RECENT CHANGES

SOFTWARE DESCRIPTION

2-WIRE NO. 1 AND NO. 1A "ESS*" SWITCH

CONTENTS PAGE

*Trademark

CONTENTS PAGE

1. GENERAL

INTRODUCTION

1.01 This section describes the program actions performed by the Recent Change Subsystem (RCSS) operating in a No. 1 or No. 1A ESS switch central office. Information unique to the No. 1A ESS switch application is so noted. Applications which are unique to the No. 1 ESS switch are not described in this document.

1.02 This section is reissued to delete the outdated materials and to include coverage of generic programs 1E7, 1AE7, and later generic programs. Revision arrows are used to emphasize these changes.

1.03 Part 6 provides a list of abbreviations and acronyms used in this section.

PURPOSE OF THE RECENT CHANGE SUBSYSTEM

1.04 The RCSS consists of a collection of Recent Change message interpretation programs. These programs accept and validate recent change input messages and generate the proper translation data for storage in translation memory. The RCSS provides maintenance personnel with a means of modifying the translation data to reflect changes in customer service requirements and in the internal system.

SCOPE OF SECTION

1.05 The information provided in this section includes a description of the following:

(a) Operation of the RCSS at the system level

(b) Operation of the RCSS at the program level

(c) Program control and data flow through the RCSS.

2. BRIEF DESCRIPTION OF PROGRAM IDENTIFICATIONS (PIDENTS) INVOLVED

2.01 The RCSS is comprised of code and message pidents. The message pidents are tables of data which precisely define the recent change messages. The control program or code pidents are recent change programs of executable code (ie, ESS switch instructions) which control the operation of the RCSS. The code pidents are listed in Table A. Many of the operations of the RCSS are repetitive in nature and are accomplished by applying fixed algorithms to the data in the message tables. Therefore, a substantial portion of the RCSS can be said to be table driven by the message pidents. The message pidents are listed in Section 231-048-301.

CONTROL (CODE) PIDENTS

2.02 The Recent Change Initialization and General Control (RCIG) program is the control program for all recent change processing. The RCIG provides the interface with the TTY system. Processing the first line of a recent change message (including internal data buffer initialization) is handled by RCIG, as is the execution control of each of the other code pidents. In addition, RCIG controls auxiliary RCSS functions in No. 1A ESS switch such as listing the rollback area (RBA) and allowing or inhibiting the various recent change message sources.

TABLE A

RCSS CODE PIDENTS

PIDENT	TITLE: Recent Change	DOCUMENT REFERENCE NO. 1 ESS SWITCH	DOCUMENT REFERENCE NO. 1A ESS SWITCH
QURC	Queuing For CCSC Changes		6A115
RCMU*	Message Update		6A230
RCIB*	Interface Block Builder		6A231
RCIG	Initialization and General Control	PR-1A300	6A300
RCIE	Input Editor	PR-1A301	6A301
RCKI	Keyword Input	PR-1A302	6A302
RCVC	Validity Check	PR-1A303	6A303
RCTF	Translation Data Format	PR-1A304	6A304
RCCH	Change Order Processor	PR-1A305	6A305
RCFI	Format Interpreter Routines	PR-1A306	6A306
RCWL	Work List Processor	PR-1A307	6A307
RCDY	Delayed Order Processor	PR-1A308	6A308
RSUB	General Subroutines	PR-1A309	6A309
RCTS	Table Subroutines	PR-1A319	6A319
RCSI	Shared Information and Input Subroutines	PR-1A320	6A320
RCxx*	(One pident per message)	PR-1A3yy	6A3yy
QUDB	Queuing for RC Control System	PR-1A830	6A830

* No. 1A ESS switch only.

† Refer to Sectin 231-048-301 for message pidents.

2.03 The Recent Change Queuing (QUDB) program is the control program for the queuing and serving of the queued recent change input message from any of the allowable sources. All clients, including TTY, are queued upon entry into the RCSS. The QUDB then determines when recent change scratch is available for use and which queued message has priority. After serving the appropriate client off its queue, RCIG is called to do further initialization and start up processing of the message.

2.04 The Recent Change Input Editor (RCIE) program provides the link between the TTY buffer and the RCSS. Services provided include character translation and character screening.

2.05 The Recent Change Keyword Input (RCKI) program performs the processing and local error-checking of the recent change message keywords.

2.06 The Recent Change Validity Check (RCVC) program verifies that the recent change message input contains a valid set of both keywords and data.

2.07 The Recent Change Translation Data Format (RCTF) program processes the NEW pass of

format interpretation, used when the building of a new translation is required by an recent change message.

2.08 The Recent Change Change Order Processor (RCCH) program processes the change (CHG) pass of format interpretation, used when a recent change message requires a change in existing translation data.

2.09 The Recent Change Format Interpreter Routines (RCFI) pident is a collection of subroutines that administer the scratch area in memory used for translation formatting.

2.10 The Recent Change Work List Processor (RCWL) program does pass 1 processing of work list entries made by the message pidents (No. 1A ESS switch only).

2.11 The Recent Change Delayed Order Processor (RCDY) program provides:

(a) Means of implementing recent changes which have been processed earlier on a delayed basis.

(b) Builds the delayed activation block (DAB)

(c) Expand DAB entry points for phone activation and TTY activation of delayed recent changes

(d) Expands the rollforward block.

2.12 The Recent Change Interface Block Builder (RCIB) program (No. 1A ESS switch only) provides the basic interface between those portions of the No. 1A ESS switch and No. 4 ESS switch RCSS used by the No. 1A ESS switch RCSS. The RCIB converts the recent change work list into a form usable by the memory update program.

2.13 The Recent Change Message Update (RCMU) program (No. 1A ESS switch only) performs the actual insertion or modification of translation data in the translation area of memory.

2.14 Queuing For Recent Change Control (QURC) program provides a means of buffering customer changeable speed calling (CCSC) list entries until the RCSS is able to process them. The entries are stored in a queue and are then processed by the RCSS as time permits. (Entries are active on queue.)

2.15 The General Subroutines (RSUB) pident is a collection of subroutines called as necessary by the other control pidents.

SHARED PIDENTS

2.16 ▶The Recent Change Control (RCCN) program assembles customers logs and performs miscellaneous maintenance in the customer-originated recent change area. In No. 1 ESS switch the recent change control program also performs recent change insertion.◀

2.17 The Table Subroutines (RCTS) pident consists of ▶assembler language subroutines accessed by the recent change message pidents to do things that are difficult or impossible to do in the recent change macro language.◀

2.18 The Shared Information and Table Subroutines (RCSI) pident consists of data tables accessed by the recent change control system to locate the correct message pident based on the message name (ie, RC:LINE), assemble keyword data, and correctly index into translations.

MESSAGE PIDENTS

2.19 The RCSS message pidents are listed in Section 231-048-301. In general, there is one message pident for each recent change message (exceptions will be noted later). Each message pident is comprised of a set of data tables used to drive the code pidents.

3. BRIEF DESCRIPTION OF RELATIONSHIP TO OTHER FUNCTIONS

INTERFACE WITH OTHER PROGRAMS

A. No. 1A ESS Switch Main Program

3.01 The ESS switch main program flags used by the RCSS are as follows:

(a) ▶Flag 447 is base level class D flag used for the queuing and input processing programs.

(b) Flag 209 is a base level class D flag used for message processing.

(c) Flag 1029 is a main program fill job to allocate additional real time to TTY recent change input processing as it becomes available (traffic load dependent).◀

(d) Flag 321 (RCMUGENT) is a base level class B flag used for updating translation data.

(f) Flag 316 (RCMUFENT) is a base level class E flag used for auxiliary program control.

3.02 The first two flags are carry-overs from No. 1 ESS switch. Due to the significant differences in the way processor input/output and main program scheduling is done in No. 1A ESS switch, the 100-ms flag used by No. 1 ESS switch for input/output is not needed. The entry rate for a class D MP flag is every 50 ms and is almost independent of traffic load. A recent change entry rate is relatively infrequent, causing a significant response delay when inputting a recent change message. Another MP flag is used to give additional entries to the RCSS on a traffic-dependent basis, thereby decreasing response time to acceptable levels except during traffic overload. The RCMUGENT flag requires a higher entry rate to ensure a rapid update of the translation data.

B. TTY Programs

3.03 The TTY programs provide the RCSS with the ability to receive recent change message inputs from the TTY and to report acceptance or reason for rejection of the input message (IM) through TTY acknowledgments and output messages (OMs). A recent change message input occurs on a line-by-line basis. The input characters are interpreted by the input/output programs after an entire line is typed, delimited with an end-of-transmission (EOT) character, and, for a DATASPEED® 40 TTY, transmitted by repositioning the cursor and operating the SEND key. The TTY service request is queued and processed as soon as the RCSS is idle.

3.04 The recent change messages are treated as messages requiring special handling. When the input/output program sees the input message verb "RC:", it passes the entire input to RCINIT for processing. If the RCSS is idle, the RCINIT initializes the RCSS to process the message. ▶The RCINIT calls QUTTYL in QUDB to determine whether or not there is space on the TTY queue for the message input. If the queue is full, an acknowledgment RL<BUSY> is printed at the TTY which input the message. If there is space on the queue, the message is put on queue and the flag is set to request queue service.◀

3.05 The RCSS output handled by the TTY programs is of two types: TTY acknowledgments and OMs. The TTY acknowledgments consist of a few American Standard Code for Information Interchange (ASCII) characters loaded by the RCSS directly into the TTY buffer. The TTY acknowledgment provides an immediate status response for the message. Data for a recent change OM is passed to the TTY programs by loading the data into scratch area in call store and initiating a TTY print call. For a description of recent change TTY acknowledgments and OMs, refer to Section 231-048-301.

INTERFACE WITH PROGRAM STORE

3.06 The RCSS consists of 12 code (or control) pidents (No. 1 ESS switch), and 15 code (or control) pidents (No. 1A ESS switch), two shared pidents (RCTS and RCSI), and a set of message pidents. The 13 control pidents and the shared pident RCTS are code pidents made up of No. 1/1A ESS switch instructions. The shared pident RCSI consists entirely of data tables, some full program store-word (24-bit) tables, and some 12-bit tables.

4. SUBSYSTEM FUNCTIONAL DESCRIPTION

4.01 Inputs to the RCSS may occur simultaneously from up to six different sources (Fig. 1):

(a) Customer lines

(b) Service order activation phone

(c) Office recent change channel

(d) Service order TTY (includes paper tape or magnetic cartridge)

(e) ▶Centrex station rearrangements (CSR) customer

(f) Inward Wide Area Telecommunication Service (INWATS) data base for busy/idle status indicator (BISI) messages.◀

These requests for RCSS service may originate for any number of reasons. Inputs from customer lines, CSR or BISI, are not under central office control. However, the end result in each case is that the translation data requires changes. (Inputs can be restricted by the central office.)

TRANSLATION DATA STORAGE

4.02 In the No. 1A ESS switch, translation data is stored in core memory in unduplicated call store. The recent changes with permanent status are stored in the translations data area of unduplicated

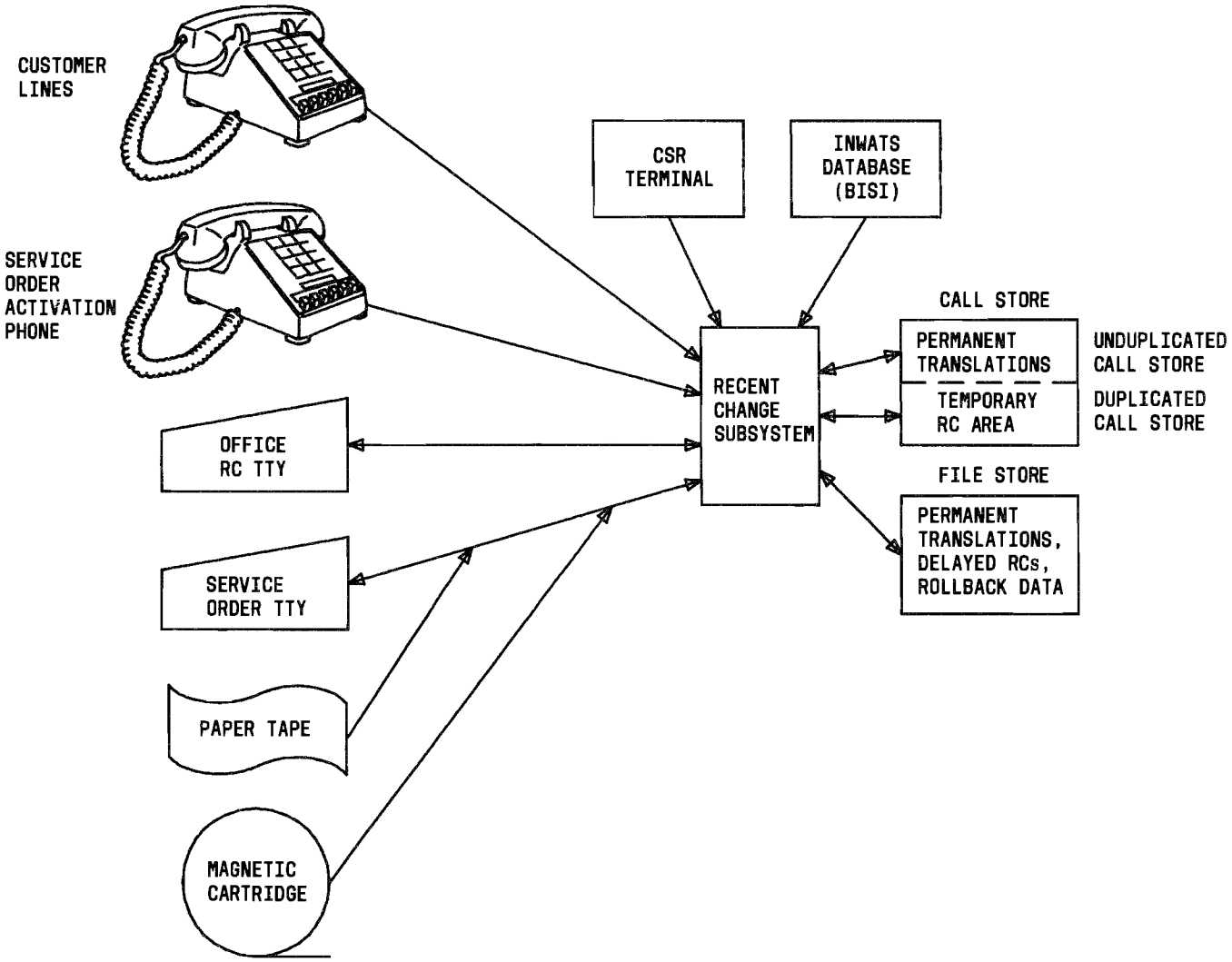

Fig. 1—Recent Change Subsystem Inputs (No. 1A ESS Switch)

protected call store with two complete backup copies stored on disk in the file stores. The recent changes that are temporary (eg, call forwarding changes) are stored in the temporary recent change area in duplicated call store, and also, in the temporary recent change backup area in file store. The file store also provides for storage of delayed recent changes and translation rollback data (ie, a copy of the original translation data, before modification, of every translation word affected by a permanent status recent change).

QUEUING OF RECENT CHANGE INPUTS

4.03 With the advent of customer changeable speed calling (CCSC), it became necessary for the RCSS to handle an increasingly large volume of inputs. The basic philosophy is that customer-originated requests for changing a speed call list entry will be queued in a searchable queue until the RCSS is free to process the entries.

4.04 The recent change queue consists of a dedicated block in duplicated call store whose size (length in call store words) is defined in parameters and, thus, is office-engineered. The queue entries are processed in the order of their occurrence in the queue. Conversely, the queue is read from bottom to top in response to a request from the speed calling

program. This ensures that the most recent information is returned for call processing purposes. For example, it is possible that two queue entries for the same speed calling list entry could exist at the same time. Reading from bottom to top yields the most recent information.

CUSTOMER LINE INPUTS

4.05 The queue control program in the RCSS interfaces directly with the call processing programs which interpret speed call list changes from customer lines. When a customer dials the proper access code to change a speed call list entry (see Feature Document 231-090-101), the call processing program decodes the request and stores the change information in a holding register. The RCSS is then requested to load the change information into the ◗ CCSC◖ queue (Fig. 2). The RCSS then ensures that the request is valid and can be loaded into the queue. Once the queue entry is processed, the speed call list change is active and can be accessed as necessary by the speed calling program.

INPUTS FROM THE OFFICE RECENT CHANGE AND SERVICE ORDER TTYs

4.06 Most of the changes to the translation data in the No. 1A ESS switch memory will be the result of recent change messages input from the TTYs, either manually, via paper, magnetic tape, or magnetic disk. Refer to Section 231-048-301 for general recent change information, including a list of all recent change messages and keywords.

4.07 The recent change messages entered via the TTY may have either immediate or delayed status. Unless entered as a delayed message, the RCSS will immediately process the message, format new or modified translation data, and then update the translation data in call store. If the recent change is of a permanent type, the translation data in unduplicated call store is changed. The translation data image in file store is then brought up to date.

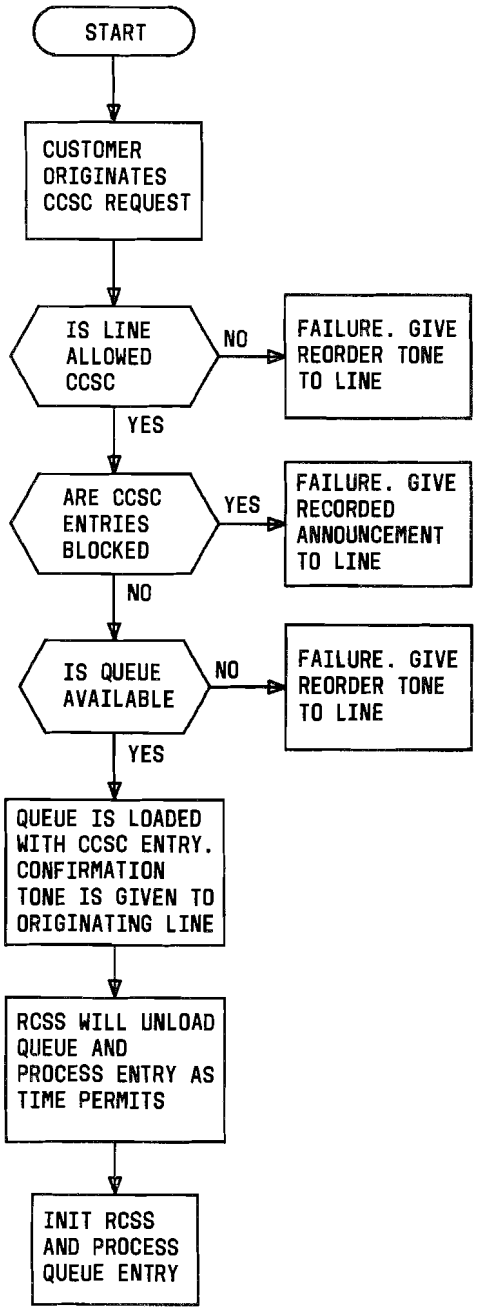

Fig. 2—Processing a CCSC Request

ROLLBACK (1A ESS SWITCH)

4.08 There will be instances where a recent change message or several recent change messages will affect the translation data in such a way as to degrade or disrupt system activities. To correct problems of this nature, it is necessary to "undo" the recent changes suspected of causing the problems by reinstating the original translation data as it appeared before the recent change message was input. This operation is termed rollback.

4.09 Each time a recent change message affecting the permanent translation data in unduplicated call store is entered, a rollback block (RBB) is

built by the RCSS and stored in the rollback area on disk. The disk rollback area is a dedicated block of disk storage whose length is Compool defined. The current size of the rollback area is 128 BINKs, where 1 BINK = 1024 (decimal) words. Each RBB, identified by an internally generated recent change order number, contains the address and original data of each translation word affected by the associated recent change message.

ROLLBACK AREA MANAGEMENT

A. Type I Rollback

4.10 Type I rollback is automatically initiated by the RCSS to remove a partially completed recent change. Each recent change message that is accepted by the RCSS and is to be inserted into the translation data base creates a single RBB in the rollback area. Each RBB contains the address and old data for each word of translations altered by the recent change. The RBB is created and stored in the file store rollback area before any updating of the translation data takes place. This is done in case the update process is, for any reason, interrupted (eg, interrupt, disk return failure, etc). If the update should fail before the entire update is complete, the RBB data is used to remove the partially completed recent change. This kind of rollback is referred to as Type I rollback.

B. Type II Rollback

4.11 Type II rollback is the process of removing or canceling recent changes which successfully update translation data and are effective with respect to call processing. Type II rollback must be requested manually, either by an input message (IM) or via the master control center (MCC).

C. Rollback Data Storage Concepts

4.12 The rollback area is logically circular, ie, the physical beginning and end of the rollback area act as though they were adjacent memory locations in a ring buffer. Thus, as RBBs are stored and the rollback area fills up, additional RBBs overlap the first RBBs stored, causing that particular rollback data to be lost. To avoid this overlap, office personnel regularly write a new translation tape. This tape is a "snapshot" of translations data at that particular point in time. When this tape dump is performed, all RBBs which were created prior to the old latest system tape are discarded, freeing up a large (approximately 40 percent) portion of the rollback area.

4.13 As shown in Fig. 3, the rollback area is presented as a pie-shaped figure to represent its circularity. Associated with the rollback area are three indexes or pointers: TAPE1, TAPE2, and NEXT. The NEXT index points to where the next RBB will be stored when a new recent change message is successfully processed. The TAPE2 index shows where the NEXT pointer was when the latest translation tape was written. The TAPE1 index, likewise, marks the position of the NEXT index when the next to the latest translation tape was written.

4.14 Looking at Fig. 3 notice that the latest translation tape dump was followed by recent change messages 13, 14, 15, 16, and 17 as indicated by the TAPE2 index. To restore translations to the state that existed when the latest tape was written, it is necessary to roll back recent changes 17, 16, 15, 14, and 13. This is a rollback to TAPE2. To restore translations to the state that existed when the next to the latest tape was written, recent changes 12, 11, 10, 9, 8, and 7 must also be rolled back. This is a rollback to TAPE1. It is not possible to roll back beyond the time of the writing of the next to the latest tape; ie, RBBs 6, 5, 4, 3, 2, and 1 cannot be rolled back and should be considered nonexistent, as indicated by the dotted lines.

4.15 When a new translation tape is written, the rollback area tape indexes are repositioned to reflect the existence of a new translation tape and the old (latest) translation tape is now the next to the latest translation tape. Refer to Fig. 3 and assume that a new translation tape has just been written and that no recent changes have been input since. The TAPE2 index is now pointing to the same position as is the NEXT index and the TAPE1 index is moved up to the old TAPE2 position. At this point, a rollback to TAPE2 (the latest translation tape) would have no effect since no recent changes have been input since the latest translation was written. A rollback to TAPE1, however, could be done. This would roll back recent changes, 17, 16, 15, 14, and 13. Rollback beyond recent change 13 (ie, the new TAPE1 index) is now impossible.

4.16 It is necessary to write new translation tapes on a regular basis to avoid filling up the rollback area and overtaking the TAPE1 or TAPE2 in-

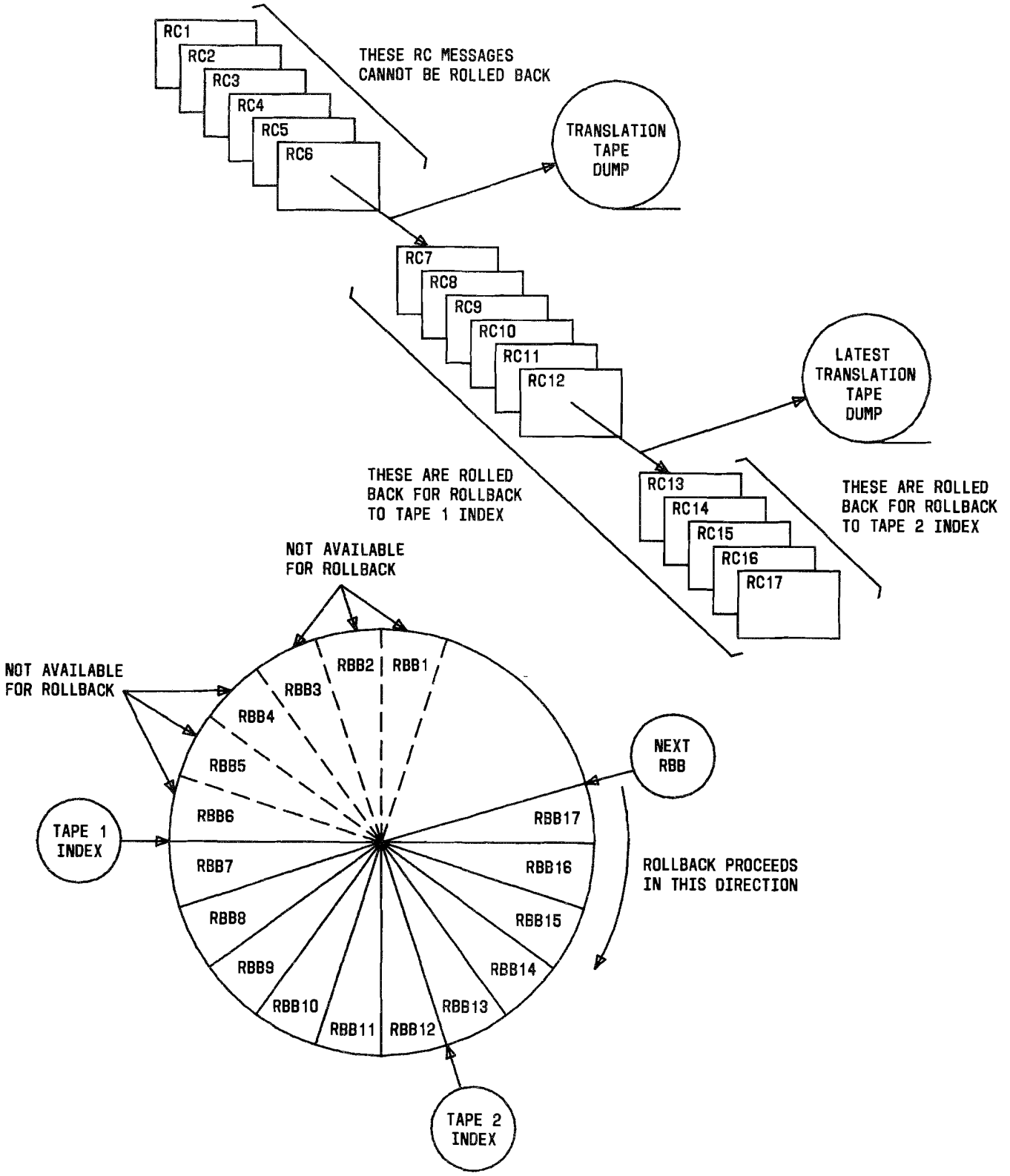

Fig. 3—Recent Change Rollback Data Storage

dexes. The frequency of writing translation tapes depends on the recent change activity level in a particular office.

4.17 The current size of the rollback area is 128K for all offices, providing storage for about 800 recent change messages, although this figure can vary considerably depending on the type of messages input.

4.18 Tape dumps may have to be performed every other day for some offices and once a week for others. The rule is that, when the portion of the rollback area used since the latest tape dump reaches 40 percent, a tape dump should be performed as soon as practical, eg, that same evening. The percentage of the rollback area used since the latest tape dump is reported by an RC:WARNING OM when the percentage exceeds 40 percent and by the REPT:RC CENSUS OM as an hourly printout or in response to the OP:RCCENSUS IM. Since no indication is given of how much of the rollback area has been used since the next to the latest tape was written, the 40 percent rule must be strictly adhered to or the TAPE1 index can be overtaken without warning. Should the TAPE1 index be overtaken, it is moved up to coincide with the TAPE2 index with a resulting REPT:RC WARNING OM stating that a rollback to the next to the latest tape (TAPE1) is no longer possible. If additional recent changes are entered beyond the 90 percent mark until the TAPE2 index is overtaken, the TAPE1 and TAPE2 indexes will be moved forward as many RBBs as necessary to store the new RBB. A REPT:RC WARNING OM will state that complete rollback to the latest system tape (TAPE2) is no longer possible. A partial rollback to TAPE2 can be performed, but the rollback is limited by the TAPE2 index which has been moved and no longer represents the state of translations that existed when the latest translation tape was written.

4.19 Type II rollback can be performed via the MCC with a Phase 4, 5, or 6 or by means of a TTY IM without going through a phase. There are four modes of Type II rollback by MCC request:

- Rollback to a particular order number
- Rollback a specified number of orders
- Rollback to the TAPE2 index
- Rollback to the TAPE1 index.

4.20 Type II rollback via the TTY is done without a phase; of course, the ESS switch must be sane enough to allow input/output via a TTY channel for this method to work. The TTY rollback is restricted to 20 decimal orders per rollback request, and only as far back as the TAPE2 index, ie, TTY rollback is limited to a rollback to the latest translation tape. Refer to the section entitled General Recent Change Information—2-Wire No. 1A ESS switch (Section 231-048-301) for more Type II rollback information.

4.21 Each RBB has an internal order number assigned to it before being stored in the rollback area. This RCSS assigned order number is completely independent of the user specified external order number entered with the order (ORD) keyword in the recent change message. The internal order number, is of the following form:

MMDDSSSS

where MM = month (1-14 octal)

DD = day of the month (1-37 octal)

SSSS = daily sequence number

(0-7777 octal)

The internal order number in addition to providing a unique identifier for each RBB, also gives information on when the RBB was created and stored in the rollback area.

4.22 Each day at midnight, the date is updated and the sequence number is reset to zero. This numbering scheme is useful in that it defines what can be called "safe rollback points." Assuming that all recent change activity is halted around the hour of midnight, due to the working hours shift change, and that translations are in a valid state so far as can be determined, it should always be safe to roll back to midnight or any particular day. This is done by rolling back through order number MMDD0000.

4.23 For example, assume that all recent change activity for June 26 ended at 10:00 p.m. and the system seemed to be in good shape. The next day recent change activity resumed and at 3:20 p.m. that afternoon serious system problems began to appear with indications of translation data mutilation due to recent change errors. The decision is made to roll back recent changes to clear the problem. Rollback of an arbitrary number of recent changes is dangerous

because the resulting state of translations at the completion of the rollback is not known. Rollback to TAPE2 or TAPE1 is maybe too drastic, possibly causing disruption of service to a large number of customers. Rollback to midnight of the previous day, a known safe point, is recommended in this situation. An MCC rollback in a phase 4, 5, or 6 through internal order number 06330000 is performed without delay resulting in the removal of the bad recent change message(s). ▶When the bad recent change is isolated and corrected, the day's recent change messages can be reentered using the rollforward procedure, floppy disk,◀ paper tape or magnetic cartridge backup. But, as a result of the rollback, ▶followed by a rollforward◀ CCSC list changes will be lost and each affected customer will have to reestablish the list. Call forwarding recent changes are kept in the temporary recent change area and produce no rollback data; consequently, they are not affected by rollback. The example illustrates the importance of the rollback data to extricate an office from service-disrupting problems caused by faulty translation data. It should emphasize the importance of keeping the rollback data clean.

4.24 An easy way to generate messy rollback data and run the risk of generating translation data errors is to try to fix a bad recent change with additional recent changes. The use of RC:PSWD is often abused in this way. The proper way to correct a bad recent change is to roll back the order and then reenter it correctly.

> ***Note:*** The following sequence is provided only as an example of a method used to roll back a bad recent change order. This is not intended for use as an operating procedure nor should it be used as such. (Refer to Section 231-048-301.)

(a) Inputs to the RCSS from delayed service order activation, the service order TTY, CCSC, CSR and BISI must be inhibited by typing:

INH:RCSOURCE DLY!

INH:RCSOURCE RCS!

INH:RCSOURCE SCV!

▶INH:RCSOURCE CSR!

INH:RCSOURCE BIS!◀

These commands will prevent additional RBBs from being created and stored in the rollback area. (Refer to the RCIG pident for RCSS control command information.) Should the recent change queue fill up during this time, customer speed calling list changes will be rejected and could cause some customer complaints. Note that call forwarding need not be inhibited since RBBs are not created for this type of recent change.

(b) List the last few orders in the rollback area by typing at the office recent change channel:

OP:RCRBLIST,LAST n! where n is a decimal integer.

The response is a list of the last n orders found in the rollback area with latest order printed first. A sample output response to OP:RCRBLIST, LAST 7! is shown in Fig. 4. Listing the last several orders in the rollback area is necessary so that, after the rollback through the bad recent change message is accomplished, it is known which recent change(s) must be reentered or rolled forward. Referring to Fig. 4, if order 06330021 is the bad message, six orders (06330026 through 06330021) must be rolled back to remove the bad recent change. The input source of these recent change orders can be recognized by the form of the external order number. If all external order numbers are assigned according to an established convention, the input source can be easily determined. For example, a service order could have an order number of at least six digits (or could have a letter prefix) to distinguish it from a delayed order which can have at most five digits with no letter prefix allowed. An SCLIST order with an external order number of 0 is obviously a customer-originated recent change, etc.

(c) Count the number of orders you wish to roll back and type the rollback message:

RCCNL:ROLLBACK,NEXT nn! where $1<n\leq 20$

In the example, n is 6. The response is:

REPT:RC WARNING ROLLBACK INITIATED AND COMPLETED THROUGH ORDER aaaaaaaa

for each order rolled back followed by a total rollback report message:

REPT:RC WARNING RC ROLLED BACK 6 ORDERS THROUGH ORDER bbbbbbbb AS REQUESTED

```
M  01   ØP:RCRBLIST   MSG   CØMPL
        RØLLBACK AREA ØRDERS SINCE THE LATEST TAPE:

            0  06330026  SCLIST
        21394  06330025  LINE
            0  06330024  SCLIST
    T2221645  06330023  LINE
    F2221645  06330022  LINE
            0  06330021  PSWD
            0  06330020  TGMEM
```

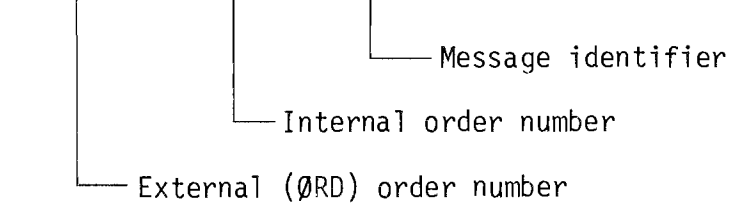

Fig. 4—Sample Output for OP:RCRBLIST OM

(d) Reenter the corrected recent change message.

(e) Reenter the recent changes which had to be rolled back to get to the bad recent change. The service bureau may have to be notified if some of the rolled-back orders were input on-line on the recent change service (RCS) order channel. Activated-delayed orders will have to be reentered from the TTY. The CCSC list changes will be corrected only after the customer discovers their list is incorrect and fixes it.

(f) Allow recent change change activity on the inhibited sources. The OP:RCCENSUS IM can be used to verify which sources are inhibited. The ALW:RCSOURCE message destroys rollforward data, and can be used to allow input once again from these sources. (Refer to IM-6A001.)

ROLLFORWARD

4.25 The use of file store to store rollforward data allows quick, automatic retrieval of data when needed. The rollforward data for a particular recent change is stored in the rollforward block (RFB) of that recent change. The RFB for a particular recent change message is created after all memory locations affected by the recent change are identified. When the recent change message passes all data checks, the updating of translation data in core and disk storage begins. One of the first stages of this update process is to create a RBB and store it in the rollback area in disk storage. It is during the generation of the RBB that the RFB is retrieved from storage and inserted into the RBB.

4.26 The number of words of rollforward data for a recent change message varies widely. In general, the length of RFB is proportional to the number of keywords input. Change orders do not follow this rule, however, since many keywords are internally generated. Current estimates indicate that on the average one-half of the rollback area is consumed by rollforward data. Since the length of the RFB is varible, an index pointer is used to point to the start of the rollback data which immediately follows the rollforward data.

4.27 Rollforward data is normally not accesssed until after a rollback has been performed. Therefore, as part of the rollback process the RFB must be extracted from the RBB and saved in an area apart from the rollback data. This area, called the rollforward stack (RFS), begins at the logical end of the rollback area. That is, the RFS is loaded from the bottom of the rollback area, whereas the rollback blocks are loaded from the top. The rollback area can thus be described as a circular "buffer". The RFS is accessed via an index pointer saved in the rollback area control block. Also located in the rollback area

control block is a 4-word block index pointing to the next RFB as well as the length, in 4-word block, of that RFB. The first RFB is linked to the second RFB and so on with a zero link indicating the end of the RFS.◀

DELAYED RECENT CHANGES

4.28 A delayed recent change message may be entered by specifying DELAY in the message format. The entire message is processed normally, including syntax and data checks, but no translation data updating is done. Instead, the message is compressed and stored in the delayed recent change area in file store. The message is not effective until the recent change message is activated. Activation is accomplished by another recent change message from the TTY or via the service order activation phone.

INPUTS FROM THE SERVICE ORDER ACTIVATION PHONE

4.29 A central office telephone set whose major class of service is "activate service order" may be used to activate delayed recent changes. The delayed recent change is activated by dialing the order (ORD) number, nnnnn, filled out to five digits with leading zeros. If no delayed message with the dialed order number is found, reorder tone is heard. ▶If the recent change interactive queue is full or another delayed activation via phone is still being processed,◀ busy tone is heard and the user may try again. If the delayed message associated with the dialed order number is found in memory, dial tone is returned and other order numbers may be dialed as appropriate. However, receipt of dial tone does not guarantee that the delayed message was activated. The TTY should be checked for an ACPT RC18 output message indicating successful activation. See Section 231-048-301.

5. PROGRAM FUNCTIONAL DESCRIPTION

RECENT CHANGE MESSAGE PROCESSING

A. General

5.01 The actual recent change messages, formats, and procedures for entering a recent change message are not described in this document. Refer to Section 231-048-302 for general recent change information, system user information, and general practice.

5.02 Each recent change message specifies an operation or set of operations which alters the contents of translation data memory. As such, each message is expressly defined in three types of tables:

(a) The message keyword table contains the keywords and data formats associated with each message.

(b) The message validity table provides the data and validity checks to be used to ensure valid data and keyword combinations.

(c) The translation format table provides the translation data layouts of the translation data to be built or modified as a result of each valid combination of keywords.

5.03 Each of the three basic table types has a set of processing routines. A combination of these tables and routines makes up three modules that are the heart of the RCSS (Fig. 5):

- Input processing
- Validity checking
- Translation data formatting.

B. New Order Processing

5.04 The input processing module examines the input message keywords, checks the associated data for proper format, and assembles the data into a convenient form for storage and access by the following modules. Only local data checking is done; any checking that requires knowledge of data in other keywords (or existing translations) is left to the validity checking module.

5.05 The validity checking module determines whether the input message is a valid combination of keywords and data. Validity checking also decides which translation format should be built as a result of the input message.

5.06 The translation data formatting module places the data received in the message into the proper format for the translation that is to be built by the message.

5.07 The message storage buffer (MSB) is a table in duplicated call store used for communication between the program modules (Fig. 5). The entire input message is stored in the MSB after the

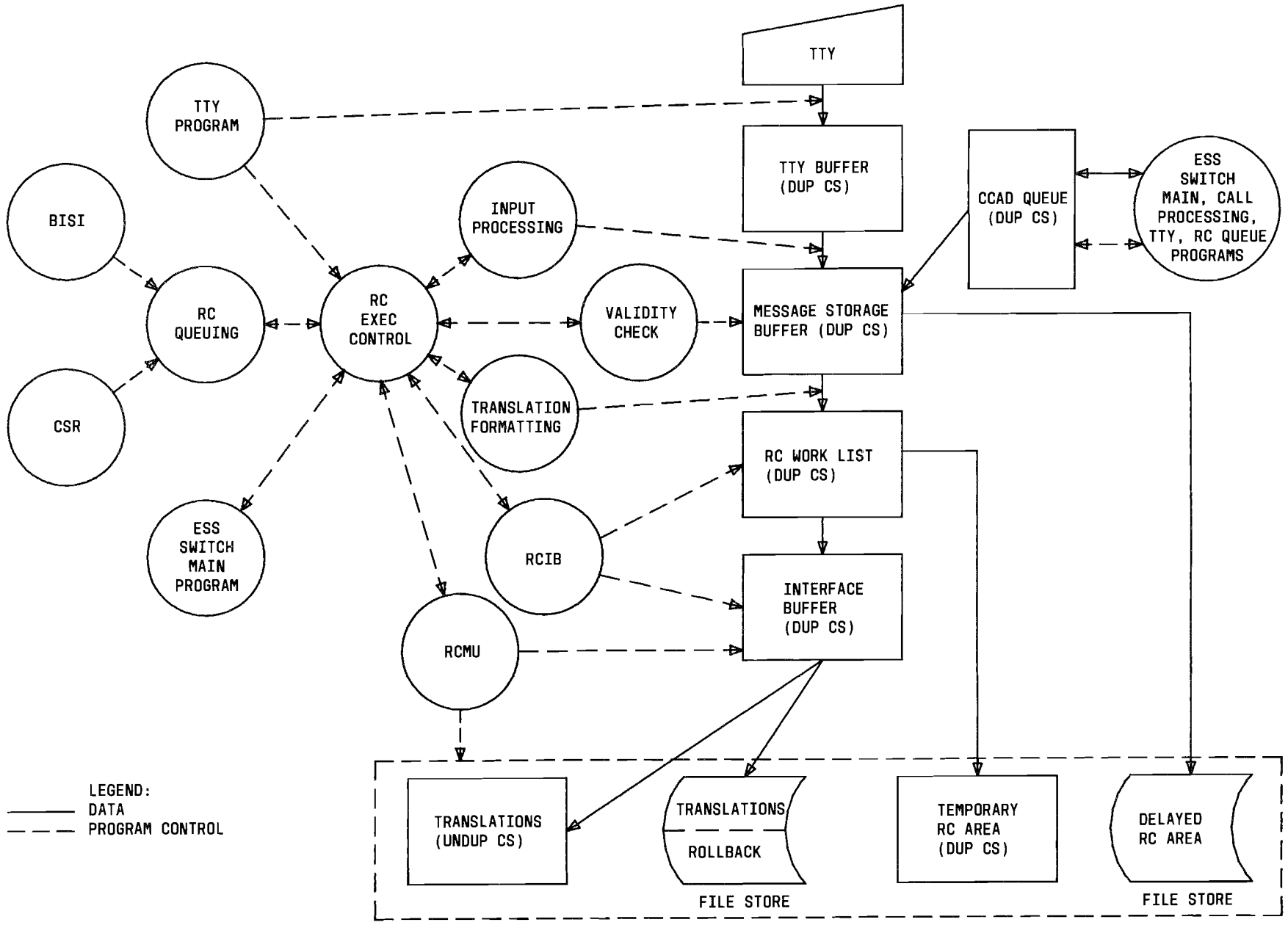

Fig. 5—No. 1A ESS Switch Recent Change Subsystem

input processing module has performed range checks and appropriate conversions on the data. This allows an arbitrary order of keywords within a message and more efficient validity checking since all of the data is available when validity checking is performed.

5.08 After validity checking is accomplished, the format module reads the MSB and builds the proper translations for storage in the recent change work list. The RCIB then formats the translation data into an interface buffer which provides temporary storage for RCMU. Rollback blocks are built and the translation data is then stored in translation memory.

5.09 The sequence just described traces the basic flow through the RCSS for a NEW order. The additional recent change order types, ie, CHG and OUT with immediate (IMMED) or DELAY options, are discussed on a basic internal system level in the following paragraphs. Detailed information is provided in the program description portion of this section.

C. Change and Out Order Processing

5.10 The change processing module allows recent change messages (CHG or OUT) to be specified for translations that are currently in the active state. For a CHG message, information from an existing translation is combined with data from the input message (IM) to produce a modified translation. For example, a customer may wish to buy call forwarding service. Using one CHG message, this feature may be added to the customer's line without respecifying or even being aware of the other features already assigned to the line. This is possible because old information is retrieved from the customer's existing translation instead of being required in the IM.

5.11 The change processing module moves all the information from an existing translation into the MSB, except where data from the IM has already been stored in the MSB by the input processing module. The combination of data from the IM and the existing translation makes the MSB look exactly as it would have looked if a NEW type recent change message had been typed to build the desired translation. After the change processing module has finished, the message can be treated as a NEW type message.

5.12 An OUT message is used to undo everything done in a NEW order. The call store blocks are returned to the idle link list and unassigned codes are entered into the translators. By using the format interpreting routines that are used to build translation formats for NEW orders, OUT order processing is able to locate all blocks that are to be returned and all translators that are to be unassigned.

D. Delay Order Processing

5.13 Delayed recent change messages are entered from the TTY and processed into the MSB (Fig. 5). The message is thoroughly checked for valid data and keywords as is any recent change message. If errors are present, normal error responses are given. After the complete recent change message is assembled in the MSB, normal message processing stops. No translation formatting is done and the translation data is not modified. Instead, the MSB image, in compressed form, is stored in a DAB on disk. The delayed recent change area on disk provides storage for approximately 500 DABs, ie, one DAB for each delayed recent change message.

5.14 A directory at the start of the delayed recent change area provides a 2-word entry for each DAB. Word one contains the order number associated with the delayed recent change message; word two contains the starting disk address and the length of the DAB. At activation time, the delayed recent change activation routines locate the DAB via the directory and expand the recent change message back into the MSB. The MSB now contains the recent change message exactly as if it had just been entered from the TTY. Message validity is checked again before any translation formats are built. Normal recent change message processing then continues.

PROGRAM DESCRIPTIONS

A. Control Pidents

◗ QUDB

5.15 The QUDB is the queue control program of the RCSS. Any message which is entered from any recent change source must be queued before it can be processed. The QUDB provides the queuing, queue service, queue release, and queue audit function for all of the RCSS queues. The following are the different queues within the RCSS:

(a) Recent Change Administrative IM Queue (No. 1A ESS switch only)

(b) Recent Change TTY Queue

(c) Recent Change Interactive Queue

(d) Recent Change Batch Queue

(e) CCSC Queue.

5.16 The recent change administrative IM queue is used for all recent change administrative IMs which require the use of RCMU. This queue has the highest priority in queue service, as the messages which use it are generally trying to clean up garbaged recent changes. These messages should therefore be given priority over normal recent change message processing. An entry on this queue will be served as soon as a recent change scratch block becomes available.

5.17 This algorithm is used because the entries on the recent change interactive queue require higher priority than the recent change batch queue and the CCSC queue. The recent change interactive queue has human client waiting for a response at the end of processing the recent change message. On the other hand, the recent change batch and CCSC queues do not have a human client waiting for a response. Therefore, their priorities are not as great as the recent change interactive queue.

5.18 The audit of the queues is an addition to Audit 4 which goes through the headcells of all of the queues and checks for any garbaged data within those queues. If bad data is found, the entire queue is reinitialized, thus removing any entries on the queue at the time, and an audit error is printed.

5.19 The recent change TTY queue is used for all recent change messages which are input from the service order or recent change office channels. This queue is given second highest priority behind the recent change administration IM queue. An entry on the recent change TTY queue will be served when a recent change scratch block becomes available. However, there should be nothing on the recent change administrative IM queue to be served and no other TTY recent change message or segmented message in input processing.

5.20 The recent change interactive queue is used for recent change messages which originate from centrex station rearrangement (CSR) clients and for delayed activation of service orders via phone.

5.21 The recent change batch queue is used for recent change messages which originate from the INWATS data base for the Busy/Idle Status Indicator (BISI) feature.

5.22 The CCSC queue is used for customer changeable speed calling changes made by customers via the phone.

5.23 The recent change interactive, recent change batch, and CCSC queues are served when a recent change scratch block becomes available and when neither the recent change administrative IM queue nor the recent change TTY queue is served. The priority of service for these three queues is as follows:

(a) Recent change interactive queue—16 out of 24 times

(b) Recent change batch queue—4 out of 24 times

(c) CCSC queue—4 out of 24 times.◀

RCIG

5.24 The RCIG is the input screening, initialization and general control module of the RCSS. It provides IM and channel screening, system initialization for message processing, control of the message processing sequence, and control of several auxiliary recent change functions such as TTY rollback and listing rollback area orders.

5.25 The RCIG screens all inputs from the following sources of recent change input:

(a) Service order TTY (including paper tape/ magnetic cartridge or magnetic disk)

(b) Office recent change TTY.

5.26 In addition, the office recent change TTY is the most important source of recent change information. Any recent change message and all recent change administrative messages can be entered via the office recent change channel. The office recent change channel is selected from among those channels specified in the recent change IM class. Currently, the channels in the recent change IM class are:

(a) LOC—local maintenance TTY

(b) REM—remote maintenance TTY

(c) SC1—switching control center TTY No. 1

(d) SC2—switching control center TTY No. 2

(e) SRM—supplementary remote maintenance.

The RCS (service order TTY) is also in the recent change IM class but cannot be selected as the office recent change channel.

5.27 After a system reinitialization (SR) or phase which zeros call store, the office recent change channel is null. To assign an office recent change channel, type at any recent change input class channel:

ALW:RCCHAN aaa! where aaa = LOC
= REM
= SC1
= SC2
= SRM

5.28 The system response to the above IM is:

REPT:RC WARNING

OFFICE RC CHANNEL = aaa

This response is accompanied by a spurt minor alarm. It is not intended that the office recent change channel be casually switched from channel to channel.

5.29 As part of its input screening function, RCIG checks that any recent change message or recent change administrative IM was input on a legal TTY channel. The legal input channels from the various IMs are shown below:

Type of Message	Legal Input Channel
RC:	RCS, Office Recent Change
ALW:RCCHAN	LOC, REM, SC1, SC2, SRM
ALW:RCSOURCE	Office Recent Change
INH:RCSOURCE	Office Recent Change
OP:RCCENSUS	RCS, Office Recent Change
OP:RCRBLIST	RCS, Office Recent Change
RCCNL:ROLLBACK	Office Recent Change
RCACT:ROLLFWD	Office Recent Change
OP:RCDYLIST	Office Recent Change
OP:RCRFLIST	Office Recent Change

When RCIG determines that a recent change message (RC: or otherwise) was input on an illegal channel, it acknowledges with a ?C TTY acknowledgment, and the RCSS is otherwise unaffected.

5.30 The RCIG also checks the status of the RCSS and the allow/inhibit flag associated with the recent change source. If the input source is inhibited, the RCSS acknowledges the request by sending an RL,<LOCKOUT> TTY acknowledgment to the input channel, where RL indicates repeat later. If the source is not inhibited but the appropriate queue is full, the RCSS returns an RL,<BUSY> TTY acknowledgment.

5.31 The RCIG performs an SR for processing recent change requests. An IM which begins with "RC:" is passed to RCIG at subroutine RCINIT from the input/output program (IOCP). After queue service of the request, control is passed to a routine (RCINIT1) to process the first input line. The entire input line is passed to RCIG in the channel memory block. From this input, RCINIT1 extracts the message index (LINE, TRUNK, PSWD, etc); type (NEW, CHG, or OUT); and the action option (IMMED or DELAY). If an input error is detected in the heading, an invalid heading (IH) or error (ER) TTY acknowledgment is given and the channel is released. If no errors are found in the first line, a time break is taken to wait for the next input line.

5.32 All IMs which are not of the new No. 1A ESS switch format, ie, all converted No. 1 ESS switch IMs, have to be screened by the RCSS in case the RCSS is expecting input on that particular channel. There is no way to distinguish between a converted No. 1 ESS switch IM and an input line of a recent change message. Thus, all nonstandard IMs are passed to RCIG from the input/output program. The RCIG routine, Recent Change TTY Message (RCTTYM), checks to see if a recent change message line is expected on the input channel. The RCTTYM compares the input channel number with the channel number stored in the usage word of each recent change scratch block, which is in use by a TTY in

input processing. If the two do not match, the input is passed to pident TTYM as a converted No. 1 ESS switch IM. If the input channel numbers match, the input is assumed to be the next line of a recent change message being processed as described in pidents RCIE and RCKI.

5.33 The general control portion of RCIG directs the flow of program control among the other control modules in the RCSS, according to the type of IM being processed. The four basic parts of the general control module are:

(a) Data tables

(b) A sequence initialization routine

(c) The sequencer

(d) A routine used to exit from the RCSS.

5.34 There are several data tables used to control the sequencing of RCSS control modules. Table GSEQNO translates the type of message (NEW, CHG, or OUT) and action option (IMMED or DELAY) into a sequence number. This sequence number is then used as an index into a second table, GSEQTAB, which consists of lists of the control modules arranged in the order in which they are to receive control for each sequence. Two call store program state words, R2PSW1 and R2PSW2, are used by general control to record the number of the module currently in control.

5.35 Sequence initialization routine translates a code in item R2ISEQ of state word R2PSW2 into a sequence number, using table GSEQNO. The R2ISEQ is set by the initialization module on the basis of the message type (NEW, CHG, OUT) and the action option (IMMED or DELAY). The module number of the initialization module is stored in item R2LUN of state word R2PSW1 to indicate that the initialization module was the last to have control.

5.36 The general control sequencer selects the flow of control between modules, based on the type of message being processed. It does this by indexing into the sequence table GSEQTAB with the sequence number in program state word R2PSW1 to get an ordered list of modules for the sequence being processed. The module number stored in program state word R2PSW1 is used as an index into the sequence to derive the next module that should get control. This module number is stored in program state word R2PSW1 to indicate that the module now has control. The traffic load of the office is then determined, and R2VSEGIN is set to indicate the amount of work that must be performed between real time breaks, according to the module that is to receive control. Control is then passed to the proper module through the subroutine linkage mechanism. Every module returns to the address in J with registers restored.

5.37 The RCSS exit routine checks item R2SUBM of program state word R2PSW1 to determine if the message just processed is complete or is a segmented message. For a segment, the RCSS is reinitialized for the next segment and the general control sequence is restarted. For a complete message, if there were no errors, an accept message is printed, the internal order number daily sequence number is incremented, and control is returned to the ESS switch main program via routine RSQUIT.

5.38 The last function of RCIG is to control the recent change administrative functions. These functions are:

(a) Listing the orders in the rollback area

(b) Inhibiting/allowing the various recent change input sources

(c) Processing the ALW:RCCHAN IM,

(d) Processing the OP:RCCENSUS IM

(e) Performing TTY rollback operations

(f) Automatically inhibiting CCSC if the recent change audit finds a problem in the recent change queue.

RCIE

5.39 The RCIE is the recent change input editor and does the initial processing of the input characters passed in the TTY buffer from IOCP. The RCIE is stimulus-controlled in that the client programs (either RCIG for first line processing or RCKI for keyword processing) turn it on by requesting a line of input. The functions performed by RCIE are as follows:

(a) Character translation, from 7-bit ASCII to recent change character code, and loading into line buffer

(b) Character screening to reject any characters not in the allowed character subset and to discard ignore characters (like RUBOUT and XOFF) and all characters within quotes

(c) Determination of end of line so a complete line can be passed back to the client

(d) Loading of input error TTY acknowledgment or line accept TTY acknowledgment.

5.40 The RCIE is structured as a state table with the translated recent change character codes used as input to get the next state and an associated work routine. For example, if the system is in the state GETLIN (get a line) and the input is "/" (slash), the next state found in the state table is FORCHK (check the input format) and the work routine will return the line to the client program. The characters in the TTY buffer are unloaded one at a time via an RSUB routine. The characters are immediately translated by the XLATE table into recent change character code and stored in the recent change input buffer (Fig. 6).

RCKI

5.41 The RCKI is the recent change keyword input program and performs the processing and local error-checking of the keywords of a recent change message. The descriptions of the valid keywords and data formats are read from 12-bit word tables, the message keyword table, and the supplementary input table. Word layouts of the 12-bit tables are contained in the program listing.

5.42 The input consists of the keyword units of the message read from the recent change input buffer (Fig. 6). Based on the format information stored in the 12-bit input tables, the keyword data is read and checked for errors. The checks performed by RCKI are local to each keyword unit. The message is rejected if a keyword is not valid for the message, or if the data (if any) has the wrong format or range. The assembled data is then stored in the MSB, and the corresponding keyword-received bit in the truth value table is set. If more than one call store word is required for keyword data storage, an MSB auxiliary block is used.

5.43 Error termination can be caused by two types of errors, user errors and internal errors. An

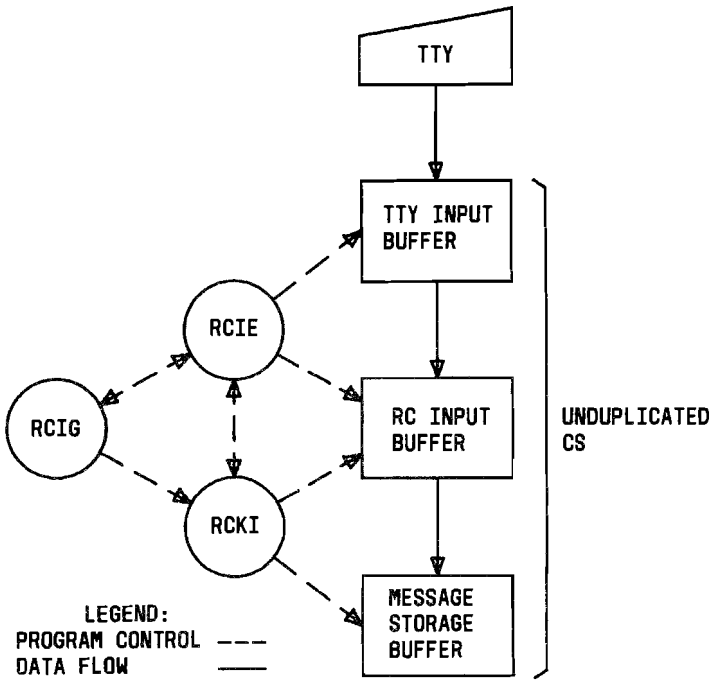

Fig. 6—RCSS Input Processing

error code is assigned to each type of user error, such as invalid keyword or invalid octal digit. The error code, along with the octal line and column where the error is located, is passed to the RSPRNT routine. The RSPRNT routine uses the error code in a correctable error TTY acknowledgment if the line ended with a / (slash), or in an INPUT type of an RC18 OM. Refer to Section 231-048-302 for error code explanations.

5.44 An internal error results from defensive programming checks. For example, if the control pushdown list (which saves 12-bit addresses while jumping within the supplementary input table) were to reach an overflow condition, an internal error would result. The program state at the time of the error is reported by an RC16 internal error OM via a call to the RSPBUG routine.

RCVC

5.45 The RCVC is the recent change validity check program. This program processes the validity tables contained in a message pident to ensure that the combination of keywords with their input data and the existing translation data (in memory) will produce a valid translation change. Valid translation changes are determined by translation and call processing restrictions and the administrative restrictions (eg, lines with centrex major class of service cannot be changed to plain old telephone service). The RCVC also forms a bit string called the format selector which is used on a bit-by-bit basis by the format pidents (RCFI, RCTF, RCCH) to build the correct formatted translation data.

5.46 The validity tests of each message are specified within a table consisting of 12-bit words. Layouts for the individual words are contained in the RCVC program listing.

5.47 Each validity section of a pident has three contiguous parts:

- Data check
- Logic check
- Format selector definition.

There can be up to three validity sections depending on the message type (NEW, CHG, or OUT). Each section is ended by an all zero 12-bit word, and if a section is not needed, a single all zero word marks its absence.

5.48 In addition, validity check processing results in a bit string stored in the format selector. This bit string defines a translation format and is used by the format section.

RCTF, RCFI, RCCH (Format Section)

5.49 The format section of the RCSS (RCTF, RCFI, and RCCH) builds images of translation data generated by NEW and CHG type messages and stores them in the recent change work list. For CHG and OUT messages, the format section retrieves existing translation data from translation memory in unduplicated call store.

5.50 The format section uses three RCSS interface tables to communicate with the rest of the system:

- MSB (R2MSB)
- Truth value bits (R2TRVAL)
- Format selector (R3FORSEL).

5.51 The MSB holds keyword input data and some data generated internally by the data check section of RCVC. The truth value bits indicate which keywords have been received as input and which data checks and nodes are true. The format selector consists of ♦five♦ call store words with the appropriate bits set to indicate the exact layout of the translation data being prepared.

5.52 Two types of data tables are used by the format section:

- Translation format table
- Change table.

Each recent change message has both types in its pident, describing the format of the translation data for that message.

5.53 The translation format table consists of 12-bit words arranged into units called operations that are interpreted by the format section. The sequence of operations in a message's translation format table provides for all possible arrangements of

translation data that can be built by that message. The 12-bit data words associated with each operation in the translation format table are generated at compile time by a SWAP macro call describing that operation. The collection of all SWAP macros used to generate the translation format table forms a language to describe the translation formats. The format selector words are used to conditionally execute the translation format operations and thereby build the translation as specified in the recent change message.

5.54 The change table for each message consists of three parts:

- Change screening table
- Change transition table
- YES/NO and data/NO keyword list.

5.55 The change screening table contains information used by the format section while retrieving translation data from translation memory. It describes some required relationships between translation data and keyword inputs as well as updating keyword, truth value between change and new pass of a message. For example, when extracting data from translations for a change order, a data item of zero ordinarily means that the item is unassigned. If a zero value indicates valid data for a keyword, this fact would be recorded in the change screening table.

5.56 The change transition table describes keywords that must be moved on CHG type messages. For example, the keyword New Telephone Number (NTN) in the RC:LINE message must be moved toTelephone Number (TN) after the old TN is unassigned so that NTN will be assigned.

5.57 The YES/NO and data/NO keyword list contains all YES/NO and data/NO keywords so that set or reset the truth values for the keywords on change messages can be performed on them after translation data retrieval.

5.58 The format section operates in two passes:

- NEW pass
- CHG pass.

5.59 The NEW pass interprets the translation format table for the message being processed under control of the format selector, to build the translation data required by the IM. This table describes to the program what input keyword, truth value bit, or format selector bit is associated with each item in the translation data. Moving the keyword data from the MSB, truth value bit, or format selector bit, the NEW pass builds the item of translation data in the recent change work list.

5.60 A CHG message is processed as an OUT message followed by a NEW message. That is, all translations are internally unassigned by a CHG pass and then modified and reassigned by a NEW pass. Interpreting the translation format table for the message being processed under control of the format selector, the CHG pass moves each item of existing translation data (from unduplicated CS) into its proper MSB location or truth value bit. As it goes through the translation data, it also generates the recent change work list entries to unassigned translations and returns all blocks involved to the idle link list. After the CHG pass, the MSB looks exactly as it would immediately after a NEW message had been input, and the data in translations has been modified so that this "NEW" message is acceptable.

5.61 The OUT messages stop after the CHG pass because all the work list entries necessary to unassign the translation have been made.

RCFI

5.62 The RCFI is a set of format interpretation subroutines that administer the scratch area in memory and decode translation format table operations. These subroutines are called by RCTF and RCCH. As each translation format table entry is decoded by RCFI routines, the format scratch area in unduplicated call store is updated with outputs for use by RCTF and RCCH as well as updating keyword, truth value between change and new pass of a message used by subsequent RCFI routines. Memory word layouts are provided in the RCFI program listing.

RCTF

5.63 The RCTF processes the NEW pass of format, using the MSB, truth value bits, format selector, and translation format table, and then constructs a translation data image in the recent change work list. The RCTF keeps track of where it is currently building translation data in a scratch word

called the present word pointer. As each translation format table operation is read for RCTF by RCFI, the present word pointer is moved, or information is entered into the word pointed to by the present word pointer, on the basis of data that has been assembled in the format scratch location by RCFI. The unit of translation data assembly by RCTF is the item, a set of one or more contiguous bits within a word. For each item described by a translation format table entry, the contents of the MSB location, truth value bit, format selector bit, or constant that goes into that item, are inserted into the word pointed to by the present word pointer.

RCCH

5.64 The RCCH processes the CHG pass of format. Using the translation format table and format selector, RCCH retrieves items from existing translations and moves them into the MSB and sets the corresponding truth value bits. The RCCH also uses the present word pointer in format scratch to keep track of which translation data word it is currently examining. As each translation format table operation is read for RCCH by RCFI, the present word pointer is moved or an item is moved from the word pointed to by the present word pointer into its appropriate MSB location or truth value bit. As each translation data item is examined, it is looked up in the change screening table to see if it needs special error screening or updating keyword, truth value between change and new pass of a message. Some translation format operations also cause recent change work list entries to be made to unassign the translator being examined and to give back associated blocks of translation memory to the idle link list of available space.

5.65 After the entire translation format table has been interpreted by RCCH, two cleanup functions are performed. First, the change transition move table is examined to see if any keywords should be moved before the NEW pass of the system. A keyword move means moving the contents of one MSB location to another. Second, the YES/NO and data/NO keywords in the YES/NO and data/NO keyword list are made to appear as they would have had they just been input.

RCWL

5.66 In the No. 1A ESS switch, RCWL performs the first pass processing of the recent change work list entries made by the format section pidents. A work list entry consists of a gauge word and two or more data words. The gauge word contains a 5-bit right-adjusted item, the gauge, which characterizes the work list entry (there are 15 different types). The RCWL has the primary task of obtaining call store space for the eventual storage of Auxiliary (AUX) block data. The work list entries are altered somewhat in the process for subsequent processing by RCIB.

RCIB and RCMU (No. 1A ESS switch Only)

5.67 The RCIB takes the translation data built in the work list and converts it into a form acceptable to RCMU. The data is formed into primary change blocks (PCB) which are stored in the interface block. Each PCB is a block of 4 to 36 words containing a call store address and 1 to 32 consecutive words of data to be written starting at that address. A single recent change message may result in several PCBs being built to overwrite translation data. The RCMU reads the interface block and writes the translation data directly into translation memory (unduplicated call store) making the recent change immediately effective. The disk image of the translation data is brought up to date, and the rollback blocks (RBBs) are built and stored in the disk rollback area.

RCDY

5.68 The RCDY processes recent change messages which are entered on a delayed basis. The MSB image of the delay recent change is compressed (after validity checking) and stored in the delayed recent change area on disk. A directory at the start of the delayed recent change area contains a 2-word entry for each delayed recent change (Fig. 7). The recent change order number (enter with the recent change message) identifies each delayed recent change message and is the first word of a 2-word entry in the directory. The second word contains the relative address which points to the DAB containing the condensed MSB image of the recent change message. The second word also contains the length of the associated DAB.

5.69 At activation time, RCDY locates the desired DAB by equating the order number specified in the activate message with the order number in a

particular 2-word entry in the directory. The start and end disk addresses are then computed from the second word in the control block entry, and the complete recent change message is expanded back into the MSB. Once the MSB is loaded with the recent change message, RCDY returns control to the normal message processing routines where validity is again checked and normal message processing resumes. The message is then processed to completion as if it had just been input.

RSUB

5.70 The RSUB consists of more than 60 subroutines used by the other control pidents. Subroutines are included in RSUB if they were presently or potentially called from more than one pident.

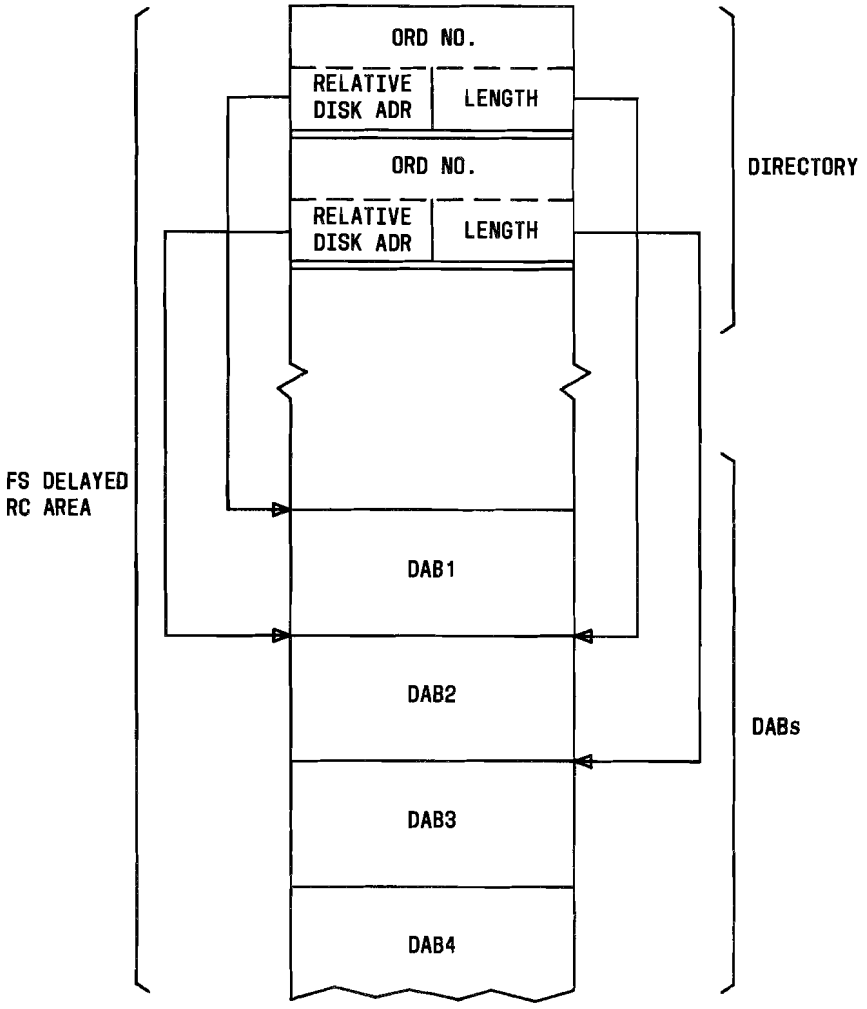

Fig. 7—Delayed Recent Change Message Storage

B. Shared Pidents

RCTS and RCSI

5.71 The shared pidents, RCTS and RCSI, are sometimes referenced while using other pidents in connection with problem analysis procedures. The RCTS consists of a collection of special purpose subroutines, used by the control pidents, which provide functions that are too specialized to be handled efficiently by the macro languages of input, data check, and format.

5.72 The RCSI consists of five data tables accessed by several code pidents during recent change message processing. The tables are as follows:

(a) Message Head Table: The message head table contains a vector for each recent change message pident. This table is used by control pident RCIG.

(b) Tag and Assignment Table: Each table entry provides the structure of a translator in terms of head table, number of bits in selector and index, PTW (primary translation word) and head table unassignment codes, and power of expansion values. These tables are used by the RSGTAG routine in control pident RSUB when generating the TAG (program store address) of a translator.

(c) Parameter Table: This table is a Transfer Vector (TV) table for the parameters used in format checking of the keyword data. This table is used by control pident RCKI.

(d) TV Table for Supplementary Input Tables: Each table entry contains relocatable addresses of two separate supplementary input table assembly routines. The table is used when the entry in the keyword table of a message pident points to a supplementary input table assembly routine in RCSI for special keyword data processing. This table is used by control pident RCKI.

(e) Supplementary Input Table: Each table entry consists of two or more 12-bit words that describe the data of a keyword and indicate whether to store the data in the MSB or MSB auxiliary area. Any of the recent change message pidents can point to entries in this table. This table is accessed by control pident RCKI, using the TV table.

C. Message Pidents

5.73 There is a separate message pident for each recent change message (with the exception of "RC:ACT:" for delayed messages, which is contained in RCDY). In general, there is a separate message pident for each type of translator (with a few exceptions such as RCLI, which affects both directory number and line equipment number translators; and TG and TGBVT, both of which affect the same translator). Each message pident is made up of 12-bit data tables which are used to drive the control pidents. There are six types of tables:

- Message Head
- ⧫Keyword Data Assembly⧫
- Keyword
- Validity
- Format
- Change.

These tables provide the decision-making information necessary to control the processing flow in the associated control pidents.

Message Head Table

5.74 The message head table contains the following:

(a) Pident message identification

(b) Relative address of associated tables and subtables

(c) Types of recent change messages allowed (NEW, CHG, OUT) unless prohibited

(d) Number of associated keyword units

(e) Number of MSB entries.

Note: The MSB is a storage facility in the CS area. The MSB consists of the MSB proper and auxiliary areas. The MSB proper consists of a 1-word entry assigned for each keyword in the recent change message, ▶1-word entries for internal keyword,◀ and the MSB auxiliary area is for use when the 1-word entry is inadequate to store the keyword data.

Keyword Data Assembly Table

5.75 ▶The Keyword Data Assembly Table (KDAT) is used for supplementary input routines which are required only for a particular recent change message. These routines are put in the KDAT table rather than in the RCSI. Consequently, the RCSI is for routines used by more than one pident. If a routine is used in only one pident, it is put in the KDAT table in the message pident.◀

Keyword Table

5.76 The keyword table contains all valid keywords for the message in condensed hashed form. (This hashed form is a number created by combining the keyword character codes according to an arbitrary, but fixed algorithm, to achieve a compact code for the total keyword.)

5.77 For each keyword, the table contains the following:

(a) Hashed form of the keyword

(b) Either a description of the data that must accompany the keyword, including a limited number of basic checks such as type of data, number of digits, for more complex forms of keyword data, or a pointer to the description of supplementary input tables located in either the KDAT table or the shared pident RCSI

(c) An indicator for allowing the keyword to be repeated in segmented messages

(d) A word containing the actual characters of the keyword (the first two and the last two characters if the keyword is more than four long) used to print back recent change messages for the ▶ OP:RCDYLIST and OP:RCRFLIST messages with the detail option. (No. 1A ESS switch only). ◀

5.78 There can also be a:

(a) Internal keyword: Scratch and storage area used by message pident.

(b) Keyword equivalence: Specifies equivalent alternative keyword name (ie, NTN synonymous to NEWTN or CFN synonymous to CFTN).

(c) Keyword collision table: If two keywords hash value are the same, both keywords are placed in the collision table. In collision table first, second, and last two letters are used for keyword name.

(d) CSR keyword equivalence: Specifies equivalent alternative keyword CSR keyword name inputs.

(e) CSR keyword collision: If two keywords hash value are the same, both keywords are placed in the CSR collision table. In CSR collision table first, second, and last two letters are used for keyword name.

(f) CLOG/AMA keyword table: Used for customer originated recent change output messages to print keyword names.

Validity Table

5.79 The number of validity tables contained in a message pident varies in accordance with the number of message types (NEW, CHG, OUT, or equivalents) allowed by the message. Typically, a validity table is provided for each message type allowed, but certain validity tables are combined in some messages, such as one combined table for both CHG or OUT messages. Each subtable name is prefixed by NEW, CHG, OUT, or equivalents. Normally, validity tables are provided for the following major functions:

(a) Data Check Table: The data check table contains the detailed checks performed on keyword data (which includes comparing keyword data with translation values, checking translation assignments, and testing for specific values of keyword data) and the algorithms to calculate internal keyword values.

(b) Logic Check Table: The logic check table contains ➧Boolean tests to be performed on previously defined truth value bits for keyword units, data checks, or logic checks. These tests check the consistency of the input keywords, and existing translations for valid combinations of keywords and data. If the tests are successful the truth bit for that test is set to true.➧ Some checks cause RC18 messages to be printed if they fail the set of tests. (Section 231-048-302)

(c) Format Selector Table: The validity format selector table contains a set of Boolean operations to be performed upon the truth value bits to derive the format selector bits.

Format Table

5.80 The translation format table contains the data format description of the translator words changed by the recent change message. Each item in a word of the affected translator is described in terms of size, displacement, auxiliary word number, etc. The new contents of the translator words and items are described in terms of:

- Truth value bits
- Format selector bits
- Data in the MSB
- Data in the MSB auxiliary area.

The RCTF uses this data to build translation data from MSB data and the MSB auxiliary area data. The RCCH uses it to reconstruct MSB data, MSB auxiliary area data, and truth value bits from old data in translations.

Change Table

5.81 The change transition screening table is only used to process a CHG or OUT message; therefore, change tables are not provided for many messages. The table contains the following:

(a) A change keyword list that is used at the end of the CHG pass of the format table to move data from one keyword to another in preparation for a NEW pass of the format table. The list allows truth table bits for the moved data to be updated.

(b) Comparison checks between old and new keyword data (a match failure results in a CHGER-type RC18 message).

(c) All YES/NO and data/NO keywords of the message.

ERROR PROCESSING

A. User-Detected Errors

5.82 The simplest form of error detection is recognition by a user, while typing the message (either on-line or perforating a tape off-line). One way to correct such an error is to use the underscore, each occurrence of which effectively backspaces one character. (Thus, LEM_N internally becomes LEN, and TN456 __ 456 becomes TN 456). The underscore, however, cannot undo the effect of a control character (!, /, %, carriage return, etc).

> ***Note:*** Underscore is for model 35 TTY. Model 43 TTY uses the backspace key. On CRTs the cursor is repositioned to the error and the correction mode.

5.83 Another way of correcting a user-detected error is to cancel the current line of input with the dollar sign ($). An ampersand (&) cancels the current message or, if segmented, the current segment of the message.

B. Input Format Errors

5.84 If a format error is made and the line is ended with a slash (/) or percent sign (%, for segmented messages only), the system responds with a correctable error TTY acknowledgment identifying the type of error and the column number (number of character spaces, in octal) of the error. The line in error is automatically canceled, and may be immediately retyped. These errors are also called correctable errors. Since they are detected by the input processing program, they can be corrected without repeating earlier input lines. Details of these TTY acknowledgments are given in Section 231-048-302.

C. Validity and Translation Errors

5.85 These errors are only detected after input is complete and the total input is checked for internal validity and consistency with translation data already existing in the system. These errors are uncorrectable in the sense that the entire message (for segmented messages, only the last segment must be reentered, but all keywords must be entered) must be reentered. Detection is announced by the TTY ac-

knowledgment "ER" following the end of message (!) or end of nonfinal segment (%), and the explanation of the error is given in subsequent output message (OM). These OMs are described in Section 231-048-302.

D. Internal Errors

5.86 The RC16 INTERR (internal error) printout, as a response to a recent change message, indicates some type of abnormal message termination. (An abnormal message termination is in contrast with a normal message termination such as accept, reject for input or validity, or reject for insufficient recent change memory). The internal error termination is accompanied by an extensive octal dump of key recent change message control and data words to permit a thorough analysis of the problem causing the termination. The analysis of the dump requires an understanding of the RCSS structure. Assuming this, the following paragraph gives clues of how to analyze an RC16 INTERR problem, with some examples.

INTERR Data

5.87 The first RC16 INTERR printout, consisting of six words, gives the central control register at the time of detecting an internal error. Of most importance here is the L register, which gives the code pident location at which the error was detected. By going to the location in the pident listing pointed to by L, one can find a description of the reason for the internal error, an interpretation of the other central control registers, as well as use of the other INTERR printouts.

5.88 ***Update Errors:*** If an error occurs during the update of the translation data (eg, a disk request did not complete) a Type I rollback is automatically initiated to remove any part of the recent change which may have been completed through translation update. Thus, if any part of the recent change update fails, the entire recent change message fails and must be reinput from the beginning. The REPT:RCFAILURE OM reports the update failure, and this normally occurs following the "IP" TTY acknowledgment printed after the end-of-message character (!). It can also occur after the segmented message character (%) since each segment is a complete recent change in itself. (See Section 231-048-302 for details.)

5.89 ***Interpreting the Central Control Registers:*** Although the contents of central control registers depend on the specific INTERR call, the programming standards adopted by the RCSS permit some general statements about their probable contents. The order of the registers are F, K, L, X, Y, and Z.

(a) Register L: This register is the address of client calling INTERR routine. Correct interpretation of the location pointed to by L is the most critical step in the interpretation of INTERR printouts. In some instances, L alone will provide all the necessary information. In others, some or all of the following INTERR printouts will need to be interpreted. Of special importance here is L within the pident RSUB. The RSUB consists of shared subroutines referenced by the other code pidents. In this case, L will normally tell what the error is, but will not specify when or where the error occurred. In this case, one must look at the state words and push-down scratch to find where the error occurred.

(b) Register X: In pidents RCKI, RCVC, RCFI, RCCH, or RCTF, register X will likely contain the 12-bit address in the table where the INTERR occurred. This permits isolation of the error to a 12-bit word in the message pident. For example, if L points to RCFI and X points to the 12-bit word following a TAG table instruction in any message pident, one can deduce with reasonable confidence that the translation data (eg, trunk network number, line equipment network) being used to create a call store address at this point is not producing a valid call store address. This can be further verified by looking in the listing of RCFI as indicated by L.

(c) Registers F, K, Y, Z: Use L and pident listing to find contents.

State Words

5.90 The second INTERR message, consisting of ten words, (twelve words for 1E7/1AE7 and later generics) prints state and pointer words of the RCSS. Of greatest significance here are the following:

(a) The R2PSW1 (4-0) equals code pident currently in control.

(b) The R2NAPDS points to the next available word of push-down scratch.

(c) The R2NAMSBA gives the next available word of the MSB auxiliary area.

(d) The R2NAWL gives the next available work list location.

(e) The R2TRVAL specifies address of the truth table (1E7/1AE7 and later generic).

(f) The R2SB specifies address of recent change scratch block being used for the input message (1E7/1AE7 and later generics).

For more details refer to Section 231-048-302.

5.91 As mentioned previously, some INTERR printouts tell what the error is, but not where it is located (especially in RSUB). In these cases, one can use C(R2NAPDS)—1 to C(R2PDS) (ie, all used push down scratch) to find a history of events leading up to the internal error. Since the scratch system gives client addresses and, in some cases, the contents of the central control registers when a client called subroutine, one can work backward from the scratch entries at C(R2NAPDS)—1 to determine the problem. (Note that one must understand the layouts of scratch seized in R2PDS.)

Input Control

5.92 The third INTERR printout gives five input control words, in call store addresses starting with 17515:

(a) R2SVCHAR—ASCII character last processed

(b) R2ISTATE—Input editor state

(c) R2BUNLD—Line buffer unload pointer (1E6/1AE6 generic programs)

(d) R2LINE—Current line (1E6/1AE6 generic programs)

(e) R2COLUMN—Current column (1E6/1AE6 generic programs)

(f) R2BUFPTR1—Line buffer load pointer (RCIE) and line buffer unload pointer for last keyword (RCKI)

(g) R2BUFPTR2—Current line buffer unload pointer for RCKI

(h) R2CCHRPOS—Current line and column.

5.93 The fourth and fifth INTERR printouts give a Boolean history of input, data check, validity, and format selection of the recent change message. By inspecting the truth value table (fourth printout), one can see which keywords were received, which data checks were true, and which validity tree nodes were true. The format selector (fifth printout) gives the selected paths through format based on input and translation data.

Push-Down Scratch

5.94 The sixth and seventh INTERR printouts give a portion of the 168-word push-down scratch. Selected for printout are the first 14 words (sixth INTERR message) and last 14 used, plus the next seven available (seventh INTERR = 21 words). In some circumstances it will be necessary to DUMP additional push-down scratch locations.

6. ABBREVIATIONS AND ACRONYMS

6.01 The following is a list of abbreviations and acronyms used within this section.

ASCII	American Standard Code for Information Interchange
BINK	Binary one thousand (memory block of 1024 (decimal) words)
BISI	Busy/Idle Status Indicator
CCSC	Customer Changeable Speed Calling
CS	Call Store
CSR	Centrex Station Rearrangement
DAB	Delayed Activation Block
EOT	End Of Transmission
FS	File Store
IM	Input Message
INWATS	Inward Wide Area Telecommunication Service
NTN	New Telphone Number
MCC	Master Control Center
MSB	Message Storage Buffer

OM	Output Message
PCB	Primary Change Block
Pident	Program Identification
PS	Program Store
PTW	Primary Translation Word
RBA	Rollback Area
RBB	Rollback Block
RC	Recent Change
RCS	Recent Change Service Order TTY
RCSS	Recent Change Subsystem
RFB	Rollforward Block
SC1	Switching Control Center TTY No. 1
SC2	Switching Control Center TTY No. 2
SR	System Reinitialization
SWAP	Switching Assembly Program
TN	Telephone Number