVE REGISTERS AND SYSTEM STATUS)
SCAN RECEIVERS
DIAL PULSE RECEIVING TRUNK SCAN AND RECEIVING FUNCTIONS
NETWORKS QUEUE PROCESSING
DO DIGIT RECEIVING FUNCTIONS FOR RECEIVERS
TIME FOR CHANGE CHECK
YES
CHECKS FOR INTERDIGITAL PERIODS AND ABANDONMENTS
NO
FAST TRUNK SCANNING
TIMING HOPPER PROCESSING - (SENDING WORK AND PERIPHERAL WORK)
ADD 1 TO SYSTEM TIMER
RESTORE REGISTERS AND SYSTEM STATUS FOR BASE LEVEL
BASE LEVEL

Fig. 17—Timed Interrupt

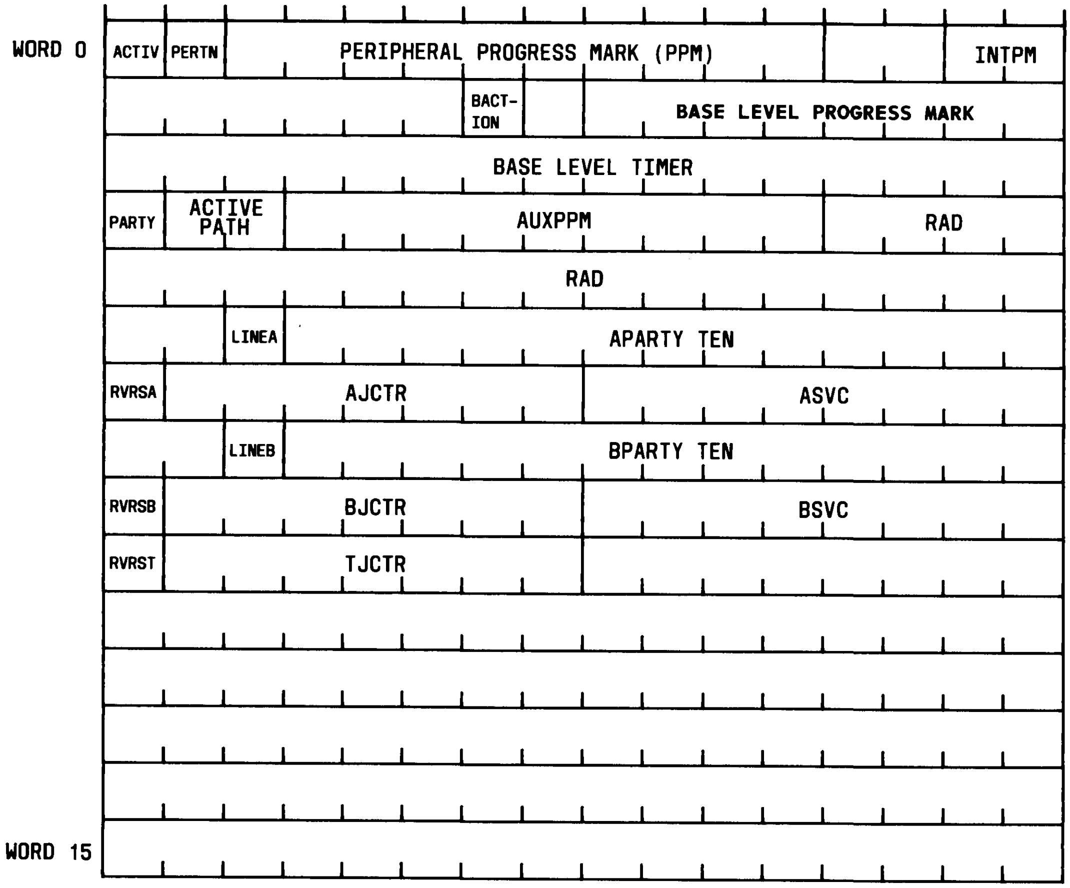


Fig. 18—Transient Call Record

NETQ -005- 01 #

00	QACTV	QSKIP	QEMPTY	QFULL	QHEAD				QACTV	QSKIP	QEMPTY	QFULL	QHEAD NEXT AVAILABLE SLOT			
01		QVRFY	QRTRY		QTAIL					QVRFY	QRTRY		QTAIL OLDEST ENTRY			
02	VERIFY SLOT								VERIFY-SLOT NETWORK VERIFY SLOT							
03	QUEUE SLOT								QUEUE-SLOT 16 SLOTS							
04	QUEUE SLOT								QUEUE SLOT							
05	QUEUE SLOT								QUEUE SLOT							
06	QUEUE SLOT								QUEUE SLOT							
07	QUEUE SLOT								QUEUE SLOT							
08	QUEUE SLOT								QUEUE SLOT							
09	QUEUE SLOT								QUEUE SLOT							
10	QUEUE SLOT								QUEUE SLOT							
11	QUEUE SLOT								QUEUE SLOT							
12	QUEUE SLOT								QUEUE SLOT							
13	QUEUE SLOT								QUEUE SLOT							
14	QUEUE SLOT								QUEUE SLOT							
15	QUEUE SLOT								QUEUE SLOT							
16	QUEUE SLOT								QUEUE SLOT							
17	QUEUE SLOT								QUEUE SLOT							
18	QUEUE SLOT								QUEUE SLOT							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Fig. 19—Network Controller/Queues

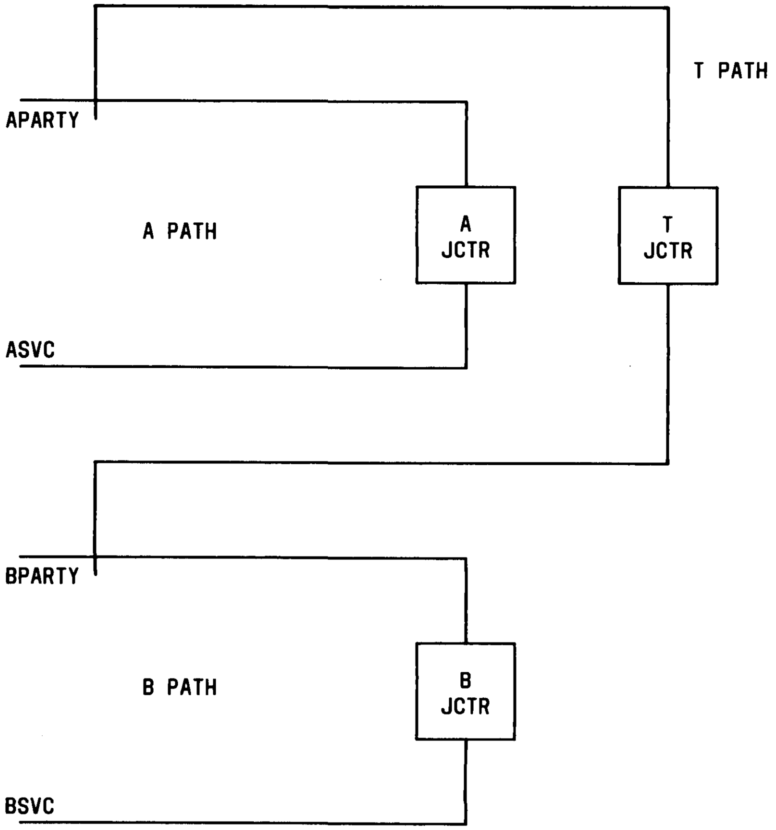


Fig. 20—Standard Network Paths

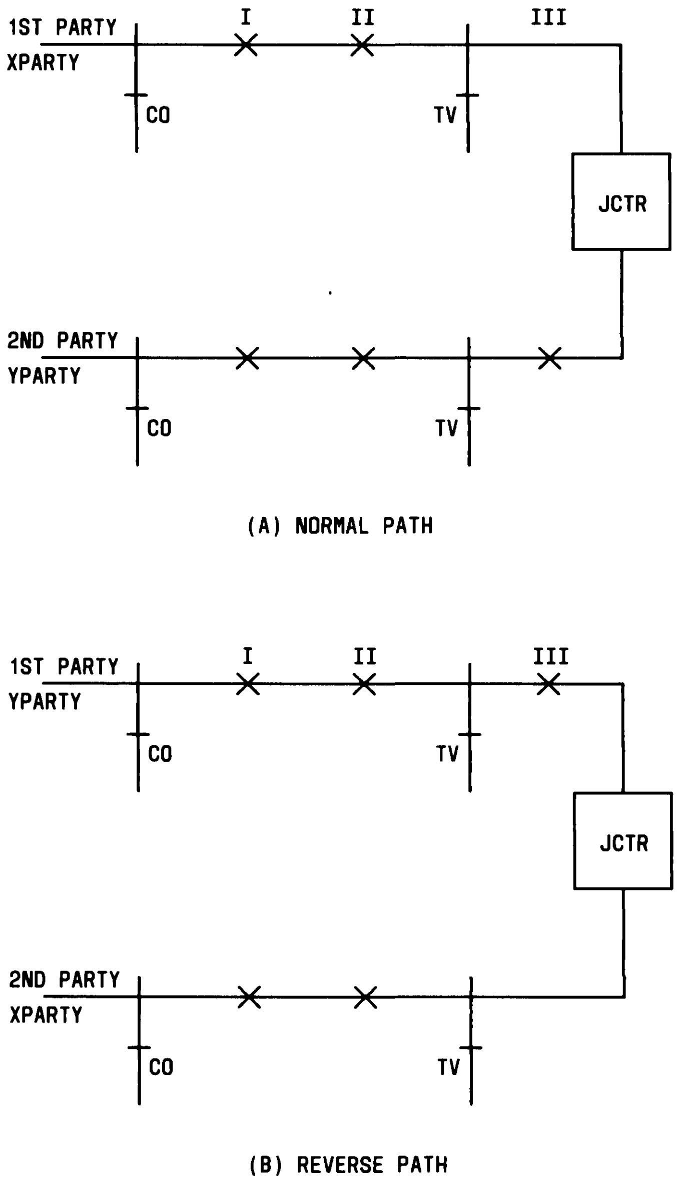


Fig. 21—Normal/Reverse Paths

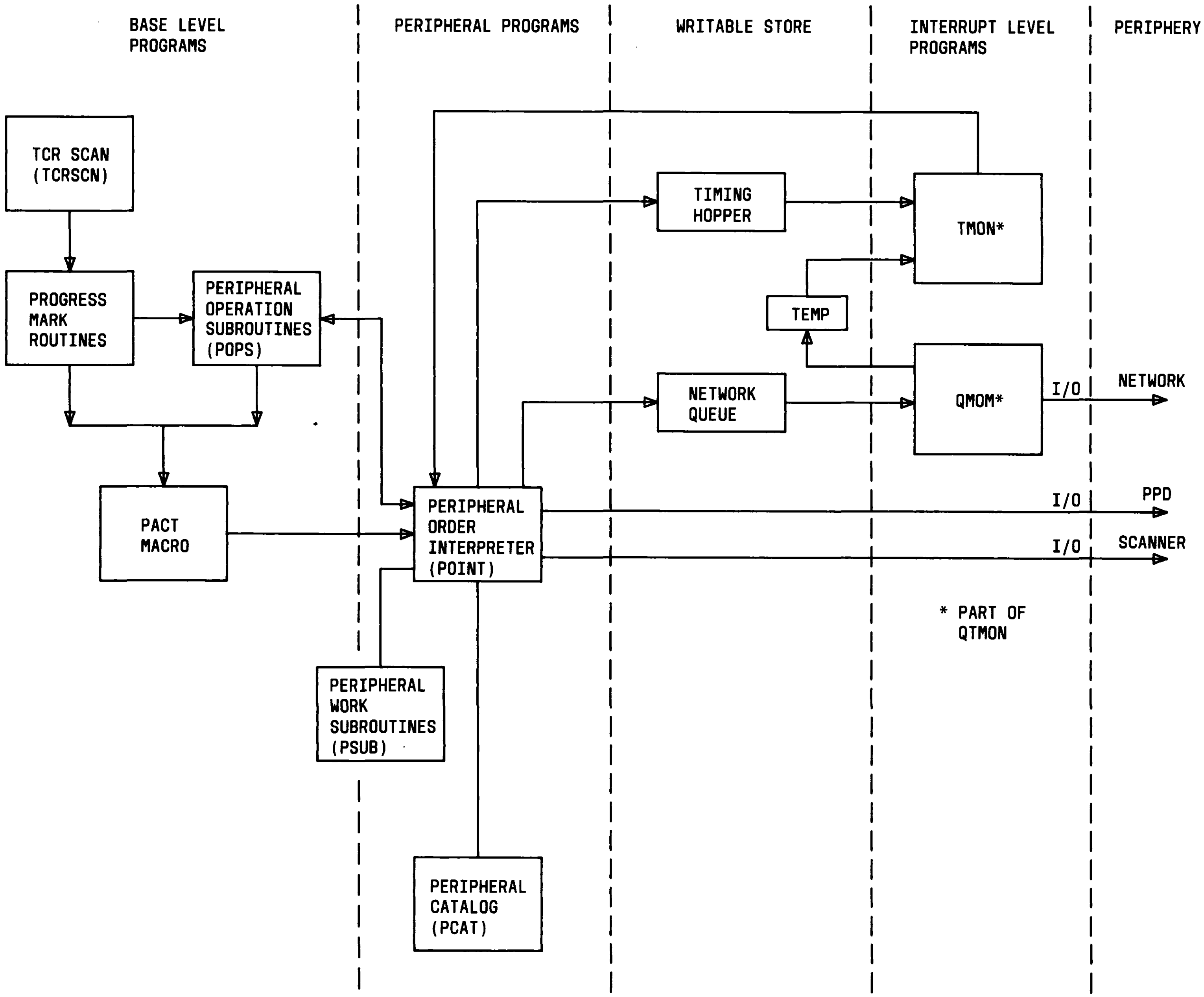


Fig. 22—Peripheral Order Program Organization

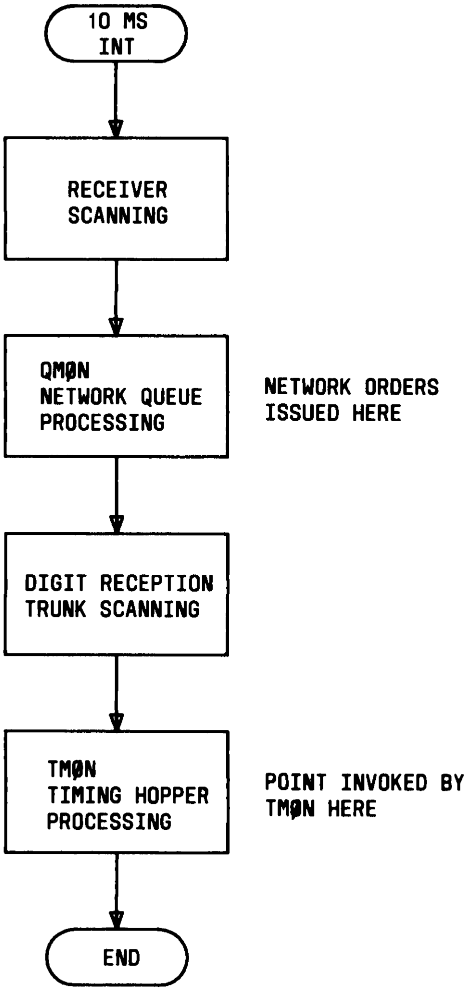


Fig. 23—Interrupt Level Structure

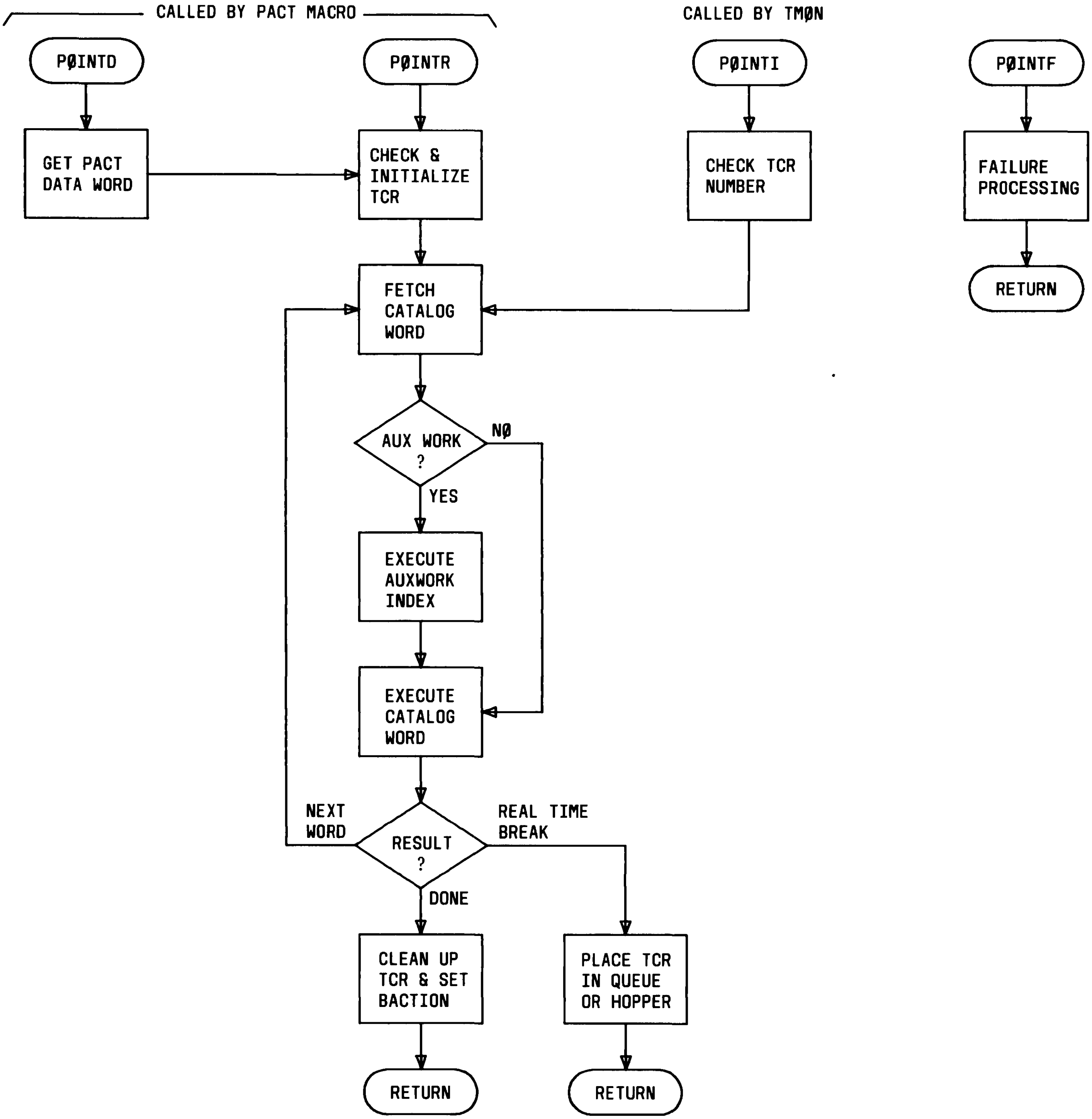


Fig. 24—Peripheral Order Interpreter

VARIABLE DATA										AUX 0	CATALOG INDEX 0-15				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

VARIABLE DATA					AUXWORK INDEX					AUX 1	CATALOG INDEX 0-15				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

VARIABLE DATA											CATALOG INDEX 16-31				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Fig. 25—Catalog Word General Format

TABLE A

ABBREVIATIONS AND ACRONYMS	
BACTION	Base Level Action
CPDR	Customer Dial Pulse Receiver
CTV	Circuit Test Vertical
DTA	Distribute Triplet Address
FCG	False Cross and Ground
JSWN	Junctor Switch Number
MF	Multifrequency
PACT	Peripheral Action
PCAT	Peripheral Catalog
PD	Peripheral Decoder
POINT	Peripheral Order Interpreter
PPD	Peripheral Pulse Distributor
SPN	Scan Point Number
SVC	Service Circuit Number
TCR	Transient Call Record
TEN	Terminal Equipment Number
TI	Hardware TImer
WTV	Wire Test Vertical

TABLE B

EXAMPLE CATALOG SEQUENCE

CATALOG WORD	FUNCTION
CONNECT CPSEQ	PACT Sequence for Simple Talk Path Connect
PATH T	Mark the Talk Path Active
DJT OPEN	Open the Junctor
NET Z	Close the Third Stage
NET OCXYOT, 1PARTY	Cut Through the First Party Side
NET OCXYOT, 2PARTY	Cut Through the Second Party Side
T 1	Delay 10 ms for Antichatter
DJT TALK	Place Junctor Into Correct Talking State
BASE	Return to Base Level

TABLE C

CATALOG INDICES

CATALOG INDICES AND SUBINDICES		
1		Enqueue for Network Order
2		Enqueue for Network Order to Previous Controller
3		Issue a Distribute Order to Active Path Junctor
4		Issue Distribute Order from TCR
5		Issue Distribute Order from TCR (DTA Only)
6		Delay for Specified Number of Milliseconds
7		Terminate Sequence and Return to Base Level
8		No Operation (Auxwork Only)
9		Terminate Recovery Sequence and Return to Base Level
17		Store Data in Auxppm Area
18		Miscellaneous Writes
	1	Mark Heavy Traffic State
	2	Set Bit in TCR
	4	Mark Path Up/Down Bits in TMR
19		Unconditional Branch/Subroutine Call
20		Return from Subroutine (Direct and Indirect)
21		Miscellaneous Branch on Condition
22		Miscellaneous Branch on Not Condition
	1	Check for Idle Test Vertical (and Seize if Idle)
	2	Check for Talk Path Active
	3	Scan Active Path Service Circuit
	4	Scan Both Sides of Active Path Junctor
	5	Scan Party Side of Active Path Junctor
	6	Check for Heavy Traffic State
	7	Scan Power Cross Test Circuit for Failure
	8	Check if Active Party is Precut Line
9		Check for Noisy Line Circuit
25		Branch on Circuit Type Match
26		Branch on Circuit Type Mismatch
27		Branch on Set Bit in TCR
28		Branch on Reset Bit in TCR

TABLE C (Contd)

CATALOG INDICES

AUXILIARY WORK INDICES	
1	Connect FCG Test Circuit
2	Scan and Release FCG Test Circuit (Abort on Failure)
3	Connect PWC Test Circuit
4	Scan and Release PWC Test Circuit (Abort on Failure)
5	Connect RV Test Circuit
6	Scan Line and Release RV Test Circuit
7	Idle the Test Vertical
8	Complement the Party Bit
9	Scan Line Circuit (Abort if On-Hook)
10	Scan Service Circuit (Abort if On-Hook)
11	Scan Both Sides of Junctor (Abort if On-Hook)
12	Scan Party Side of Junctor (Abort if On-Hook)
13	Complement the Active Path and Party
14	Reset Party Bit (First Party)
15	Set Party Bit (Second Party)
16	Abort Origination if Test Vertical Unavailable
17	Seize Test Vertical, Abort if Unavailable