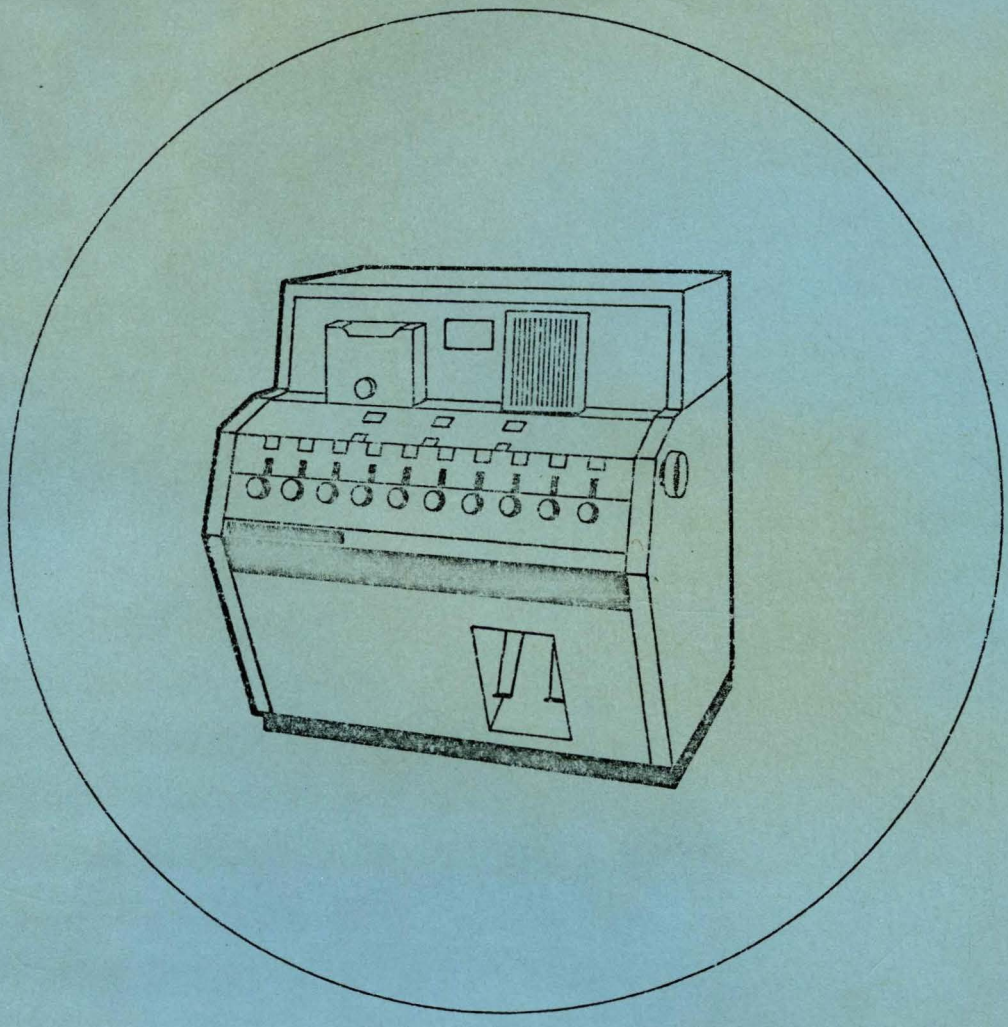
Collectadata-30 Terminal Module



logic manual

SYSTEM TEN BY SINGER

TABLE OF CONTENTS

PREFACE

This document describes the Collectadata-30 Terminal Module, which is an interdependent part of the Manufacturing Information System. Variations in system configurations are explained. Collectadata-30 (CD-30) terminals with their own IOC's and partitions may be intermixed with Model 100 Job Information Stations and/or Model 105 Attendance Stations in the system configuration. Code conversion to any output code is available but optional. Messages from the terminals may be verified by checking terminal ID numbers and/or message length. The message can be written to 7- or 9-track tape or to disc, off-line or on-line.

It is assumed that the user of this manual will be familiar with System Ten concepts. He will probably be responsible for the interface between the terminal partitions and the rest of the MIS configuration.

INTRODUCTION

The main purpose of the Manufacturing Information System (MIS) is to process data collected from terminals located in different manufacturing areas. The data is processed by the System Ten CPU and is written on customer-specified output media. The Collectadata-30 Module has been designed as an interdependent part of the system to process these data transactions from Collectadata-30 terminals.

Core allocation, hardware requirements, and software associated with the CD-30 partitions are described in detail in this manual. The examples used throughout are based on a standard CD-30 Terminal Module; possible options for output media, code conversion, and transaction validity checks are specified where applicable.

Section 1

GENERAL DESCRIPTION

The Collectadata-30 (CD-30) program module provides the interface between the CD-30 terminals and the System Ten CPU. It examines the data from each terminal as it enters the CPU, checking each message against a terminal/transaction table that resides in each CD-30 partition for correct message length, terminal ID and/or transaction validity. Portions of the incoming data are converted from the CD-30 code to the System Ten USASCII 6-bit subset for CPU processing; the entire output may or may not be converted to the appropriate output code before being written to an output device. Errors are logged on a 7102 Communications Terminal or Model 70 Work Station in addition to being logged on the output device as an error record.

In this system there can be a varying number of CD-30 partitions per CPU. Each partition must have its own IOC and can have a theoretical maximum of twenty terminals (ten is a more realistic number). Therefore, the CD-30 Terminal Module must be present in each CD-30 partition with only the identifying partition number and related table information being changed for each. At the start of each CD-30 partition program, tests are made to see whether a terminal transaction table update message has been entered from the communication terminal, whether the system is inhibited, and whether there is a service request from a CD-30 terminal. A service request will cause the program to initiate the reading process and switch partitions. When data transfer has been completed, control will be returned to the CD-30 partition program where condition indicators are tested to determine if the read was normal. If not, the error will be identified and the message routed to the appropriate error handling routines.

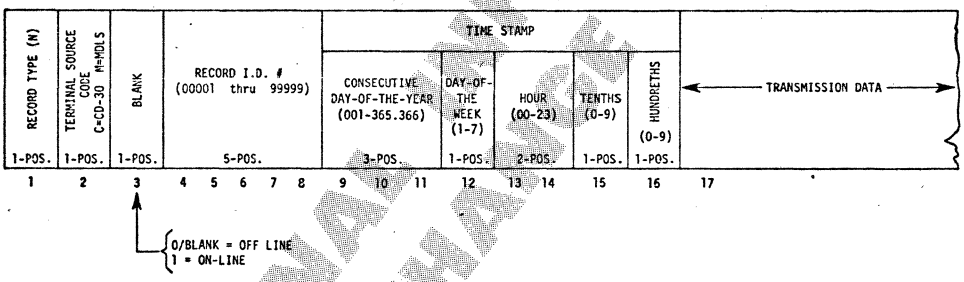
The number of characters transferred into the CD-30 terminal partition buffer is determined for use in code conversion if used, "wrong-message-length" checks, and output to disc or magnetic tape. Validation checks are performed on the terminal ID number and/or transaction code with a resulting ACK or NAK being sent to the terminal, which will then be released from the line.

The terminal message, which optionally may or may not be converted to an output code, is prefixed with the Friden standard record identification, time-stamped, and sent to the write routine. As errors are encountered, selected data is extracted from the terminal message and sent to an error buffer in Common from which error messages are printed on the communications terminal.

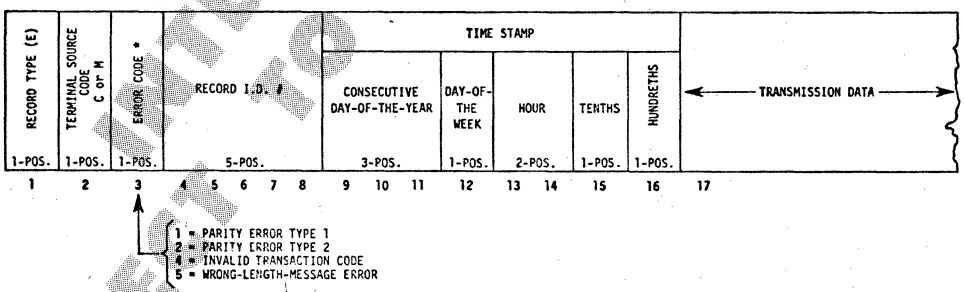
The CD-30 Terminal Module consists of three routines: two in each CD-30 partition (CD-30 Main Routine--CDSTR, and Terminal/Transaction Check--TCSTRT) and one in Common (Table Update--CDCHK).

SYSTEM TEN OUTPUT RECORD FORMAT

NORMAL RECORD



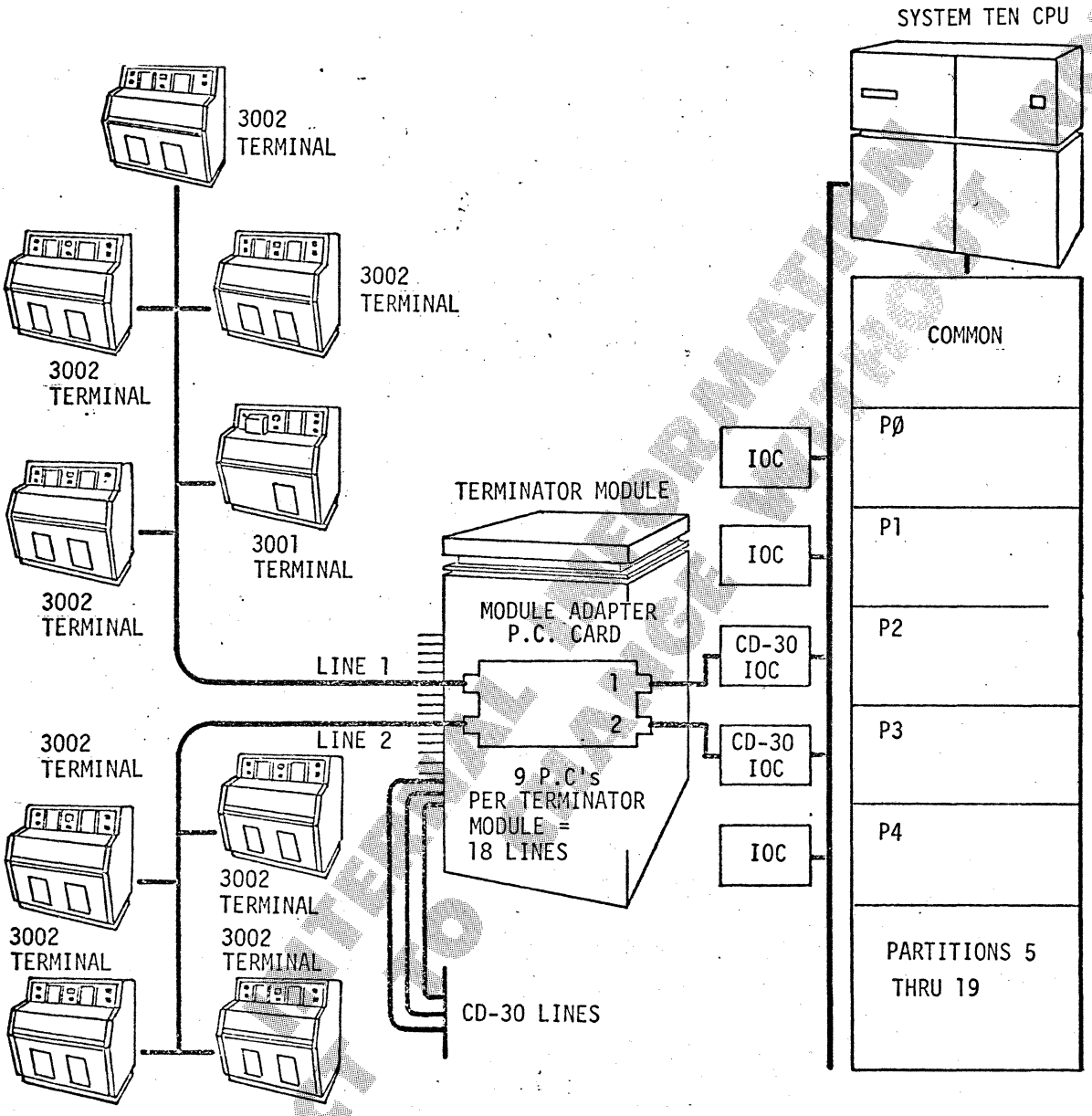
ERROR RECORD



FORMAT OF THE ERROR BUFFER (ERRBUF)

ERROR ID CONTAINS
- PAR1 - parity error
- PAR2 - CD-30 fault
- PAR3 - ACK not received
- TRAN - transaction code error
- WLM - wrong length message

*Only those buffers that are filled are printed. Thus 1, 2, or 3 error messages can be printed at one time in the above format.



SAMPLE CD-30 CONFIGURATION

Section 2

HARDWARE REQUIREMENTS

In addition to a System Ten CPU with at least 2K of core for each CD-30 partition, the following hardware is required.

Terminals: Collectadata-3001, 3002, 3022 (practical maximum is ten per CD-30 partition).

Collectadata Terminator Module: To convert Collectada 90v signal levels to System Ten 5v logic levels (one per IOC). Each PC card in the terminator module will accommodate 2 lines (cables).

Collectadata I/O Channels: One per CD-30 cable to serve as interface to the System Ten CPU from the terminator module.

A system with magnetic tape output can use a maximum of 18 partitions for terminals. An on-line disc system can use a maximum of 17 partitions for terminals provided partition 19 is used by the SCA Program Module. An SCA requires two partitions but will actually occupy only one if put in Partition 19 because it will use the PC card that follows this partition. However, each additional SCA will require two full partitions and thus will decrease the maximum devoted to CD-30 or MDCS terminals.

It is assumed that the system in which this module functions will also include a communications terminal in the standard partition (Partition 0), a digital clock in Partition 1, and at least one output device--magnetic tape or disc. The system is not limited to CD-30 terminals, however; it may also include MDCS terminals (Model 100 Job Information Station and/or Model 105 Attendance Station) in different partitions.

Section 3

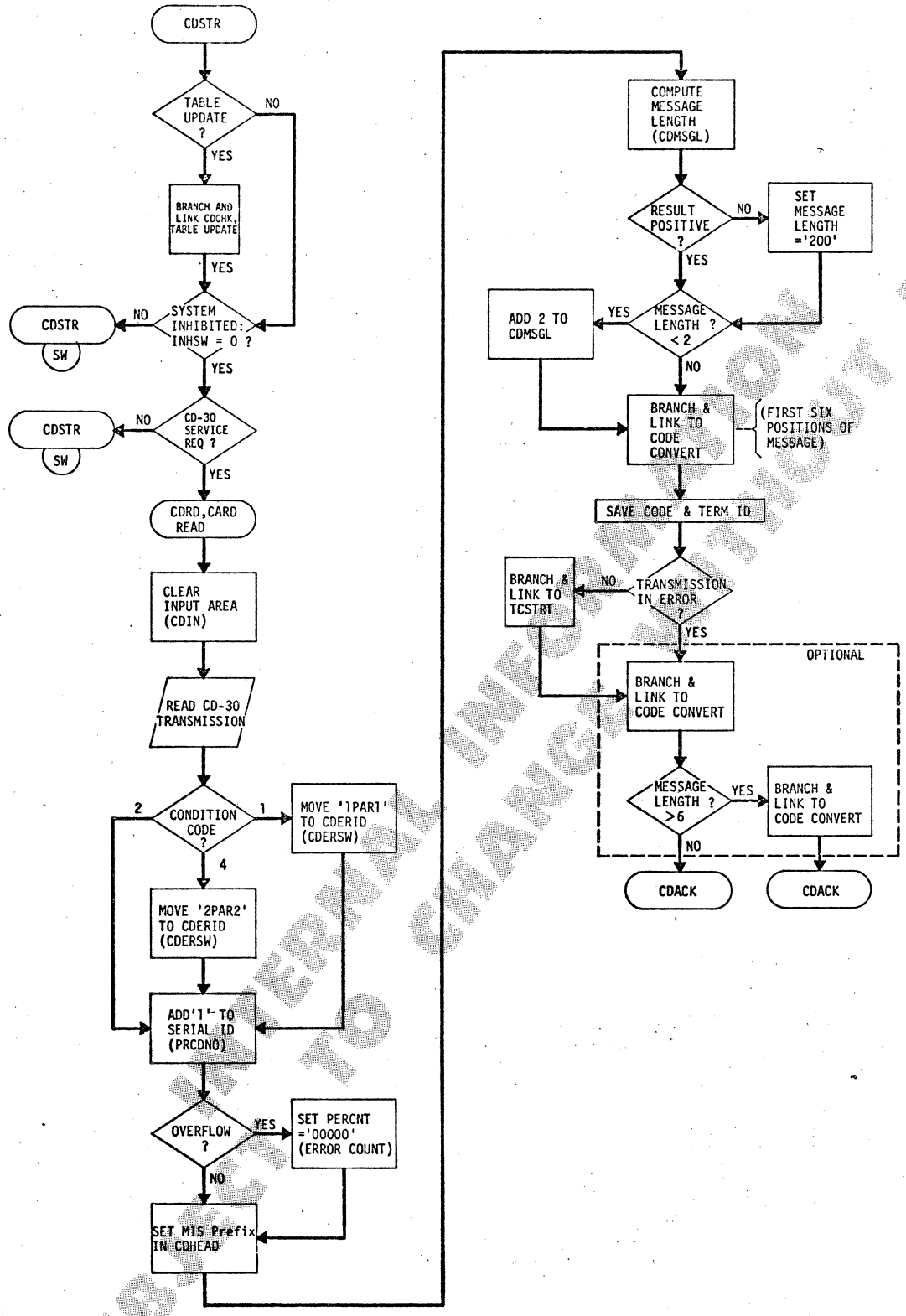
CORE REQUIREMENTS AND ALLOCATION

CORE REQUIREMENTS

Each CD-30 Terminal Module (Main Line Program CDSTR and Terminal/Transaction Check TCSTRT) requires approximately 1200 core locations of core in each CD-30 partition. The Table Update Routine, CDCHK, requires approximately 1700 core locations. This does not include labels in common that are defined by the Control Module.

ALLOCATION

CD-30 partitions may be loaded starting with Partition 2 because, in a normal MIS configuration, Partition 0 and Common hold the Control Module and Partition 1 the Clock Module. There should be one CD-30 partition for each ten terminals although there may be fewer than ten terminals in any one partition. By expanding a CD-30 partition to 3K the user may expand the program and Terminal/Transaction Table to accommodate up to twenty CD-30 terminals as long as the same number of terminal devices has been attached to that particular CD-30 cable.



CD-30 Terminal Read Routine - CDSTR

this chart belongs on p. 6-2

Section 4

MODULE NARRATIVE

The detailed logic of each routine required to process a message from a CD-30 terminal is described in Section 9, "Routine Logic". This section gives an overall view of the process in more general terms. The program used in this manual is based on a system configuration that includes a digital clock, CD-30 terminals, and at least one disc/tape I/O device. In this particular application, Partition 0 is reserved for control functions, Partition 1 for the clock, and the balance of partitions for the terminals.

Several major functions are required to process a terminal transaction. The terminal/transaction table (TCTABL), located in each CD-30 partition is of major importance because it is used to verify the terminal message by checking its length; it can be used to verify the terminal ID number and can also be used to route the message to on-line or off-line storage. This routing function is not used in the example program. Code conversion is required to convert portions of the message from the original CD-30 code to the System Ten USASCII 6-bit subset for processing by the CPU; all of the message may be optionally converted to the code required by the user for output. (Messages are sent to the SCA in USASCII. CD-30 code is considered as USASCII by the SCA, but requires further code conversion in the host computer.) All errors except terminal acknowledgement errors are noted and recorded along with the message, and all errors are reported on the communications terminal so that they can be corrected later. The program records parity and fault errors after reading the message, inability to transmit ACK or NAK to the terminal, and invalid message lengths or terminal ID numbers.

Processing Sequence

Two tests are made when the CPU enters the CD-30 partition: any changes to the partition terminal/transaction table (TCTABL) that have been entered from the communications terminal are executed, and a check is made for a service request from a CD-30 terminal. If no request has been

signalled, the CPU switches to the next partition. If there is a service request, the CD-30 transmission is read into a 200-position partition buffer for processing.

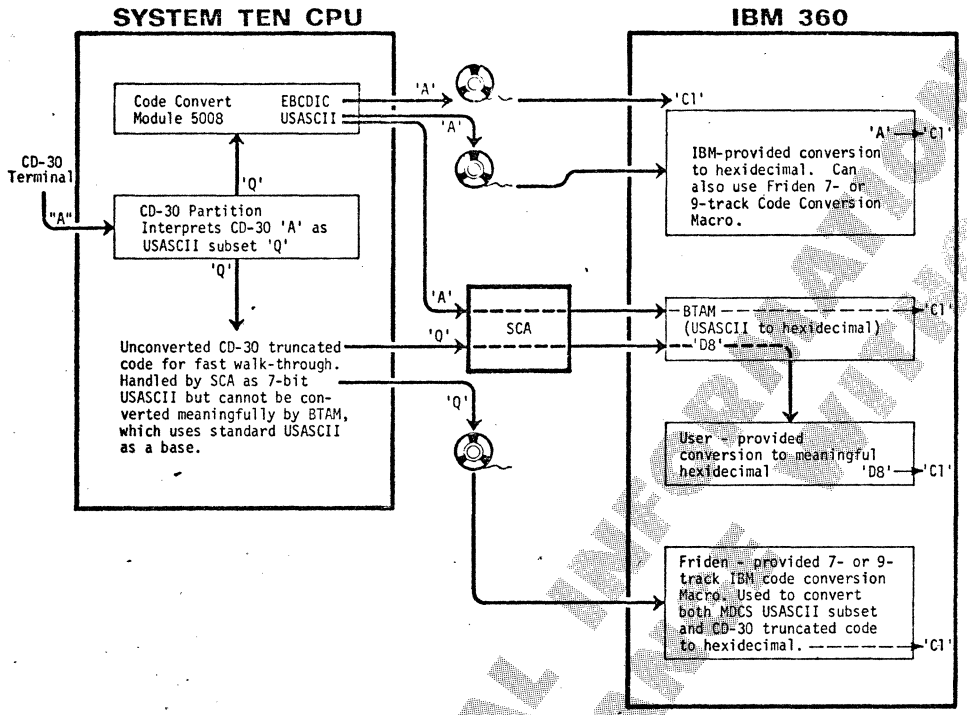
If a fault or parity error is detected after the message is read, appropriate codes are moved into a holding area to be added to the message when it is written. The message length is computed next so that it can be verified later with terminal/transaction table TCTABL to be sure that the input data lengths from the terminal are correct.

Code conversion is done in several steps. First the transaction code and terminal ID are converted from the CD-30 code to 6-bit System Ten USASCII for internal processing of error messages if errors are detected. If no read errors have been encountered, the calculated message length is compared to the acceptable length for that transaction code and terminal ID in TCTABL (single-entry tables do not require the terminal ID check). Again, an appropriate error code is moved into a holding area to be added to the message later if an error is encountered.

At this time the entire 22-character standard MIS header may be optionally converted from System Ten USASCII to the required output code. The header consists of identifying information including transaction code, terminal ID number, record ID numbers, and time stamp. Next, the rest of the message may be optionally converted from CD-30 code to an output code. As data is written to the SCA in USASCII, header conversion is unnecessary for this type of transmission.

Receipt of the message by the CPU is acknowledged by sending ACK or NAK to the terminal, which, if properly received, will drop the terminal from the line. If these messages connot be transmitted to the terminal, an error code is moved to a holding area, and the line is held high, making all terminals on that line inoperative.

The holding area is checked for error codes; if there is none, the transaction message is written to the output device by a routine in Common. Errors require processing by the CD-30 partition. One of the error buffers is filled with error code, partition number, terminal ID, transaction code, message length, and time stamp. On the next pass through Partition 0, the buffer will be printed. If the message being processed has incurred terminal-acknowledgement errors only, it is now sent to the write routine as a valid message; the error, however, is noted on the communications terminal to indicate the possible need for terminal or IOC repair. If another error has occured, the message is adjusted to reflect the error, and then the message is written as received from the terminal.

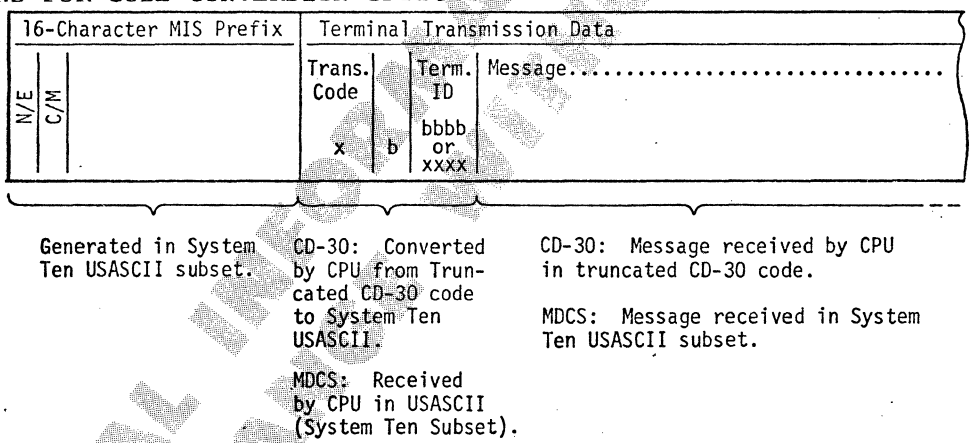


CD-30	(USASCII subset) SYSTEM TEN			(Hexidecimal) IBM 360
A	Q	EBCDIC conversion to A (tape)	360 hardware converts EBCDIC to hexidecimal C1	C1(A)
A	Q	USASCII conversion to A (tape)	IBM - provided, converts USASCII to hexidecimal C1	C1(A)
A	Q	Unconverted subset: Q (tape)	Friden - provided conversion of Q to hexidecimal C1	C1(A)
A	Q	Unconverted subset: Q (SCA)	360 BTAM converts Q to hexidecimal D8; User must provided for conversion of D8 to C1	C1(A)

Section 5

MODULE VARIATIONS

MODIFICATIONS FOR CODE CONVERSION OPTIONS



SYSTEM TEN OUTPUT RECORD FORMAT

Code Convert Logic Manual 5008 explains the code conversion process in detail. The amount of code conversion performed for the output record will vary according to the customer's needs. The conversion tables can be adapted to meet any requirement.

The first six characters (may later be increased to ten in this module) of the terminal transmission (transaction code and terminal ID number) are always converted to System Ten USASCII for use by the CPU. The balance of the terminal transaction can be stored on tape or disc as received but will be virtually meaningless unless converted either before ultimate output or in the host computer.

The SCA will accept unconverted, truncated CD-30 code as USASCII and the BTAM routine will process the transmission as USASCII. However, since only the first twenty-two characters (16-character prefix, transaction code, and terminal ID) are in USASCII, the balance of the message must be converted further to be meaningful. This is a user responsibility.

CD-30 and MDCS (JIS terminals and Attendance terminals) records can be mixed on the same tape or disc. If the CD-30 transmissions are still in unconverted, truncated CD-30 code, the second character of the MIS prefix (C = CD-30, M = MDCS) must be tested to determine which record to process through further code conversion in the host computer. The Friden-provided 7- or 9-track IBM Code Conversion Macro will do this for tape. If unconverted CD-30 transactions are mixed with JIS transactions and sent through the SCA, it is the user's responsibility to sort them out for further code conversion.

MODIFICATIONS FOR TERMINAL ID AND/OR TRANSACTION VALIDITY CHECK

Four types of transaction tables can be used currently in the MIS configuration to verify terminal ID and/or transaction validity: by function, these are single-entry, multiple-entry, and modulus remainder tables. (See facing example.)

- Single-entry tables are used when all terminals on the partition have the same transaction message lengths (and routing indicators, if used) for the same codes. Because the terminal ID number does not have to be checked, table look-up is eliminated.

- Multiple-entry tables are used when the terminals on the line differ in transaction message length (and routing indicators, if used). Because terminal numbers must be checked and verified, this routine takes longer than the single-entry method.

- Modulus remainder tables can be of either the single- or multiple-entry type; they use the modulus-four remainder of the message length rather than the message length itself.

The choice of modulus remainder or message length usually depends on how the customer has configured his system.

The tables are initially loaded into core with the rest of the CD-30 partition program. They are defined in the DM statements. The first four positions are the terminal ID number. As terminal ID is never checked in single-entry table transactions, these four positions can be any number or can be blanks in the single-entry table. The rest of the table entry consists of information in multiples of four positions, one group of four for each transaction code, arranged in ascending order by code number (see facing examples). The left three positions of each group of four contain the acceptable message length (or modulus remainder), the right position contains the routing indicator (0/blank = off-line, 1 = on-line). If routing indicators are not used (as they are not in this example program), this space must contain a blank, 0, or 1, anyway.

The detailed procedure for updating the table from the communications terminal is explained in Section 7, "How to Update Transaction Table, TCTABL".

Single-entry Table using Message Lengths

```
TCTABL    DM    0C36
          DM    C36'bbbb or
                C36'9999076 076 076 076 076 076 076 042 '
                   Term.    Code    Code    Code    Code (Attendance
                    No.       2       4       6       8  Transaction)
                        Code    Code    Code    Code
                          1       3       5       7
TCEOT     DM    C4':::::'
```

Single-entry Table using Modulus Remainders

```
TCTABL    DM    0C36
          DM    C36'bbbb002 001 000 002 003 001 000 001 '
TCEOT     DM C4':::::'
```

Multiple-entry Table using Message Lengths (Modulus Remainders can be used)

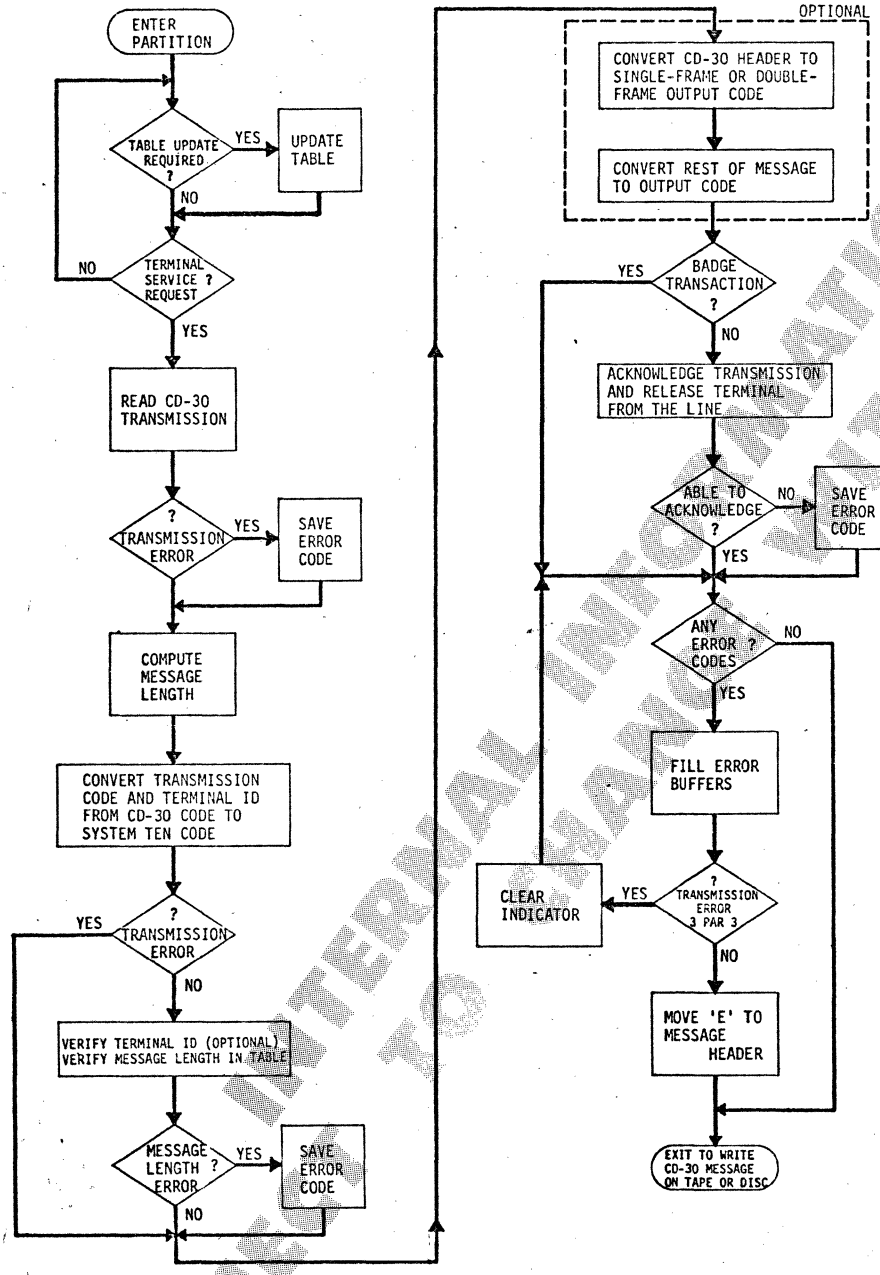
```
TCTABL    DM    0C360
          DM    C36'9999076 076 076 076 040 076 076 042 '
          DM    C36'8888076 076 085 076 076 076 076 042 '
          DM    C36'7777020 076 085 076 076 076 076 042 '
          DM    C36'6666076 076 085 034 076 076 076 042 '
          DM    C36'5555035 049 085 076 068 076 076 042 '
          DM    C36'4444040 026 063 076 076 080 067 042 '
          DM    C36'3333076 035 059 076 058 076 076 042 '
          DM    C36'2222070 046 038 076 080 076 076 042 '
          DM    C36'1111072 059 048 076 035 058 076 042 '
          DM    C36'0000035 067 076 054 076 089 076 042 '
TCEOT     DM    C4':::::'
```

No table entry can be shorter than 36 positions. If the length of an unused code is specified as a blank in the table, that transaction code is considered invalid if used by mistake and will result in a "4TRAN4" error. When these unused codes are needed, they can be given a valid length.

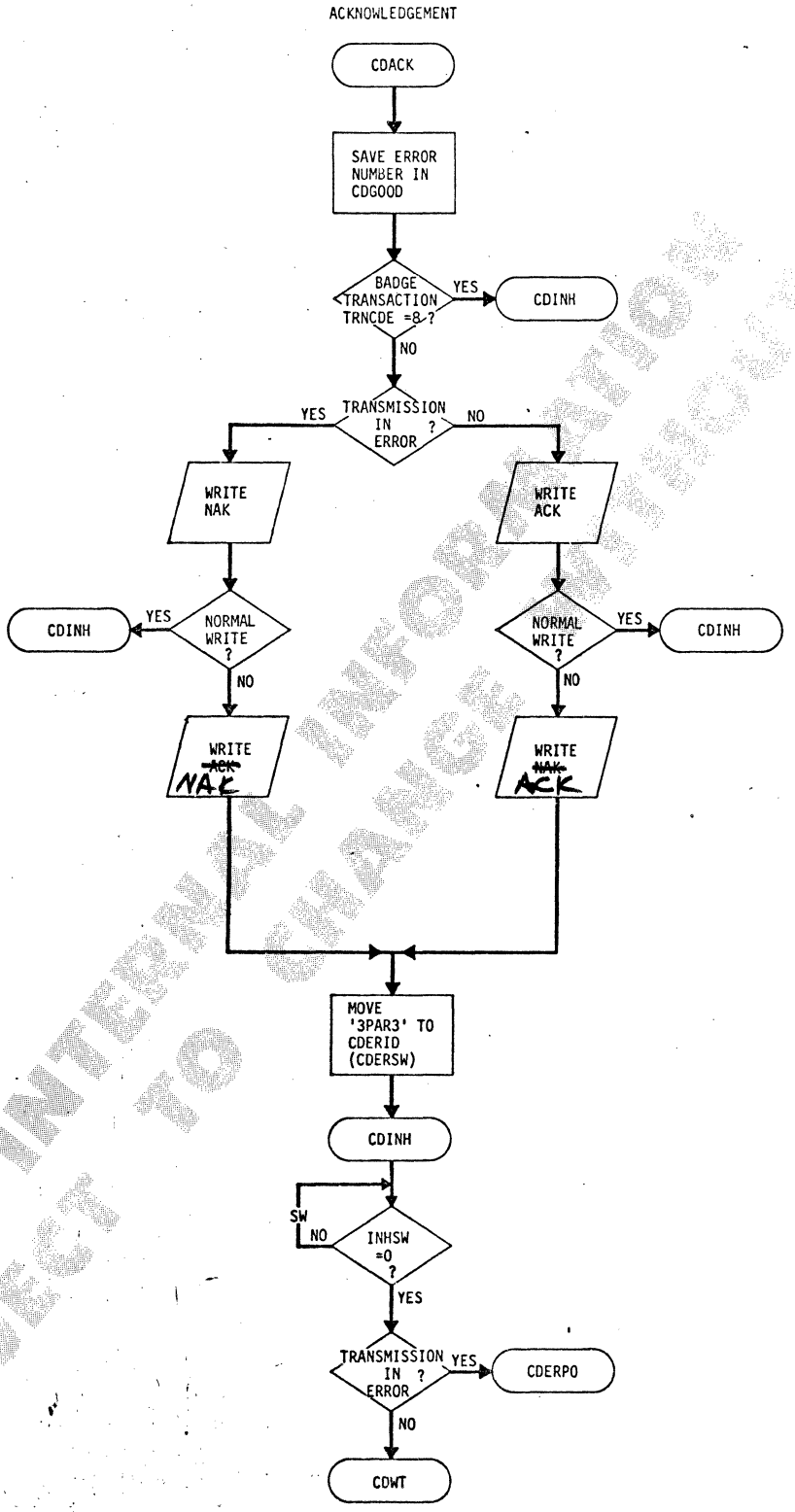
EXAMPLES OF TRANSACTION TABLE,
TCTABL

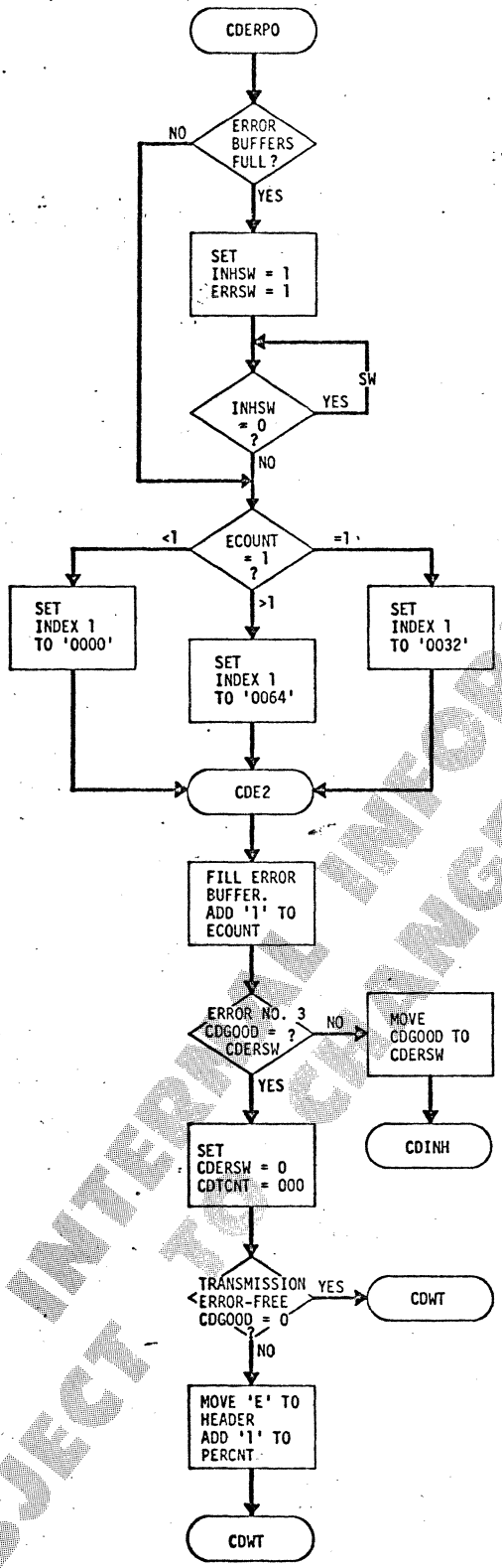
Section 6

PROGRAM LISTING & FLOW CHARTS

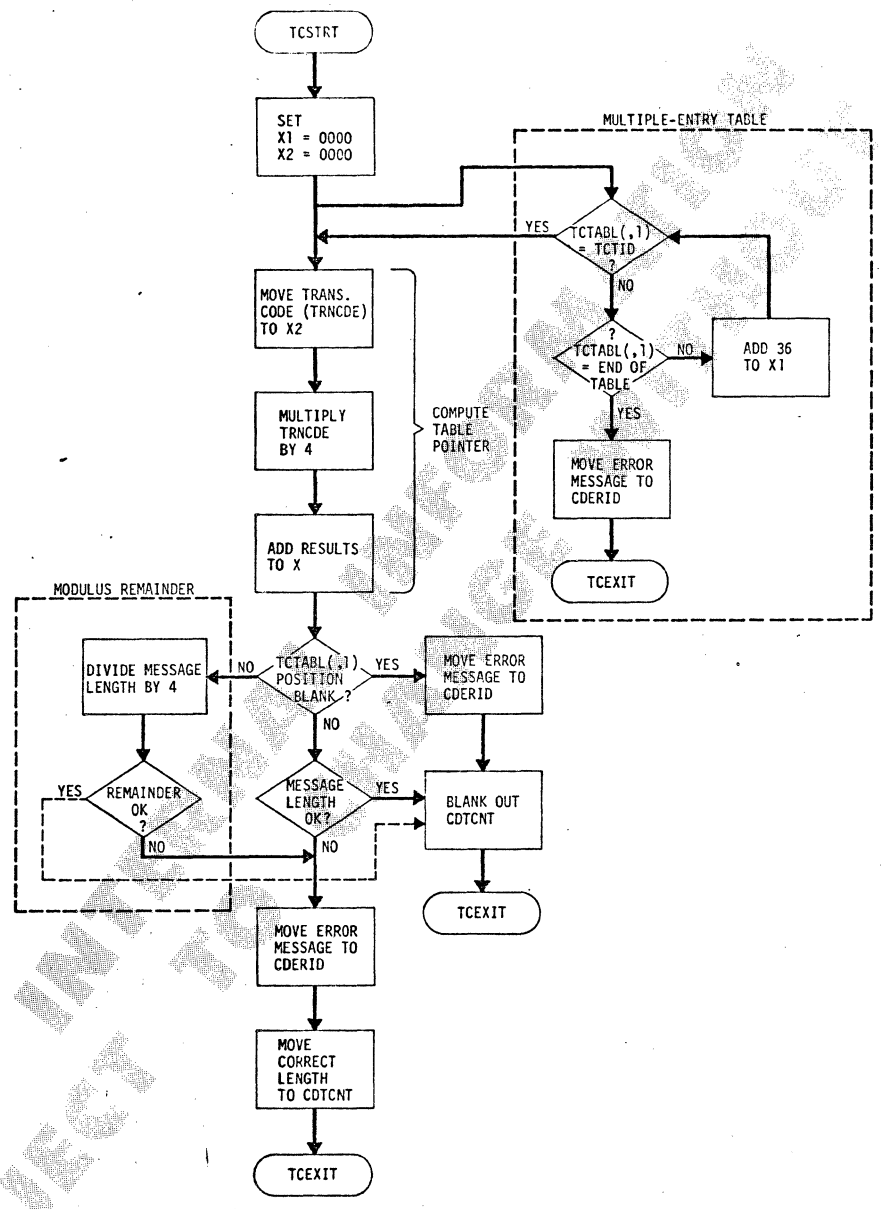


this chart should face p. 4-1





ERROR Processing Routine--CDERPO



Terminal Transmission Check Routine - TCSTRT

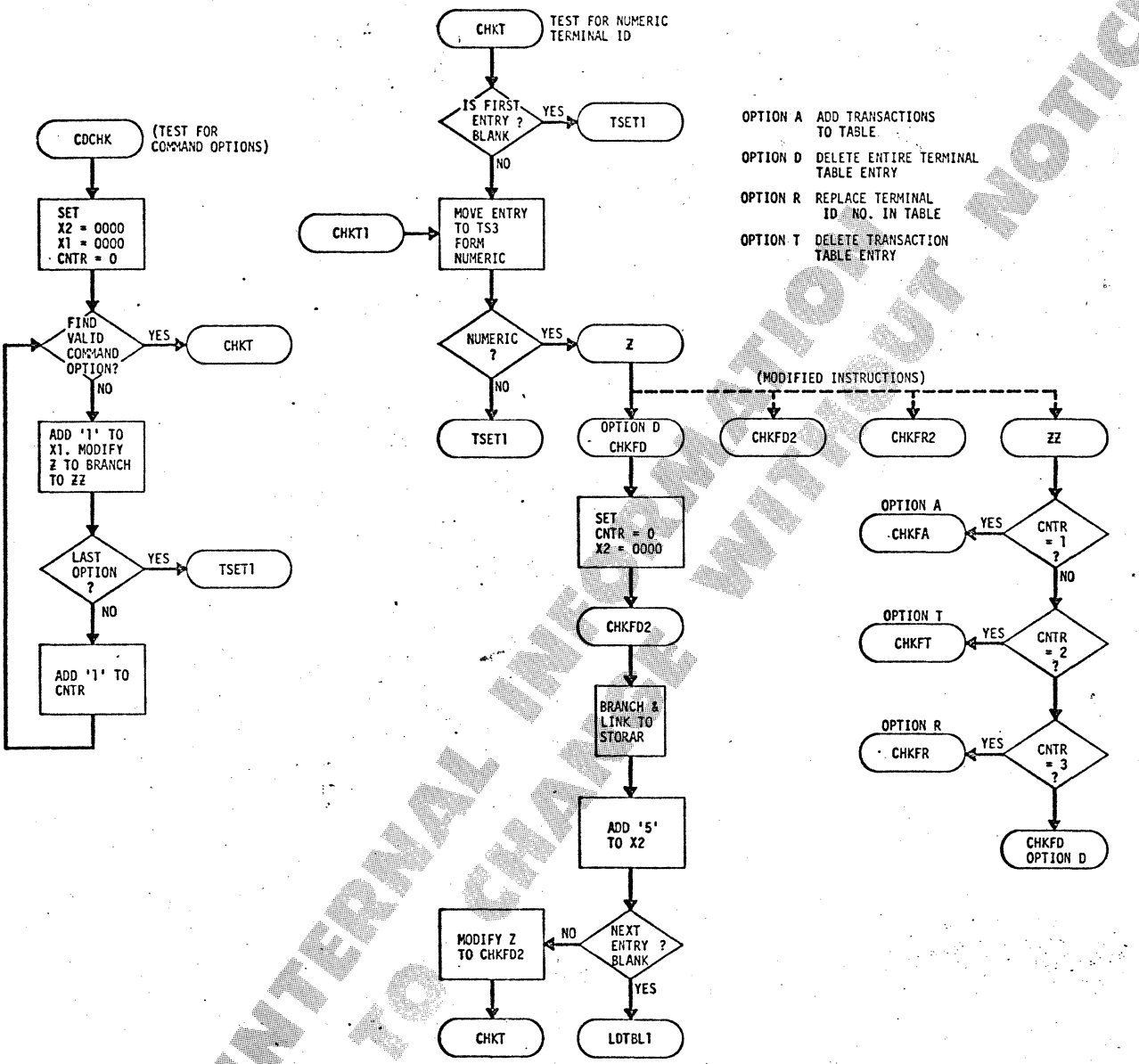
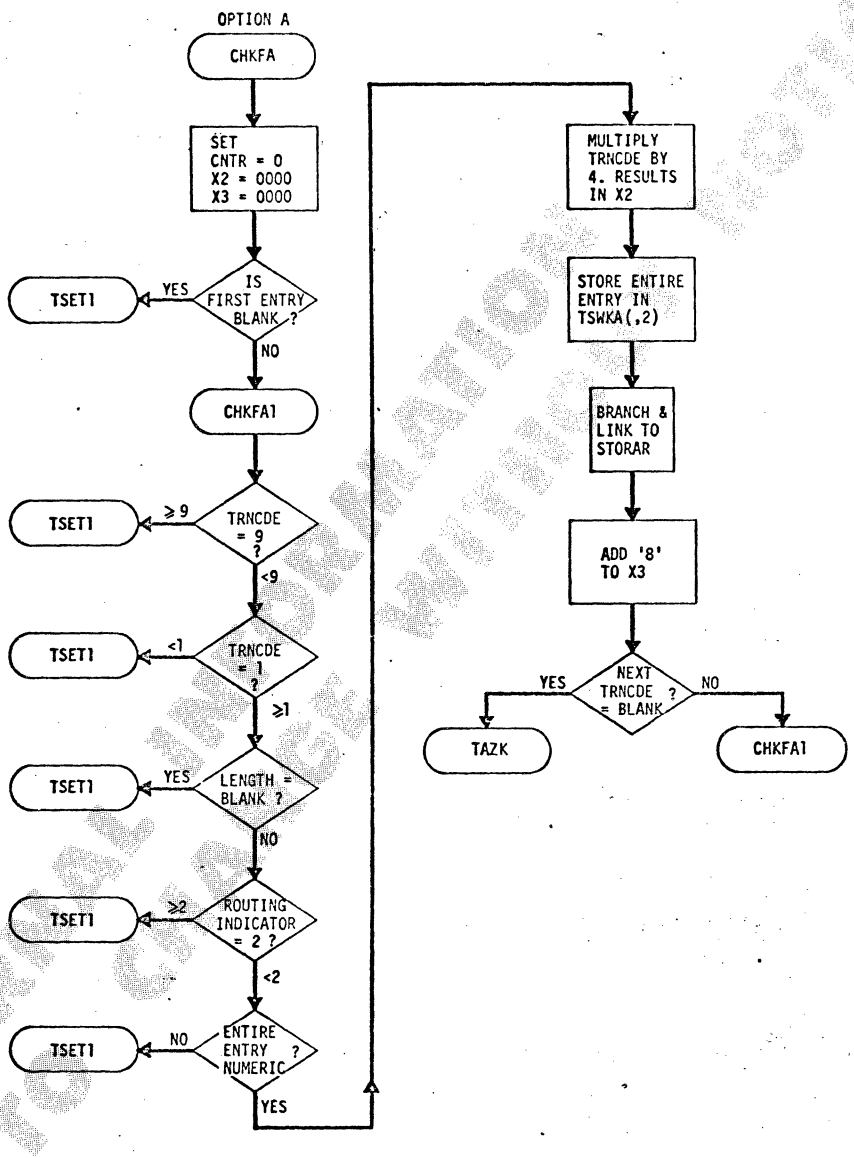
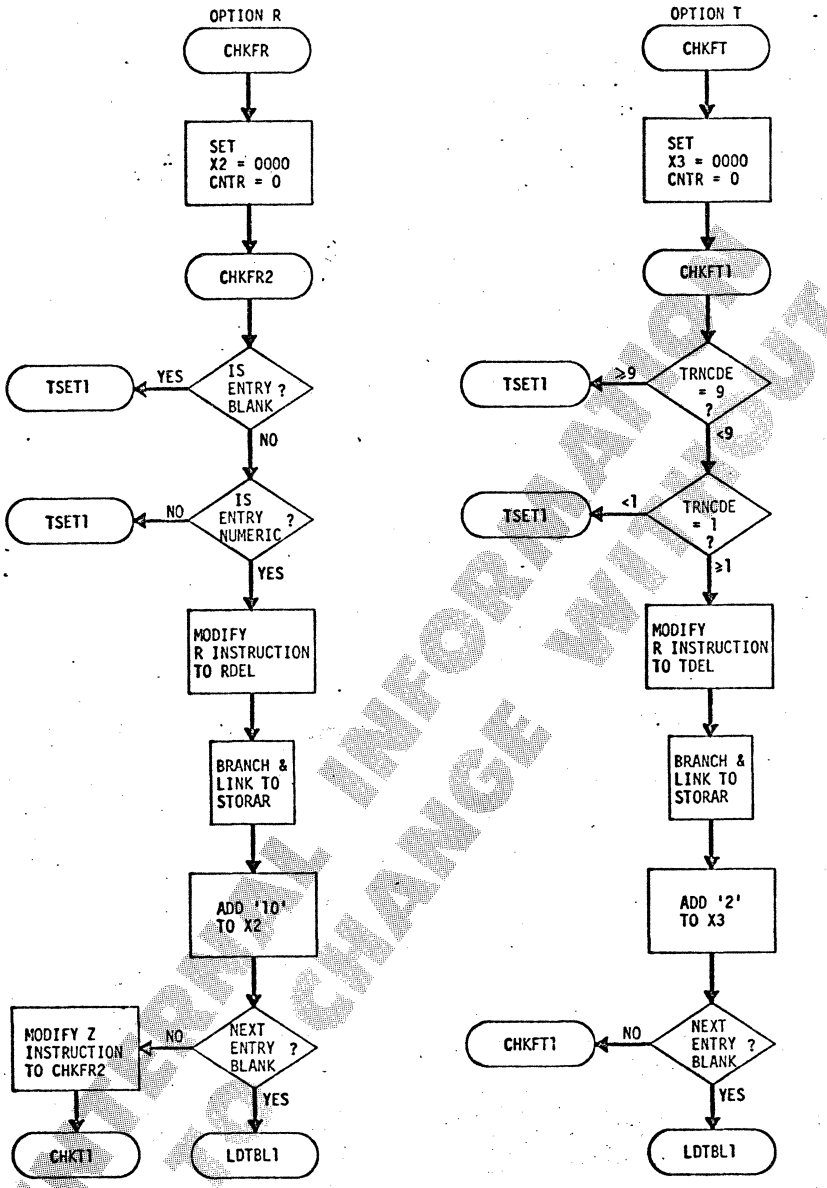


Table Update Routine - CDCHK

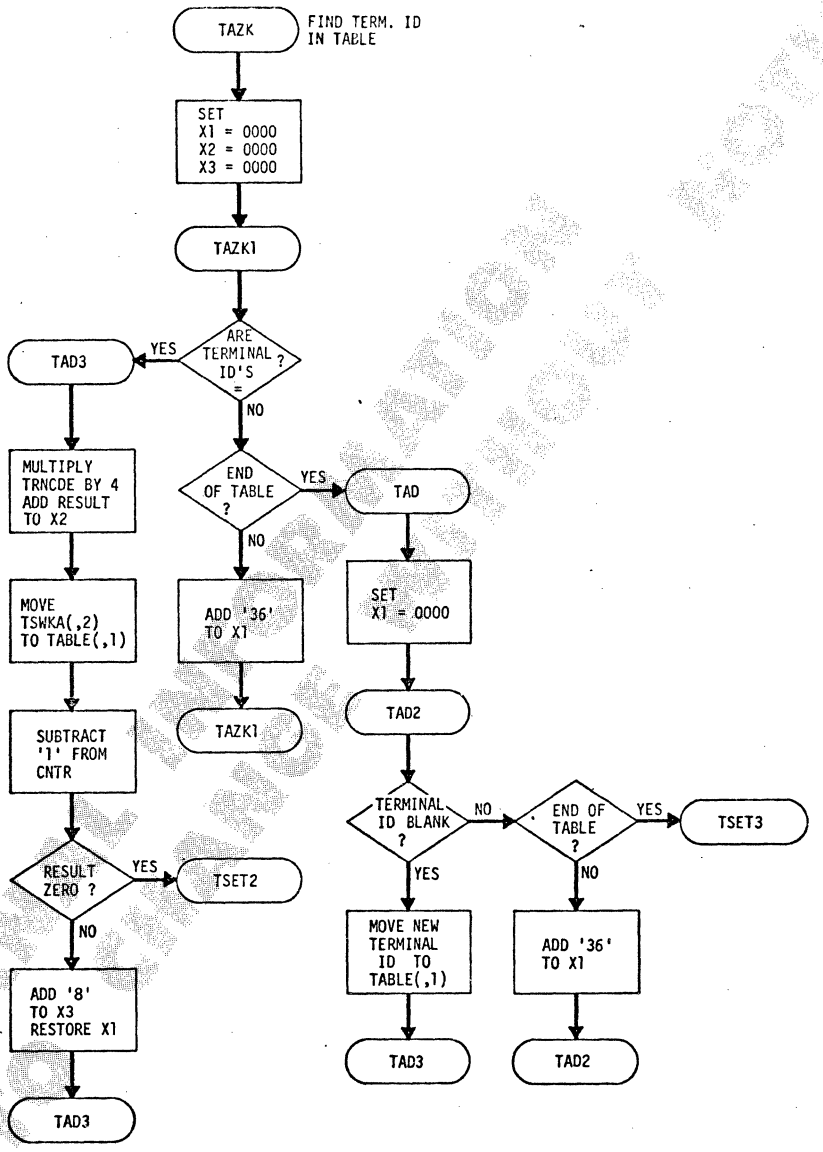
(This flow chart is for the CDCHK routine used with a multiple-entry table; see listing.)



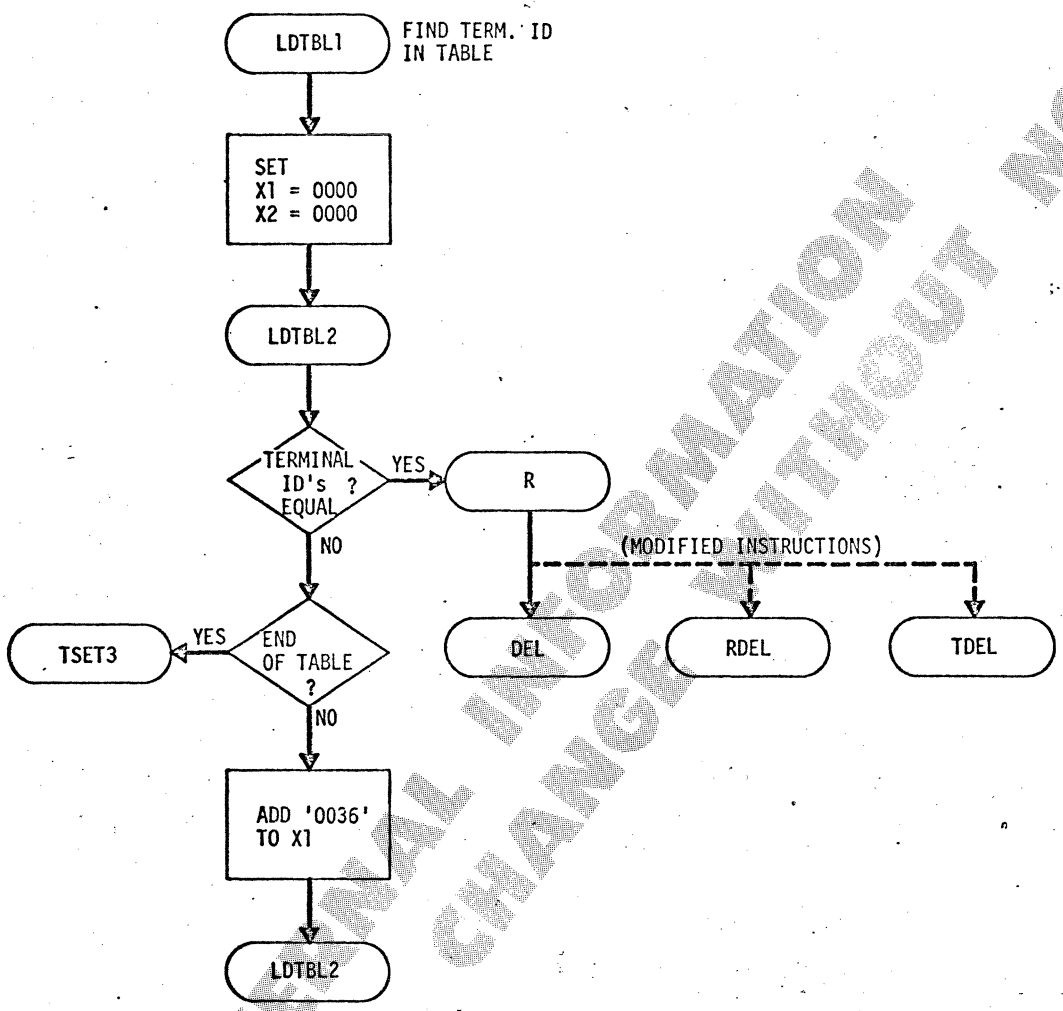
CDCHK - OPTION A (Add Transaction Codes to Table)



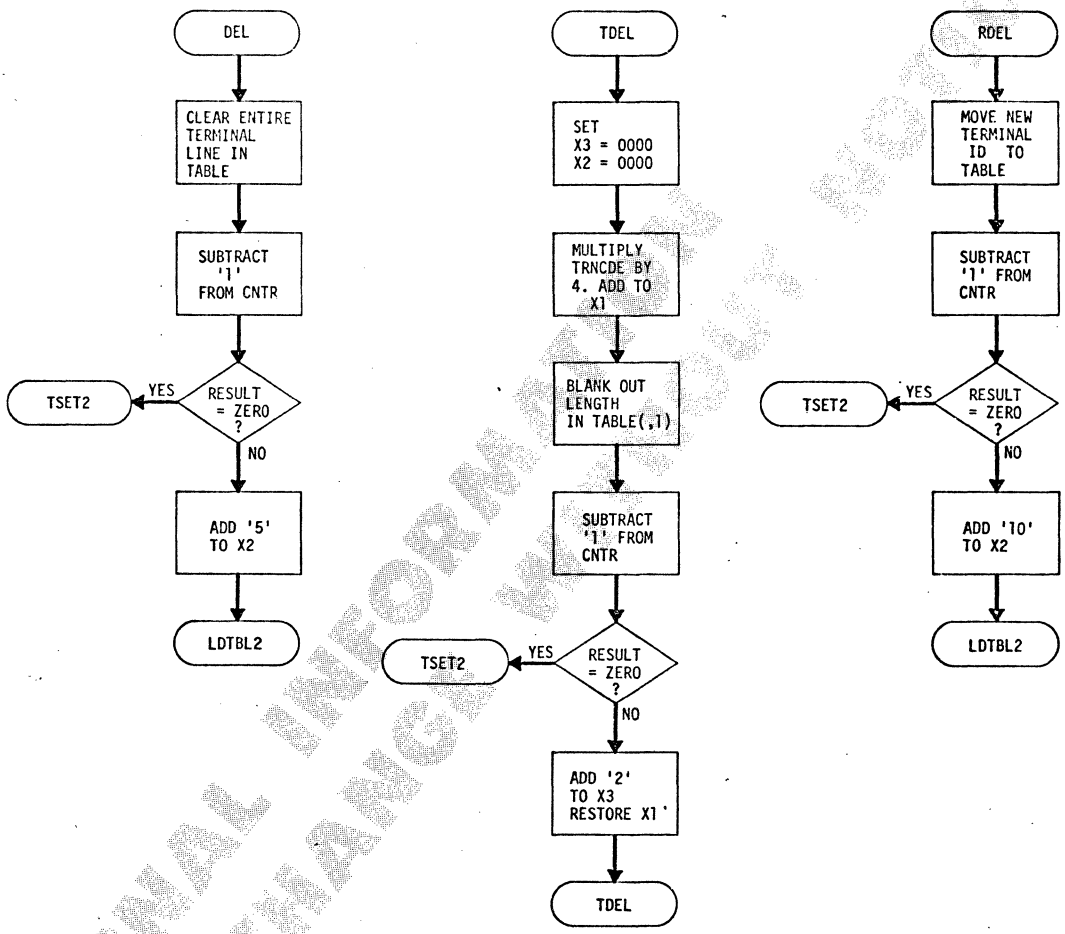
CDCHK - OPTION R (Replace Terminal ID Number)



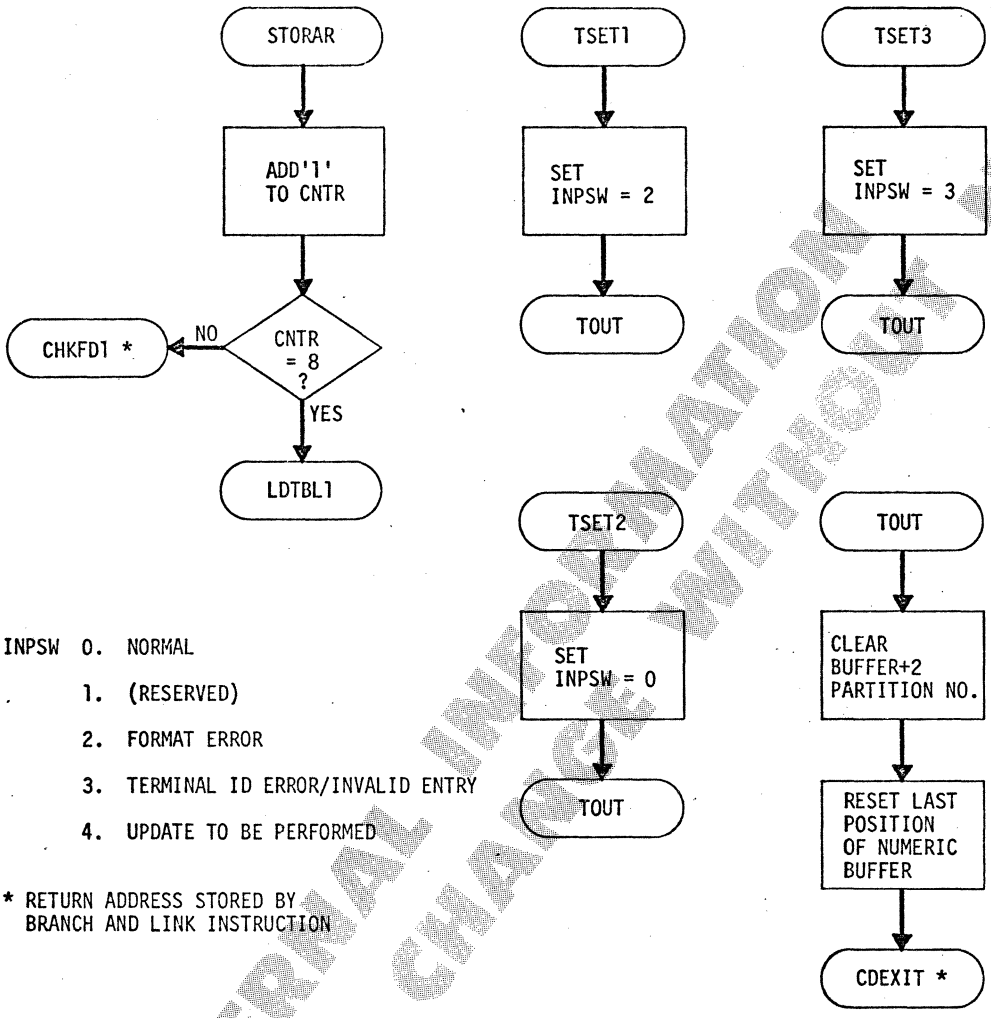
CDCHK - TAZK (Find Terminal ID in Table)



CDCHK - LDTBL1 (Find Terminal ID in Table)



CDCHK - DEL, TDEL, RDEL



INPSW 0. NORMAL

1. (RESERVED)

2. FORMAT ERROR

3. TERMINAL ID ERROR/INVALID ENTRY

4. UPDATE TO BE PERFORMED

* RETURN ADDRESS STORED BY BRANCH AND LINK INSTRUCTION

CDCHK - Miscellaneous Routines

The following listing is an example of the program with provisions for a multiple-entry Terminal ID/Transaction Table. It converts the entire CD-30 transmission to single-frame output code before writing the transmission to magnetic tape. Variations in handling terminal ID/transaction checking, code conversion, and writing to tape or disc are discussed in other sections.

```
*      ************************************************************
*
*
*
*
*
*       MULTIPLE-ENTRY CDCHK, USING MESSAGE LENGTH
*
*      TERMINAL TRANSACTION TABLE UPDATE ROUTINE
*
*
*
*    1.REPLACE TERMINAL ID NO. -  MAX = 8 ENTRIES
*
*                       PR03-9999-0701-8888-0702/
*      COMMAND OPTION  *
*      PARTITION NO.    **
*      EXISTING TERMINAL ID****      ****
*      NEW TERMINAL ID          ****      ****
*
*    2.DELETE ENTIRE TERMINAL TABLE ENTRY
*
*                       PD03-9999-8888-7777-6666/
*      COMMAND OPTION  *
*      PARTITION NO.    **
*      TERMINAL TO DELETE  **** **** **** ****
*
*    3.DELETE TRANSACTION TABLE ENTRY
*
*                       PT03-9999-8-7-6-5-4-3-2-1/
*      COMMAND OPTION  *
*      PARTITION NO.    **
*      TERMINAL ID          ****
*      TRANSACTIONS TO DELETE   * * * * * * * * *
*
*
*    4.ADD TRANSACTION CODES TO TABLE
*
*                       PA03-9999-8-012-1-7-076-1/
*      COMMAND OPTION  *
*      PARTITION NO.    **
*      TERMINAL ID          ****
*      TRANSACTION CODE          *       *
*      NEW LENGTH                   ***     ***
*      ROUTING INDICATOR                *       *
*
*      INPSW                  MESSAGE STATUS SWITCH
*                             =0 MEANS NORMAL
*                             =1 (RESERVED)
*                             =2 MEANS FORMAT ERROR
*                             =3 MEANS TERMINAL ID ERROR/INVALID ENTRY
*                             =4 UPDATE TO BE PERFORMED
*      ************************************************************
*
*      TABLE UPDATE ROUTINE
*
*      TEST FOR POSSIBLE COMMAND OPTIONS
*
CDCHK  MC ZEROS,X2                CLEAR INDEX 2
```

```
         MC ZEROS,X1                 CLEAR INDEX 1
         MC ZEROS(1),CNTR            SET COUNTER TO ZERO
A1       C  A(1,1),BUFFER+1          IS THIS A VALID FUNCTION
         BC CHKT(2)                  IF YES, GOTO CHKT
         A  ONE,X1+3(1)              ADD '1' TO INDEX 1
         MN ADCH9,Z+1                MODIFY INSTRUCTION Z TO BRANCH TO
         C  A(1,1),ASTER             IS THIS THE LAST OPTION
         BC TSET1(2)                 IF YES, GOTO ERROR ROUTINE
         A  ONE,CNTR                 IF NO, ADD 1 TO COUNTER
         BC A1(5)                    AND LOOP BACK TO A1
*
*
*        TEST TERMINAL ID TO BE NUMERIC
*
CHKT     C  BUFFER+5(1,2),BLNKS      IS NEXT TERMINAL ID BLANK?
         BC TSET1(2)                 IF YES, GOTO ERROR ROUTINE
CHKT1    MC BUFFER+5(4,2),TS3        IF NO, MOVE TERMINAL ID TO TS3
         PN TS3(5),TS4(5)            FORM NUMERIC
         C  TS3(4),TS4               IS ENTRY NUMERIC?
Z        BC CHKFD(2),TSET1(5)        IF NO, GOTO ERROR ROUTINE
ZZ       C  ONE,CNTR                 IF COUNTER IS ONE
         BC CHKFA(2)                 GOTO ADD TRANSACTION LENGTH IN TAB
         C  TWO,CNTR                 IF COUNTER IS TWO
         BC CHKFT(2)                 GO TO DELETE TRANSACTION LENGTH IN
         C  THREE,CNTR               IF COUNTER IS THREE
         BC CHKFR(2)                 GOTO CHANGE TERMINAL ID IN TABLE
*
*        DELETE ENTIRE TERMINAL TABLE ENTRY  -  OPTION D
*
CHKFD    MC ZEROS(1),CNTR            ELSE SET COUNTER TO ZERO
         MC ZEROS,X2                 SET INDEX 2 TO 0
CHKFD2   BC CHKFD1+1(6),STORAR(5)     BRANCH AND LINK TO STORAR
         A  FIVE,X2+2(2)             ADD 5 TO INDEX 2
         C  BUFFER+5(1,2),BLNKS      IS TRANSACTION CODE BLANK?
         BC LDTBL1(2)                IF YES, GOTO FIND TERMINAL ID
         MN ADCH9,Z+1                IF NOT, MODIFY INSTRUCTION Z TO CH
         BC CHKT(5)                  AND LOOP BACK TO CHKT
*
         ORG *-5
ADCH9    DM A'CHKFD2'
*
*
*        DELETE ENTIRE TERMINAL ENTRY IN TABLE
*
DEL      MC BLNKS(36),TCTABL(,1)     BLANK OUT ENTIRE TERMINAL TABLE EN
         S  ONE,CNTR                 SUBTRACT 1 FROM COUNTER
         BC *+10(3),TSET2(2)         IF ZERO, GOTO NORMAL EXIT ROUTINE
         A  FIVE,X2                  IF POSITIVE, ADD 5 TO INDEX 2
         BC LDTBL2(5)                BRANCH AND SEARCH FOR TERMINAL ID
*
*
*        REPLACE TERMINAL ID  -  OPTION R
*        TEST FOR NUMERIC ENTRY
*
CHKFR    MC ZEROS,X2                 SET INDEX 2 TO 0
         MC ZEROS(1),CNTR            SET COUNTER TO 0
CHKFR2   C  BUFFER+10(1,2),BLNKS     IS NEXT ENTRY BLANK?
         BC TSET1(2)                 IF YES, GOTO ERROR ROUTINE
         MC BUFFER+10(4,2),TS3       IF NO, MOVE ENTRY TO TS3
         PN TS3(5),TS4(5)            FORM NUMERIC
```

```
         C   TS3(4),TS4                IS ENTRY NUMERIC?
         BC  **+10(2),TSET1(5)         IF NO, GOTO ERROR ROUTINE
         MN  ADC11,R+1                 MODIFY INSTRUCTION R TO BRANCH TO
         BC  CHKFD1+1(6),STORAR(5)     BRANCH AND LINK TO STORAR
         A   TEN,X2+2(2)               ADD 10 TO INDEX 2
         C   BUFFER+5(1,2),BLNKS       IS NEXT ENTRY BLANK?
         BC  LDTBL1(2)                 IF YES, GOTO FIND TERMINAL ID
         MN  ADCHNG,Z+1                IF NO, MODIFY INSTR. Z TO CHKFR2
         BC  CHKT1(5)                  BRANCH TO TEST FOR NUMERIC ENTRY
**
         ORG *-5
ADCHNG   DM  A'CHKFR2'
*
*
*        REPLACE TERMINAL ID
*
RDEL     MC  BUFFER+10(4,2),TCTABL(,1) MOVE ENTRY TO TABLE
         S   ONE,CNTR                  SUBTRACT 1 FROM COUNTER
         BC  **+10(3),TSET2(2)         IF ZERO, GOTO NORMAL EXIT ROUTINE
         A   TEN,X2                    IF POSITIVE, ADD 10 TO INDEX 2
         BC  LDTBL2(5)                 BRANCH TO FIND ANOTHER TERMINAL ID
*
         ORG   *-5
ADC11    DM  A'RDEL'
*
*        DELETE TRANSACTION TABLE ENTRY  -  OPTION T
*
*        TEST TRANSACTION CODE (1-8)
*
CHKFT    MC  ZEROS,X3                  SET INDEX 3 TO 0
         MC  ZEROS(1),CNTR             SET COUNTER TO 0
CHKFT1   C   BUFFER+10(1,3),NINE       IS NEXT TRANSACTION CODE = 9?
         BC  **+10(1),TSET1(5)         IF YES, GOTO ERROR ROUTINE
         C   BUFFER+10(1,3),ONE        IF NO, IS IT LESS THAN ONE?
         BC  TSET1(1)                  IF YES, GOTO ERROR ROUTINE
         MN  ADC10,R+1                 MODIFY INSTR. R TO TDEL
         BC  CHKFD1+1(6),STORAR(5)     BRANCH AND LINK TO STORAR
         A   TWO,X3+2(2)               ADD TWO TO INDEX 3
         C   BUFFER+10(1,3),BLNKS      IS THERE ANOTHER TRANSACTION CODE?
         BC  LDTBL1(2),CHKFT1(5)       IF NO, GO FIND TERMINAL ID
*                              IF YES, LOOP BACK AND TEST NEXT TRAN CDE
*
*
*        DELETE TRANSACTION IN TABLE
*
TDEL     MC  ZEROS,X3                  SET INDEX 3 TO 0
         MC  ZEROS,X2                  SET INDEX 2 TO 0
TDEL1    MC  FOUR,X2+2                 SET INDEX 2 TO '0040'
         M   BUFFER+10(1,3),X2+2(1)    MULTIPY TRANSACTION CODE BY 4
         A   X2+2(2),X1(4)             ADD INDEX 2 TO INDEX 1
         MC  BLNKS(4),TCTABL(,1)       CLEAR APPROPRIATE LENGTH ENTRY
         S   ONE,CNTR                  SUBTRACT 1 FROM COUNTER
         BC  **+10(3),TSET2(2)         IF ZERO, GO TO NORMAL OUT ROUTINE
         A   TWO,X3                    IF POSITIVE, ADD 2 TO INDEX 3
         S   X2+2(2),X1(4)             RESTORE CONTENTS OF INDEX 1
         BC  TDEL+10(5)                LOOP BACK TO DELETE NEXT TRAN CDE
*
         ORG *-5
ADC10    DM  A'TDEL'
*
```

```
*       ADD TRANS. LENGTH TO TABLE  -  OPTION A
*
*       TEST TRANSACTION CODE (1-8)
*
ChkFA   MC  ZEROS(1),CNTR           SET COUNTER TO ZERO
        MC  ZEROS,X2                SET INDEX 2 TO ZERO
        MC  ZEROS,X3                SET INDEX 3 TO ZERO
        C   BUFFER+10(1,3),BLNKS    IS TRANSACTION CODE  ENTRY BLANK?
        BC  TSET1(2)                IF YES, GOTO ERROR ROUTINE
ChkFA1  C   BUFFER+10(1,3),NINE     IF NO, IS TRANSACTION CODE=9?
        BC  *+10(1),TSET1(5)        IF YES, GOTO ERROR ROUTINE
        C   BUFFER+10(1,3),ONE      IF NO, IS TRAN CDE LESS THAN 0?
        BC  TSET1(1)                IF YES, GOTO ERROR ROUTINE
        MC  BUFFER+10(1,3),TS3      IF NO, MOVE TRAN CDE TO TS3
        C   BUFFER+12(1,3),BLNKS    IS ASSOCIATED LENGTH BLANK?
        BC  TSET1(2)                IF YES, GOTO ERROR ROUTINE
        MC  BUFFER+12(3,3),TS3+1    IF NO, MOVE LENGTH TO TS3
        C   BUFFER+16(1,3),TWO      IS ROUTING INDICATOR <2?
        BC  *+10(1),TSET1(5)        IF NO, GOTO ERROR ROUTINE
        PK  TS3(5),TS4(5)           IF YES, FORM NUMERIC
        C   TS3(4),TS4              IS ENTRY NUMERIC?
        BC  *+10(2),TSET1(5)        IF NO, GOTO ERROR ROUTINE
        MC  BUFFER+16(1,3),TS3+4    IF YES, MOVE ROUTING INDICATOR TO
        MC  FOUR,X2+2               SET INDEX 2 TO '0040'
        M   BUFFER+10(1,3),X2+2(1)  MULTIPLY TRAN CDE BY 4
        MC  TS3+1(4),TSWKA(,2)      MOVE TS3 TO TSWKA
        BC  ChkFD1+1(6),STORAR(5)   BRANCH AND LINK TO STORAR
        A   EIGHT,X3+2(2)           ADD 8 TO INDEX 3
        C   BUFFER+10(1,3),BLNKS    IS NEXT TRAN CDE BLANK?
        BC  TAZK(2),ChkFA1(5)       IF YES, GOTO FIND TERMINAL ID
*                                   IF NO, LOOP BACK TO TEST NEXT TRAN CDE
*
*
*       FIND TERINAL ID IN TABLE
*
TAZK    MC  ZEROS,X1                SET INDEX 1 TO 0
        MC  ZEROS, X2               SET INDEX 2 TO 0
        MC  ZEROS,X3                SET INDEX 3 TO 0
TAZK1   C   BUFFER+5(4),TCTABL(,1)  ARE TERMINAL ID'S EQUAL?
        BC  TAD3(2)                 IF YES, GOTO TAD3
        C   TCTABL(4,1),TCEOT       IF NO, IS THIS THE END OF TABLE?
        BC  TAD(2)                  IF YES, GOTO TAD
        A   TCONS1,X1               IF NO, ADD 36 TO INDEX 1
        BC  TAZK1(5)                LOOP BACK AND CONTINUE SEARCH
*
*
*
*
TAD     MC  ZEROS,X1                SET INDEX 1 TO 0
TAD2    C   TCTABL(4,1),BLNKS       IS TERMINAL ID BLANK?
        BC  *+10(2),*+30(5)         IF NO, GO TEST FOR END OF TABLE
        MC  BUFFER+5(4),TCTABL(,1)  IF YES, MOVE NEW TERMINAL ID TO TA
        BC  TAD3(5)                 BRANCH TO ADD ASSOCIATED TRAN CDE
        C   TCTABL(4,1),TCEOT       IS THIS THE END OF THE TABLE?
        BC  TSET3(2)                IF YES, GO SIGNAL TERMINAL ID ERRO
        A   TCONS1,X1               IF NO, ADD 36 TO INDEX 1
        BC  TAD2(5)                 LOOP BACK AND CONTINUE SEARCH
**
*       ADD NEW TRANSACTION CODE LENGTH AND ROUTING INDICATOR
*
```

```
TAD3   MC FOUR,X2+2              SET INDEX 2 TO '0040'
       M  BUFFER+10(1,3),X2+2(1) MULTIPLY TRAN CDE BY 4
       A  X2+2(2),X1(4)          ADD INDEX 2 TO INDEX 1
       MC TSWKA(4,2),TCTABL(,1)  MOVE LENGTH, ROUTING INDICATOR TO
       S  ONE,CNTR               SUBTRACT 1 FROM COUNTER
       BC *+10(3),TSET2(2)       IF ZERO, GOTO NORMAL EXIT ROUTINE
       A  EIGHT,X3               IF POSITIVE, ADD 8 TO INDEX 3
       S  X2+2(2),X1(4)          RESTORE INDEX 1
       BC TAD3(5)                LOOP BACK AND CONTINUE
**
**
*      KEEP COUNT OF CONSECUTIVE ENTRIES IN MESSAGE
*
STORAR A  ONE,CNTR               ADD 1 TO COUNTER
       C  CNTR,EIGHT             IS COUNTER =8?
CHKFD1 BC CHKFD1(1),LDTBL1(5)    IF NO,RETURN
*                          IF YES, GOTO FIND TERMINAL ID
**
*
*      FIND TERMINAL ID IN TABLE
*
LDTBL1 MC ZEROS,X1               SET INDEX 1 TO 0
       MC ZEROS,X2               SET INDEX 2 TO 0
LDTBL2 C  BUFFER+5(4,2),TCTABL(,1) ARE TERMINAL ID'S EQUAL?
R      BC DEL(2),*+10(5)         IF YES, GOTO MODIFIED ADDRESS
       C  TCTABL(4,1),ICEOT      IF NO, IS THIS END OF TABLE?
       BC TSET3(2)               IF YES, GOTO ERROR ROUTINE
       A  TCONST,X1              ADD 36 TO INDEX 1
       BC LDTBL2(5)              AND LOOP BACK TO LDTBL2
*
*
*      ERROR ROUTINE
*
TSET1  MC TWO,INPSW              SET INPSW TO SIGNAL INVALID ENTRY
       BC TOUT(5)                BRANCH TO EXIT ROUTINE
*
       ORG *-5
ADCNS1 DM A'ZZ'
*
*
*      NORMAL EXIT ROUTINE
*
TSET2  MC ZERO,INPSW             SET INPSW TO SIGNAL NORMAL
       BC TOUT(5)                BRANCH TO EXIT ROUTINE
*
       ORG   *-5
A      DM    0C4                COMMAND OPTION LIST
       DM    C'D'               MEANS DELETE ENTIRE TERMINAL ENTRY
       DM    C'A'               MEANS ADD TRANSACTION CODES  CNTR=1
       DM    C'T'               MEANS DELETE TRANSACTION CODES CNTR=2
       DM C'R'                  MEANS REPLACE TERMINAL ID   CNTR=3
ASTER  DM C'*'                  END OF POSSIBLE OPTIONS
*
*
*      TERMINAL ID ERROR ROUTINE
*
TSET3  MC THREE,INPSW            SET INPSW TO SIGNAL TERMINAL ERROR
TOUT   MC ZEROS(2),BUFFER+2      REMOVE SIGNAL FOR TABLE UPDATE
       MC ZEROS(1),TS3+4         RESET FORM NUMERIC FIELD
CDEXIT BC CDEXIT(5)              RETURN ADDRESS
```

```
*
        ORG   *-5
CNTR    DM C'0'                 COUNTER - USED TO INDICATE COMMAND OPTN
*                               - USED TO INDICATE NUMBER OF CONSECUTIVE
*                               ENTRIES
TS3     DM C'00000'
TS4     DM C'00000'
TEN     DM C'10'
TSMKA   DM C36' '
BLNKS   DM C36' '
*
*
*
*
```

```
          TITLE  'BASIC CD-30 MODULE'
          COMMON
CDNC      DM     C'NC'          HEADER PREFIX N=NORMAL,C=CD30
PCERPT    DM     C' '           ERROR NUMBER
*         CDNC MUST BE FOLLOWED BY THE FOLLOW SYSTEM CONSTANTS
*         PCERPT  DM  C' '
*         PRCONC  DM C5'00000'
*         SCTIME DM N8
E         DM     C'E'
BLANK     DM C7' '
*
*         CONSTANTS
*         TO CALL CODE CONVERT
*
SIZE1     DM     N'006'                ** CONVERT (CD-30 TO BDP)
IADDR1    DM     A'CDIN'               ** THE FIRST 6 CHARS OF INP
TADDR1    DM     A'TBL1'               ** USING TBL1
*                                      **
SIZE2     DM N'022'                    ** CONVERT (BDP TO OUTPUT)
IADDR2    DM     A'CDHEAD'             **  THE 16 CHARS OF CDHEAD
*                                      ** AND THE FIRST 6 CHARS OF IN
TADDR2    DM     A'TBL2'               **  INPUT USING TBL2
*                                      **
IADDR3    DM A'CD6'                    ** CONVERT (CD30 TO OUTPUT)
*                                      ** THE REMAINING MESSAGE
TADDR3    DM     A'TBL3'               **  USING TBL3
*
*
*
*

*         BEGINNING OF CD-30 PARTITION
*
*         THE PURPOSE OF THIS MODULE IS TO READ CD-30 TRANSMISSIONS;
*         PREFIX THE TRANSMISSION WITH TIME STAMP AND ROUTING INFOR.;
*         CHECK THAT TRANSMISSION FOR ERRORS AND FOR WRONG LENGTH MESSAGE
*         RESPOND TO CD-30 TERMINAL; AND TO STORE TRANSMISSIONS IN
*         ERROR INTO AN ERROR BUFFER TO BE PRINTED ON 7102.
*         ALL TRANSMISSIONS IN THIS LISTING ARE CONVERTED TO OUTPUT
*         CODE AND ARE WRITTEN TO AN OUTPUT DEVICE.
*
          NORMAL
          ORG    0000
          DM    7C9'&&---000'
          ORG    0000
          BC TATTLE(5)
          ORG    *-4
          DM     A'CDSTR'
*
          ORG    0011
          DM     C4'0000'             INDEX REGISTER 1
          ORG    0021
          DM     C4'0000'             INDEX REGISTER 2
CD5P      DM     C4                   FIVE TIMES PARTTN NUMBER
          ORG    0031
          DM     C4'0000'             INDEX REGISTER 3
          ORG    0047
CDPNO     DM     C2                   CD PARTTN NUMBER
*
```

```
*       CONSTANTS
*
FOUR5   DM C'0004'
SIXTY4  DM     N'0064'
CDBR    DM     N'0101'              B REGISTER CONSTANT
CDMIN   DM    N'0002'           MINUMUM MESSAGE LENGTH
TCCNS1  DM C'0036'
INTY2   DM     N'0032'
*
*       SWITCHS AND STORAGE AREAS
*
CDTCNT  DM     C'0000'              MESSAGE LENGTH EXPECTED
CDMXLN  DM     N'0199'                MAXIMUM INPUT LENGTH MINUS 1
CDERSW  DM     0C1                    LOCAL ERROR SWITCH
CDERID  DM     C'0    '               ERROR DESCRIPTION FIELD
ERRID   DM C'0'
        ORG  *-1
CDGOOD  DM     N'0'              SECOND ERROR SWITCH
TRNCDE  DM C'0'                  TRANSACTION CODE
        DM C1
TCTID   DM C'0000'               TERMINAL ID
*
*
*       INPUT/OUTPUT AREA
*
CDMSGL  DM     C4                     MESSAGE LENGTH
CDHEAD  DM C16                   MIS HEADER PREFIX
CDIN    DM 0C200                 INPUT\OUTPUT AREA FOR CD-30 TRANS.
        DM C6
CD6     DM C194
*
*                                ERROR CODES
*
CDPAR1  DM     C'1PAR1'          ERROR NO. 1 - PARITY ERROR
CDPAR2  DM C'2PAR2'              ERROR NO. 2   FAULT
CDPAR3  DM C'3PAR3'              ERROR NO. 3   TRANSMISSION (ACK/NAK)
*
*       TERMINAL READ MODULE
*
CDSTR   C   CDPNO,BUFFER+2       DOES TABLE NEED UPDATE?
        BC CDEXIT+1(6),CDCHK(2)  IF YES,GO TO TABLE UPDATE ROUTINE
*
        C   INHS#,ZERO           IS SYSTEM INHIBITED?
        BC *+10(2),CDSTR(8)      IF YES,BRANCH & SWITCH PARTITION
*
        BC CDRD(7),CDRD(0)       CD-30 SERVICE REQUEST?
        BC CDSTR(8)              IF NO,SWITCH PARTITION
*
CDRD    MC BLANK,CDIN            CLEAR 1ST 6 CHARS OF INPUT AREA
        R  CDIN(0),0200(5)       READ CD-30 TRANSMISSION
        BC *+40(2),*+30(1)
*
        MC CDPAR2,CDERID         INDICATE ERROR 2 - FAULT ON READ
        BC *+20(5)
*
        MC CDPAR1,CDERID         INDICATE ERROR NO. 1 - PARITY ON R
        A  ONE,PRCDNC            ADD 1 TO RECORD SERIAL #
        BC *+10(4),*+20(5)       OVERFLOW?
        MC ZEROS(5),PERCNT       IF YES,SET ERROR COUNT TO ZERO
```

```
*
        MC CDNC(16),CDHEAD
*
        MC CDBR,X2                  *
        A  CDBP,X2                  *
        MC CDBXLN,CDMSGL            * COMPUTE MESSAGE LENGTH
        S  CN(,2),CDMSGL            *
        BC **20(3),**10(5)          IF RESULT IS NOT POSITIVE
        MC CDNIN+2(4),CDMSGL        SET CDMSGL TO '200'
        C  CDMSGL,CDNIN             IS MESSAGE LENGTH .< 2?
        BC **10(1),**20(5)          IF NO, SKIP NEXT INSTRUCTION
        A  TWO,CDMSGL               IF YES,ADD 2 TO MESSAGE LENGTH
*
*
        MC SIZE1,CCSIZE             SET LENGTH FOR CODE CONVERT TO 6
        MC IADDR1,X1                SET REG1 TO INPUT ADDR = CDIN
        MC IADDR1,X2                SET REG2 TO OUTPUT ADDR = CDIN
        MN TADDR1,CC8+1             SET A=ADDR TO TABLE = TBL1
        BC CC4+1(6),CC1(5)          BRANCH & LINK TO CODE CONVERT
        MC CDIN(6),TRNCDE           SAVE TRNCDE AND TERMINAL ID
*
        C  CDERSW,ZERO              IS TRANSMISSION IN ERROR?
        BC **20(3)                  IF YES,BRANCH AROUND CALL TO TERMI
*                             CHECK MODULE
        BC TCFXIT+1(6),TCSTRT(5)    BRANCH AND LINK TO TERMINAL CHECK
*
        MC SIZE2,CCSIZE             SET LENGTH FOR CODE CONVERT TO 22
        MC IADDR2,X1                SET REG1 TO INPUT ADDR = CDHEAD
        MC IADDR2,X2                SET REG2 TO OUTPUT ADDR = CDHEAD
        MN TADDR2,CC8+1             SET A=ADDR TO TABLE = TBL2
        BC CC4+1(6),CC1(5)          BRANCH AND LINK TO CODE CONVERT
*
*
        MC CDMSGL,X1                MOVE MESSAGE LENGTH TO INDEX 1
        S  SIX,X1                   SUBTRACT 6 FROM INDEX 1
        BC **60(1)                  IF MINUS,BRANCH AROUND NEXT CALL T
*
*
        MC X1+1(3),CCSIZE           SET LENGTH TO REMAINING LENGTH OF
        MC IADDR3,X1                SET REG1 TO INPUT ADDR = CDIN+6
        MC IADDR3,X2                SET REG2 TO OUTPUT ADDR = CDIN+6
        MN TADDR3,CC8+1             SET A=ADDR TO TABLE = TBL3
        BC CC4+1(6),CC1(5)          BRANCH & LINK TO CODE CONVERT
*
*
*       ACKNOWLEDGEMENT MODULE
*
CDACK   MC CDERID(1),CDGOOD         SAVE LOCAL ERROR SWITCH
*                             BEFORE ERROR 3 CAN OCCUR
        C  TRNCDE,EIGHT             IS THIS A BADGE TRANSACTION?
        BC CDINH(2)                 IF YES,BRANCH TO CDINH
        C  CDERSW,ZERO              IS TRANSMISSION IN ERROR?
        BC **50(2)
        W  0(2),02(3)               WRITE NAK
        BC CDINH(2)
        W  0(2),02(3)               WRITE NAK AGAIN
        BC CDINH-10(5)
        W  0(1),02(3)               WRITE ACK
        BC CDINH(2)
        W  0(1),02(3)               WRITE ACK AGAIN
```

```
          MC CDPAR3,CDERID            INDICATE ERROR 3
     *
     *
     CDINH  C  INHSW,ZERO             IS SYSTEM INHIBITED?
            BC **10(2),CDINH(8)       IF YES BRANCH AND SWITCH
            C  CDERSW,ZERO            IS TRANSMISSION IN ERROR?
            BC CDERPO(3),CDWT(5)      IF NO, BRANCH TO WRITE ROUTINE
     *                              IF YES,CONTINUE
     *
     *
     *
     *
     *       ERROR PROCESSING MODULE
     *
     CDERPO C  ECOUNT,THREE           ERROR BUFFERS FILLED?
            BC CDE1(1)
            MN ONE,INHSW             IF YES,TURN ON INHIBIT SWITCH
            MN ONE,ERRSW             TURN ON ERROR SWITCH
            C  INHSW,ZERO            WAIT FOR ERROR BUFFERS
            BC **10(2),*-10(8)       TO BE PRINTED ON /102
     CDE1   C  ECOUNT,ONE            WHICH BUFFER IS EMPTY?
            BC **30(2),**50(3)
            MC ZEROS,X1              SET INDEX TO FILL FIRST ERROR BUFF
            BC CDE2(5)
            MC THTY2,X1              SET INDEX TO FILL SECOND ERROR BUF
            BC CDE2(5)
            MC SIXTY4,X1             SET INDEX TO FILL THIRD ERROR BUFF
     *
     *       FILL ERROR BUFFER
     *
     CDE2   MC CDERID+1(4),ERRBUF+7(,1) ERROR ID (4)
            MC CDPRO,ERRBUF+12(,1)   PARTITION NUMBER
            MC SCHRS,ERRBUF+15(,1)   HOURS (2)
            MC SCL1+2(1),ERRBUF+18(,1) TENTHS
            MC SCL1+3(1),ERRBUF+20(,1) HUNDREDS
            MC TCTID,ERRBUF+22(,1)   TERMINAL NUMBER
            MC TRNCDE,ERRBUF+27(,1)  TRANSACTION CODE
            MC CDMSGL+1(3),ERRBUF+29(,1) MESSAGE LENGTH
            MC CDTCNT+1(3),ERRBUF+33(,1) EXPECTED MESSAGE LENGTH
     *
            A  ONE,ECOUNT            ADD 1 TO ERROR COUNTER
            C  CDGOOD,CDERSW         HAS ERROR 3 OCCURED?
            BC **30(2)
            MC CDGOOD,CDERSW         IF YES,RESTORE LOCAL ERROR SWITCH
            BC CDINH+20(5)           AND RETURN TO ERROR PROCESSING
     *                              ROUTINE IF NECESSARY
     *
            MC ZERO,CDERSW           CLEAR THE LOCAL ERROR SWITCH
            MC ZEROS(3),CDTCNT       CLEAR MODULUS REMAINDER AREA
            C  CDGOOD,ZERO           IS TRANSMISSION IN ERROR?
     *                              CDGOOD CONTAINS ERROR NO. 1,2,4, OR 5.
            BC CDWT(2)               IF NO, GO TO WRITE ROUTINE
            MC E,CDHEAD              MOVE E TO HEADER
            MC CDGOOD,CDHEAD+2       MOVE ERROR NO. TO 3RD POSITION OF
            A  ONE,PERCNT            ADD 1 TO ERROR RECORD COUNT
     *
     *                              TAPE WRITE ROUTINE
     *
     CDWT   A SIXTN,CDMSGL
            MC CDMSGL,X2
            BC WTEXIT+1(6),WTSTR(5)
            BC CDSTR(5)
     WTBUF  DM A'CDHEAD'
```

```
*       TERMINAL TRANSACTION CHECK MODULE
*
*       THE PURPOSE OF THIS ROUTINE IS TO CHECK THE MESSAGE LENGTH
*       OF THE TRANSMISSION ACCORDING TO THE TRANSACTION CODE AND
*       A PREDEFINED TABL.
*
TCSTRT MC  ZEROS,X1
       MC  ZEROS,X2                    SET INDEX 2
*
       MC  TRNCDE(1),X2+2              *
       M   FOUR,X2+2(1)                *  COMPUTE TABLE POINTER
       A   X2+2(2),X1(4)               *
       C   TCTABL(3,1),BLANK           IS POSITION IN TABLE BLANK?
       BC  TCERR2(2)                   IF YES,SIGNAL TRNCDE ERROR.
*
       C   CDMSGL+1(3),TCTABL(,1)      DOES MESSAGE LENGTH CHECK?
       BC  TCEXIT-10(2)                IF YES,EXIT.
*
       MC  TCIWLM,CDERID               IF NO,SIGNAL WRONG LENGTH MSG
       MC  TCTABL(3,1),CDTCNT+1        MOVE CURRECT LENGTH TO CDTCNT
       BC  TCEXIT(5)                   EXIT
*
*
TCERR2 MC  TCITC,CDERID                SIGNAL TRNCDE ERROR
       MC  BLANK(4),CDTCNT             BLANK OUT CDTCNT
*
TCEXIT BC  TCEXIT(5)                   EXIT
*
TCTABL DM  0C36                  TERMINAL TRANSACTION TABLE
       DM  C36'9999076 076 076 076 076 076 076 042 '
TCEOT  DM  C':::::'
TCITC  DM  C'4TRAN'                    TRANSACTION CODE INVALID
TCIWLM DM  C'5WLM '                    WRONG LENGTH MESSAGE
*      END OF ROUTINE
*
*
*
```

```
*****************************************************************
*                                                               *
*       GENERAL FUNCTIONAL DESCRIPTION                          *
*                                                               *
*           EACH CD-30 PARTITION CONTAINS A TRANSACTION TABLE.  *
*         THIS TABLE CONTAINS AN APPROPRIATE MODULUS FOUR       *
*         REMAINDER FOR EACH TRANSACTION THAT IS VALID FOR THE  *
*         TERMINALS CONNECTED TO THE PARTITION.  THIS PARTICULAR *
*         TABLE ACCOMODATES UP TO EIGHT TRANSACTIONS, SEVEN OF  *
*         WHICH ARE VALID.  AT LOAD TIME, EACH CD-30 PARTITION  *
*         TRANSACTION TABLE IS INITIALIZED WITH THE APPROPRIATE *
*         REMAINDERS FOR EACH VALID TRANSACTION.                *
*                                                               *
*         IN ADDITION,  THE MODULUS REMAINDERS WHICH ARE INITIALLY *
*         STORED IN THE TABLE MAY BE READILY DELETED ANY TIME IT *
*         IS NECESSARY TO ALTER THE VALID TRANSACTIONS SPECIFIC  *
*         TO EACH MASTER CD30 PARTITION.  NEW TRANSACTIONS OR    *
*         NEW REMAINDERS FOR EXISTING TRANSACTIONS CAN BE ADDED AT *
*         ANY TIME.                                             *
*                                                               *
*****************************************************************

*****************************************************************
*                                                               *
*       METHOD                                                  *
*                                                               *
*        TRANSACTION DELETION                                   *
*                                                               *
*          ALL OPERATIONS ARE ACCOMPLISHED VIA A 7102 TERMINAL  *
*          KEYBOARD.  TO DELETE A TRANSACTION REMAINDER AND ITS *
*          ASSOCIATED ROUTING INDICATOR, OBTAIN A SERVICE REQUEST *
*          ON THE 7102 KEYBOARD AND ENTER DATA ACCORDING TO THE *
*          FOLLOWING FORMAT:                                    *
*                                                               *
*                        PTXX-X                                 *
*                                                               *
*                          WHERE                                *
*                                                               *
*                            XX IS A TWO DIGIT PARTITION NUMBER IN *
*                                 THE RANGE 02 THROUGH 19.      *
*                                                               *
*                             X  IS A ONE DIGIT TRANSACTION CODE *
*                                  IN THE RANGE 1 THROUGH 8.    *
*                                                               *
*          THE OPERATOR MAY EITHER SPECIFY ONE "PT" FOR EACH    *
*          TRANSACTION THAT IS TO BE DELETED OR HE MAY SPECIFY ONE *
*          "PT" FOLLOWED BY FROM TWO TO EIGHT SEPARATE TRANSACTIONS. *
*          THE COMPLETION OF AN ENTRY IS SIGNALLED BY DEPRESSING *
*          "CONTROL /" (UNIT SEPERATOR) ON THE 7102.            *
*          THE PROGRAM WILL THEN DELETE THE APPROPRIATE TRANSACTION *
*          REMAINDER AND ROUTING INDICATOR FROM THE TABLE.      *
*                                                               *
*                                                               *
*          BOTH OF THE FOLLOWING DEMONSTRATE VALID TRANSACTION  *
*          DELETION FORMATS:                                    *
*                                                               *
*                             PT02-4/                           *
*                             PT02-1/                           *
*                             PT02-8/                           *
*                                                               *
*****************************************************************
```

```
*                                                                     *
*                              OR                                     *
*                                                                     *
*                         PT02-4-1-8/                                 *
*                         PT02-4-1-5-2-3-6-8-7/                       *
*                                                                     *
*        A TRANSACTION DELETION (PT) MUST ALWAYS PRECEEDE A           *
*        TRANSACTION ADDITION (PA), UNLESS A TABLE ENTRY IS           *
*        INITIALIZED TO BLANKS OR AN ENTRY IS DELETED AND A NEW       *
*        VALUE IS NOT IMMEDIATELY ADDED.                              *
*                                                                     *
*      TRANSACTION ADDITION                                           *
*                                                                     *
*        AFTER OBTAINING A SERVICE REQUEST ON THE 7102, THE           *
*        OPERATOR MAY ADD TRANSACTION REMAINDERS AND ASSOCIATED       *
*        ROUTING INDICATORS BY ENTERING THE FOLLOWING DATA:           *
*                                                                     *
*                      PAXX-X-XXX-X                                   *
*                                                                     *
*                         WHERE                                       *
*                                                                     *
*                           XX IS A TWO DIGIT PARTITION NUMBER IN     *
*                              THE RANGE 02 THROUGH 19.               *
*                                                                     *
*                            X IS A ONE DIGIT TRANSACTION NUMBER      *
*                              IN THE RANGE 1 THROUGH 8.              *
*                                                                     *
*                          XXX IS A THREE DIGIT MODULUS REMAINDER     *
*                              IN THE RANGE 000 THROUGH 003.          *
*                                                                     *
*                            X IS A ONE DIGIT ROUTING INDICATOR       *
*                              WHICH MAY BE A BLANK, ZERO, OR         *
*                              ONE.                                   *
*                                                                     *
*        THE OPERATOR MAY ENTER A SEPARATE "PA" CODE FOR EACH         *
*        TRANSACTION HE WISHES TO ENTER, OR HE MAY ENTER ONE "PA"     *
*        AND SPECIFY FROM TWO TO EIGHT TRANSACTIONS AND THEIR         *
*        ASSOCIATED REMAINDERS AT ONE TIME.  THE COMPLETION OF        *
*        AN ENTRY IS SIGNALLED BY DEPRESSING "CONTROL /" (UNIT        *
*        SEPARATOR) ON THE 7102.  THE PROGRAM WILL THEN ENTER THE     *
*        DATA IN THE APPROPRIATE PLACE IN THE TRANSACTION TABLE.      *
*        BOTH OF THE FOLLOWING DEMONSTRATE VALID ADDITION FORMATS:    *
*                                                                     *
*                         PA02-4-000- /                               *
*                         PA02-2-001- /                               *
*                         PA02-8-003- /                               *
*                                                                     *
*                              OR                                     *
*                                                                     *
*                   PA02-4-000- -2-001- -8-003- /                     *
*                                                                     *
*                                                                     *
*        AT THE PRESENT TIME THE ROUTING INDICATOR HAS NO             *
*        FUNCTION.  HOWEVER, A BLANK ZERO, OR ONE MUST BE             *
*        ENTERED IN THE SPECIFIED POSITION EACH TIME A TRANSACTION    *
*        REMAINDER IS ADDED.                                          *
*                                                                     *
*        NOTE THAT THE ACTUAL VALUES OF THE TRANSACTION CODES         *
*        FOR BOTH "PA" AND "PT" MAY BE ENTERED IN ANY NUMBERIC        *
```

```
*****************************************************************
*                                                               *
*       ERROR CHECKING                                          *
*                                                               *
*          BEFORE A TRANSACTION REMAINDER AND ROUTING INDICATOR IS *
*          ADDED TO OR DELETED FROM THE TABLE A SERIES OF CHECKS *
*          ARE PERFORMED ON THE DATA ENTERED.  TWO TYPES OF ERROR *
*          MESSAGES ARE POSSIBLE:                               *
*                                                               *
*                    WLM TABLE ERR (OR INVALID ENTRY)           *
*                    FORMAT ERROR                               *
*                                                               *
*          FOR "PA" THESE MESSAGES MAY BE INTERPRETED AS FOLLOWS: *
*                                                               *
*                    WLM TABLE ERR                              *
*                                                               *
*                         1.  THE OPERATOR ATTEMPTED TO ADD A   *
*                             NEW TRANSACTION REMAINDER TO THE  *
*                             TABLE WITHOUT DELETING THE        *
*                             ONE ALREADY RESIDING THERE.       *
*                                                               *
*                         2.  THE TWO DIGIT PARTITION NUMBER WAS *
*                             A) NOT IN THE RANGE 02-19.        *
*                             B) CONTAINED NON NUMERIC CHARACTERS *
*                             C) WAS NOT PUNCHED ACCORDING TO THE *
*                                SPECIFIED FORMAT.              *
*                                                               *
*                    FORMAT ERROR                               *
*                                                               *
*                         1.  THE TRANSACTION CODE, THE MODULUS *
*                             REMAINDER, AND/OR THE ROUTING     *
*                             INDICATOR CONTAINED NON-NUMERIC   *
*                             CHARACTERS.                       *
*                         2.  A TRANSACTION CODE ENTERED WAS NOT IN THE*
*                             RANGE 1-8.                        *
*                         3.  A ROUTING INDICATOR ENTERED WAS NOT A *
*                             BLANK, ZERO OR ONE.               *
*                         4.  THE DATA WAS NOT INPUTED ACCORDING TO *
*                             THE REQUIRED "PA" FORMAT.         *
*                                                               *
*                                                               *
*          FOR "PT" THESE MESSAGES MAY BE INTERPRETED AS FOLLOWS: *
*                                                               *
*                    WLM TABLE ERR                              *
*                                                               *
*                         1.  THE TWO DIGIT PARTITION NUMBER WAS *
*                             A) NOT IN THE RANGE 02-19.        *
*                             B) CONTAINED NON NUMERIC CHARACTERS *
*                             C) WAS NOT PUNCHED ACCORDING TO THE *
*                                SPECIFIED FORMAT.              *
*                                                               *
*                    FORMAT ERROR                               *
*                                                               *
*                         1.  A TRANSACTION CODE ENTERED WAS NON-NUMERIC*
```

```
*                    2.  A TRANSACTION CODE ENTERED WAS NOT IN THE*
*                        RANGE 1-8.                               *
*                    3.  THE DATA WAS NOT INPUTED ACCORDING TO    *
*                        THE REQUIRED "PT" FORMAT.                *
*                                                                 *

*  DETERMINE THE TYPE OF OPERATION TO BE PERFORMED: TRANSACTION
*     ADDITION OR TRANSACTION DELETION.
*
CDCHK   C   BUFFER+1(1),T       IS THIS A TRANSACTION DELETION.
        BC  CHKFT(2)            IF SO, BRANCH TO CHECK FORMAT FOR PT.
        C   BUFFER+1(1),A       IS THIS A TRANSACTION ADDITION.
        BC  CHKFA(2),TSET1(5)   IF SO, BRANCH TO CHECK FORMAT FOR PA.
*                               OTHERWISE MOVE A TWO TO INPSW AND PRINT
*                               FORMAT ERROR TO INDICATE INVALID
*                               OPERATION ATTEMPTED.
*
*  TRANSACTION DELETION
*
CHKFT   MC  ZEROS,X1            ZERO INDEX REGISTER ONE.
CHKFT2  C   BUFFER+5(1,1),NINE  WAS THE TRANSACTION CODE ENTERED LESS
*                               THAN NINE.
        BC  *+10(1),TSET1(5)    CONTINUE IF IT IS, OTHERWISE MOVE A TWO
*                               TO INPSW AND PRINT FORMAT ERROR.
        C   BUFFER+5(1,1),ONE   WAS THE TRANSACTION CODE LESS THAN ONE.
        BC  TSET1(1)            CONTINUE IF IT IS, OTHERWISE MOVE A TWO
*                               TO INPSW AND PRINT FORMAT ERROR.
*                               DELETE THE REMAINDER AND ROUTING
*                               INDICATOR FOR THE TRANSACTION SPECIFIED.
        BC  CHKFT9+1(6),LDTBL1(5)
        A   TWO,X1+2(2)         INCREMENT INDEX ONE BY TWO.
        C   BUFFER+5(1,1),BLK   CHECK TO SEE IF ADDITIONAL ENTRIES ARE I
*                               THE BUFFER.
        BC  TSET2(2),CHKFT2(5)  IF THERE ARE NONE, MOVE A ONE TO INPSW
*                               TO INDICATE THE OPERATION WAS VALID AND
*                               HAS BEEN COMPLETED. OTHERWISE, BRANCH
*                               TO CHKFT2 AND CHECK THE FORMAT OF THE
*                               NEXT ENTRY.
*
*  TRANSACTION ADDITION
*
CHKFA   MC  ZEROS,X2            ZERO INDEX REGISTER TWO.
CHKFA2  C   BUFFER+5(1),NINE    CHECK THE TRANSACTION CODE ENTERED TO SE
*                               IF IT IS LESS THAN NINE.
        BC  *+10(1),TSET1(5)    CONTINUE IF IT IS, OTHERWISE MOVE A TWO
*                               TO INPSW AND PRINT FORMAT ERROR.
        C   BUFFER+5(1),ONE     WAS THE TRANSACTION CODE ENTERED LESS
*                               THAN ONE.  IF SO, CONTINUE.
        BC  TSET1(1)            OTHERWISE, MOVE A TWO TO INPSW AND
*                               PRINT FORMAT ERROR.
        MC  BUFFER+5(1),XS1     MOVE THE TRANSACTION CODE TO A TEMPORARY
*                               STORAGE AREA.
        C   BUFFER+7(1),BLK     CHECK THE BUFFER TO SEE IF A REMAINDER
*                               WAS ENTERED.
*                               NOTE:  FULL CHARACTER COUNTS WILL ALSO BE
*                               ACCEPTED.
        BC  TSET1(2)            IF NOTHING WAS ENTERED, MOVE A TWO TO
*                               INPSW AND PRINT FORMAT ERROR.
        MC  BUFFER+7(3),XS1+1   MOVE THE REMAINDER (OR CHARACTER COUNT)
*                               TO THE NEXT AVAILABLE LOCATION IN THE
```

```
*                                  TEMPORARY STORAGE AREA.
        C   BUFFER+11(1),TWO       IS THE ROUTING INDICATOR LESS THAN TWO?
        BC  *+10(1),TSET1(5)       IF SO, CONTINUE PROCESSING. OTHERWISE
*                                  MOVE A TWO TO INPSW AND PRINT FORMAT
*                                  ERROR.
        FN  XS1,XS2                CHECK TO SEE IF THE TRANSACTION CODE
*                                  AND THE REMAINDER ARE NUMERIC.
        C   XS1(4),XS2
        BC  *+10(2),TSET1(5)       IF SO, BRANCH TO LDTBL2 TO ADD
*                                  THE DATA TO THE TABLE.  OTHERWISE, MOVE
*                                  A TWO TO INPSW AND PRINT FORMAT ERROR.
        MC  BUFFER+11(1),XS1+4     MOVE THE ROUTING INDICATOR TO THE
*                                  STORAGE AREA.
        BC  CHKFA1+1(6),LDTBL2(5)
        A   EIGHT,X2+2(2)          INCREMENT INDEX TWO BY EIGHT.
        C   BUFFER+5(1,2),BLK      CHECK TO SEE IF THE BUFFER CONTAINS
*                                  ADDITIONAL ENTRIES.
        BC  TSET2(2)               IF NOT, MOVE A ONE TO INPSW TO INDICATE
*                                  THE OPERATION IS VALID AND HAS BEEN
*                                  COMPLETED.
        MC  BUFFER+5(7,2),BUFFER+5 OTHERWISE, OVERLAY THE FIRST ENTRY I
*                                  THE BUFFER WITH THE NEXT SET OF DATA.
        BC  CHKFA2(5)              BRANCH TO CHKFA2 AND CHECK THE FORMAT OF
*                                  THE NEXT ENTRY.
*
*  TABLE DELETION
*
LDTBL1  MC  ZEROS,X3               ZERO INDEX REGISTER THREE.
        MC  BUFFER+5(1,1),X3+2     MOVE THE TRANSACTION CODE TO INDEX
*                                  REGISTER THREE.
        M   FOUR(1),X3+2(1)        MULTIPLY THE TRANSACTION CODE BY FOUR
*                                  TO DETERMINE THE POSITIONS IN THE TABLE
*                                  TO BE DELETED.
        MC  BLK(4),TCTABL(,3)      MOVE BLANKS TO THE APPROPRIATE POSITIONS
CHKFT9  BC  CHKFT9(5)              RETURN TO THE CHKFT2 MAINLINE TO CHECK
*                                  FOR MORE ENTRIES IN THE BUFFER.
        ORG *-5
TST2    DM  C'00'
BLK     DM  C'  '
*
*  TABLE ADDITION
*
LDTBL2  MC  ZEROS,X3               ZERO INDEX REGISTER THREE.
        MC  BUFFER+5(1),X3+2       MOVE THE TRANSACTION CODE TO INDEX
*                                  REGISTER THREE.
        M   FOUR(1),X3+2(1)        MULTIPLY THE INDEX REGISTER BY FOUR TO
*                                  DETERMINE WHERE IN THE TABLE THE DATA
*                                  IS TO BE ADDED.
        C   TCTABL(1,3),BLK        CHECK THOSE LOCATIONS IN THE TABLE TO
*                                  MAKE SURE THEY ARE BLANK.
        BC  *+10(2),TSET3(5)       IF SO, MOVE THE CONTENTS OF THE STORAGE
*                                  AREA TO THE TABLE.  IF NOT, MOVE A THREE
*                                  TO INPSW AND PRINT WLM TABLE ERROR.
        MC  XS1+1(4),TCTABL(,3)
CHKFA1  BC  CHKFA1(5)              RETURN TO THE CHKFA1 MAINLINE AND
*                                  CONTINUE PROCESSING.
*
*  SET INPSW TO INDICATE FORMAT ERROR
*
TSET1   MC  TWO,INPSW              MOVE A TWO TO INPSW
```

```
          BC TOUT(5)
          ORG *-5
T         DM C'T'
A         DM C'A'
TST1      DM C'00'
*
*  SET INPSW TO INDICATE TRANSACTION ADDITION OR DELETION WAS
*     SUCCESSFULLY COMPLETED.
*
TSET2     MC ZERO,INPSW          MOVE A ZERO TO INPSW
          BC TOUT(5)
          ORG *-5
XS1       DM C'00000'
*
*  SET INPSW TO INDICATE THE OPERATOR ATTEMPTED TO ADD DATA TO
*     THE TABLE WITHOUT DELETING THE INFORMATION ALREADY THERE.
*
TSET3     MC THREE,INPSW         MOVE A THREE TO INPSW.
          BC TOUT(5)
          ORG *-5
XS2       DM C'00000'
TOUT      MC ZEROS(2),BUFFER+2   ZERO THE PARTITION NUMBER IN THE BUFFER.
          MC ZERO,XS1+4
CDEXIT    BC CDEXIT(5)           EXIT FROM THE ROUTINE.
*
```

Section 7

LOADING AND OPERATING INSTRUCTIONS

ORG ADDRESSES

CM, located at 0000 in Common, should be the only program data in Common below address 0320; the CDCHK routine can go anywhere in Common that does not conflict with other phases of the MIS program or the restricted register area.

The first DM statement ORG'd at 0000 in each CD-30 partition is an aid in loading the program and has no bearing on the program itself. The distinctive text card that results from this constant makes it easy to identify the CD-30 Partition object deck. The instruction to branch to TATTLE must be located at 0000 in partition also; if a program check (usually an illegal instruction in this case) sets the P register to 0000, the program will branch to TATTLE for recovery. TATTLE is discussed in the Basic Control Module Logic Manual 5004.

CD5P, and CDPNO must be at the addresses shown in the listings. Except as noted in Section 7, the location of the rest of the DM statements in partition is not critical. CD5P and CDPNO will differ for each CD-30 partition.

LOADING THE PROGRAM

A multiprogram loader should be used to load each CD-30 partition. Each deck of instructions to be placed in a particular partition should be preceded by a parameter card. Common and Partition 0 must be loaded last and coordinated with other phases of the program.

HOW TO UPDATE TERMINAL TRANSACTION TABLE TCTABL

Each partition table is initialized with appropriate information when the program is loaded. See Section 5, "Modifications for Terminal ID and/or Transaction Validity Check": also see figures 7-1 and 7-2 in this section. All modifications to TCTABL are entered via the 7102 or Model 70 keyboard.

1. Obtain a service request at communications terminal.

2. Enter appropriate format for update information, each entry followed by the unit separator.

3. Make correction if error message is printed at the communications terminal.

Format of Entries

Entries may be made in series as shown in the examples (Fig. 7-1 and Fig. 7-2) or individually. For instance, PT03-9999-8-7-6-5.../ can be entered PT03-9999-8/, PT03-9999-7/, PT03-9999-6/, etc. The actual values of the transaction codes can be in any numeric order within one format, but command options can not be mixed in one entry. Routing information is not used at this time; a blank, 0, or 1 must be entered in this position.

An addition to the table must be preceded by a deletion unless the required positions are blank already.

Errors

"WLM TABLE ERR"

1. Operator attempted to add (PA option) new transaction data to table before deleting that already residing in that portion.

2. The two-digit partition number was:

   a. not in range 02-19

   b. not numeric

   c. not in specified format.

"FORMAT ERROR"

1. Format contained non-numeric characters other than command option.

2. Transaction code not in range 1-8.

3. Routing indicator not blank, 0, or 1.

4. Data not in format required for that option.

```
*****************************************************************
*                                                               *
*     GENERAL FUNCTIONAL DESCRIPTION                            *
*                                                               *
*      EACH CD-30 PARTITION CONTAINS A TRANSACTION TABLE.       *
*     THIS TABLE CONTAINS AN APPROPRIATE MODULUS FOUR           *
*     REMAINDER FOR EACH TRANSACTION THAT IS VALID FOR THE      *
*     TERMINALS CONNECTED TO THE PARTITION.  THIS PARTICULAR    *
*     TABLE ACCOMODATES UP TO EIGHT TRANSACTIONS, SEVEN OF      *
*     WHICH ARE VALID.  AT LOAD TIME, EACH CD-30 PARTITION      *
*     TRANSACTION TABLE IS INITIALIZED WITH THE APPROPRIATE     *
*     REMAINDERS FOR EACH VALID TRANSACTION.                    *
*                                                               *
*     IN ADDITION,  THE MODULUS REMAINDERS WHICH ARE INITIALLY  *
*     STORED IN THE TABLE MAY BE READILY DELETED ANY TIME IT    *
*     IS NECESSARY TO ALTER THE VALID TRANSACTIONS SPECIFIC     *
*     TO EACH MASTER CD30 PARTITION.  NEW TRANSACTIONS OR       *
*     NEW REMAINDERS FOR EXISTING TRANSACTIONS CAN BE ADDED AT  *
*     ANY TIME.                                                 *
*                                                               *
*****************************************************************
     EJECT
*****************************************************************
*                                                               *
*     METHOD                                                    *
*                                                               *
*      TRANSACTION DELETION                                     *
*                                                               *
*        ALL OPERATIONS ARE ACCOMPLISHED VIA A 7102 TERMINAL    *
*        KEYBOARD.  TO DELETE A TRANSACTION REMAINDER AND ITS   *
*        ASSOCIATED ROUTING INDICATOR, OBTAIN A SERVICE REQUEST *
*        ON THE 7102 KEYBOARD AND ENTER DATA ACCODRING TO THE   *
*        FOLLOWING FORMAT:                                      *
*                                                               *
*                     PTXX-X                                    *
*                                                               *
*                        WHERE                                  *
*                                                               *
*                          XX IS A TWO DIGIT PARTITION NUMBER IN*
*                               THE RANGE 02 THROUGH 19.        *
*                                                               *
*                           X  IS A ONE DIGIT TRANSACTION CODE  *
*                               IN THE RANGE 1 THROUGH 8.       *
*                                                               *
*        THE OPERATOR MAY EITHER SPECIFY ONE "PT" FOR EACH      *
*        TRANSACTION THAT IS TO BE DELETED OR HE MAY SPECIFY ONE*
*        "PT" FOLLOWED BY FROM TWO TO EIGHT SEPARATE TRANSACTIONS.
*        THE COMPLETION OF AN ENTRY IS SIGNALLED BY DEPRESSING  *
*        "CONTROL /" (UNIT SEPERATOR) ON THE 7102.              *
*        THE PROGRAM WILL THEN DELETE THE APPROPRIATE TRANSACTION
*        REMAINDER AND ROUTING INDICATOR FROM THE TABLE.        *
*                                                               *
*        BOTH OF THE FOLLOWING DEMONSTRATE VALID TRANSACTION    *
*        DELETION FORMATS:                                      *
*                                                               *
*                          PT02-4/                              *
*                          PT02-1/                              *
*                          PT02-8/                              *
*                                                               *
*                             OR                                *
*                                                               *
*                          PT02-4-1-8/                          *
*                          PT02-4-1-5-2-3-6-8-7/                *
*                                                               *
*        A TRANSACTION DELETION (PT) MUST ALWAYS PRECEEDE A     *
*        TRANSACTION ADDITION (PA), UNLESS A TABLE ENTRY IS     *
*        INITIALIZED TO BLANKS OR AN ENTRY IS DELETED AND A NEW *
*        VALUE IS NOT IMMEDIATELY ADDED.                        *
*                                                               *
*      TRANSACTION ADDITION                                     *
*                                                               *
*        AFTER OBTAINING A SERVICE REQUEST ON THE /102, THE     *
*        OPERATOR MAY ADD TRANSACTION REMAINDERS AND ASSOCIATED *
*        ROUTING INDICATORS BY ENTERING THE FOLLOWING DATA:     *
*                                                               *
*                          PAXX-X-XXX-X                         *
*                                                               *
*                             WHERE                             *
*                                                               *
*                               XX IS A TWO DIGIT PARTITION NUMBER IN
*                                    THE RANGE 02 THROUGH 19.   *
*                                                               *
*                                X IS A ONE DIGIT TRANSACTION NUMBER
*                                    IN THE RANGE 1 THROUGH 8.  *
*                                                               *
*                               XXX IS A THREE DIGIT MODULUS REMAINDER
*                                    IN THE RANGE 000 THROUGH 003.
*                                                               *
*                                X IS A ONE DIGIT ROUTING INDICATOR
*                                    WHICH MAY BE A BLANK, ZERO, OR
*                                    ONE.                       *
*                                                               *
*        THE OPERATOR MAY ENTER A SEPARATE "PA" CODE FOR EACH   *
*        TRANSACTION HE WISHES TO ENTER, OR HE MAY ENTER ONE "PA"
*        AND SPECIFY FROM TWO TO EIGHT TRANSACTIONS AND THEIR   *
*        ASSOCIATED REMAINDERS AT ONE TIME.  THE COMPLETION OF  *
*        AN ENTRY IS SIGNALLED BY DEPRESSING "CONTROL /" (UNIT  *
*        SEPARATOR) ON THE 7102.  THE PROGRAM WILL THEN ENTER THE
*        DATA IN THE APPROPRIATE PLACE IN THE TRANSACTION TABLE.*
*        BOTH OF THE FOLLOWING DEMONSTRATE VALID ADDITION FORMATS:
*                                                               *
*                          PA02-4-000- /                        *
*                          PA02-2-001- /                        *
*                          PA02-8-003- /                        *
*                                                               *
*                             OR                                *
*                                                               *
*                   PA02-4-000- -2-001- -8-003- /               *
*                                                               *
*        AT THE PRESENT TIME THE ROUTING INDICATOR HAS NO       *
*        FUNCTION.  HOWEVER, A BLANK ZERO, OR ONE MUST BE       *
*        ENTERED IN THE SPECIFIED POSITION EACH TIME A TRANSACTION
*        REMAINDER IS ADDED.                                    *
*                                                               *
*        NOTE THAT THE ACTUAL VALUES OF THE TRANSACTION CODES   *
*        FOR BOTH "PA" AND "PT" MAY BE ENTERED IN ANY NUMBERIC  *
```

```
*     ************************************************************
*
*
*
*
*
*       MULTIPLE-ENTRY CDCHK, USING MESSAGE LENGTH
*
*     TERMINAL TRANSACTION TABLE UPDATE ROUTINE
*
*
*
*   1.REPLACE TERMINAL ID NO. -  MAX = 8 ENTRIES
*
*                     PR03-9999-0701-8888-0702/
*     COMMAND OPTION  *
*     PARTITION NO.    **
*     EXISTING TERMINAL ID****       ****
*     NEW TERMINAL ID          ****       ****
*
*   2.DELETE ENTIRE TERMINAL TABLE ENTRY
*
*                     PD03-9999-8888-7777-6666/
*     COMMAND OPTION  *
*     PARTITION NO.    **
*     TERMINAL TO DELETE  **** **** **** ****
*
*   3.DELETE TRANSACTION TABLE ENTRY
*
*                     PT03-9999-8-7-6-5-4-3-2-1/
*     COMMAND OPTION  *
*     PARTITION NO.    **
*     TERMINAL ID         ****
*     TRANSACTIONS TO DELETE  * * * * * * * *
*
*
*   4.ADD TRANSACTION CODES TO TABLE
*
*                     PA03-9999-8-012-1-7-076-1/
*     COMMAND OPTION  *
*     PARTITION NO.    **
*     TERMINAL ID         ****
*     TRANSACTION CODE         *       *
*     NEW LENGTH                  ***     ***
*     ROUTING INDICATOR                *       *
*
*     INPSW                  MESSAGE STATUS SWITCH
*                            =0 MEANS NORMAL
*                            =1 (RESERVED)
*                            =2 MEANS FORMAT ERROR
*                            =3 MEANS TERMINAL ID ERROR/INVALID ENTRY
*                            =4 UPDATE TO BE PERFORMED
*     ************************************************************
```

Fig. 7-2 Multiple-entry Transaction Table Formats

Section 8

LABELS

The following labels are defined by DM statements in the program. An asterisk indicates that the user must supply his own definition for the term. The items do not have to be in this order in the program unless so indicated.

NUMERIC CONSTANTS

Label	DM	EXPLANATION
BLNKS	C36' '	
ZERO	N'0'	
ZEROS	N'0000'	
ONE	N'1'	
TWO	N'2'	
THREE	N'3'	
FOUR	N'4'	
FIVE	N'5'	
EIGHT	N'8'	
NINE	N'9'	
TEN	C'10'	
CDMIN	N'0002'	Minimum message length.
CDMXLN	N'0199'	Maximum input length minus 1.
THTY2	N'0032'	Used to set index register 1 to fill second error buffer.

Label	DM	EXPLANATION
SIXTY4	N'0064'	Used to set index register 1 to fill third error buffer.
TCCNS1	N'0036'	Used to increment index register 1 in Table Update routine.
SIZE1 SIZE2	N'006' N'022'	Used to call Code Conversion Module.
*CDPNO	C2	CD-30 Partition number.
*CD5P	C4	5 times partition number--us5d in computing message length.
CDBR	N'0101'	B-register base constant--used in computing message length.
*TCTABL	0C36/0C360	Terminal transaction table. Values depend on message length used in system (See Section 5). This is followed in core by the table used.
	C36'. . .	actual table is entered here..'
	C36'. . .	each entry is 36 positions long'.
TCEOT	C':::'	End of TCTABL. TCTABL and close up TCEOT must follow this sequence in the DM statements.

ALPHABETIC CONSTANTS

A	0C4	Command Option List--used in Table Update.
	C'D'	Delete entire terminal entry.
	C'A'	Add transaction codes (CNTR=1).
	C'T'	Delete transaction codes (CNTR=2).
	C'R'	Replace terminal ID (CNTR=3).
ASTER	C'*'	End of possible options. A through ASTER must be in this order in the DM statements.

Label	DM	EXPLANATION
ADC10	A'TDEL'	Address constant for TDEL, the routine used to delete transaction table codes. routine used to replace terminal ID in the transaction table.
ADCH9	A'CHKFD2'	Address constant for CHKFD2, a routine used in Table Update to delete terminal entry.
ADCHNG	A'CHKFR2'	Address constant for CHKFR2, a Table Update routine to replace terminal ID.
ADCNG1	A'ZZ'	Address constant for ZZ, a routine in Table Update.
CM	OC4	The first four positions of Common. Modified by X2,CM is used to determine the contents of the B-register for computing CD-30 message length in CDRD.
IADDR1 IADDR2 IADDR3	A'CDIN' A'CDHEAD' A'CD6'	Used as address constants in Code Conversion routine. See Code Convert Logic Manual 5008.
CDNC	C'NC'	Header prefix: N (Normal), C (CD-30).
CDPAR1	C'1PAR1'	Error No. 1--parity error on Read.
CDPAR2	C'2PAR2'	Error No. 2--fault on Read.
CDPAR3	C'3PAR3'	Error No. 3--unable to transmit ACK or NAK to CD-30 terminal.
TCITC	C'4TRAN'	Transaction code invalid.
TCIWLM	C'5WLM'	Indicates wrong length message.

The preceding five constants are moved into CDERID (error description field), the first position of which is also CDERSW (local error switch).

Label	DM	EXPLANATION
INPSW	N'0'	Input message status switch set in Table Update routine. Updaze routine. 0=Normal, 1= (Reserved), 2=Format Error, 3=Terminal ID Error/Invalid Entry, 4=Update to be Performed.
CDERSW	0C1	Local error switch, the first character of CDERID, which follows it in core.
CDERID	C'0bbbb'	Error description field into which error codes are moved.
CDGOOD	N'0'	Second local error switch.
CNTR	C'0'	Used to indicate command option in Table Update; also used to indicate number of consecutive entries in Table Update message.
ECOUNT	N'0'	Error buffer counter.
ERRSW	N'0'	Error buffer switch.
INHSW	N'2'	System inhibit switch; will inhibit system if not set to zero.
PERCNT	C'00000'	Error record counter.
PRCDNO	C'00000'	Record serial number.

STORAGE AREAS

*BUFFER	C75' '	Input area for update message from 7102. This length depends on the system used.
TS3 TS4	C'00000' C'00000'	Used primarily for testing for numeric entries in the Table Update routine.
TSWKA	C36' '	Holds length and routing indicators in Table Update routine before these are moved to TCTABL.
CDHEAD	C16	MIS header prefix. Filled with data stored in Common starting with CDNC.

LABELS

Label	DM	EXPLANATION
CDIN CD6	0C200 C6 C194	CDIN through CD6--output/input area for CD-30 transmissions for single-frame output code, input area for double-frame output code.
ERRBUF	0C100	Used to hold error messages to be printed on 7102; contains three buffers. See listing for detailed definition of contents.
WTBUF	A'CDHEAD'	Address constant for entire message to be written to tape in single-frame code; used by tape Write Module. See Appendix B for address constants to use for double-frame code and for output to disc.

VARIABLES

SCHRS	N2	Part of clock time. Used in header prefix for transaction recording on tape and error messages to the 7102. See Clock Module Logic Manual 5006.
CDMSGL	C4	CD-30 message length.
CDTCNT	C'0000'	CD-30 message length expected. Length is moved from TCTABL to CDTCNT in TCSTRT subroutine.
TRNCDE	C'0' C1	Transaction code.
TCTID	C'0000'	Terminal ID. TRNCDE through TCTID are the first six characters of the CD-30 transmission.
X1 X2 X3	C'0000' C'0000' C'0000'	Index registers used to modify instruction address.
CCSIZE	N3	Length of input field--used in code conversion and set by SIZE1 and SIZE2. See Code Convert Logic Manual 5008.

Section 9

ROUTINE LOGIC--COMMON

TABLE UPDATE ROUTINE

This routine updates the data in TCTABL, which is contained in each CD-30 partition. The particular TCTABL used is chosen according to the criteria discussed in Section 5, "Modifications for Terminal ID and/or Transaction Validity Check". The explanations here are based on a multiple-entry table using message lengths rather than modulus remainders. Listings for single-entry table using modulus remainders start on page 6-25. The single-entry table does not require that the program test terminal ID numbers.

The message length initially stored in TCTABL may be deleted when it is necessary to alter the valid transactions specific to a CD-30 partition. New transactions or new message lengths for existing transactions can be added at any time.

All update operations are entered from the 7102 or Model 70 communications terminals after obtaining a service request; the maximum number of transactions entered at one time is eight. The format shown in the program listing may be used or each transaction may be entered separately, each followed by the unit separator /. The routing indicator in the table entries has no significance at this time, but a blank, 0, or 1 must be entered in the specified position. The actual values of the transaction codes can be entered in any order.

Before the table is updated, the 7102 message is checked for possible errors and the result is placed in the message status switch, INPSW, where

0 = Normal	3 = Terminal ID Error or Invalid Entry
1 = Reserved for future Use	
2 = Format Error	4 = Update to be Performed

The contents of this switch are used by the Control Module to write appropriate update message status at the 7102.

```
*
*       TEST FOR POSSIBLE COMMAND OPTIONS
*
CDCHK   MC  ZEROS,X2            CLEAR INDEX 2
        MC  ZEROS,X1            CLEAR INDEX 1
        MC  ZEROS(1),CNTR       SET COUNTER TO ZERO
A1      C   A(1,1),BUFFER+1     IS THIS A VALID FUNCTION
        BC  CHKT(2)             IF YES, GOTO CHKT
        A   ONE,X1+3(1)         ADD '1' TO INDEX 1
        MN  ADDRG1,Z+1          MODIFY INSTRUCTION Z TO BRANCH TO
        C   A(1,1),ASTER        IS THIS THE LAST OPTION
        BC  TSET1(2)            IF YES, GOTO ERROR ROUTINE
        A   ONE,CNTR            IF NO, ADD 1 TO COUNTER
        BC  A1(5)               AND LOOP BACK TO A1
*
```

CDCHK

Test Possible Command Options

BUFFER contains PA03-9999-8-012-1/

Explanation:

To determine transaction validity and subroutine routing, the second character in BUFFER is compared to function codes D,A,T,R,* stored in A. Each time the program loops through this routine, 1 is added to register 1 and to CNTR to use in ZZ (CHKT routine) for routing codes A,T, a d R. The program will branch to CHKT when the code is identified. If the program loops four times and has not encountered a valid code, it goes to error routine TSET1 and moves a 2 into message status switch INPSW to indicate a format error.

External References:

A, ADCNG1, ASTER, BUFFER, CNTR, ONE, ZEROS--Common

X1, X2--Partition

Referencing Routines:

CDSTR

System Constants Affected:

None

Exits:

a. Normal: CHKT, TSET1

b. Abnormal: None

```
*
*       TEST TERMINAL ID TO BE NUMERIC
*
CHKT    C   BUFFER+5(1,2),BLNKS     IS NEXT TERMINAL ID BLANK?
        BC  TSET1(2)                IF YES, GOTO ERROR ROUTINE
CHKT1   MC  BUFFER+5(4,2),TS3       IF NO, MOVE TERMINAL ID TO TS3
        FN  TS3(5),TS4(5)           FORM NUMERIC
        C   TS3(4),TS4              IS ENTRY NUMERIC?
Z       BC  CHKFD(2),TSET1(5)       IF NO, GOTO ERROR ROUTINE
ZZ      C   ONE,CNTR                IF COUNTER IS ONE
        BC  CHKFA(2)                GOTO ADD TRANSACTION LENGTH IN TAB
        C   TWO,CNTR                IF COUNTER IS TWO
        BC  CHKFT(2)                GO TO DELETE TRANSACTION LENGTH IN
        C   THREE,CNTR              IF COUNTER IS THREE
        BC  CHKFR(2)                GOTO CHANGE TERMINAL ID IN TABLE
*
```

CHKT

Test for Numeric Terminal ID and Route to Proper Subroutine

BUFFER contains PA03-9999.../

Explanation:

The first time through CHKT, index register 2 is zero so the fifth character in BUFFER is the one checked. If it is not blank, the four characters indicated by BUFFER+5(4,2) are moved to TS3, the numeric bits in TS3 are moved to TS4, and the two are compared. An unequal compare means that the four characters are not all numeric and at Z the program will branch to error routine TSET1 (INPSW = 2, Format Error).

If the entry is numeric and a PD (delete entire terminal entry) transaction is being checked, the program will branch to CHKFD because Z has not been modified by the address constant ADCNG1. A, T, or R transactions, having looped through CDCHK at least once (thereby causing Z to be modified), will branch to ZZ instead of to CHKFD. The content of CNTR is used to route these transactions to CHKFA, CHKFT, or CHKFR.

External References:

BUFFER, BLNKS, TS3, TS4, CNTR, ONE, TWO, THREE,

ZEROS--Common

Referencing Routines:

CDCHK, CHKFD2

System Constants Affected:

None

Exits:

a. Normal: TSET1, CHKFA, CHKFT, CHKFR

b. Abnormal: None

```
*       DELETE ENTIRE TERMINAL TABLE ENTRY  -  OPTION D
*
CHKFD   MC ZEROS(1),CNTR            ELSE SET COUNTER TO ZERO
        MC ZEROS,X2                 SET INDEX 2 TO 0
CHKFD2  BC CHKFD1+1(6),STORAR(5)    BRANCH AND LINK TO STORAR
        A  FIVE,X2+2(2)             ADD 5 TO INDEX 2
        C  BUFFER+5(1,2),BLNKS      IS TRANSACTION CODE BLANK?
        BC LDTBL1(2)                IF YES, GOTO FIND TERMINAL ID
        MN ADCH9,Z+1                IF NOT, MODIFY INSTRUCTION Z TO CH
        BC CHKT(5)                  AND LOOP BACK TO CHKT
*
        ORG *-5
ADCH9   DM A'CHKFD2'
*
*
*       DELETE ENTIRE TERMINAL ENTRY IN TABLE
*
DEL     MC BLNKS(36),TCTABL(,1)     BLANK OUT ENTIRE TERMINAL TABLE EN
        S  ONE,CNTR                 SUBTRACT 1 FROM COUNTER
        BC *+10(3),TSET2(2)         IF ZERO, GOTO NORMAL EXIT ROUTINE
        A  FIVE,X2                  IF POSITIVE, ADD 5 TO INDEX 2
        BC LDTBL2(5)                BRANCH AND SEARCH FOR TERMINAL ID
*
```

CHKFD/DEL

Delete Entire Terminal Table Entry--Option D

BUFFER contains PD03-9999-8888-7777 /

(Terminal Numbers Only)

Explanation:

The first instruction is a branch and link to STORAR to count the number of entries in the message; 1 is added to CNTR. If CNTR = 8, the program branches to LDTBL1 to find the terminal ID's in TCTABL; CNTR < 8 will cause the program to link back. Register 2 (X2) is incremented by 5 so that BUFFER+5(1,2) now refers to the eleventh character in the 7102 message stored in BUFFER. If this position is not blank, the program loops from CHKT through CHKFD2 again until it does encounter a blank in the X2-modified address or until CNTR in STORAR=8. At this point it branches to LDTBL1 to find the terminal ID's in TCTABL and then to DEL for deletion of the entry or entries.

External References:

ZEROS, CNTR, STORAR, FIVE, BUFFER, BLNKS, ADCH9--Common

X2--Partition

Referencing Routines:

Z in CHKT

System Constants Affected:

None

Exits:

a. Normal: LDTBL1, CHKT

b. Abnormal: None

```
*       DELETE ENTIRE TERMINAL ENTRY IN TABLE
*
DEL     MC  BLNKS(36),TCTABL(,1)     BLANK OUT ENTIRE TERMINAL TABLE EN
        S   ONE,CNTR                 SUBTRACT 1 FROM COUNTER
        BC  *+10(3),TSET2(2)         IF ZERO, GOTO NORMAL EXIT ROUTINE
        A   FIVE,X2                  IF POSITIVE, ADD 5 TO INDEX 2
        BC  LDTBL2(5)                BRANCH AND SEARCH FOR TERMINAL ID
*
*
```

DEL

Delete Entry in Table--Option D

Explanation:

Blanks are moved into the 36 spaces occupied by TCTABL(,1), identified in LDTBL2 as the terminal table entry to be deleted. CNTR is decremented by 1, X2 is incremented by 5, and the program loops again from LDTBL2 until all terminal entries indicated in the 7102 message have been deleted. (With a single-entry table, this routine would be performed only once). When CNTR = 0, the program exits to TSET2.

External References:

BLNKS, ONE, CNTR, FIVE--Common

TCTABL, X2--Partition

Referencing Routines:

LDTBL2

System Constants Affected:

None

Exits:

a. Normal: TSET2, LDTBL2

b. Abnormal: None

```
*
*       REPLACE TERMINAL ID  -  OPTION R
*       TEST FOR NUMERIC ENTRY
*
CHKFR   MC  ZEROS,X2                    SET INDEX 2 TO 0
        MC  ZEROS(1),CNTR               SET COUNTER TO 0
CHKFR2  C   BUFFER+10(1,2),BLNKS        IS NEXT ENTRY BLANK?
        BC  TSET1(2)                    IF YES, GOTO ERROR ROUTINE
        MC  BUFFER+10(4,2),TS3          IF NO, MOVE ENTRY TO TS3
        FN  TS3(5),TS4(5)               FORM NUMERIC
        C   TS3(4),TS4                  IS ENTRY NUMERIC?
        BC  *+10(2),TSET1(5)            IF NO, GOTO ERROR ROUTINE
        MN  ADC11,R+1                   MODIFY INSTRUCTION R TO BRANCH TO
        BC  CHKFD1+1(6),STORAR(5)       BRANCH AND LINK TO STORAR
        A   TEN,X2+2(2)                 ADD 10 TO INDEX 2
        C   BUFFER+5(1,2),BLNKS         IS NEXT ENTRY BLANK?
        BC  LDTBL1(2)                   IF YES, GOTO FIND TERMINAL ID
        MN  ADCHNG,Z+1                  IF NO, MODIFY INSTR. Z TO CHKFR2
        BC  CHKT1(5)                    BRANCH TO TEST FOR NUMERIC ENTRY
**
        ORG *-5
ADCHNG  DM  A'CHKFR2'
*
*
*       REPLACE TERMINAL ID
*
RDEL    MC  BUFFER+10(4,2),TCTABL(,1)   MOVE ENTRY TO TABLE
        S   ONE,CNTR                    SUBTRACT 1 FROM COUNTER
        BC  *+10(3),TSET2(2)            IF ZERO, GOTO NORMAL EXIT ROUTINE
        A   TEN,X2                      IF POSITIVE, ADD 10 TO INDEX 2
        BC  LDTBL2(5)                   BRANCH TO FIND ANOTHER TERMINAL ID
*
        ORG   *-5
ADC11   DM  A'RDEL'
*
```

CHKFR/RDEL

Replace Terminal ID--Option R

Test for Numeric Entry

BUFFER contains PR03-9999-0701-8888-0702/

Explanation:

The program enters from CHKT(ZZ) when CNTR = 3. X2 and CNTR are set to zero, then the eleventh position of BUFFER+10(1,2) is checked for a blank. If not blank, the new terminal ID nummber (four characters) is moved to TS3 to be checked for numeric as explained in CHKT. If the entry is all numeric, R in the LDTBL1 routine is modified to branch to RDEL (replace terminal ID) and there is a branch and link to STORAR to count the number of entries in the message.

X2 is incremented by 10 and the twenty-first character is checked for blanks to see if the message continues or not. A blank sends the program to LDTBL1 to find the terminal ID in TCTABL; a character causes Z in CHKT to be modified to return to CHKFR2 and a branch to CHKT1 to see if this next four-character block is numeric.

External References:

ZEROS, CNTR, BUFFER, TS3, TS4, ADC11, STORAR, TEN,

BLNKS, ADCHNG--Common

X2--Partition

Referencing Routines:

CHKT (ZZ) when CNTR = 3

System Constants Affected:

None

Exits:

a. Normal: TSET1, LDTBL1, CHKT1

b. Abnormal: None

```
*       REPLACE TERMINAL ID
*
RDEL    MC BUFFER+10(4,2),TCTABL(,1) MOVE ENTRY TO TABLE
        S  ONE,CNTR                  SUBTRACT 1 FROM COUNTER
        BC *+10(3),TSET2(2)          IF ZERO, GOTO NORMAL EXIT ROUTINE
        A  TEN,X2                    IF POSITIVE, ADD 10 TO INDEX 2
        BC LDTBL2(5)                 BRANCH TO FIND ANOTHER TERMINAL ID
*
        ORG   *-5
ADC11   DM A'RDEL'
```

RDEL

Replace Terminal ID

Explanation:

Four characters are moved from BUFFER+10(4,2) (the new terminal ID NUMBER) to TCTABL(,1) (X1 identified in LDTBL1) and the program either exits to TSET2 or loops again through LDTBL2. Since index register 2 (X2) has been incremented by 10, the next terminal in TCTABL will be checked by LDTBL2, which will increment register 1 (X1) to indicate the position of the next terminal number. This last step applies to multiple-entry tables only.

External Refernces:

BUFFER, ONE, CNTR, TEN--Common

TCTABL--Partition

Referencing Routines:

LDTBL2 (R is modified by AADC11)

System Constant Affected:

None

Exits:

a. Normal: TSET2, LDTBL2

b. Abnormal: None

```
*       DELETE TRANSACTION TABLE ENTRY  -  OPTION T
*
*       TEST TRANSACTION CODE (1-8)
*
CHKFT   MC  ZEROS,X3                    SET INDEX 3 TO 0
        MC  ZEROS(1),CNTR               SET COUNTER TO 0
CHKFT1  C   BUFFER+10(1,3),NINE         IS NEXT TRANSACTION CODE = 9?
        BC  *+10(1),TSET1(5)            IF YES, GOTO ERROR ROUTINE
        C   BUFFER+10(1,3),ONE          IF NO, IS IT LESS THAN ONE?
        BC  TSET1(1)                    IF YES, GOTO ERROR ROUTINE
        MN  ADC10,R+1                   MODIFY INSTR. R TO IDEL
        BC  CHKFD1+1(6),STORAR(5)       BRANCH AND LINK TO STORAR
        A   TWO,X3+2(2)                 ADD TWO TO INDEX 3
        C   BUFFER+10(1,3),BLNKS        IS THERE ANOTHER TRANSACTION CODE?
        BC  LDTBL1(2),CHKFT1(5)         IF NO, GO FIND TERMINAL ID
*                               IF YES, LOOP BACK AND TEST NEXT TRAN CDE
*
*
*       DELETE TRANSACTION IN TABLE
*
TDEL    MC  ZEROS,X3                    SET INDEX 3 TO 0
        MC  ZEROS,X2                    SET INDEX 2 TO 0
TDEL1   MC  FOUR,X2+2                   SET INDEX 2 TO '0040'
        M   BUFFER+10(1,3),X2+2(1)      MULTIPY TRANSACTION CODE BY 4
        A   X2+2(2),X1(4)               ADD INDEX 2 TO INDEX 1
        MC  BLNKS(4),TCTABL(,1)         CLEAR APPROPRIATE LENGTH ENTRY
        S   ONE,CNTR                    SUBTRACT 1 FROM COUNTER
        BC  *+10(3),TSET2(2)            IF ZERO, GO TO NORMAL OUT ROUTINE
        A   TWO,X3                      IF POSITIVE, ADD 2 TO INDEX 3
        S   X2+2(2),X1(4)               RESTORE CONTENTS OF INDEX 1
        BC  TDEL+10(5)                  LOOP BACK TO DELETE NEXT TRAN CDE
*
        ORG *-5
ADC10   DM  A'TDEL'
*
```

CHKFT/TDEL

Delete Transaction Entry--Option T

Test Transaction Code

BUFFER contains PT03-9999-8-7-6-5/
^

Explanation:

This routine uses the one-digit transaction code to determine which transaction entry to delete. The program branches to this routine from CHKT (ZZ) when CNTR = 2; CNTR and index register 3 (X3) are cleared as the first step.

If BUFFER=10(1,3) (the eleventh position the first time through) is < 1 or = > 9, the program branches to error routine TSET1 because only codes 1 - 8 are valid. Otherwise, instruction R in LDTBL2 is modified to branch to TDEL, and a branch and link to STORAR to increment CNTR is executed. X3 is incremented by 2 to check for the next transaction code, and the program either loops to CHKFT to test the next code or branches to LDTBL1 to find the terminal ID in TCTABL.

External References:

ZEROS, CNTR, BUFFER, NINE, ONE, ADC10, STORAR, TWO, BLNKS--Common

X3--Partition

Referencing Routines:

CHKT (ZZ), LDTBL2 (R is modified to ADC10)

System Constants Affected:

None

Exits:

a. Normal: TSET1, LDTBL1

b. Abnormal: None

```
*
*       DELETE TRANSACTION IN TABLE
*
TDEL    MC  ZEROS,X3                    SET INDEX 3 TO 0
        MC  ZEROS,X2                    SET INDEX 2 TO 0
TDEL1   MC  FOUR,X2+2                   SET INDEX 2 TO '0040'
        M   BUFFER+10(1,3),X2+2(1)      MULTIPY TRANSACTION CODE BY 4
        A   X2+2(2),X1(4)               ADD INDEX 2 TO INDEX 1
        MC  BLNKS(4),TCTABL(,1)         CLEAR APPROPRIATE LENGTH ENTRY
        S   ONE,CNTR                    SUBTRACT 1 FROM COUNTER
        BC  *+10(3),TSET2(2)            IF ZERO, GO TO NORMAL OUT ROUTINE
        A   TWO,X3                      IF POSITIVE, ADD 2 TO INDEX 3
        S   X2+2(2),X1(4)               RESTORE CONTENTS OF INDEX 1
        BC  TDEL+10(5)                  LOOP BACK TO DELETE NEXT TRAN CDE
*
        ORG *-5
ADC10   DM  A'TDEL'
*
```

TDEL

Delete Specific Transaction Information in Table

BUFFER contains PT 03-9999-8-7-6/

TCTABL contains '9999076 076 076 076 076 076 076 042 '

Explanation:

Index register X2, and X3 are cleared at the start of this routine; X3 is used to find the next transaction in the communications terminal message. X1 (computed by LDTBL1) and X2 are used to find the proper code length entry in TCTABL (the entries are in ascending sequence by code number).

BUFFER+10(1,3), the eleventh position the first time through, is multiplied by 4; the product is stored in X2 (in our example, 8 x 4 = 32, so X2 becomes 0032). If terminal 9999 is the first entry in TCTABL, X1 will be 0000. When X2 is added to X1, X1 will be 0032. The next instruction, the first time through, moves four blanks into TCTABL(,1) (X1 is 32), the thirty-third position that is the starting address of the four digits of information associated with transaction code 8 and terminal 9999.

External References:

ZEROS, FOUR, BUFFER, BLNKS, ONE, CNTR, TWO--Common

X1, X2, X3, TCTABL--Partition

Referencing Routines:

R in LDTBL2 modified by ADC10

System Constants Affected:

None

Exits:

a. Normal: TSET2

b. Abnormal: None

```
*       ADD TRANS. LENGTH TO TABLE  -  OPTION A
*
*       TEST TRANSACTION CODE (1-8)
*
CHKFA   MC ZEROS(1),CNTR              SET COUNTER TO ZERO
        MC ZEROS,X2                   SET INDEX 2 TO ZERO
        MC ZEROS,X3                   SET INDEX 3 TO ZERO
        C  BUFFER+10(1,3),BLNKS       IS TRANSACTION CODE ENTRY BLANK?
        BC TSET1(2)                   IF YES, GOTO ERROR ROUTINE
CHKFA1  C  BUFFER+10(1,3),NINE        IF NO, IS TRANSACTION CODE=9?
        BC *+10(1),TSET1(5)           IF YES, GOTO ERROR ROUTINE
        C  BUFFER+10(1,3),ONE         IF NO, IS TRAN CDE LESS THAN 0?
        BC TSET1(1)                   IF YES, GOTO ERROR ROUTINE
        MC BUFFER+10(1,3),TS3         IF NO, MOVE TRAN CDE TO TS3
        C  BUFFER+12(1,3),BLNKS       IS ASSOCIATED LENGTH BLANK?
        BC TSET1(2)                   IF YES, GOTO ERROR ROUTINE
        MC BUFFER+12(3,3),TS3+1       IF NO, MOVE LENGTH TO TS3
        C  BUFFER+16(1,3),TWO         IS ROUTING INDICATOR <2?
        BC *+10(1),TSET1(5)           IF NO, GOTO ERROR ROUTINE
        PN TS3(5),TS4(5)              IF YES, FORM NUMERIC
        C  TS3(4),TS4                 IS ENTRY NUMERIC?
        BC *+10(2),TSET1(5)           IF NO, GOTO ERROR ROUTINE
        MC BUFFER+16(1,3),TS3+4       IF YES, MOVE ROUTING INDICATOR TO
        MC FOUR,X2+2                  SET INDEX 2 TO '0040'
        M  BUFFER+10(1,3),X2+2(1)     MULTIPLY TRAN CDE BY 4
        MC TS3+1(4),TSWKA(,2)         MOVE TS3 TO TSWKA
        BC CHKFD1+1(6),STORAR(5)      BRANCH AND LINK TO STORAR
        A  EIGHT,X3+2(2)              ADD 8 TO INDEX 3
        C  BUFFER+10(1,3),BLNKS       IS NEXT TRAN CDE BLANK?
        BC TAZK(2),CHKFA1(5)          IF YES, GOTO FIND TERMINAL ID
*                                     IF NO, LOOP BACK TO TEST NEXT TRAN CDE
*
```

CHKFA

Add transactions to Table--Option A

Test Transaction Codes 1 - 8

BUFFER contains PA03-9999-8-012-1-7-076-1/

Explanation:

This routine can be used for inserting a new terminal ID into a table as well as for changing length and routing information.

X2, X3, and CNTR are cleared and BUFFER+10(1,3) (the eleventh position the first time through) is checked for a valid transaction code, which is moved to TS3 if found. Then the next three digits (BUFFER+12(3,3)) are moved to TS3+1 if they are not blank. If the routing indicator (BUFFER+16(1,3)) is less than 2, the code and length entries are tested for numeric by comparing TS3 and TS4 as explained in CHKT. If they are numeric, the routing indicator is moved to TS3.

Next, the transaction code in BUFFER+10(1,3) is multiplied by 4 and the result is put in index register 2; in this example, X2 will become 0032. Length and routing indicators are moved to TSWKA(,2) as modified by register 2 for storage until TCTABL is updated. A branch and link to STORAR increments CNTR. The program either branches to find the terminal ID in TCTABL or adds 8 to X3 and looks at the next transaction code.

External References:

ZEROS, CNTR, BUFFER, BLNKS, NINE, ONE, TS3, TS4, FOUR,

TSWKA, STORAR, EIGHT, TAZK, CHKAFA1--Common

X2, X3--Partition

Referencing Routine:

CHKT (ZZ) when CNTR = 1

System Constants Affected:

None

Exits:

a. Normal: TSET1, TAZK

b. Abnormal: None

```
*
*       FIND TERINAL ID IN TABLE
*
TAZK    MC  ZEROS,X1                SET INDEX 1 TO 0
        MC  ZEROS,X2                SET INDEX 2 TO 0
        MC  ZEROS,X3                SET INDEX 3 TO 0
TAZK1   C   BUFFER+5(4),TCTABL(,1)  ARE TERMINAL ID'S EQUAL?
        BC  TAD3(2)                 IF YES, GOTO TAD3
        C   TCTABL(4,1),TCEOT       IF NO, IS THIS THE END OF TABLE?
        BC  TAD(2)                  IF YES, GOTO TAD
        A   TCCNS1,X1               IF NO, ADD 36 TO INDEX 1
        BC  TAZK1(5)                LOOP BACK AND CONTINUE SEARCH
*
*
*
*
TAD     MC  ZEROS,X1                SET INDEX 1 TO 0
TAD2    C   TCTABL(4,1),BLNKS       IS TERMINAL ID BLANK?
        BC  *+10(2),*+30(5)         IF NO, GO TEST FOR END OF TABLE
        MC  BUFFER+5(4),TCTABL(,1)  IF YES, MOVE NEW TERMINAL ID TO TA
        BC  TAD3(5)                 BRANCH TO ADD ASSOCIATED TRAN CDE
        C   TCTABL(4,1),TCEOT       IS THIS THE END OF THE TABLE?
        BC  TSET3(2)                IF YES, GO SIGNAL TERMINAL ID ERRO
        A   TCCNS1,X1               IF NO, ADD 36 TO INDEX 1
        BC  TAD2(5)                 LOOP BACK AND CONTINUE SEARCH
**
```

TAZK/TAD

Find Terinal ID in Table

BUFFER contains PA03-9999-8-012-1/

Explanation:

BUFFER+5(4), positions 6 through 9, is compared to TCTABL(,1); if unequeal, the length of each terminal entry in TCTABL (36) is added to register 1 and the search continues. There are several possible branches from this routine. If ID's match, a branch to TAD3 inserts new length and routing indicator after the terminal ID. If the end of the table is reached without finding a matching ID, a branch to TAD sets X1 back to zero and the search starts over to look for a blank ID rather than a matching one. If there is a blank ID in TCTABL, the new ID is moved there and processing continues in TAD3; if there is no blank, the program branches to error routine TSET3.

External References:

ZEROS, BUFFER, BLNKS--Common

X1, X3, TCTABL, TCCNS1, TCEOT--Partition

Referencing Routines:

CHKFA1

System Constants Affected:

None

Exits:

a. Normal: TAD3, TAD2, TAZK1, TSET3

b. Abnormal: None

```
*       ADD NEW TRANSACTION CODE LENGTH AND ROUTING INDICATOR
*
TAD3    MC  FOUR,X2+2                   SET INDEX 2 TO '0040'
        M   BUFFER+10(1,3),X2+2(1)      MULTIPLY TRAN CDE BY 4
        A   X2+2(2),X1(4)               ADD INDEX 2 TO INDEX 1
        MC  TSWKA(4,2),TCTABL(,1)       MOVE LENGTH, ROUTING INDICATOR TO
        S   ONE,CNTR                    SUBTRACT 1 FROM COUNTER
        BC  *+10(3),TSET2(2)            IF ZERO, GOTO NORMAL EXIT ROUTINE
        A   EIGHT,X3                    IF POSITIVE, ADD 8 TO INDEX 3
        S   X2+2(2),X1(4)               RESTORE INDEX 1
        BC  TAD3(5)                     LOOP BACK AND CONTINUE
**
```

TAD3

Add New Transaction Length and Routing Indicator to TCTABL

BUFFER contains PA03-9999-1-012-1...

TCTABL contains 9999076 076 076 076 076 076 076 042

Explanation:

The transaction code is multiplied by 4 and the result is put in register 2--the value is 4 in this example. The value of X2 is now added to X1, the pointer for TCTABL. Length (012) and routing indicator (1) were moved into TSWKA as modified by X2 in routine CHKFA1 and are now moved to TCTABL(,1).

In our example 076b will be replaced by 0121. If CNTR is not zero, X3 is incremented by 8, X1 is restored to its value before TAD3, and the program loops back to process the next part of the 7102 message; BUFFER+10(1,3) now points to the nineteenth position and so on. When CNTR = 0, the program exits to TSET2.

External References:

FOUR, BUFFER, TSWKA, ONE, CNTR, EIGHT--Common

X1, X2, X3, TCTABL--Partition

Referencing Routines:

TAZK1, TAD2

System Constants Affected:

None

Exits:

a. Normal: TSET2

b. Abnormal: None

```
**
*       KEEP COUNT OF CONSECUTIVE ENTRIES IN MESSAGE
*
STORAR A   ONE,CNTR                    ADD 1 TO COUNTER
       C   CNTR,EIGHT                  IS COUNTER =8?
CHKFD1 BC CHKFD1(1),LDTBL1(5)          IF NO,RETURN
*                              IF YES, GOTO FIND TERMINAL ID
**
```

STORAR

Count Consecutive Entries in Message

Explanation:

This routine is entered on branch-and-link instructions from CHKFD, CHKFR, CHKFT, and CHKFA. At the beginning of each of these routines, CNTR is set to 0. Each time the program proceeds normally through one of these routines, the branch-and-link to STORAR increments CNTR by 1 and compares the result to 8. If the counter is less than 8, the program returns to the address stored in CHKFD1; otherwise, it goes to LDTBL1.

The contents of CNTR are utilized in DEL, RDEL, TDEL, and TAD3 in changing TCTABL.

External References:

ONE, EIGHT, CNTR--Common

Referencing Routines:

CHKFD, CHKFR, CHKFT, CHKFA

System Constants Affected:

None

System Constants Affected:

None

Exits:

a. Normal: LDTBL1, return address stored in CHKFD1

b. Abnormal: None

```
*
*       FIND TERMINAL ID IN TABLE
*
LDTBL1  MC  ZEROS,X1                    SET INDEX 1 TO 0
        MC  ZEROS,X2                    SET INDEX 2 TO 0
LDTBL2  C   BUFFER+5(4,2),TCTABL(,1)    ARE TERMINAL ID'S EQUAL?
R       BC  DEL(2),*+10(5)              IF YES, GOTO MODIFIED ADDRESS
        C   TCTABL(4,1),TCEOT           IF NO, IS THIS END OF TABLE?
        BC  TSET3(2)                    IF YES, GOTO ERROR ROUTINE
        A   TCCNS1,X1                   ADD 36 TO INDEX 1
        BC  LDTBL2(5)                   AND LOOP BACK TO LDTBL2
*
```

LDTBL1, LDTBL2

Find Terminal ID in Table

Explanation:

This routine is used to find the appropriate terminal entry in TCTABL to update; X2 points to the buffer contents and X1 points to the contents of TCTABL. X1 and X2 begin at '0000'.

The four digits in TCTABL(,1) are compared with those in BUFFER+5(4,2). If they are equal, the program branches to DEL, TDEL, or RDEL according to how it has been modified by the referencing routine. If they are not equal and the end of TCTABL has not been reached, 36 (the length of each TCTABL entry) is added to X1 and the search continues. If the end of TCTABL is reached with no match, the program branches to error routine TSET3.

External References:

ZERO, BUFFER--Common

X1, X2, TCTABL, TCEOT, TCCNS1--Partition

Referencing Routines:

CHKFD1, CHKFD2, CHKFR2, RDEL, CHKFT1, TDEL, DEL

System Constants Affected:

None

Exits:

a. Normal: DEL(modified), TSET3

b. Abnormal: None

```
*
*       ERROR ROUTINE
*
TSET1   MC TWO,INPSW                    SET INPSW TO SIGNAL INVALID ENTRY
        BC TOUT(5)                      BRANCH TO EXIT ROUTINE
*
        ORG *-5
ADCNG1  DM A'ZZ'
*
*
*       NORMAL EXIT ROUTINE
*
TSET2   MC ZERO,INPSW                   SET INPSW TO SIGNAL NORMAL
        BC TOUT(5)                      BRANCH TO EXIT ROUTINE
*
        ORG   *-5
A       DM     0C4              COMMAND OPTION LIST
        DM     C'D'             MEANS DELETE ENTIRE TERMINAL ENTRY
        DM     C'A'             MEANS ADD TRANSACTION CODES  CNTR=1
        DM     C'T'             MEANS DELETE TRANSACTION CODES CNTR=2
        DM C'R'                 MEANS REPLACE TERMINAL ID   CNTR=3
ASTER   DM C'*'                 END OF POSSIBLE OPTIONS
*
*
*       TERMINAL ID ERROR ROUTINE
*
TSET3   MC THREE,INPSW                  SET INPSW TO SIGNAL TERMINAL ERROR
TOUT    MC ZEROS(2),BUFFER+2            REMOVE SIGNAL FOR TABLE UPDATE
        MC ZEROS(1),TS3+4               RESET FORM NUMERIC FIELD
CDEXIT  BC CDEXIT(5)                    RETURN ADDRESS
*
```

TSET1, TSET2, TSET3

Exit Routines

Explanation:

TSET1 moves 2 into the message status switch INPSW to signal a format error, then branches to TOUT. TSET2 moves 0 into INPSW to signal a normal entry, then branches to TOUT. TSET3 moves 3 into INPSW to signal a terminal error or invalid entry and falls through to TOUT.

TOUT moves 00 into BUFFER+2, the partition number, to remove the signal for table update, clears the field used to form numeric, and returns to CDEXIT in CDSTR at the beginning of the CD-30 Terminal Read module.

External References:

ZEROS, ONE, TWO, THREE, INPSW, BUFFER--Common

Referencing Routines:

DEL, CHKFR2, RDEL, CHKFT1, TDEL1, CHKFA1, TAD2, TAD3,

LDTBL2

System Constants Affected:

INPSW

Exits:

a. Normal: CDEXIT

b. Abnormal: None

Blank Page

ROUTINE LOGIC--CD-30 PARTITION

This is the main line program that processes all transactions from the terminal, then branches to an appropriate output routine.

```
*
*       TERMINAL READ MODULE
*
CDSTR   C   CDPNO,BUFFER+2          DOES TABLE NEED UPDATE?
        BC  CDEXIT+1(6),CDCHK(2)    IF YES,GO TO TABLE UPDATE ROUTINE
*
        C   INHSW,ZERO              IS SYSTEM INHIBITED?
        BC  *+10(2),CDSTR(8)        IF YES,BRANCH & SWITCH PARTITION
*
        BC  CDRD(7),CDRD(0)         CD-30 SERVICE REQUEST?
        BC  CDSTR(8)                IF NO,SWITCH PARTITION
*
```

CDSTR

Beginning of CD-30 Partition

Explanation:

This is the first routine in each CD-30 partition. If a table update has been entered on the 7102 or Model 70 communications terminal, this routine performs a branch and link to CDCHK, the table update routine in Common. If the system is inhibited (INHSW ≠ 0), the routine branches back to CDSTR and switches to the next partition. If there is a service request from one of the CD-30 terminals on this IOC, the routine branches to CDRD; if not, back to CDSTR with a partition switch.

External References:

CDCHK, INHSW, ZERO, BUFFER--Common

CDRD--Partition

Referencing Routines:

TATTLE--Common

System Constants Affected:

None

Exits:

a. Normal: CDSTR,CDRD

b. Abnormal: None

```
CDRD    MC  BLANK,CDIN              CLEAR 1ST 6 CHARS OF INPUT AREA
        R   CDIN(0),0200(5)         READ CD-30 TRANSMISSION
        BC  **40(2),**30(1)
*
        MC  CDPAR2,CDERID           INDICATE ERROR 2 = FAULT ON READ
        BC  **20(5)
*
        MC  CDPAR1,CDERID           INDICATE ERROR NO. 1 = PARITY ON R
        A   ONE,PRCDNO              ADD 1 TO RECORD SERIAL #
        BC  **10(4),**20(5)         OVERFLOW?
        MC  ZEROS(5),PERCNT         IF YES,SET ERROR COUNT TO ZERO
*
        MC  CDNC(16),CDHEAD
*
        MC  CDBR,X2                 *
        A   CDSP,X2                 *
        MC  CDMXLN,CDMSGL           * COMPUTE MESSAGE LENGTH
        S   CM(,2),CDMSGL           *
        BC  **20(3),**10(5)         IF RESULT IS NOT POSITIVE
        MC  CDMIN+2(4),CDMSGL       SET CDMSGL TO '200'
        C   CDMSGL,CDMIN            IS MESSAGE LENGTH < 2?
        BC  **10(1),**20(5)         IF NO, SKIP NEXT INSTRUCTION
        A   TWO,CDMSGL              IF YES,ADD 2 TO MESSAGE LENGTH
*
*
        MC  SIZE1,CCSIZE            SET LENGTH FOR CODE CONVERT TO 6
        MC  IADDR1,X1               SET REG1 TO INPUT ADDR = CDIN
        MC  IADDR1,X2               SET REG2 TO OUTPUT ADDR = CDIN
        MN  TADDR1,CC8+1            SET A-ADDR TO TABLE = TBL1
        BC  CC4+1(6),CC1(5)         BRANCH & LINK TO CODE CONVERT
        MC  CDIN(6),TRNCDE          SAVE TRNCDE AND TERMINAL ID
*
        C   CDERSW,ZERO             IS TRANSMISSION IN ERROR?
        BC  **20(3)                 IF YES,BRANCH AROUND CALL TO TERMI
*                                   CHECK MODULE
        BC  TCEXIT+1(6),TCSTRT(5)   BRANCH AND LINK TO TERMINAL CHECK
*
        MC  SIZE2,CCSIZE            SET LENGTH FOR CODE CONVERT TO 22
        MC  IADDR2,X1               SET REG1 TO INPUT ADDR = CDHEAD
        MC  IADDR2,X2               SET REG2 TO OUTPUT ADDR = CDHEAD
        MN  TADDR2,CC8+1            SET A-ADDR TO TABLE = TBL2
        BC  CC4+1(6),CC1(5)         BRANCH AND LINK TO CODE CONVERT
*
*
        MC  CDMSGL,X1               MOVE MESSAGE LENGTH TO INDEX 1
        S   SIX,X1                  SUBTRACT 6 FROM INDEX 1
        BC  **60(1)                 IF MINUS,BRANCH AROUND NEXT CALL T
*
*
        MC  X1+1(3),CCSIZE          SET LENGTH TO REMAINING LENGTH OF
        MC  IADDR3,X1               SET REG1 TO INPUT ADDR = CDIN+6
        MC  IADDR3,X2               SET REG2 TO OUTPUT ADDR = CDIN+6
        MN  TADDR3,CC3+1            SET A-ADDR TO TABLE = TBL3
        BC  CC4+1(6),CC1(5)         BRANCH & LINK TO CODE CONVERT
*
```

CDRD

Read Transmission

Explanation:

This routine reads the CD-30 transmission into the cleared input/output area CDIN. CDIN is usually 200 characters long even though some programs, such as the one used here, do not use all 200 positions.

Normal Read. 1 is added to the five-digit record serial number PRCDNO (on PRCDNO overflow, the partition error counter PERCNT is set to zero), and the MIS 16-character prefix is moved into CDHEAD, which immediately precedes CDIN in core.

Next, the message length is computed as follows. CDBR, the B register base constant of 0101, is moved into X2, then CD5P (five times the partition number) is added to X2. Using partition 04 as an example, this would be

	X2
CDBR = 0101	0101
CD5P = 0020	+0020
	012T

The constant CDMXLN, maximum message length minus 1 (or 0199), is moved to CDMSGL, the message length. The contents of CM(,2) are subtracted from this. Since CM is defined as the first four positions of Common, modifying CM by the contents of X2 will give us address 0121 in Common, which is the address of the B register for Partition 04. The B register contains the number of characters remaining to be transferred, less one.

If the result of this subtraction is not positive, 200 is moved to message length CDMSGL and, if either positive or negative in the preceding subtraction, CDMSGL is now compared to minimum message length CDMIN (0002). If CDMSGL is less than 2, 2 is added to it. This last transaction is required because the IBM 360, which is sometimes used, is apt to ignore a record of sixteen characters or less. Since the MIS header prefix is sixteen characters long and since we want a record of at least eighteen characters in this example, adding 2 to CDMSGL will ensure that it is read.

Next, the first six characters of the terminal transmission (transaction code and terminal ID) are converted to System Ten USASCII and stored in TRNCDE and TCTID. At this point the program executes a branch and link to TCSTRT, Terminal Transaction Check, using the transaction code and the predefined TCTABL to see if the message length is correct for that terminal ID and transaction code. Next, the first

```
*
CDRD    MC BLANK,CDIN              CLEAR 1ST 6 CHARS OF INPUT AREA
        R  CDIN(0),0200(5)         READ CD-30 TRANSMISSION
        BC *+40(2),*+30(1)
*
        MC CDPAR2,CDERID           INDICATE ERROR 2 - FAULT ON READ
        BC *+20(5)
*
        MC CDPAR1,CDERID           INDICATE ERROR NO. 1 - PARITY ON R
        A  ONE,PRCDNO              ADD 1 TO RECORD SERIAL #
        BC *+10(4),*+20(5)         OVERFLOW?
        MC ZEROS(5),PERCNT         IF YES,SET ERROR COUNT TO ZERO
*
        MC CDNC(16),CDHEAD
*
        MC CDBR,X2                 *
        A  CD5P,X2                 *
        MC CDMXLN,CDMSGL           * COMPUTE MESSAGE LENGTH
        S  CM(,2),CDMSGL           *
        BC *+20(3),*+10(5)         IF RESULT IS NOT POSITIVE
        MC CDMIN+2(4),CDMSGL       SET CDMSGL TO '200'
        C  CDMSGL,CDMIN            IS MESSAGE LENGTH < 2?
        BC *+10(1),*+20(5)         IF NO, SKIP NEXT INSTRUCTION
        A  TWO,CDMSGL              IF YES,ADD 2 TO MESSAGE LENGTH
*
*
        MC SIZE1,CCSIZE            SET LENGTH FOR CODE CONVERT TO 6
        MC IADDR1,X1               SET REG1 TO INPUT ADDR - CDIN
        MC IADDR1,X2               SET REG2 TO OUTPUT ADDR - CDIN
        MN TADDR1,CC8+1            SET A-ADDR TO TABLE - TBL1
        BC CC4+1(6),CC1(5)         BRANCH & LINK TO CODE CONVERT
        MC CDIN(6),TRNCDE          SAVE TRNCDE AND TERMINAL ID
*
        C  CDERSW,ZERO             IS TRANSMISSION IN ERROR?
        BC *+20(3)                 IF YES,BRANCH AROUND CALL TO TERMI
*                                  CHECK MODULE
        BC TCEXIT+1(6),TCSTRT(5)   BRANCH AND LINK TO TERMINAL CHECK
*
        MC SIZE2,CCSIZE            SET LENGTH FOR CODE CONVERT TO 22
        MC IADDR2,X1               SET REG1 TO INPUT ADDR - CDHEAD
        MC IADDR2,X2               SET REG2 TO OUTPUT ADDR - CDHEAD
        MN TADDR2,CC8+1            SET A-ADDR TO TABLE - TBL2
        BC CC4+1(6),CC1(5)         BRANCH AND LINK TO CODE CONVERT
*
*
        MC CDMSGL,X1               MOVE MESSAGE LENGTH TO INDEX 1
        S  SIX,X1                  SUBTRACT 6 FROM INDEX 1
        BC *+60(1)                 IF MINUS,BRANCH AROUND NEXT CALL T
*
*
        MC X1+1(3),CCSIZE          SET LENGTH TO REMAINING LENGTH OF
        MC IADDR3,X1               SET REG1 TO INPUT ADDR - CDIN+6
        MC IADDR3,X2               SET REG2 TO OUTPUT ADDR - CDIN+6
        MN TADDR3,CC8+1            SET A-ADDR TO TABLE - TBL3
        BC CC4+1(6),CC1(5)         BRANCH & LINK TO CODE CONVERT
```

sixteen characters of the output (MIS header, CDHEAD) are converted from System Ten USASCII to output code and then any remaining characters in the message are converted from CD-30 code to output code for later recording. These last two conversions are optional; see Section 5, "Modifications for Code Conversion".

Faulty Read. Condition codes are checked immediately after the Read instruction. On condition code 1, 'PAR1' is moved into the error description field CDERID (the first character of which is also CDERSW) to indicate a parity error. Condition code 4 will cause '2PAR2' to be moved into CDERID, indicating a fault on read. The program proceeds normally until after the first code conversion. If CDERSW ≠ 0 at this point, the program does not go to TCSTRT to verify message length but does continue with code conversion routines because this data must be in proper code to be written on the tape with the errors noted. This last step is optional; see Section 5.

External References:

ZERO, ONE, TWO, SIX, PRCDNO, PERCNT, CM, SIZE1, SIZE2,

CCSIZE, IADDR1, IADDR2, IADDR3, CC1, CC4, CC8, TADDR1,

TADDR2, TADDR3--Common

TCSTRT, CDPAR1, CDPAR2--Partition

Referencing routines:

CDSTR

System Constants Affected:

PRCDNO, PERCNT

Exits:

a. Normal: CDACK

b. Abnormal: None

```
*
*       ACKNOWLEDGEMENT MODULE
*
CDACK   MC CDERID(1),CDGOOD          SAVE LOCAL ERROR SWITCH
*                                  BEFORE ERROR 3 CAN OCCUR
        C  TRNCDE,EIGHT              IS THIS A BADGE TRANSACTION?
        BC CDINH(2)                  IF YES,BRANCH TO CDINH
        C  CDERSW,ZERO               IS TRANSMISSION IN ERROR?
        BC *+50(2)
        W  0(2),02(3)                WRITE NAK
        BC CDINH(2)
        W  0(2),02(3)                WRITE NAK AGAIN
        BC CDINH+10(5)
        W  0(1),02(3)                WRITE ACK
        BC CDINH(2)
        W  0(1),02(3)                WRITE ACK AGAIN
        MC CDPAR3,CDERID             INDICATE ERROR 3
*
```

CDACK

Acknowledgement

Explanation:

This routine responds to the CD-30 transmission by sending or attempting to send a NAK or ACK to the terminal. The first character of CDERID (CDERSW) is moved to CDGOOD to save error code 1, 2, 4, or 5 if any was noted. Then several tests are made. If transmission code TRNCDE is 8, indicating a badge transaction, the program branches to CDINH, a routine that directs it to either error processing routine CDERPO or to output write routine CDWT.

Next, CDERSW is checked for error codes. If there is none, ACK is written on the CD-30 terminal and the program branches to CDINH if the write was normal. The same procedure is followed with NAK if there are error codes stored in CDERSW. If neither ACK nor NAK was received in a normal way by the CD-30 terminal, another write attempt is made and '3PAR3' is moved into CDERID to indicate a lack of response from the terminal.

The program now falls through to CDINH.

External References:

ZERO, EIGHT--Common

TRNCDE, CDPAR3, CDERID, CDGOOD--Partition

Referencing Routines:

None

System Constants Affected:

None

Exits:

a. Normal: CDINH

B. Abnormal: None

```
*
CDINH   C   INHSW,ZERO              IS SYSTEM INHIBITED?
        BC  *+10(2),CDINH(8)        IF YES BRANCH AND SWITCH
        C   CDERSW,ZERO             IS TRANSMISSION IN ERROR?
        BC  CDERPO(3),CDWT(5)       IF NO, BRANCH TO WRITE ROUTINE
*                                 IF YES,CONTINUE
*
```

CDINH

Explanation:

This routine directs the program to CDERPO for error processing if an error code is stored in CDERSW and to the appropriate write routine if no errors have been noted.

External References:

ZERO, CDWT, INHSW--Common

CDERSW, CDERPO--Partition

Referencing Routines:

CDACK, CDE2

System Constants Affected:

None

Exits:

a. Normal: CDERPO, CDWT

b. Abnormal: None

```
*
*
*
*       ERROR PROCESSING MODULE
*
CDERPO  C   ECOUNT,THREE            ERROR BUFFERS FILLED?
        BC  CDE1(1)
        MN  ONE,INHSW               IF YES,TURN ON INHIBIT SWITCH
        MN  ONE,ERRSW               TURN ON ERROR SWITCH
        C   INHSW,ZERO              WAIT FOR ERROR BUFFERS
        BC  *+10(2),*-10(8)         TO BE PRINTED ON /102
CDE1    C   ECOUNT,ONE              WHICH BUFFER IS EMPTY?
        BC  *+30(2),*+50(3)
        MC  ZEROS,X1                SET INDEX TO FILL FIRST ERROR BUFF
        BC  CDE2(5)
        MC  THTY2,X1                SET INDEX TO FILL SECOND ERROR BUF
        BC  CDE2(5)
        MC  SIXTY4,X1               SET INDEX TO FILL THIRD ERROR BUFF
*
*       FILL ERROR BUFFER
*
CDE2    MC  CDERID+1(4),ERRBUF+7(,1) ERROR ID (4)
        MC  CDPNO,ERRBUF+12(,1)      PARTITION NUMBER
        MC  SCHRS,ERRBUF+15(,1)      HOURS (2)
        MC  SCL1+2(1),ERRBUF+18(,1)  TENTHS
        MC  SCL1+3(1),ERRBUF+20(,1)  HUNDREDS
        MC  TCTID,ERRBUF+22(,1)      TERMINAL NUMBER
        MC  TRNCDE,ERRBUF+27(,1)     TRANSACTION CODE
        MC  CDMSGL+1(3),ERRBUF+29(,1) MESSAGE LENGTH
        MC  CDTCNT+1(3),ERRBUF+33(,1) EXPECTED MESSAGE LENGTH
*
        A   ONE,ECOUNT              ADD 1 TO ERROR COUNTER
        C   CDGOOD,CDERSW           HAS ERROR 3 OCCURED?
        BC  *+30(2)
        MC  CDGOOD,CDERSW           IF YES,RESTORE LOCAL ERROR SWITCH
        BC  CDINH+20(5)             AND RETURN TO ERROR PROCESSING
*                                ROUTINE IF NECESSARY
*
        MC  ZERO,CDERSW             CLEAR THE LOCAL ERROR SWITCH
        MC  ZEROS(3),CDTCNT         CLEAR MODULUS REMAINDER AREA
        C   CDGOOD,ZERO             IS TRANSMISSION IN ERROR?
*                                CDGOOD CONTAINS ERROR NO. 1,2,4, OR 5.
        BC  CDWT(2)                 IF NO, GO TO WRITE ROUTINE
        MC  E,CDHEAD                MOVE E TO HEADER
        MC  CDGOOD,CDHEAD+2         MOVE ERROR NO. TO 3RD POSITION OF
        A   ONE,PERCNT              ADD 1 TO ERROR RECORD COUNT
```

CDERPO

Error Processing

Explanation:

This routine checks the three error buffers as errors are noted. ECOUNT = 3 indicates that the error buffers are full so 1 is moved into INHSW and ERRSW to turn them on. The program executes a branch and switch routine until the errors are printed on the communications terminal when the CPU is processing the standard partition entrance routine, INIT, and INHSW is turned off (back to 0).

If the error buffers are not full, the program branches to or falls through to CDE1 to determine which buffer is empty; CDE2 is the routine used to fill the empty one by moving pertinent information into ERRBUF. ECOUNT is the incremented by 1.

CDGOOD is compared to CDERSW to see if error 3 (faulty transmission of ACK or NAK) has occurred. As the contents of CDERID (CDERSW) have been moved to CDGOOD in CDACK before error 3 could have occurred, an unequal compare indicates that error 3 was the one just processed in CDE2. CDGOOD is moved back to CDERSW and the program branches back to CDINH for further error processing if necessary to record errors 1, 2, 4, or 5 if they have occured.

If CDGOOD equals CDERSW (no ACK or NAK transmission errors or such errors already processed), 0 is moved into CDERSW and CDTCNT. CDGOOD is now compared to 0. An equal compare indicates that error 3 was the only one encountered and, since this has already been processed, the program branches to CDWT to be recorded on tape or disc. On an unequal compare, appropriate error identifications are moved into CDHEAD and the program branches to CDWT. The "E" (for error) that is moved into CDHEAD must be defined in the DM statements as a System Ten USASCII character with a bit structure that will result in the letter E in the output code.

External References:

ZERO, ONE, THREE, FIVE, INHSW, ERRSW, ECOUNT, PERCNT,

ERRBUF, SCHRS, SCHL1, CDWT SPACE--Common

THTY2, SIXTY4, CDERID, CDPNO, TCTID, TRNCDE, CDMSGL

CDMSGL, CDTCNT, CDGOOD, CDERSW, CDINH--Partition

Blank Page

Referencing Routines:

CDINH

System Constants Affected:

ECOUNT, INHSW, ERRSW, PERCNT

Exits:

a. Normal: CDINH, CDWT

b. Abnormal: None

```
*                                  TAPE WRITE ROUTINE
*
CDWT.   A SIXTN,CDMSGL
        MC CDMSGL,X2
*       BC WTEXIT+1(6),WTSTR(5)
        BC CDSTR(5)
WTBUF   DM A'CDHEAD'
```

```
*                                  DISC WRITE ROUTINE
WTBUF   DM A 'CDMSGL'
CDWT    A  SIXTN,CDMSGL            ADD LENGTH OF PREFIX TO CDMSGL
        C  DISCSW,ONE              IS DISC PACK FULL?
        BC *+10(2),*+20(5)          IF NO, CONTINUE
        BC *-20(8)                  IF YES, CONTINUE IN WAIT LOOP
        C  DINDC,ZERO              IS DISC BUSY?
        BC *+10(2),*-10(8)         IF NO, CONTINUE
*                                  IF YES,CONTINUE IN WAIT LOOP
        MC WTBUF,DINPT1            SET WRITE BUFFER AREA
*
        BC DSTREX+1(6),DSTORE(5)   BRANCH AND LINK TO DISC WRITE
        C  DISCSW,ZERO             IS WRITE GOOD?
        BC CDSTR(2),CDWT+10(8)     IF YES, BRANCH TO BEGINNING
*                                  OF PARTITION
*                                  IF NO, TRY AGAIN
```

CDWT

Link To Write Routines

Explanation:

Tape Output

Sixteen, the length of the header prefix, is added to the message length which is then moved to X2 for use by the Tape Write Module. A branch-and-link is executed to WTSTR, the beginning address of the Tape Write Module. On return, the program branches to CDSTR at the start of the CD-30 partition.

External References:

SIXTN, WTEXIT, WTSTR--Common

CDSTR--Partition

Referencing Routines:

CDINH, CDSTR

System Constants Affected:

None

Exits:

a. Normal: CDSTR

b. Abnormal: None

Disc Output

The first instruction adds the length of the header prefix to the message length contained in CDMSGL. The standard MIS header prefix is 16 characters; if the prefix is a different length, 'SIXTN' must be redefined. The status of the disc is checked and, when the disc is available, the address of the output storage area is moved to the write buffer.

The program executes a branch-and-link to DSTORE in the Disc Write Routine. If the write is good, the program branches back to partition beginning (CDSTR); if bad, it tries the write again.

External References:

ZERO, ONE, SIXTN, DISCSW, DINDC, DINPT1--Common

CDMSGL, CDSTR, WTBUF--Partition

Referencing Routines:

CDINH, CDERPO

System Constants Affected:

None

Exits:

a. Normal: CDSTR

b. Abnormal: None

```
*         SINGLE-ENTRY, MODULUS REMAINDER TCSTRT
*
*
*
*       TERMINAL TRANSACTION CHECK MODULE
*
*       THE PURPOSE OF THIS ROUTINE IS TO CHECK TRANSACTION CODE,
*       TERMINAL NUMBER, AND LENGTH OF MESSAGE (MODULUS REMAINDER)
*       OF EVERY CD-30 TRANS AGAINST A PREDEFINED TABLE.
TCSTRT MC  ZEROS,X1
TCWK1  MC  TRNCDE(1),X1+2
       M   FOUR,X1+2(1)
       C   TCTABL(3,1),BLANK
       BC  TCERR2(2)
TCWK2  MC  CDMSGL+1(3),TCTS1
       D   FOUR,TCTS1(2)
       C   TCTS1+2(1),TCTABL+2(3,1)
       BC  TCEXIT-10(2)
TCERR3 MC  TCIWLM,CDERID
       MC  TCTABL(3,1),CDTCNT+1
       BC  TCEXIT(5)
       ORG     *-5
TCITC  DM  C'4TRAN'
TCERR2 MC  TCITC,CDERID
       BC  *+10(5)
       MC  BLANK(4),CDTCNT
TCEXIT BC  TCEXIT(5)
       ORG     *-5
TCB    DM      C'00'
TCTABL DM C'    000 002 001 001 000 002 001     '
TCIWLM DM C'5WLM '
TCTS1  DM C'0000'
*
*
*
*
```

```
*        MULTIPLE-ENTRY TCSTRT WITH MODULUS REMAINDER
*
*
TCSTRT MC ZEROS,X1                SET INDEX 1
       C  TCTID,TCTABL(,1)        COMPARE TCTID & TERM.NO.IN TCTABL
       BC TCWK1(2)                IF EQUAL, FIND TRANSACTION CODE
       C  TCEOT,TCTABL(,1)
       BC TCERR1(2)               IF EQUAL, GO TO ERROR ROUTINE
       A  TCCNS1+2(2),X1          IF NOT,ADD 36 TO INDEX 1
       BC TCSTRT+10(5)            GO TO TEST NEXT TERM ID NO.
*
       ORG    *-5
TCTERM DM     C'6TERM'
*
TCERR1 MC TCTERM,CDERID           SIGNAL TERMINAL ERROR
       BC TCERR2+10(5)            EXIT
*
TCWK1  MC ZEROS,X2                SET INDEX 2
*
       MC TRNCDE(1),X2+2          *
       M  FOUR,X2+2(1)            * COMPUTE TABLE POINTER
       A  X2+2(2),X1(4)           *
       C  TCTABL(3,1),BLANK       IS POSITION IN TABLE BLANK?
       BC TCERR2(2)               IF YES,SIGNAL TRNCDE ERROR.
*
       MC CDMSGL+1(3),TCTS1
       D  FOUR,TCTS1(2)
       C  TCTS1+2(1),TCTABL+2(,1)
       BC TCEXIT-10(2)            IF YES,EXIT.
*
       MC TCIWLM,CDERID           IF NO,SIGNAL WRONG LENGTH MSG
       MC TCTABL(3,1),CDTCNT+1    MOVE CORRECT LENGTH TO CDTCNT
       BC TCEXIT(5)               EXIT
*
       ORG  *-5
TCITC  DM C'4TRAN'                TRANSACTION CODE INVALID
*
TCERR2 MC TCITC,CDERID            SIGNAL TRNCDE ERROR
       MC BLANK(4),CDTCNT         BLANK OUT CDTCNT
*
TCEXIT BC TCEXIT(5)               EXIT
*
       ORG  *-5
TCIWLM DM C'5WLM '                WRONG LENGTH MESSAGE
FOURS  DM C'0004'
TCCNS1 DM C'0036'
TCTS1  DM C'0000'
*
TCTABL DM 0C360                   TERMINAL TRANSACTION TABLE
       DM C36'9999001 003 001 002 003 003 003 000 '
       DM C36'8888000 001 002 003 000 002 003 000 '
       DM C36'7777000 001 003 001 002 002 002 000 '
       DM C36'6666001 003 003 002 002 001 001 000 '
       DM C36'5555002 001 002 001 001 001 003 000 '
       DM C36'4444003 002 002 001 001 002 003 002 '
       DM C36'3333002 003 003 001 000 000 002 001 '
       DM C36'2222000 002 003 001 002 000 001 002 '
       DM C36'1111002 001 003 002 001 000 001 003 '
       DM C36'0000001 003 002 002 001 003 000 001 '
TCEOT  DM C':::::'
*
*       END OF ROUTINE
*
```

```
*       TERMINAL TRANSACTION CHECK MODULE
*
*       THE PURPOSE OF THIS ROUTINE IS TO CHECK THE MESSAGE LENGTH
*       OF THE TRANSMISSION ACCORDING TO THE TRANSACTION CODE AND
*       A PREDEFINED TABL.
*
TCSTRT MC  ZEROS,X1
       MC  ZEROS,X2                    SET INDEX 2
*
       MC  TRNCDE(1),X2+2              *
       M   FOUR,X2+2(1)                *  COMPUTE TABLE POINTER
       A   X2+2(2),X1(4)               *
       C   TCTABL(3,1),BLANK           IS POSITION IN TABLE BLANK?
       BC  TCERR2(2)                   IF YES,SIGNAL TRNCDE ERROR.
*
       C   CDMSGL+1(3),TCTABL(,1)      DOES MESSAGE LENGTH CHECK?
       BC  TCEXIT-10(2)                IF YES,EXIT.
*
       MC  TCIWLM,CDERID               IF NO,SIGNAL WRONG LENGTH MSG
       MC  TCTABL(3,1),CDTCNT+1        MOVE CORRECT LENGTH TO CDTCNT
       BC  TCEXIT(5)                   EXIT
*
*
TCERR2 MC  TCITC,CDERID                SIGNAL TRNCDE ERROR
       MC  BLANK(4),CDTCNT             BLANK OUT CDTCNT
*
TCEXIT BC  TCEXIT(5)                   EXIT
*
TCTABL DM  0C36                  TERMINAL TRANSACTION TABLE
       DM  C36'9999076 076 076 076 076 076 076 042 '
TCEOT  DM  C':::::'
TCITC  DM  C'4TRAN'                      TRANSACTION CODE INVALID
TCIWLM DM  C'5WLM '                      WRONG LENGTH MESSAGE
*       END OF ROUTINE
*
*
*
```

```
*
*
*
*        MULTIPLE-ENTRY TCSTRT
*
*
TCSTRT MC ZEROS,X1                    SET INDEX 1
       C  TCTID,TCTABL(,1)            COMPARE TCTID & TERM.NO.IN TCTABL
       BC TCWK1(2)                    IF EQUAL, FIND TRANSACTION CODE
       C  TCEOT,TCTABL(,1)
       BC TCERR1(2)                   IF EQUAL, GO TO ERROR ROUTINE
       A  TCCNS1+2(2),X1              IF NOT,ADD 36 TO INDEX 1
       BC TCSTRT+10(5)                GO TO TEST NEXT TERM ID NO.
*
       ORG    *-5
TCTERM DM     C'6TERM'
*
TCERR1 MC TCTERM,CDERID               SIGNAL TERMINAL ERROR
       BC TCERR2+10(5)                EXIT
*
TCWK1  MC ZEROS,X2                    SET INDEX 2
*
       MC TRNCDE(1),X2+2              *
       M  FOUR,X2+2(1)                *  COMPUTE TABLE POINTER
       A  X2+2(2),X1(4)               *
       C  TCTABL(3,1),BLANK           IS POSITION IN TABLE BLANK?
       BC TCERR2(2)                   IF YES,SIGNAL TRNCDE ERROR.
*
       C  CDMSGL+1(3),TCTABL(,1)      DOES MESSAGE LENGTH CHECK?
       BC TCEXIT-10(2)                IF YES,EXIT.
*
       MC TCIWLM,CDERID               IF NO,SIGNAL WRONG LENGTH MSG
       MC TCTABL(3,1),CDTCNT+1        MOVE CORRECT LENGTH TO CDTCNT
       BC TCEXIT(5)                   EXIT
*
       ORG  *-5
TCITC  DM C'4TRAN'                        TRANSACTION CODE INVALID
*
TCERR2 MC TCITC,CDERID                SIGNAL TRNCDE ERROR
       MC BLANK(4),CDTCNT             BLANK OUT CDTCNT
*
TCEXIT BC TCEXIT(5)                   EXIT
*
       ORG  *-5
TCIWLM DM C'5WLM '                        WRONG LENGTH MESSAGE
FOURS  DM C'0004'
TCCNS1 DM C'0036'
*
TCTABL DM 0C36                    TERMINAL TRANSACTION TABLE
       DM C36'9999076 076 076 076 076 076 076 042 '
       DM C36'8888076 035 076 076 076 076 076 042 '
       DM C36'7777076 076 015 076 076 076 076 042 '
       DM C36'6666076 076 076 065 076 076 076 042 '
       DM C36'5555076 076 076 076 024 076 076 042 '
       DM C36'4444076 076 076 076 076 035 076 042 '
       DM C36'3333076 076 076 076 076 076 024 042 '
       DM C36'2222076 076 076 076 076 024 076 042 '
       DM C36'1111076 076 076 076 035 076 076 042 '
       DM C36'0000076 076 076 024 076 076 076 042 '
TCEOT  DM C':::::'
*
*       END OF ROUTINE
*
```

TCSTRT

Terminal Transaction Check

Explanation:

This routine uses transaction code TRNCDE and one of the four versions of TCTABL (see Section 5 for explanation) to verify the message length that was calculated in CDRD. The first seven instructions in the multiple-entry TCSTRT use the terminal ID number to find the proper TCTABL entry. The single-entry tables do not use these instructions. TRNCDE is multiplied by 4 and the product is placed in index register 2, which is added to X1, to use in finding the location in TCTABL that contains the information pertaining to this transaction--acceptable message length and routing indicator (the latter is not used in this particular program). In modulus-remainder versions, the message length is divided by four and the remainder is checked instead of the actual message length.

Invalid transaction code (4TRAN) and wrong-length message (5WLM) indicators are moved into CDERID if required. If the message is the wrong length, the correct length is moved into CDTCNT to be used later in the error processing routine for printing an error message on the communications terminal.

The program links back to CDRD to continue with code conversion.

External References:

BLANK, ZEROS, FOUR--Common

X1, X2, TRNCDE, TCTABL, TCTID, CDMSGL, TCIWLM,

CDTCNT--Partition

Referencing Routines:

CDRD

System Constants Affected:

None

Exits:

a. Normal: TCERR2, TCEXIT

b. Abnormal: None

APPENDIX A

TABLE OF INTERNAL CODES

Code Convert Logic Manual 5008 gives a comprehensive explanation of the code conversion routine and code conversion tables.

SYSTEM TEN CHARACTER	CD-30 TERMINAL OUTPUT FRIDEN BCD	CD-30 (TRUNCATED)	SYSTEM TEN USASCII	USASCII	USASCII 7-TRACK TAPE CODE	7-TRACK BCD TAPE CODE	GE-400 SERIES	EBCDIC (BINARY)	SYSTEM TEN CHARACTERS (DOUBLE FRAME)	HEX
	8 7 6 5 4 3 2 1		7 5 4 3 2 1	7 6 5 4 3 2 1	B A 8 4 2 1	B A 8 4 2 1	B A 8 4 2 1	0 1 2 3 4 5 6 7		
0	6	0	5	6 5	A	8 2		1 1 1 1 0 0 0 0	0 0	F 0
1	1	!	5 1	6 5 1	A 1	1	1	0 0 0 1	1 0	F 1
2	2	"	5 2	6 5 2	A 2	2	2	0 0 1 0	2 0	F 2
3	5 2 1	#	5 2 1	6 5 2 1	A 2 1	2 1	2 1	0 0 1 1	3 0	F 3
4	3	$	5 3	6 5 3	A 4	4	4	0 1 0 0	4 0	F 4
5	5 3 1	%	5 3 1	6 5 3 1	A 4 1	4 1	4 1	0 1 0 1	5 0	F 5
6	5 3 2	&	5 3 2	6 5 3 2	A 4 2	4 2	4 2	0 1 1 0	6 0	F 6
7	3 2 1	'	5 3 2 1	6 5 3 2 1	A 4 2 1	4 2 1	4 2 1	0 1 1 1	7 0	F 7
8	4	(	5 4	6 5 4	A 8	8	8	1 0 0 0	8 0	F 8
9	5 4 1	)	5 4 1	6 5 4 1	A 8 1	8 1	8 1	1 0 0 1	9 0	F 9
A	7 6 1	Q	7 1	7 1	B 1	B A 1	A 1	1 1 0 0 0 0 0 1	1 L	C 1
B	7 6 2	R	7 2	7 2	B 2	B A 2	A 2	0 0 1 0	2 L	C 2
C	7 6 5 2 1	S	7 2 1	7 2 1	B 2 1	B A 2 1	A 2 1	0 0 1 1	3 L	C 3
D	7 6 3	T	7 3	7 3	B 4	B A 4	A 4	0 1 0 0	4 L	C 4
E	7 6 5 3 1	U	7 3 1	7 3 1	B 4 1	B A 4 1	A 4 1	0 1 0 1	5 L	C 5
F	7 6 3 2	V	7 3 2	7 3 2	B 4 2	B A 4 2	A 4 2	0 1 1 0	6 L	C 6
G	7 6 3 2 1	W	7 3 2 1	7 3 2 1	B 4 2 1	B A 4 2 1	A 4 2 1	0 1 1 1	7 L	C 7
H	7 6 4	X	7 4	7 4	B 8	B A 8	A 8	1 0 0 0	8 L	C 8
I	7 6 5 4 1	Y	7 4 1	7 4 1	B 8 1	B A 8 1	A 8 1	1 0 0 1	9 L	C 9
J	7 5 1	A	7 4 2	7 4 2	B 8 2	B 1	B 1	1 1 0 1 0 0 0 1	1 M	D 1
K	7 5 2	B	7 4 2 1	7 4 2 1	B 8 2 1	B 2	B 2	0 0 1 0	2 M	D 2
L	7 2 1	C	7 4 3	7 4 3	B 8 4	B 2 1	B 2 1	0 0 1 1	3 M	D 3
M	7 5 3	D	7 4 3 1	7 4 3 1	B 8 4 1	B 4	B 4	0 1 0 0	4 M	D 4
N	7 3 1	E	7 4 3 2	7 4 3 2	B 8 4 2	B 4 1	B 4 1	0 1 0 1	5 M	D 5
O	7 4 3 2	F	7 4 3 2 1	7 4 3 2 1	B 8 4 2 1	B 4 2	B 4 2	0 1 1 0	6 M	D 6
P	7 5 3 2 1	G	7 5	7 5	B A	B 4 2 1	B 4 2 1	0 1 1 1	7 M	D 7
Q	7 5 4	H	7 5 1	7 5 1	B A 1	B 8	B 8	1 0 0 0	8 M	D 8
R	7 4 1	I	7 5 2	7 5 2	B A 2	B 8 1	B 8 1	1 0 0 1	9 M	D 9
S	6 5 2	J	7 5 2 1	7 5 2 1	B A 2 1	A 2	B A 2	1 1 1 0 0 0 1 0	2 N	E 2
T	6 1	K	7 5 3	7 5 3	B A 4	A 2 1	B A 2 1	0 0 1 1	3 N	E 3
U	6 5 3	L	7 5 3 1	7 5 3 1	B A 4 1	A 4	B A 4	0 1 0 0	4 N	E 4
V	6 3 1	M	7 5 3 2	7 5 3 2	B A 4 2	A 4 1	B A 4 1	0 1 0 1	5 N	E 5
W	6 3 2	N	7 5 3 2 1	7 5 3 2 1	B A 4 2 1	A 4 2	B A 4 2	0 1 1 0	6 N	E 6
X	6 5 3 2 1	O	7 5 4	7 5 4	B A 8	A 4 2 1	B A 4 2 1	0 1 1 1	7 N	E 7
Y	6 5 4	P	7 5 4 1	7 5 4 1	B A 8 1	A 8	B A 8	1 0 0 0	8 N	E 8
Z	6 4 1	Q	7 5 4 2	7 5 4 2	B A 8 2	A 8 1	B A 8 1	1 0 0 1	9 N	E 9
SP	5	SP		6		A	A	0 1 0 0 0 0 0 0	0 4	4 0
!			1	6 1	1	B 8 2	B A 8 4 2 1	0 1 0 1 1 0 1 0	J 5	5 A
"			2	6 2	2		B A 8 4 2	0 1 1 1 1 1 1 1	O 7	7 F
#			2 1	6 2 1	2 1	8 2 1	8 2 1	0 1 1 1 1 0 1 1	K 7	7 B
$			3	6 3	4	8 2 1	B 8 2 1	0 1 0 1 1 0 1 1	K 5	5 B
%	7 5 4 2 1	K	3 1	6 3 1	4 1	A 8 4	B A 8 4	0 1 1 0 1 1 0 0	L 6	6 C
&			3 2	6 3 2	4 2	B A	A 8 2	0 1 0 1 0 0 0 0	O 5	5 0
'			3 2 1	6 3 2 1	4 2 1		B 8 4 2 1	0 1 1 1 1 1 0 1	M 7	7 D
(			4	6 4	8	A 8 4	A 8 4 1	0 1 0 0 1 1 0 1	M 4	4 D
)			4 1	6 4 1	8 1	B A 8 4	B 8 4 1	0 1 0 1 1 1 0 1	M 5	5 D
*			4 2	6 4 2	8 2	B 8 4	B 8 4	0 1 0 1 1 1 0 0	L 5	5 C
+			4 2 1	6 4 2 1	8 2 1	B A	B A	0 1 0 0 1 1 1 0	N 4	4 E
,	6 5 4 2 1	;	4 3	6 4 3	8 4	A 8 2 1	B A 8 2 1	0 1 1 0 1 0 1 1	K 6	6 B
-	7	@	4 3 1	6 4 3 1	8 4 1	B	B 8 2	0 1 1 0 0 0 0 0	O 6	6 0
.	7 6 4 2 1	[	4 3 2	6 4 3 2	8 4 2	8 A 8 2 1	A 8 2 1	0 1 0 0 1 0 1 1	K 4	4 B
/	6 5 1	\	4 3 2 1	6 4 3 2 1	8 4 2 1	A 1	B A 1	0 1 1 0 1 0 0 1	I 6	6 1
:			5 4 2	6 5 4 2	A 8 2	8 4 1	8 4 1	0 1 1 1 1 0 1 0	J 7	7 A
;			5 4 2 1	6 5 4 2 1	A 8 2 1	B 8 4 2	B 8 4 2	0 1 0 1 1 1 1 0	N 5	5 E
=			5 4 3 1	6 5 4 3 1	A 8 4 1	8 2 1	B A 8 4 1	0 1 1 1 1 1 1 0	N 7	7 E
>			5 4 3 2	6 5 4 3 2	A 8 4 2	8 4 2	8 4 2	0 1 1 0 1 1 1 0	N 6	6 E
?			5 4 3 2 1	6 5 4 3 2 1	A 8 4 2 1	B A 8 2	8 4 2 1	0 1 1 0 1 1 1 1	O 6	6 F
@			7	7	B	8 4	8 4	0 1 1 1 1 1 0 0	L 7	7 C
[			7 5 4 2 1	7 5 4 2 1	B A 8 2 1	B A 8 4 1	8 2	0 1 0 0 1 1 0 1	M 4	4 D
			7 5 4 3	7 5 4 3	B A 8 4	A 8 4 2	A 8 4 2 1	0 1 1 0 1 1 1 0	N 6	6 E
]			7 5 4 3 1	7 5 4 3 1	B A 8 4 1	B 8 4 1	A 8 4	0 1 0 1 1 1 0 1	M 5	5 D
<			7 5 4 3 2 1	7 5 4 3 2 1	B A 8 4 2 1		B A 8 2	0 1 1 0 1 1 0 1	M 6	6 D
			5 4 3	6 5 4 3	A 8 4	B A 8 4 2	A 8 4 2	0 1 0 0 1 1 0 0	L 4	4 C
			7 5 4 3 2	7 5 4 3 2	B A 8 4 2		B	0 1 0 0 1 1 0 1	L 4	4 C

This chart shows the relative codes for the System Ten characters listed on the left. The second column shows the original 8-bit representation of this character that the CD-30 sent. (For these characters, the eighth bit is off; therefore, no eighth bit is shown here.) The third column shows the character that System Ten reads, because of the standard truncation of CD-30 code (bits 8 and 5 are ignored, bit 6 is accepted as the fifth bit in System Ten USASCII). The remaining columns show the representation of the character on the far left, in the type of code indicated at the top of each column.

APPENDIX B

SPECIFIC PROGRAM CHANGES REQUIRED FOR DIFFERENT COLLECTADATA-30 TERMINAL PARTITIONS AND FOR DIFFERENT OPTIONS

Numeric Constants

CD5P	Five times the partition number. Different for each partition in which the CD-30 program module resides.
CDPNO	Number of the partition in which that program resides.
TCTABL	Terminal transaction table. Length depends on table used (see Section 5 for explanation of table variations).

Storage Areas

BUFFER	Must be large enough to hold table update message from communications terminal.

Code Conversion

CDOUT	Output area for double-frame code conversion. Must be large enough to contain MIS header prefix CDHEAD and entire terminal message after conversion to double-frame code. CDIN (input area for double-frame conversion programs and input/output area for single-frame conversion programs) and CDOUT must be used as address constants in the Code Convert Module 5008.

The first call to Code Convert is used to convert transaction code and terminal ID (TRNCDE and TCTID) to System Ten USASCII. The second and third calls to Code Convert are optional. The values in the Code Convert tables will depend on the output code. See Code Convert Logic Manual 5008 for tables and constants.

CDERPO--Error Processing Routine

An instruction near the end of this routine moves an "E" (for error) to the MIS header prefix, CDHEAD:

```
MC  E,CDHEAD.
```

CDHEAD is either in USASCII or has been converted to an output code; therefore, operand A ('E' here) must be defined as a constant with a bit structure that will result in the letter E in the output code.

Terminal/Transaction Tables

Section 5 explains how to set up single- and multiple-entry transaction tables. The constant TCTABL must be defined according to which type of table is used. The first instructions of TCSTRT, Terminal/Transaction Check routine, must also be modified. If the multiple-entry table is used, the initial instructions that are used to locate terminal ID in TCTABL must be included. Listings of the various modifications of TCSTRT are shown in the routine logic discussion of TCSTRT, page 9- .

Tape/Disc Output

Label		DM	Use For:
WTBUF	DM	A'CDOUT'	Tape output of double-frame code
	DM	A'CDHEAD'	Tape output of single- rame code
	DM	A'CDMSGL'	Disc output: CDMSGL must be DM'd immediately before CDOUT or CDHEAD.
SIXTN			A numeric constant of '16' in the first instruction in the CDWT routine. This constant must be the length of the message prefix. If other than the standard 16-character MIS prefix is used, this constant must be redefined or changed.

APPENDIX C

CONSTANTS IN COMMON USED BY CDCHK AND CDSTR THAT ARE DEFINED IN THE CONTROL MODULE

BLANK	THREE	
CM	TWO	
E	ZERO	
ECOUNT	ZEROS	
EIGHT	CDNC	MIS 16-character prefix
ERRBUF	PCERPT	
ERRSW	PRCDNO	
FIVE	SCDAYR	
FOUR	SCDAWK	
INHSW	SCHRS	
INPSW	SCL1	
NINE	SCTIME	
ONE		
PERCNT		
SEVEN		
SIX		
SPACE		
TBLP		