Copyright © 1982 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 4, April 1982
Printed in U.S.A.

No. 10A Remote Switching System:

Host Software

By D. W. BROWN, J. J. DRISCOLL, and F. M. LAX

(Manuscript received April 29, 1980)

The Remote Switching System (RSS) *is a distributed control system which has the call-processing control in the host* ESS. *This design provides the capability of easily giving* RSS *lines the same features that are available to host* ESS *lines. A significant amount of new software is required in the host* ESS *to provide the control of the 10A* RSS. *This article describes the* RSS *host call-processing functions and the administrative and message handling software necessary to provide this control.*

I. INTRODUCTION

A local Electronic Switching System (ESS) provides the call control for a 10A Remote Switching System (RSS). The 10A RSS acts as a slave executing orders sent to it from the host ESS and reports events, such as line originations, to the host. A major advantage of this type of distributed control is that the complex tasks of call processing can use existing host software and share host equipment and trunking facilities. This sharing of host ESS software provides the capability of easily providing RSS lines with the sophisticated features that are offered to host ESS lines.

Figure 1 shows the system configuration consisting of a host ESS office, a 10A RSS (remote terminal), a data link controller, interconnecting voice channels, and data links. The data links are used for communication between the host ESS and the 10A RSS, and provide the means by which orders from the host are transmitted to the 10A RSS and acknowledgments are returned to the host. Voice channels are used to provide the RSS lines with access to the host network and are selected dynamically. In the No. 1 ESS RSS host implementation, the

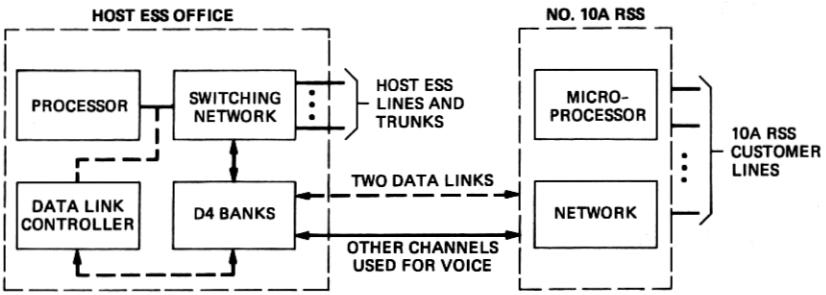

Fig. 1—Remote switching system configuration using digital carrier.

data link controller is a Peripheral Unit Controller/Data Link (PUC/DL).

The host ESS software structure for RSS is influenced by five basic system requirements:

(*i*) All existing host ESS line features should be capable of being provided to RSS lines with comparable service performance.

(*ii*) The effects on the host capacity to process non-RSS calls should be minimized.

(*iii*) Although the first RSS development uses a No. 1 ESS for a host, the design should be portable to other local ESSs with minimum development.

(*iv*) New host features should be able to be provided to RSS lines with minimum development.

(*v*) The 10A RSS hardware and firmware must be identical for all host machines.

The first requirement is met by making maximum use of the existing host call processing, translation and administration programs, with modifications wherever necessary. This also tends to minimize the overall development since a major portion of these programs are independent of RSS. Modifications, where necessary, are done in a manner to minimize the cost in processor real time to non-RSS calls and functions [Requirement (*ii*)].

The reduction of the cost of development over different ESS machines is realized by recognizing that a major portion of any software development is spent on requirements, planning, design, and testing, with a lesser portion of the time spent on the actual program coding. Fundamentally, all the ESS machines have similar software facilities to perform switching functions, although the method of implementation may differ. These factors are used in the RSS by producing host software requirements and designs that strive for machine independence to maximize portability between different host machines. Since the programs are functionally equivalent, higher level test plans are also portable between the machines.

This paper describes the RSS host software architecture, call-processing functions, and the database administration and integrity facilities. The host software structures to provide RSS resource administration and to handle RSS messages are also described.

Although most of the designs and functions are discussed from a host-independent viewpoint, the data linking and message handling structures discussed in Section V apply specifically to the No. 1 ESS design. This is because the input/output (I/O) structure of the different local ESSs are quite different, and the data link controller used in No. 1 ESS (PUC/DL) will differ from other systems. However, the data link protocol and the message structures are the same for all systems so that the same 10A RSS is used for all host ESSs.

II. RSS HOST SOFTWARE ARCHITECTURE

An overview of the major components of the RSS host software is given in Fig. 2. The design of the software architecture is such that all software was put into one of the following three categories:

(*i*) Hooks in existing programs—These are basically decision functions in the host programs that make a decision concerning some aspect of the 10A RSS feature. They are required in host programs that are shared by RSS where a special action is required for RSS lines. For example, in Fig. 2 hooks are required at appropriate places in the host

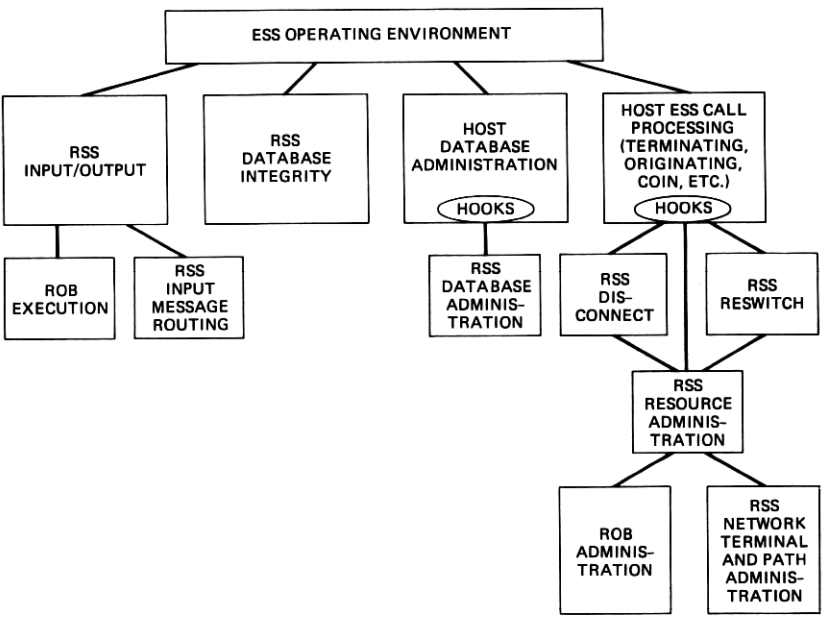

Fig. 2—Remote switching system host software architecture overview.

call-processing programs to invoke routines to perform RSS administrative functions. These hooks are coded in the host ESS standard language and the location of these hooks is quite dependent on the host software structure; thus, they are not directly portable to other hosts. However, high-level structure charts and functional descriptions may be portable since the same RSS actions are generally required in a call sequence independent of the host.

(*ii*) Unique RSS, host-dependent software—This is host software that performs unique RSS functions but whose design is different for different host ESSs. In Fig. 2, the RSS I/O and Remote Order Buffer (ROB) administration functions fall into this category. Software may be put into this category to utilize some aspect of the existing host system or to save on host resources, such as real-time. Structure charts and functional descriptions are highly portable to other hosts; however, pseudocode and state diagrams normally are not.

(*iii*) Unique RSS, host-independent software—This is host software that performs unique RSS functions whose implementation is host-independent since they only interact with the host 10A RSS database and are not dependent on the host operating system or database. This class of programs is highly portable to other host machines, even at a detailed level. The structure charts, flow charts, functional descriptions state diagrams, and even the pseudocode generated during the development process should be portable. To increase the portability of this code to other hosts, these programs were written in a high-level language. The RSS Network Terminal and Path Administration programs in Fig. 2 fall into this category.

The host software components shown in Fig. 2 are discussed in the following sections.

III. RSS RESOURCE ADMINISTRATION

The processing of RSS calls requires the allocation and management of resources that are physically located at the 10A RSS or shared between the host ESS and the 10A RSS. These resources include the channels that interconnect the host and 10A RSS, receiver off-hook (ROH) tone circuits located at the 10A RSS and the 10A RSS network crosspoints. Also, the status of RSS lines is maintained at the host. Several factors were considered in deciding whether to place these functions in the 10A RSS or in the host ESS. These factors are

(*i*) The effect on service because of the additional time delay if the 10A RSS has to be interrogated to determine line status, and to hunt voice channels and network paths.

(*ii*) The additional software development required if the host ESS programs have to take a real-time break to interrogate the 10A RSS to obtain a line's status. Host software is structured around data that are accessed without taking a real-time break.

(*iii*) The duplication of development effort that is required to provide the same functions in several host ESS systems.

Factor (*iii*) indicates that the overall development effort would be reduced by placing the line, channel and 10A RSS network path administration in the 10A RSS. However, the service criteria and the effect on the existing host software were judged to be more important; therefore, these functions are allocated to the host ESS.

The overall development effort is reduced by making the program and data structure designs independent of the host ESS. Thus, they are highly portable between ESS machines. This section describes the data structures and programs required to administer the RSS resources.

3.1 Data structures

Data structures are required in the host ESS memory (call store) to record the status of the RSS facilities and to provide for their administration. These data structures are contained in one contiguous memory block, called the RSS Path Memory Block. To simplify the engineering of the office, the RSS Path Memory Block is provided in one of three sizes corresponding to three basic network sizes of the 10A RSS. Each of the substructures in the Path Memory Block is described below.

3.1.1 Network map

The network map is only provided in one size (for a fully equipped 10A RSS network) and contains a status bit for each A-link and junctor in the 10A RSS network.

3.1.2 Path memory remote record

A path memory remote (PMR) record is provided for each possible line and channel network appearance. Since the 10A RSS network can be equipped in three different sizes, considerable ESS memory is saved by also providing PMRs in substructure sizes corresponding to the network size. The PMRs contain information about the state of the terminal and a pointer which is used to link the PMR to another PMR or to point to another memory block (call register or path memory for junctor (PMJ) involved in the call.

3.1.3 Path memory for junctor record

A PMJ record is a block of call store that is associated with a junctor in the 10A RSS network and is provided for each equipped junctor. It is used to store path and terminal information when the junctor is in a network path. A PMJ also contains a state and pointer which is used to link to another PMJ or to a call register.

3.1.4 Remote miscellaneous scan point status map

Scan points are provided at the 10A RSS for use in alarms, make-busy keys, and stop hunt keys, etc. The remote unit periodically scans these scan points and reports any changes to the host ESS via the data link. The RSS Path Memory Block contains a scan point status map which has a bit for each possible remote scan point and is updated to indicate the present state of the scan points. Host ESS programs determine the state of a scan point by interrogating the map in the host rather than sending a data link message to the remote terminal.

3.1.5 Channel head cells

The channel head cells contain memory for linking the idle voice channel PMRs onto a one-way linked list, as well as for traffic usage and peg counts. One head cell is provided for the channels terminating on each of the two modules of the 10A RSS.

3.2 Programs

The responsibility for managing the RSS resources resides within the host in the RSS terminal and network administration programs. These programs are designed to provide the functions required by client programs, such as hunting channels, hunting 10A RSS network paths, and fetching or changing the state of a line. In general, the client program is isolated from the data structures since the administration programs provide the interface with the RSS resources. This technique of "data hiding" keeps the data structure access confined to the portable programs.

3.2.1 Network terminal administration programs

The 10A RSS network terminal administration program contains routines for channel hunting, channel idling, changing the state of a terminal (line or channel), and for determining the state of a terminal. These routines have the responsibility for maintaining the state and pointer fields of the line or channel PMR (see Fig. 3) to indicate the status of the terminal.

The PMRs for idle channels are put on a one-way linked list, with the last channel idled at the bottom of the list. Channels are selected from the top of the list, thus providing rotation among channel usage. A linked list is provided for each of the two frames of the 10A RSS (see Section 3.1.5). A channel PMR on an idle link list has the state field (see Fig. 3) set "idle"; the pointer field contains the remote equipment number (REN) of the next channel on the idle linked list. The pointer field of the last PMR contains an "end" code.

The first choice on selecting an idle channel is to select from the same module on which the REN that is to be connected to it appears.

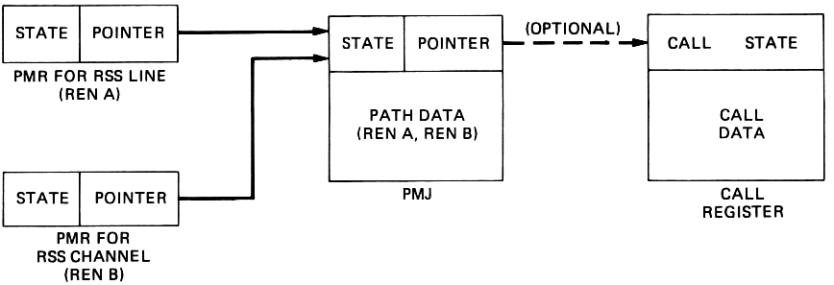

Fig. 3—Two-terminal RSS path memory configuration.

This conserves the usage of the limited number of intramodule junctors. However, if there are no idle channels on the desired module, an attempt is made to find an idle channel on the other module.

3.2.2 Network path administration programs

The main functions of the 10A RSS network path administration programs are to hunt and idle network paths between a given pair of RSS network terminals. In the course of performing these functions, these routines administer three types of data structures: the RSS Network Map, PMRs, and PMJs. A path hunt is performed by selecting an idle junctor and A-links (using the busy/idle status in the network map) so that when the corresponding network crosspoints are closed, a talking path exists between the two given network terminals. This hunt is exhaustive in that it looks for all possible combinations of junctors and A-links that could be used to form a path. The selected junctor and A-links are marked busy in the network map so that they are not selected by the next path hunt request.

A record of the path configuration between two terminals is maintained in the PMRs of the two terminals and the PMJ for the junctor included in the path. Figure 3 shows how the path memory elements are set up. The state field of the PMRs indicates that the PMR is in a path and the pointer fields contain pointers to the PMJ. The usage of the state and pointer fields of the PMJ is similar to that of the PMR. The pointer field may optionally be empty or it may be set up to point to a call register. The state field indicates which situation applies. Other data fields in the PMJ contain the REN of each of the associated terminals in the path. This structure permits the entire path to be reconstructed from the data in the PMRs and PMJ. Path trace routines, included as part of the network path administration programs, accept as input the REN of either terminal and trace through the structure to obtain the other REN and the junctor associated with the path. The identity of the A-links are also known since only one A-link can connect a given network terminal to a given end of the junctor.

More complex RSS network path configurations arise during special switching actions, such as when two RSS lines that are connected together through the ESS network are in the process of being "re-switched-down" (to be described in Section 6.4) so that they are connected solely through the RSS network. The process of linking PMRs and PMJs is extended for these cases to include all the path elements. Figure 4 shows how the elements are linked for a 4-terminal path where two pairs of terminals (A1-C1 and A2-C2) are connected and a reserved path (marked busy in memory but idle in hardware) exists between A1 and A2. As with the two-terminal path, all the RENs and path element identities can be obtained by tracing the structure given any REN associated with the path.

Routines that idle paths perform the reverse function of the path hunt routines. Given any REN associated with the path to be idled, a path trace is performed to obtain all the RENs and the junctor used in the path. The A-link and junctor status bits in the network map, the PMRs, and the PMJ, are then marked idle.

IV. DATABASE ADMINISTRATION

In addition to the generic programs, ESS machines have an extensive database that provides the information necessary for these programs to process calls and perform maintenance and administration functions. These data are called translation data and contain information such as line features, call routing and charging, and equipment configurations. Many functions, such as routing and charging, are not af-

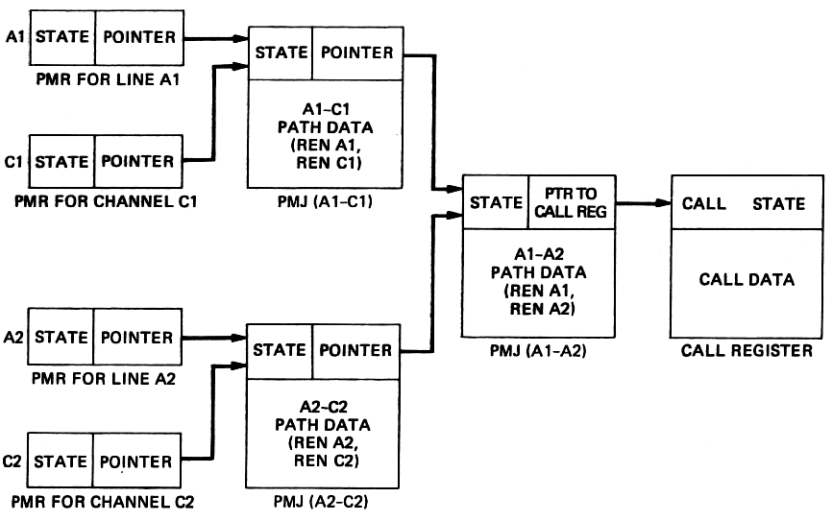

Fig. 4—Four-terminal RSS path memory configuration.

fected by the RSS feature and, therefore, most translation data are not affected either. However, changes and additions to the existing host translation data are required where it is necessary to identify line terminations and scan or distributor points as being RSS items. Also new translators are required for new RSS items. This section describes the modifications that are required for the host ESS translators, the new translators, and the programs to administer them.

4.1 Translations

Translations refers to the office-dependent data that describes the office and customer characteristics, the programs for accessing the data, and the means for changing the data. All translation data are stored in the host ESS and items that are required at the remote terminal are periodically transmitted to it via the data link. Since the host ESS performs all of the call-processing control functions, such as routing, charging and trunk selection, most of the existing translations are not affected by RSS.

The additional translations required for RSS fall into the categories of new translations and modifications to existing translations. The new translators required are the

(*i*) REN translators, which contain the originating translation data for RSS lines. If the REN is associated with an RSS channel, then the translators contain the associated ESS host line equipment numbers.

(*ii*) Remote scan point number (RMSN) translations, which define the scan point usage, are used to determine the action to be taken when a scan point changes state.

(*iii*) Remote terminal (i.e., 10A RSS) translators that contain data on the equipage and stand-alone* features for the RSS.

Both the REN and RMSN translators have corresponding host translators, and the data substructures for the new RSS translators are made identical to the ESS translators. This permits the existing host programs that access and change these translators to be used for the RSS translators with only small modifications. Another important consideration is the requirement that service features available to the host ESS lines be also available to RSS lines. By using the same data structures, this requirement is easily met and new features offered in the future will be available on RSS lines with minimum additional development. It also lessens the impact upon the client call-processing programs since the data returned by the access routines have the same formats. New programs are required to administer the remote terminal translator since it has no ESS counterpart.

---

* Stand-alone refers to the operation whereby the 10A RSS processes intra-RSS calls when communications with the host ESS are lost.

In addition to the new translators, several existing translators require modifications for RSS. An indicator has been added to the directory number translator output to specify whether the called party is an RSS or an ESS line. Several other translators that can contain equipment numbers require indicators to specify whether the equipment number is remote or local. For example, multiline hunt lists can contain a mixture of RSS and ESS lines. Similarly, some translators require an indication that scan and distributor points be specified as either remote or local.

After initial call setup, the host call-processing programs use the line equipment number (LEN) of the ESS end of the voice channel in the call register data area rather than the REN for the line involved in the call. This permits the existing host software to handle RSS calls without making extensive changes. However, since the REN of the RSS line is required for some call actions, such as disconnect and coin functions, a means must be provided to obtain this line REN from the ESS LEN of the channel. This means is provided by way of the LEN translator which contains the channel REN for the RSS channel. The path memory associated with the channel REN can then be traced, as described in Section 3.2.2, to obtain the RSS line REN from the PMJ used in the RSS line to channel path.

Corresponding to the RSS translation data discussed above, the host recent change and service order programs are modified to enable these data to be changed by the operating company. The same recent change message formats used for host lines are also used for RSS lines, with different keywords to indicate items that are RSS-related—for example, LEN is replaced by REN in the No. 1 ESS application. Additional keywords are added where a new item is required, and new messages are provided for the unique 10A RSS translation data. The flexibility of the ESS recent change programs permits this to be easily implemented.

4.2 Remote terminal translation data update

Although the host call-processing programs use the translation data stored in the host, there is a subset of translation data that is required at the remote terminal for use during both normal operation with the host and during stand-alone operation. These data are

(*i*) Multiparty ringing option (ac/dc or superimposed).

(*ii*) Ground start applique circuit number for ground start and coin lines.

(*iii*) Hardware equipage information, such as the network size and channel interface board equipage.

In addition to the above data items, stand-alone operation, whereby the 10A RSS processes intra-RSS calls when communications with the

host ESS have been lost, requires additional data to perform the following translations:

(*i*) Emergency directory number routing data. Up to four, 3-, 6-, or 10-digit directory numbers (such as 911) can be specified for special routing during stand-alone operation.

(*ii*) Terminating directory number translations to yield the REN and type of ringing.

(*iii*) Multiline hunting lists.

(*iv*) Originating line translation data which includes the type of dialing, distributor point assignments and major class. The major class stored at the remote terminal is a subset of the major classes served by the host. It can be individual, two-party, multiparty, coin, manual, or unassigned. The major class of each RSS line, as stored in the host translation database, is mapped into one of these major classes before the data are sent to the 10A RSS.

The 10A RSS copy of an individual subscriber's translation data is updated whenever a recent change on that line is entered into the host. At that time, a translation data update message, which contains all the data pertaining to that line, is sent to the remote terminal. A total update of all remote terminal translation data is done whenever the remote terminal requests it. This is done on a routine basis, once a day, and whenever the remote terminal suspects that the data may have become mutilated. The update can also be requested manually from the host teletypewriter.

V. THE RSS MESSAGE HANDLING

5.1 Hardware overview

Figure 5 is a diagram of the 10A RSS No. 1 ESS host interface and the hardware components involved in the transmission of data between the remote terminal and the host. Communication between the two machines takes place over a pair of low-speed data links that share the same transmission facilities as the voice channels that interconnect the remote terminal with the host. Each data link is placed on a separate transmission facility for reliability. Where carrier facilities are used, the links are assigned to a dedicated voice channel on the carrier system with each link being assigned to a separate carrier terminal.

Both RSS links are 2400 bps synchronous links. The on-line link is active and carries the entire data traffic between the host and 10A RSS, while the off-line link is maintained in a standby state as a spare. The on-line, off-line status of the links is determined by the host office based on error information accumulated by the software responsible for running the links. At the remote terminal, the link is interfaced to the microprocessor through a data link interface circuit (DLI). The DLI provides a small amount of data buffering and performs a number of

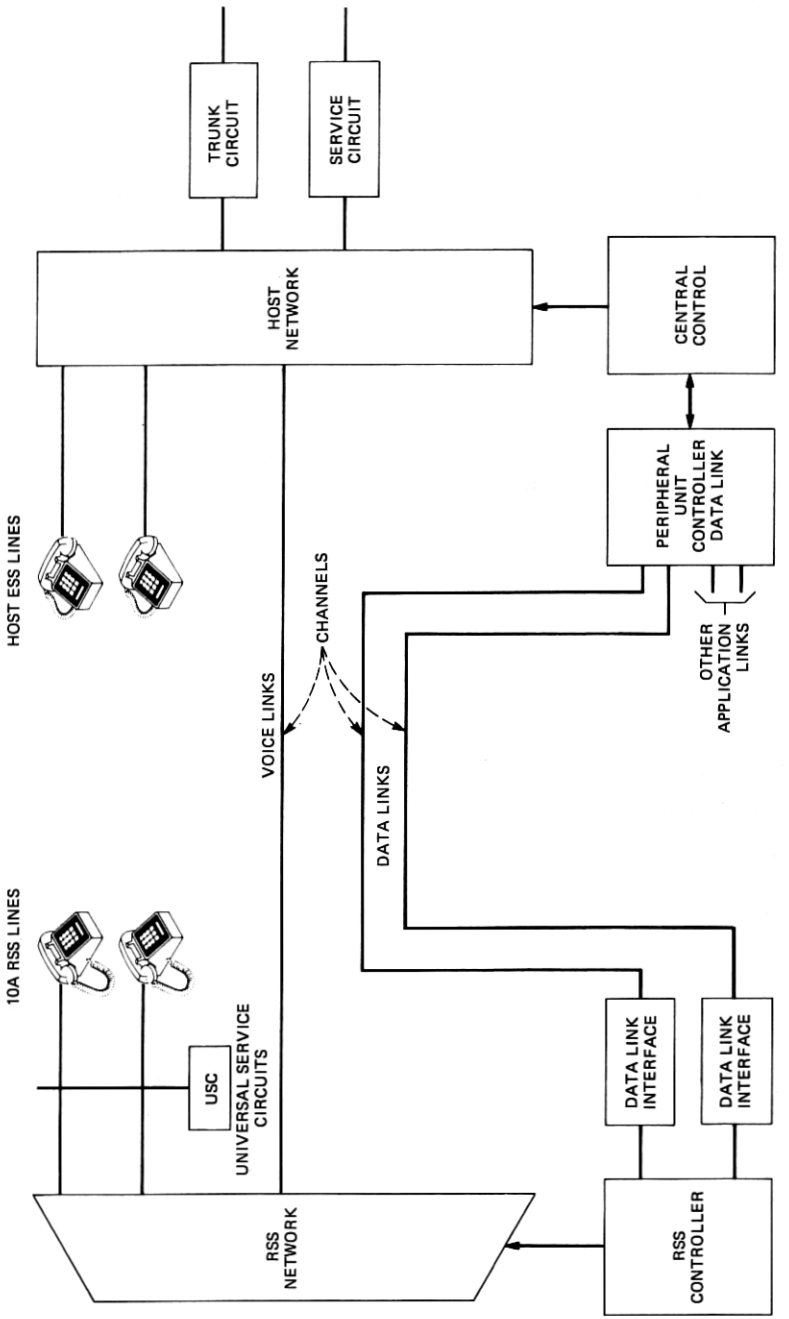

Fig. 5—No. 10A RSS—No. 1 ESS host interface.

control functions essential to implementing the synchronous link protocol.

At the host, the link interfaces with a functionally similar line interface unit that is part of the PUC/DL. The PUC/DL is a peripheral unit controller that has been equipped with the hardware and firmware to serve as a data link terminal for up to 16 data links. The function of the PUC/DL is to provide the control for physically transmitting and receiving data on the links and to provide data buffering for the host office. The data being transmitted and received by the PUC/DL are buffered on a per-link basis within the terminal. Sufficient data buffering is provided to allow the host to efficiently exchange large blocks of data with the terminal on a schedule that is efficient to the host.

5.2 Software overview

The routines that control the data transmission between the remote terminal and host are located in the remote terminal, the PUC/DL, and the host office. The organization of these programs is illustrated in Fig. 6. There are two basic functions to be performed. Data must be transferred reliably over the link and an interprocess communication system must be provided to allow software processes in the host to communicate with processes in the remote terminal. These two functions are provided by the data link protocol software and a set of message routing routines. The protocol software provides virtually error-free transmission of data over the link by executing a set of error-detection and error-correction procedures. The message routines allow a process in one machine to direct a message to a process in the other. These two systems are largely independent and bear a hierarchical relationship to one another in the sense that the message routing routines rely on the link protocol routines to accurately transmit data from one end of the link to the other.

5.3 Data link protocol

The protocol routines are executed in the remote terminal and the PUC/DL. The 10A RSS application uses the link level portion of the X.25 protocol to control the link. This is an industry standard protocol which is suitable for other link applications furnished from the PUC/DL in addition to RSS. It is a bit-oriented protocol designed for synchronous link operation. To provide for error detection and correction, the data to be transmitted are segmented into numbered blocks termed frames. The frame format is shown in Fig. 6. As frames are transmitted, they are sequentially numbered and a cyclic check code is computed over the data in the frame. The frame number is transmitted in a control byte at the beginning of the frame, and the check code is appended to the end. The frame numbering scheme makes it possible

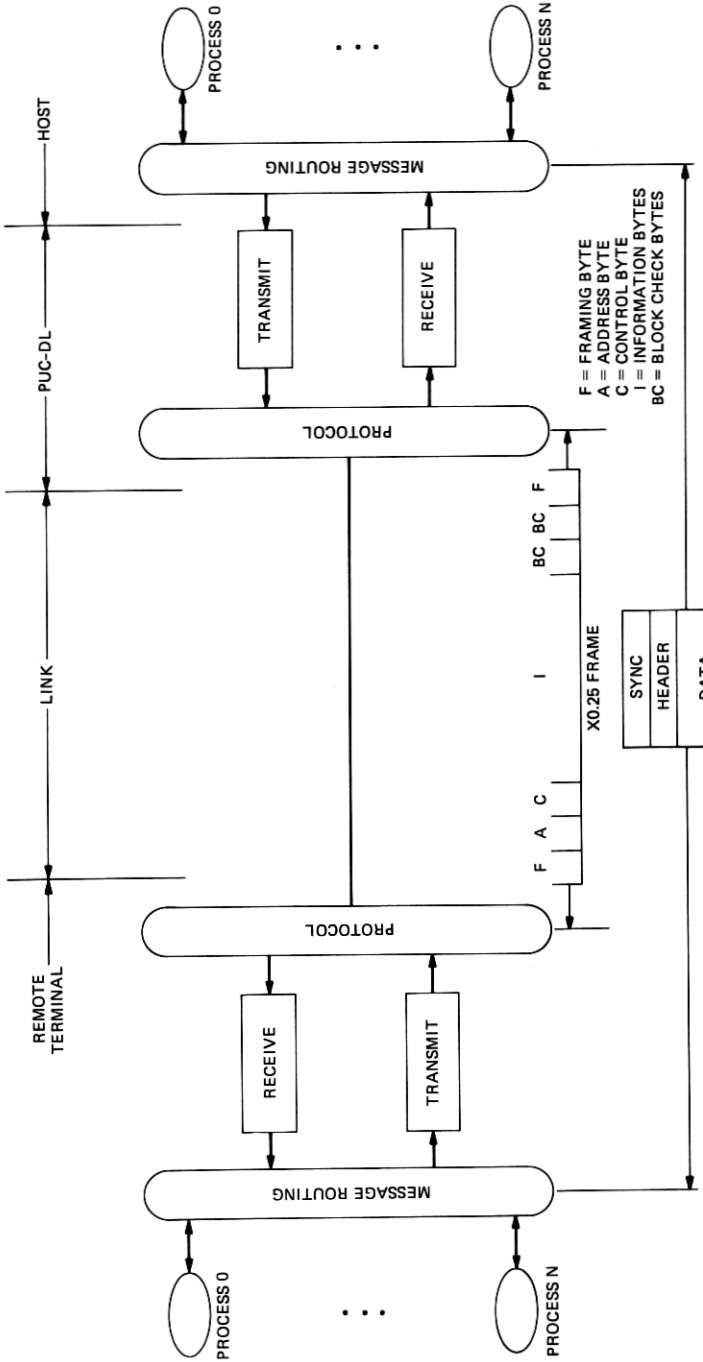

Fig. 6—Data transmission program overview.

to identify frames for retransmission and to detect missing frames in the received data. As frames are processed at the receiving end of the link, the cyclic check code is recomputed and compared to the one transmitted with the frame. A mismatch indicates that a transmission error has occurred. A positive acknowledgment is returned to the data transmitter for all frames received correctly, and a retransmission request is returned if a frame is received in error. In response to a retransmission request, the data transmitter will retransmit all the frames it has previously transmitted starting with the one in error.

The protocol software at the PUC/DL has the additional function of providing link status reports to the host machine. Error conditions such as high transmission error rate, frame acknowledgment time outs, and loss of data carrier are monitored by the protocol and reported to the host. The transmission error rate is determined from the number of retransmission commands received and sent by the PUC/DL protocol program. From these data, the host data link state control can take action to remove a link from service if it becomes inoperative or if its throughput is restricted because of excessive data errors.

5.4 Message routing routines

The routines that are responsible for routing data between individual processes in the two machines are executed by the remote terminal and the host processor. These programs assume that data received from the link protocol programs are error free and that any additional error control procedures for detecting transmission errors are unnecessary. These programs are designed to transmit data between buffers associated with client programs in the two machines. A program having data to transmit will load the data into its associated buffer. When the data are completely assembled, the buffer will be activated for the message routing routines and the data will then be transferred to a buffer associated with the destination program.

To facilitate the data transfer, the message routines assemble the data to be transmitted into messages that can be addressed to a particular destination. The RSS message format is shown in Fig. 7. A message is comprised of 16-bit data words, which is a convenient length for both the PUC/DL and the remote terminal processors. The two initial words are a SYNC word and a message header. The SYNC word denotes the start of the message and the header contains the address information necessary to route the message to a particular client program and buffer. It is possible for multiple buffers to be allocated to certain programs, such as the remote terminal call-processing routines, so that multiple processes can be executed concurrently. The message address structure allows data to be routed to an individual buffer associated with a client program by providing a CI

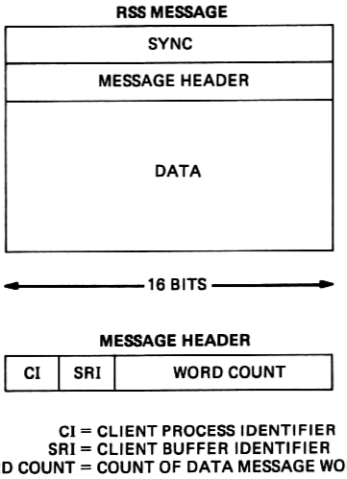

Fig. 7—Remote switching system message structure.

field to identify the program and an SRI field to identify one of its associated buffers. The word count in the header, along with the SYNC word, enables the message routing routines to parse individual messages from the received data supplied by the protocol programs.

5.5 Host data reception

The routines and buffers in the host that handle messages transmitted from the remote terminal are pictured in Fig. 8. The data received by the PUC/DL from the remote terminal are buffered in a RAM memory within the PUC/DL that is accessed from the host in the same manner as an existing scan memory. Message data in the PUC is then transferred to the host programs in two stages. Data are first transferred to a receive buffer in host call store by an interrupt level routine that executes every 25 ms. It is executed frequently enough at this rate to ensure that data from the link will not overflow the scan memory in the PUC/DL. This is necessary since the PUC/DL is not equipped to initiate a data transfer into the host when data are received from the link. In a No. 1 ESS host equipped with a signal processor, this is performed by a signal processor program.

A second routine, executed by the host Central Control (CC) at base level, is responsible for unloading the call store receive buffers and routing messages to destination client buffers. Individual messages are defined in the receive buffer by scanning for the SYNC word at the beginning of a message and using the message word count in the header to locate the final word.

Once a complete message has been received, the client identity (CI)

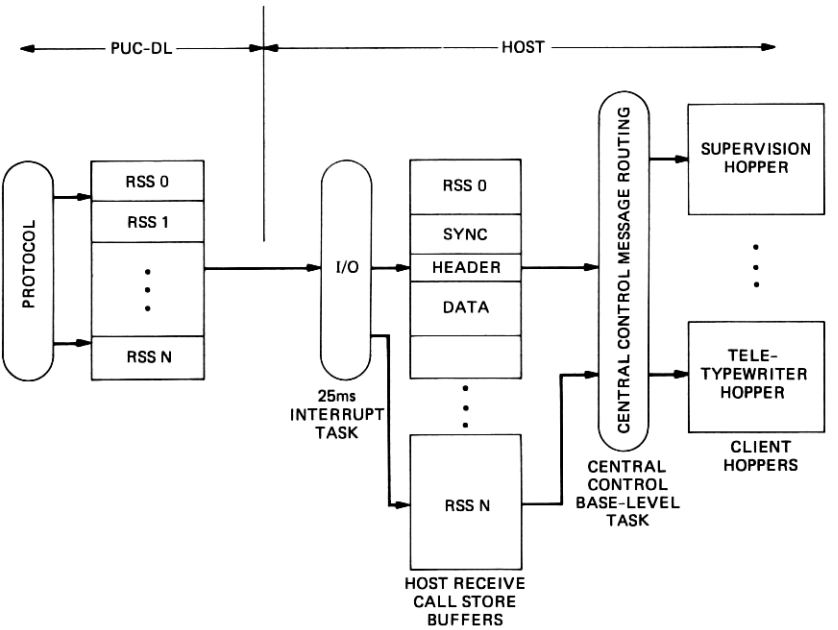

Fig. 8—Host message reception.

and SRI indexes (see Fig. 7) are used to locate the client buffers. A set of routing tables, as shown in Fig. 9, are supplied for each RSS. The primary route table is indexed with the CI in the message header. There are two types of entries in this table. If the client program has a single buffer associated with it, the entry contains the address of the load head cell for the buffer. The load pointer in the head cell enables the routing program to transfer the message from the receive buffer to the client buffer.

If the client program has multiple buffers, the primary route table entry is the address of a subroute table associated with the client program. The subroute tables are indexed with the SRI entry in the message header to obtain the load head cell address for a specific destination buffer. After the message transfer is complete, the client program is alerted to the fact that a message has arrived and is available to be processed.

5.6 Host data transmission

The buffering technique adopted for the transmission of peripheral orders to the remote terminal was designed to be compatible with the structure of the existing host software. The order buffering and message transmission must be tailored to the host software structure if the existing call processing and maintenance routines are to be preserved.

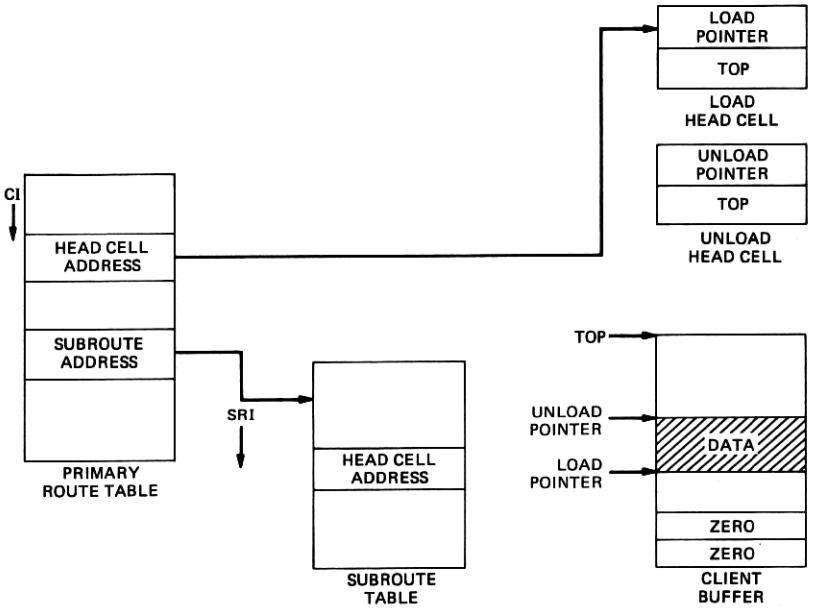

Fig. 9—Message routing tables

A few remarks on the nature of the call processing programs are necessary to understand the interface requirements.

The host call-processing programs are divided into call segments that process an input from a subscriber or a peripheral circuit to completion in one real-time segment (Fig. 10). All the peripheral orders required to process the input are generated by the program segment, but their execution is carried out by a separate set of input/output (I/O) programs. Peripheral operations in the host or remote terminal take tens or hundreds of milliseconds to execute, which precludes their direct execution within the call segment. During execution, a typical program segment may generate several remote terminal peripheral messages that are destined for different RSSs. In most cases, it will also generate a number of orders to be executed in the host periphery in conjunction with the remote terminal actions. The execution routines must be able to coordinate the host-remote terminal peripheral actions and coordinate the transmission of the remote terminal orders to the different RSSs. Frequently, it is necessary to execute the remote terminal-host peripheral actions in a predefined sequence where either the host or the remote terminal action must occur first.

The call processing and maintenance programs are coupled to the I/O programs through a set of I/O buffers that are loaded with the peripheral orders to be executed. All host peripheral orders are

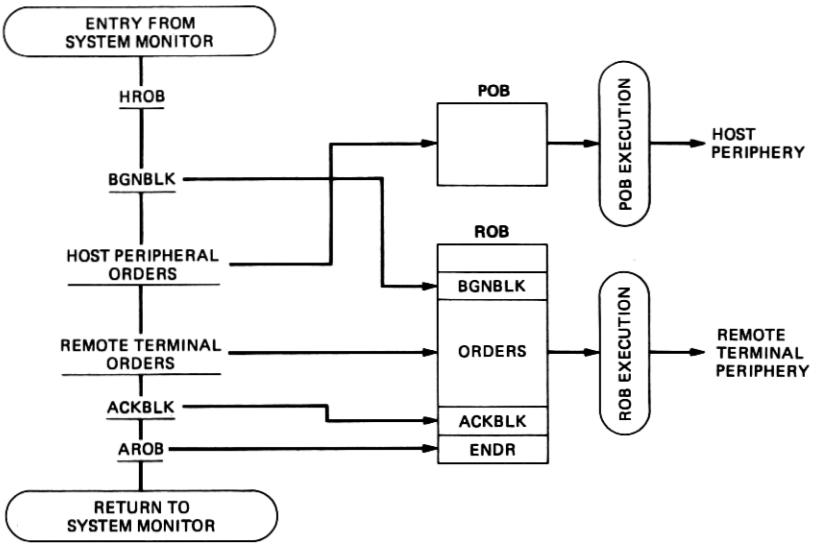

Fig. 10—Remote switching system call segment

buffered in peripheral order buffers (POBs) where they are executed by the POB execution programs, which are designed to handle the timing requirements presented by the host periphery.

A set of remote order buffers (ROBs) are provided in the host machine to buffer the peripheral orders being transmitted to the remote terminals. They are loaded and administered by the call processing programs in the same manner as the peripheral order buffers. Before a set of remote orders is generated, an ROB must be seized from a common pool provided in the host. The orders to be sent to the remote terminal are loaded in the ROB via a set of order macros which provide a high-level interface with call processing. When order loading is complete, the ROB is activated and the I/O routines transmit the orders to the remote terminal. An individual ROB may be used to send orders to any RSS. An administration macro is called before the orders are loaded to specify the identity of the remote terminal to receive the orders. The host can have any number of ROBs pending to send orders to an RSS; however, each remote terminal has a fixed set of eight ROB RECORDS that are used to store orders received from an ROB at the host (Fig. 11). The ROB RECORDS are buffers associated with the peripheral order execution program in the remote terminal that executes the orders transmitted from an ROB. Before the orders are transmitted, an idle ROB RECORD in the remote terminal must be selected by the host and the SRI in the message header must be set up to route the orders to this particular ROB RECORD.

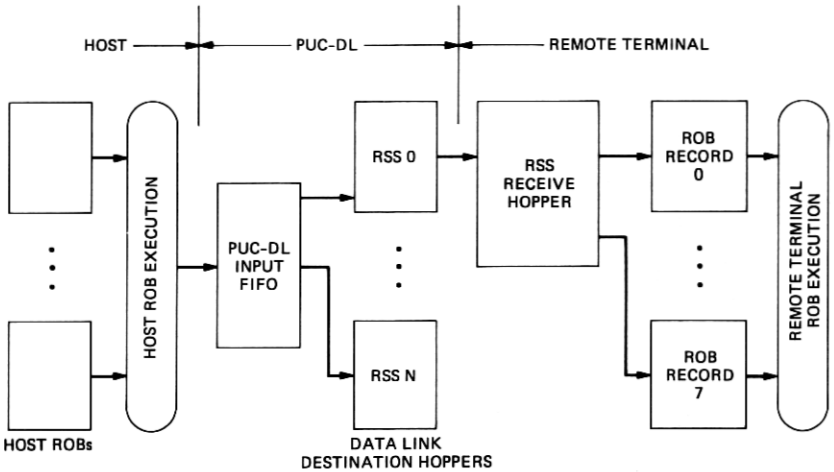

Fig. 11—Host data transmission.

5.6.1 The ROB execution protocol

A rudimentary protocol has been established to coordinate the activities of the call processing routines at the host with the execution of ROB orders at the remote terminal. Several orders will be grouped together into a single message at the host to be transmitted to the remote terminal. The orders in the message are executed at the remote terminal and upon completion an acknowledgment is returned to the host. The acknowledgment will indicate whether all the orders in the message were successfully executed or not and must be received by the host before any further actions will be permitted on this call. If the remote terminal encounters a failure in executing an order, the acknowledgment message will specify the failed order to the host and the remote terminal will suspend execution of all remaining orders in the ROB RECORD.

It is essential to receive a positive confirmation on the status of the orders for several reasons. If an order failure occurs, the host fault recovery routines can be scheduled to clear the call from the system. In addition, the acknowledgment allows the host to correctly sequence any other peripheral actions with the remote terminal orders. When the acknowledgment is received, the host can activate an associated POB or return to a call-processing program to implement the next action on the call. The restriction that additional RSS orders will not be transmitted until the acknowledgment is received also prevents the host from transmitting multiple sets of orders for the same call that would be executed in an arbitrary sequence at the remote terminal.

5.6.2 The ROB Entries

Two types of entries are loaded into an ROB. There are peripheral orders to be transmitted to the remote terminal and subroutine calls to be executed by the ROB execution program. The ROB subroutines are used to implement the message protocol and control the data transmission to the remote terminal. Figure 10 depicts the ROB loading and administration operations involved in a call segment for an RSS call. An ROB is hunted at the beginning of the segment (HROB), and a Begin-Block (BGNBLK) subroutine address is loaded in the ROB by the BGNBLK macro. When the BGNBLK routine is executed by the ROB execution program, it will identify the RSS, the subsequent ROB orders are to be sent to, and will also select an ROB RECORD at the remote terminal to receive the transmitted orders. Following BGNBLK, the set of orders to be executed at the RSS are loaded in the ROB by a set of order macros that format the order data and place it in the ROB. After the last order is loaded, an Acknowledgment-Block (ACKBLK) macro is called to terminate the loading process. This ACKBLK macro will set a flag in the last order loaded in the ROB to indicate that an acknowledgment should be returned by the remote terminal after the order is completed. In addition, an ACKBLK routine is loaded in the ROB to process the acknowledgment message when it is returned. If all orders were successfully executed, the ACKBLK routine passes control to the next entry in the ROB. If a failure is indicated, the fault recovery routines will be initiated to tear down the call. The final entry in the ROB is made when the ROB is activated for execution. The Activate-ROB macro activates the ROB to permit the ROB execution routines to process it and loads the ROB with the address of an END-ROB (ENDR) routine. Upon completion of the ROB activities, ENDR will initiate the next program segment required for the call. This may be the POB execution program or it may be a predefined call processing segment that will handle the next stage of the call.

Sequencing the execution of ROBs and POBs is necessary to control the order in which the host and remote terminal actions are carried out. This control is provided by the ENDR routines that are loaded in the POB and ROB on activation. The ENDR routines provide the capability for ROBs and POBs to be sequentially executed in either order or for the ROB and POB to be executed simultaneously. Simultaneous execution is used where it is important to minimize call setup delays and where the host and remote terminal actions can occur independently of one another. Control will be returned to the call processing client after both the ROB and POB actions have been fully completed.

VI. THE RSS CALL PROCESSING CONTROL

The RSS host call-processing software provides an ESS central office

with the capability to supply ESS features to lines served by the RSS. Since most of the call-processing functions for RSS lines are performed by the host ESS office, a full family of ESS features can be provided to the remote subscribers. The RSS call-processing software resident in the host ESS provides the means of controlling a remotely located switching system by taking advantage of existing equipment and control capability in the ESS. Firmware in the remote terminal supplements the host call-processing software appropriately. All call-processing control resides in the host ESS and any required actions at the RSS are requested via data link messages to the RSS. This permits the host to exercise total call control.

6.1 Originating call

A line originating in the RSS is first recognized during line scanning performed by the RSS microprocessor. The RSS line-scanning program in the remote terminal recognizes the line off-hook, performs timing to ensure the origination was not a transient "hit," and sends an origination request data link message to the host ESS. The host performs originating translations on the line. If service is allowed, it marks the RSS line busy and hunts an idle voice channel between the RSS and ESS. It also hunts a path through the RSS network from the originating line to the selected voice channel, and selects a customer digit receiver in the host, along with a host network path from the voice channel to the receiver. A Remote Order Buffer (ROB) is then executed to send appropriate data link messages to the remote terminal to set up the RSS network path. Messages are also sent to set the line supervision mode to repeat supervision of the originating line over the channel in the dialing (fast repeat) mode. This ensures that the dialed digits will be received properly by the host digit receiver. If the originating line is identified in translations as a two-party line or a line associated with a sleeve lead (or remote distributor point), appropriate data link messages are also included in the above ROB to perform a party test or operate the remote distributor points, respectively.

Upon successful completion of the data link orders in the remote terminal, the host then executes orders to its periphery [via a Peripheral Order Buffer (POB) mechanism] to set up the host network path between the voice channel and receiver to provide dial tone.

Processing of the call from this time on proceeds basically the same way as an origination by a host line. Digits are collected and analyzed by the same host software used to process host calls. At the completion of dialing and digit collection, a data link message is sent from the host to the remote terminal to set the line supervision mode to repeat the supervision of the originating line over the channel in the talking (slow repeat) mode to conserve remote terminal microprocessor real-time capacity.

The RSS originating call, from this point on, is routed and completed normally (excluding terminations to RSS lines) just as non-RSS line origination processing. This originating call configuration is depicted in Fig. 12 for a call that terminates in the host office. Upon answer by the called party or, for calls to other offices upon completion of outpulsing, the talking connection is established from the voice channel through the host network. In Fig. 12, this is shown by completion of the junctor (JCT) connections. The RSS answer timing, billing, traffic, and other administrative functions are all performed by the host just as for non-RSS calls. If the call terminates to an RSS line, special terminating RSS functions are performed as discussed in the following sections. When either the calling or called parties disconnect, disconnect functions are performed as discussed in Section 6.5.

6.2 Terminating call

An RSS terminating call is recognized when the host ESS performs the called number [terminating directory number (DN)] translations from digits collected from an originating host line or trunk. The RSS lines are distinguished from host lines by special RSS indicators in the terminating line translation output. After the translation is completed, special actions, as with the RSS originating call, are required to set up ringing. The host hunts an idle voice channel to the RSS, hunts a path in the RSS network between the voice channel and the terminating line, and seizes idle host ringer (whose function is explained below) and audible service circuits with associated host network paths to the voice channel and originating line or trunk, respectively. In addition,

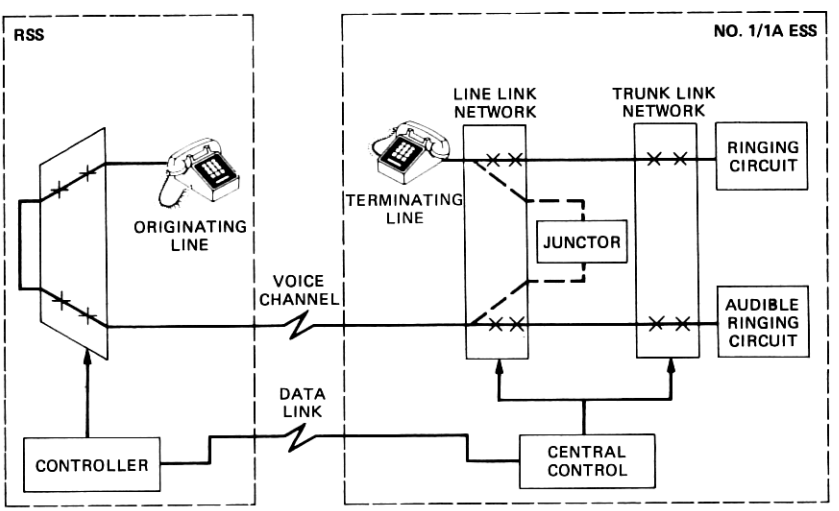

Fig. 12—Remote switching system originating call.

a talking path through the ESS network between the voice channel and the originating line or trunk is reserved.

An ROB is activated to send data link messages to the remote terminal to connect the terminating line to the voice channel and apply ringing to the line. Upon receipt of the data link orders, the remote terminal selects an idle universal service circuit (USC), along with a metallic bus and time slot to provide the type of ringing specified in the data link message. Supervision of the line is transferred across the voice channel to the host in the fast repeat mode. As with the originating call setup procedure, appropriate data link messages are included in the above ROB and sent to the remote terminal to operate sleeve leads or remote distributor points if so indicated in the output from the terminating line translations.

Upon successful execution of the ROB data link orders in the remote terminal, the host executes a POB to set up paths in the host network from the voice channel and the originating line or trunk to its associated service circuit. Power cross and low-line-resistance tests are done on the voice channel from the host ringing circuit. The host ringing circuit is then left in a state to monitor ring trip sent by the remote terminal over the voice channel to the host. Actual ringing is applied to the line by the USC at the remote terminal; the ESS host ringing circuit does not apply ringing voltage to the voice channel, but is only used to monitor for ring trip. This call configuration, as shown in Fig. 13, maximizes use of the existing terminating call sequences in the host.

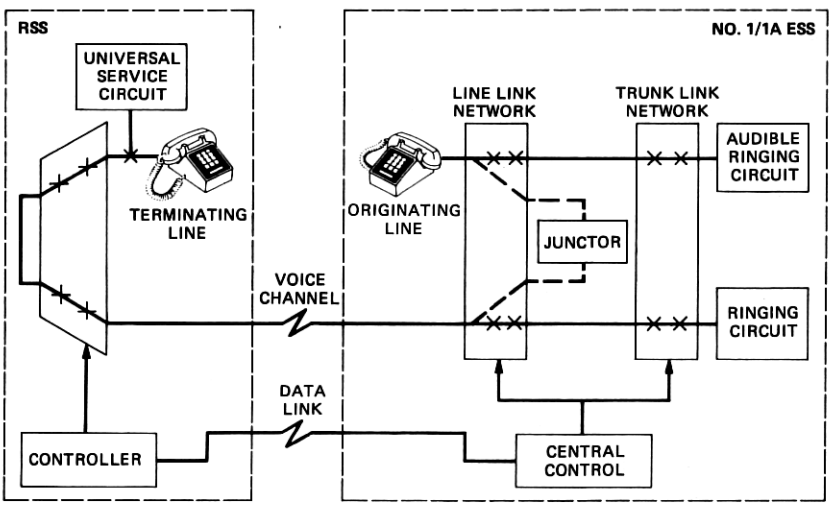

Fig. 13—Remote switching system terminating call.

When the called party answers, the remote terminal automatically releases and idles the ringing facilities (USC, metallic access bus, and time slot) and relays the ring trip report (off-hook signal) of the line across the voice channel and sets the supervisory mode to slow repeat.

The host ESS detects answer over the voice channel at the ringing service circuit, tears down the ringing and audible circuit connections in the host, and sets up the talking path that was previously reserved between the voice channel and the originating line or trunk. If the originating line in the RSS terminating call description is actually another voice channel to the same RSS as the terminating line, the call is considered an intra-RSS call and special actions are invoked as described in Section 6.4.

Disconnect actions are identical to those for the RSS originating call, except for the disconnect timing associated with the terminating versus the originating party.

6.3 The RSS reverting calls

The RSS reverting calls involving a call between two parties on a party line are handled similar to host reverting calls. However, ringing is provided similarly to the way it is applied on RSS terminating calls, with the exception that two time slots are needed in the RSS remote terminal so that ringing can be applied to both customers. Each RSS has its own ringing office option as defined in translations. This RSS ringing option, which can be either ac/dc or superimposed, is completely independent of the host ESS office ringing option, or any other RSS served by the same host. The RSS universal service circuit has the capability to provide either ringing option under firmware control.

6.4 Intra-RSS call

An intra-RSS call, where both the originating and terminating parties are served by the same RSS, is handled initially as a combination of an RSS originating call and an RSS terminating call. After answer, the host initially establishes a talking path within its network between the two voice channels. Immediately following the establishment of this talking connection, certain RSS actions are invoked to reswitch-down the call so that the talking connection resides entirely within the RSS network, thus releasing the ESS network path and voice channels for use on other calls. The call is initially set up through the host network, then followed by a reswitch-down. This two-step process maximizes the use of existing host line-to-line call setup routines and provides for a well-defined interface with the reswitch-down software module.

The reswitch-down action is initiated when the host hunts an intra-RSS network path between the originating and terminating lines. An ROB is activated to send data link orders to the remote terminal to

disconnect both line-to-channel network paths and connect the two lines through the RSS network. The supervisory mode of the RSS lines is set to scan for either a disconnect or switchhook flash, depending on the features associated with each line. Since the intra-RSS connection is entirely within the RSS, a change in supervisory state of the line must be reported over the data link to the host. The sequence of intra-RSS call configurations including reswitch-down is illustrated in Fig. 14.

If a network path in the RSS is not available, or one of the lines is an RSS coin line, the intra-RSS call is not reswitched-down and remains connected through the host ESS network. Intra-RSS calls involving coin lines are not reswitched-down in order to utilize host coin disconnect routines and thus, simplify disconnect actions.

The use of various custom calling services or other special services requires a reswitch-up of an intra-RSS call to establish a talking path between the two parties via the host network using two voice channels. This allows existing host software and equipment to be utilized to provide these customer services. The following operations require a reswitch-up operation on an intra-RSS call.

(*i*) A flash by an RSS customer to add on a third party.

(*ii*) A terminating call to one of the two parties of an intra-RSS call that has the call waiting-terminating feature.

(*iii*) A busy verification test of one of the two parties of an intra-RSS call by an operator.

When the host determines that a reswitch-up function must be performed, for any of the reasons given above, the host seizes two idle voice channels to the RSS and hunts a path between them in the host network; the host also hunts a path between the two voice channels and both lines in the remote terminal. A POB is executed in the host to set up a talking connection between the two voice channels. Following this, an ROB is executed to send data link messages to the remote terminal to disconnect the intra-RSS talking connection, connect each line to a voice channel, and set the supervisory state of each line to the talking mode (repeat supervision of the line over its respective voice channel). Once the intra-RSS call is reswitched-up to a talking connection via the host ESS network, the original service requested can be provided just as it would to a normal line-to-line connection of two host lines.

6.5 Disconnect functions

Disconnect actions for RSS calls are a function of the particular call configuration involved, i.e., intra-RSS calls or RSS calls through the host network. For call configurations involving both RSS and ESS paths, the ESS disconnect programs control the call disconnect actions. The same disconnect sequence that is performed on ESS host lines is used on RSS

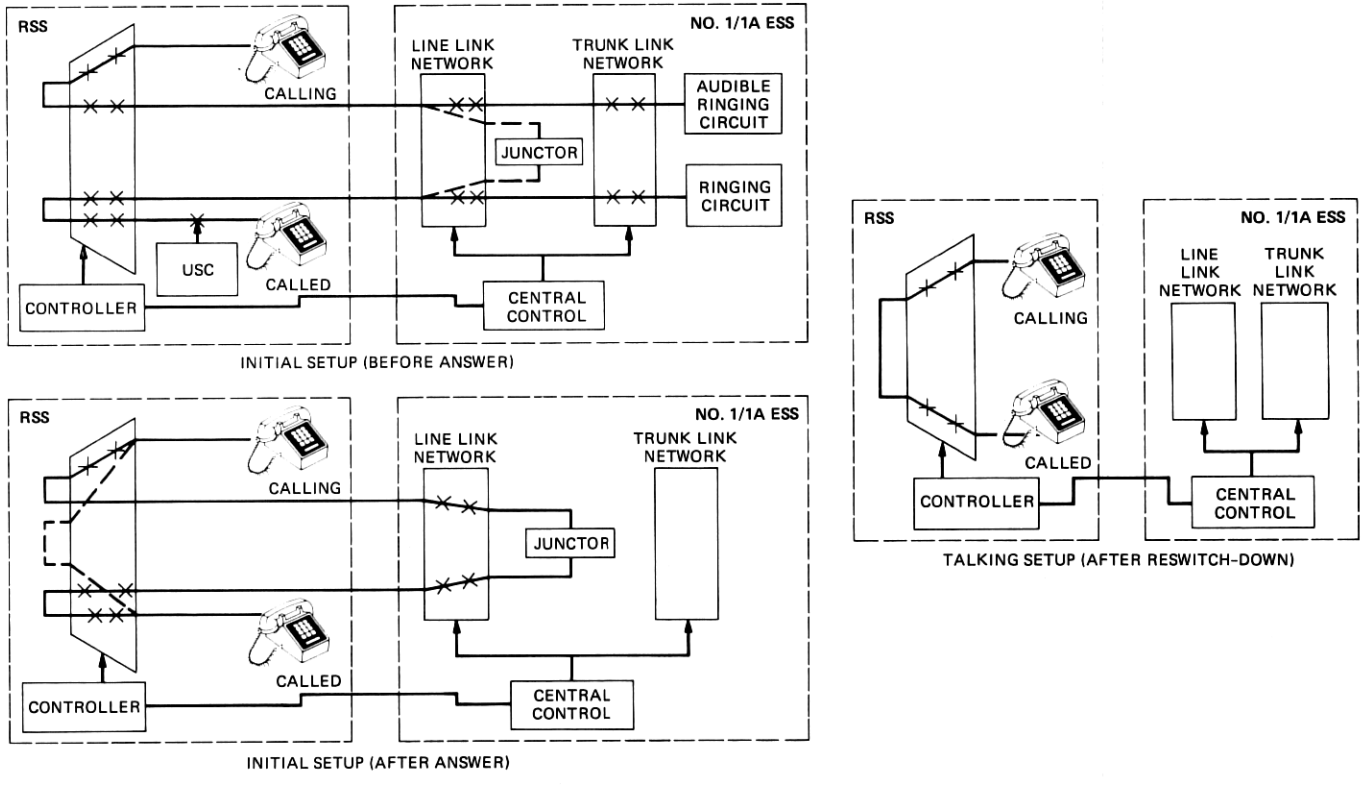

Fig. 14—Intra-RSS connection sequence.

channels that terminate on the host line network. This disconnect control strategy provides the ability to centrally recognize an RSS channel during normal host disconnect processing. This recognition occurs when ESS programs perform a restore verify action on the RSS channel (restore the channel's line ferrod to the idle state and verify that it can detect an off-hook signal). At this point in the host disconnect processing, unique host RSS disconnect modules are invoked to disconnect the RSS network path. The normal host disconnect program and RSS disconnect module then autonomously complete their respective disconnect actions. The only common resource between the two control programs is the RSS channel. The ESS host disconnect program administers the host end of the channel on its network, and the RSS host disconnect program administers the RSS end of the channel on its network. After both ends of the channel are idled, the channel is then available in the host for reallocation to a new call.

In the case of an intra-RSS call, where the call configuration involves a connection totally within the RSS network and no channels or ESS network paths are involved, the RSS lines are supervised in the 10A RSS. Hits, flashes, and on-hooks are detected by the RSS; flashes and on-hooks are reported to the ESS host via data link message. These messages are routed to unique RSS disconnect control modules in the host for proper processing. In the case of an on-hook, these programs perform the proper disconnect timing and then execute ROBs to disconnect the intra-RSS network paths and idle the lines involved. The intra-RSS call is reswitched-up on receipt of a flash message (as discussed in Section 6.4).

VII. DATABASE INTEGRITY

In an electronic switching system, the status of resources and telephone calls are recorded in temporary memory. These data can become mutilated because of program bugs, hardware errors or program design errors, resulting in the loss of resources or system degradation. The problem is further complicated in RSS because parts of the new data structures in the host ESS are duplicated in the remote terminal. The new structures are

(*i*) PMRs for lines
(*ii*) PMRs for channels
(*iii*) Network map
(*iv*) Remote order buffers (ROBs) and
(*v*) Remote miscellaneous scan point map.

The actual data stored in the two copies of these structures are not identical in all cases since the host and remote terminal do not perform identical functions. For example, the state stored in a line PMR at the remote terminal represents the supervisory state of the line (origina-

tion, repeat supervision onto a channel, high and wet, etc.), whereas the state in the host PMR represents both the status of the line (idle, busy, maintenance) and the type of path memory configuration, as discussed previously. However, there is a mapping between the two sets of states in that the host line state implies the possible supervisory states of the line. Deviations from this mapping should only be because of the time lag of the data link.

To ensure the integrity of the data structures, the first step is to prevent as many errors as possible. The RSS software applies many of the techniques that have been successfully used in other ESS projects to avoid potential causes of errors. Some of these are good documentation, standardized program interfaces, structured design, structured programming, a high-level programmer's language, and a standardized data definition language. In addition, access to the new data structures introduced into the host is limited to the administrative programs that have the responsibility for that particular database.

The second step is to make the programs as error tolerant as possible since data errors will still occur. The main technique for this is defensive programming. The degree to which a data error is propagated through the system depends upon how the programs use the data. To have a minimal effect upon the system, programs should account for bad data. Some specific types of defensive coding techniques are

(*i*) Range checks on data to prevent overindexing tables.
(*ii*) Accounting for all possible subroutine return code values.
(*iii*) Use of symbolic definitions for data values.
(*iv*) Accounting for all possible program inputs.
and
(*v*) Invalid data value checks.

Despite the preventive and defensive techniques that are employed, errors can still occur in the data. Programs are required to detect these errors and restore the facilities to the proper condition to avoid system degradation. The audit programs are responsible for detecting and correcting data errors and the initialization programs are responsible for restoring system facilities when the degradation is severe enough to cause major system degradation.

7.1 Audit programs

The integrity of the data structures is checked and corrected by a set of audit programs. Each audit program is individually tailored to a specific data structure or group of data structures and determines if the data items follow certain established rules. If the checks fail, the audit programs idle all resources (both software and hardware) associated with the particular error and print error messages on the teletypewriter. The audit programs also aid in the initialization of the data structures.

As previously discussed, the RSS system has new data structures in the host ESS that are duplicated in the remote terminal. From a call processing viewpoint, the remote terminal functions as a slave to the host, executing the various orders sent to it. The remote terminal updates its databases as a result of the orders received from the host. Orders involving lines or channels cause the remote terminal to set its PMR to the supervisory state appropriate to the order. The network map bits are set to busy or idle when it receives a setup or tear-down order, respectively. Although the host is the controlling entity, calls can be lost or service can be denied a customer if the data stored in the remote terminal are not consistent with the host data. For example, if the host has a line in the origination state and the remote terminal has supervision turned off on that line, the customer will not be able to originate since the remote terminal would never detect the off-hook. Thus, a means must be established to ensure that service or facilities cannot be lost because of differences between the two data structures. In addition, the data structures must be internally consistent within each entity independent of the synchronization problem.

The audit philosophy adopted for RSS is that each entity (host or remote terminal) will maintain the integrity of its databases independently. If the host finds a discrepancy in its database, it corrects the problem by idling all host resources involved and instructs the remote terminal to put its facilities into a known (usually idle) state. If the remote terminal finds a discrepancy in its database, it sends a message to the host and the host audit programs initiate the actions given above.

Thus there are three classes of audits associated with the RSS system:

(*i*) Host audits that maintain the internal integrity of the data structure in the host.

(*ii*) Remote terminal audits that maintain the internal integrity of the data structures in the remote terminal.

(*iii*) Audits that guarantee that the host and remote terminal data structures are consistent.

Audit classes 1 and 3 are discussed below.

7.1.1 Host audits

Existing host audits are extended to include the new data structures and new data values introduced with RSS. The main audit modifications are for the RSS path memory, the RSS network map, channels, and ROBs.

The RSS path memory and network map are audited by making the following checks:

(*i*) Point-to point-back checks are performed between PMRs and PMJs; that is, a PMJ that is pointed to by a PMR must contain the REN associated with the PRM.

(*ii*) Point-to point-back from a PMR or PMJ to a call register if linkage to a call register is indicated. In this case, a PMJ contains the REN of one of the terminals involved in the path and the call register contains an REN or LEN of the channel.

(*iii*) The junctor busy-idle bit in the network map is checked to ensure that it is idle if and only if the PMJ is idle.

(*iv*) Each of the network map bits that is marked busy is checked to guarantee that it is in a valid path.

Additional checks are performed on channels to ensure that every idle, equipped channel is on the idle link list, to ensure that the idle link list only contains equipped idle channels and that the associated host line bit for the channel is idle if the PMR is idle. The latter check requires timing since the two ends of the channel are idled autonomously, and thus the status of the two ends can be out of step for a short interval.

The ROBs are audited by periodically rebuilding the idle link list and by timing busy ROBs. If an ROB remains busy for an extensive length of time, the ROB and any associated call register are idled. All paths and circuits are also idled. The corrective action taken by the host audits is to idle facilities (hardware and software) in the host and to send orders to the remote terminal to cause the facilities at that end to be idled.

7.1.2 Synchronization between host and remote terminal

The problem of maintaining the data structures in the host and remote terminal in synchronization is greatly simplified by taking advantage of the normal system operation. The remote terminal updates its data in response to orders from the host. Bits in the remote copy of the network map are marked busy or idle in response to orders to set up or tear down network paths. Thus, no network map audit is required between the host and remote terminal since the host does the hunting and idling of paths and the remote copy will trend towards the proper state even if it does temporarily get out of step.

Similarly, ROBs are controlled from the host end and normal operation will result in the remote copy being brought back into step with the host.

The remote terminal audits check that all equipped lines have supervision turned on. Any equipped lines that are unsupervised are reported to the host via a data link message. If the host audits determine that the line state really calls for supervision to be on, the line and any associated resources are idled.

Periodically, the host sends a copy of its version of the remote scan point map to the remote terminal which overwrites its map with the host's data. If the hardware state of the scan point differs from the

map, the normal scan program will detect this as a change and report it to the host, resulting in both copies being brought back into step.

7.2 Remote terminal initialization

System initialization programs are responsible for correcting errors that prevent the system programs from cycling correctly. The initialization programs are usually executed as a consequence of errors being detected by the processor check circuits that monitor the sanity of the system operation. In the remote terminal, the primary checks for monitoring proper program operation are the system sanity timer which monitors the main program cycle time, the write protect circuitry which prevents illegal writes into program store and certain peripheral error checks which detect attempts to access unequipped areas of the periphery. If any of these errors are detected, it is indicative of an error in the system database and the method of recovery is to initialize a segment of the data and then return to the normal program cycle.

Since the initialization process inherently destroys a portion of the call-processing data, a corresponding set of calls will be lost, and it becomes a requirement for the initialization program to release the peripheral circuits associated with these calls. Any network links, channels, or service circuits employed on these calls must be idled by the initialization program before a return to normal system operation is begun. This is accomplished by releasing all the circuits that are marked idle in the initialized database.

Whenever an error is detected by a fault detection circuit, a processor interrupt is generated that executes the fault recovery programs. Various fault recovery actions are taken, depending on the type of errors detected and their frequency. As the result of a write-protect error or a sanity timer timeout, the off-line processor complex may be switched on-line and some degree of data initialization performed, depending on the number of errors that have been detected within the recent past.

The amount of data that is initialized on the first error is small. As successive errors are detected the serverity of the initialization is increased with the consequent loss of progressively greater numbers of calls. The ultimate action is to initialize the entire database and restore the system to an idle state. The goal of handling the fault recovery in stages, with increasingly more severe initializations, is to restore the system to a working mode with the loss of a minimal number of calls.

At either the remote terminal or the host, there are three fundamental levels of initialization: a minimal clear, a transient clear, and a stable clear. A minimal clear involves the initialization of the variable data associated with the active processes in the system and has the potential of disrupting at most several calls. A transient clear will

initialize all the data in the system that is related to any call in progress. All calls in the process of being established or disconnected will be lost; however, calls in a stable talking state will be preserved. The final stage of initialization is a stable clear where the entire database is reinitialized and all calls are lost.

For the RSS system, the initialization process is somewhat more involved than normal because the host and remote terminal databases are interrelated and an initialization level (phase) in one machine affects the database of the other system. Although the host machine is responsible for the control of the remote terminal on a call-related basis, the operation of the processors in the two machines is fairly autonomous with respect to their instantaneous activities. This is the situation for fault detection where the two systems are entirely independent. Each machine is responsible for initializing and carrying out is own fault recovery actions and the level of the accompanying initialization will be solely determined by the conditions within the machine that detected the error. In effect, either machine is able to initiate any level of initialization on its own database independently of the other. However, as part of the initialization procedure, the hardware and software within the two machines must be synchronized so that the databases reflect a consistent set of calls. The calls that were destroyed in one machine must be reported to the other so they can be cleared from that system also. For example, a host transient clear will destroy a number of calls that involve remote terminal lines. These calls must be cleared in the remote terminal so that periphery and call records are in agreement with those in the host.

The exact procedure for synchronizing the two machines will depend on which system has initiated the phase. Since the host is in charge of call control, its call records are regarded as the master copy and the remote terminal state is brought into agreement with its set of records. The host is, therefore, in charge of synchronizing the two machines.

Whenever a phase occurs in the host, the synchronization procedure is straightforward. The host will initialize its database and periphery and will then send initialization orders to the remote terminal to bring it into agreement with its updated records. When the remote terminal undergoes an initialization, it reports the level of the initialization to the host. In some instances, on a stable clear for example, this is sufficient information to allow the host to initialize its database. In the case of a transient clear, it is also necessary to transmit a map of the lines that are in a transient state in the remote terminal. From the data specifying the initialization level and the map of transient lines, the host is able to update its call records and periphery. Once this is accomplished, it will conduct an initialization of the remote terminal in the same manner as for a host-initiated phase.

VII. SUMMARY

The host RSS software supplies a local ESS with the capability to control a 10A RSS. The major call control resides in the host, with the 10A RSS acting as a slave in executing orders from the host.

This paper has described the major host software functions required for implementing the RSS feature on a host ESS. These major functions are RSS resource and data administration, the RSS message handling, and call processing and database integrity. The host RSS software is structured to meet system requirements to provide easy portability among local ESSs, to provide the capability of giving RSS lines the same features as host ESS lines, to minimize the effects on host capacity for non-RSS calls and to easily make new host features available to RSS lines.

IX. ACKNOWLEDGMENTS

In addition to the authors of the other papers in this issue of *The Bell System Technical Journal*, many others contributed to the design and implementation of the 10A RSS host software. Those who deserve special mention are R. T. Broeren, W. I. Czajkowski, E. J. DiVecchio, R. T. Emery, G. A. Inberg, G. W. Lambrecht, S. A. Lottes, and L. H. Ringwald.