INTERCOMM

UTILITIES USERS GUIDE

ISOGON
330 Seventh Avenue · New York, New York 10001

LICENSE: INTERCOMM TELEPROCESSING MONITOR

Utilities Users Guide

Publishing History

Publication	Date	Remarks
First Edition	September 1973	This manual corresponds to Intercomm Release 6.0.
IPN 021	December 1973	Errata and general updates.
IPN 026	January 1974	Errata and general updates.
IPN 031	February 1974	Errata.
IPN 042	April 1974	General updates.
2nd Printing	April 1974	Incorporating IPNs 021, 026, 031 and 042.
IPN 045	April 1974	Release 6.1 updates.
IPN 057	July 1974	Errata.
3rd Printing	July 1974	Errata.
IPN 068	August 1974	General updates.
4th Printing	September 1974	Incorporating IPN 068.
IPN 079	December 1974	Release 6.2 updates.
5th Printing July	1975	Incorporating IPN 079.
IPN 093	April 1976	Release 7.0 updates.
6th Printing	April 1976	Incorporating IPN 093.
IPN 135	September 1978	Release 8.0 updates and general revisions.
SPR 174	June 1980	Revisions.
7th Printing	August 1980	Incorporating SPR 174.

PREFACE

Intercomm is a state-of-the-art teleprocessing monitor system of SDA, operating under the control of IBM 360/370 Operating Systems (MFT, MVT, VS). Intercomm monitors the transmission of mes- sages from terminals, concurrent message processing, centralized access to I/O files, and the routine utility operations of editing input mes- sages and formatting output messages, as required.

This document provides a general introduction to and a detailed specification for use of the Intercomm System Utilities. The Edit and Output Utilities provide both terminal and format independence to an application program (message processing subsystem) executing under Intercomm control. The Change/Display Utilities provide predefined transactions for entry at a terminal to perform file inquiry and file maintenance. The utilities are generalized; each installation codes tables using Intercomm macros to specify the precise use of the utility for their applications.

This document is intended for both system and application pro- grammers. The following prerequisite Intercomm publications support the material contained herein:

- Concepts and Facilities
- COBOL Programmers Guide
- PL/1 Programmers Guide
- Assembler Language Programmers Guide

A User Review Form is included at the back of this manual. We welcome recommendations, suggestions and reactions to this or any Intercomm publication.

INTERCOMM PUBLICATIONS

GENERAL INFORMATION MANUALS

Concepts and Facilities

Planning Guide

APPLICATION PROGRAMMERS MANUALS

Assembler Language Programmers Guide

COBOL Programmers Guide

PL/1 Programmers Guide

SYSTEM PROGRAMMERS MANUALS

Basic System Macros

BTAM Terminal Support Guide

Installation Guide

Messages and Codes

Operating Reference Manual

System Control Commands

CUSTOMER INFORMATION MANUALS

Customer Education Course Catalog

Technical Information Bulletins

User Contributed Program Description

FEATURE IMPLEMENTATION MANUALS

Autogen Facility

ASMF Users Guide

DBMS Users Guide

Data Entry Installation Guide

Data Entry Terminal Operators Guide

Dynamic Data Queuing Facility

Dynamic File Allocation

Extended Security System

File Recovery Users Guide

Generalized Front End Facility

Message Mapping Utilities

Multiregion Support Facility

Page Facility

Store/Fetch Facility

SNA Terminal Support Guide

Table Facility

TCAM Support Users Guide

Utilities Users Guide

EXTERNAL FEATURES MANUALS

SNA LU6.2 Support Guide

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

The INTERCOMM System Utilities (EDIT, OUTPUT, CHANGE/DISPLAY) are optional system components provided to simplify application program design and coding logic. The Utilities are table-driven; their actual operation is specified by user-coded tables generated via INTERCOMM macro instructions. A general description of each utility follows:

THE EDIT UTILITY is used to prepare messages from on-line terminals for processing by the individual application programs. It handles the overall processing of the incoming message, (e.g., stripping of T/P related control characters) and, in addition, edits and places the individual parameters in the message into a predefined format. The module is driven by the EDIT Control Table (one table entry related to each independent message to be processed by the module) which defines the characteristics of each input message, the related editing function to be performed, and the format of the corresponding output (edited) message for either COBOL, PL/1, FORTRAN or BAL message processing subsystems.

THE OUTPUT UTILITY is designed to provide simplified generation and revision of output formats to telecommunication devices without impact to the individual application subsystems. Each application program may use the module to generate its terminal display formats or hard copy reports by passing to OUTPUT the data fields that vary from message to message. The OUTPUT module will in turn:

- Select the format to be used based upon table specifications;
- Locate the application variable data to go into the format;
- Format the output message based on an device dependent constraints such as line-ending sequences, and line and buffer size.
- Insert the necessary T/P control characters related to the transmission of the message; and
- Output the actual message(s) to the Telecommunications module for transmission to the designated terminal.

THE DISPLAY and CHANGE UTILITY are intended to allow a remote terminal operator to display an individual file record (for BDAM, ISAM or VSAM files) in a fixed character format on his terminal, and then in turn to modify selected fields within the file record that he had just displayed. The module, similar to EDIT and OUTPUT, is totally table-driven. There is no program modification required for any fixed format file

records. The only item required by the user for the display of record data (binary, hexadecimal or character) is the Format Description Table entries defining the characteristics of the individual file records.

Choice of which Utilities are in use at INTERCOMM execution time is a matter of using INCLUDE control cards for the individual modules in the INTERCOMM link-edit. The EDIT Utility operates as a system service routine and is CALLed whenever EDITing functions are required. CHANGE/DISPLAY and OUTPUT operate as message processing subsystems and hence are scheduled for execution by the INTERCOMM monitor. Terminal input transactions for CHANGE/DISPLAY are queued, scheduled, and processed automatically by the standard INTERCOMM facilities. Application subsystems (and INTERCOMM modules as well) create and queue messages to be processed by the OUTPUT Utility via standard INTERCOMM service routines.

There is one set of operational tables in the production environment. Hence, the INTERCOMM System Manager has the final responsibility for controlling the tables used at execution time. This person (or persons) must ensure the accuracy of all new table entries created by application personnel and may specify which particular table entries may be core or disk resident considering system throughput and response time requirements.

Certain manufacturer's terminals have hardware operating features necessitating special coding conventions for the EDIT and/or OUTPUT Utility Tables. These terminals and the procedures for defining their use with the Utilities are described in a separate section.

All tables for the Utilities are summarized in Appendix A, Appendices B, C, D contain table-oriented macro coding specifications.

GENERAL DESCRIPTION

The EDIT Utility is provided with INTERCOMM to relieve the user's application program from terminal and format dependent considerations. All messages entering the computer from remote terminals have three standard characteristics which make all application programs unnecessarily complex. They are as follows:

1. Teleprocessing control characters are embedded throughout the message.
2. In many cases, the terminal operator need not enter all fields in a message. Application program logic is required to determine which fields were entered in each input message.
3. If the operator enters fields in error, application program logic is required to handle these errors by analyzing them and then either rejecting the total message or accepting the message while omitting processing of the erroneous parameters.

The EDIT Utility provides an interface under which the application programmer need not be greatly aware of these problems.

The EDIT Control Module, in conjunction with standard INTERCOMM supplied subroutines and optional user-supplied EDIT subroutines, provides field-by-field editing of all fields input by the remote terminal operator. The entire input message is edited and formatted as specified by the user in the EDIT Control Table. Errors relating to invalid fields supplied by the terminal operator are diagnosed by the Editing modules and, when appropriate, error messages are sent to the terminal operator.

Every transaction entered from a remote location is identified by a 4-character transaction identifier (verb). Editing of an incoming message is performed when specified by the Verb Definition in the INTERCOMM BTAM Front-End Verb Table or based upon low-value (X'ØØ') in the Verb/Message Identifier (VMI) in the message header. The EDIT Utility is activated when:

- Message processing is to be initiated by a subsystem coded in a high level language. The INTERCOMM language interface module CALLS the EDIT Utility. (A subsystem receives only successfully EDITed messages.)

- A BAL subsystem CALLs the EDIT utility directly. (The BAL subsystem logic defines recovery from unsuccessful EDITing.)

. The INTERCOMM BTAM Front-End CALLs the EDIT Utiliity. (Messages not successfully EDIT'ed are not queued for message processing.)

Messages created by one subsystem and queued for processing by another subsystem may be EDIT'ed if desired by setting the message header VMI to low-value (X'ØØ') and creating message text in a valid format for EDIT.

Figure 1 illustrates the options for input message formats, and EDITed results.

INPUT FORMAT OPTIONS

The EDIT Utility can accept input from a remote terminal in either of four formats called the Keyword Format, the Positional Format, the Positional by line Format, and the Positional Within Keyword Format.

Each mode of input has advantages that the user should weigh in determining which format to use for each transaction input to INTERCOMM subsystems. Note that it is NOT mandatory for subsystems (application programs) executing under INTERCOMM to use the EDIT Utility; this Utility can be completely bypassed for any given transaction type. Any mixture of input modes may be used for different transaction types. For example, assuming an installation that has 5 different transaction types, the possibilities are that: 2 transaction types are not edited at all; 1 transaction type is entered in Keyword Format; 2 transaction types are entered in Positional Format.

In this section, the conventions for input message formats are denoted by:

□ the field separator character defined in the installation's system parameter list (SPA).

△ the new line, carriage return, or carriage return/line feed sequence of the terminal.

○ the End of Message sequence of the terminal (EOT, EOB, ETX, ENTER, etc.).

In general □ and △ are interchangeable.

Keyword Format

When data is to be supplied in the Keyword format, the message is entered in the following manner:

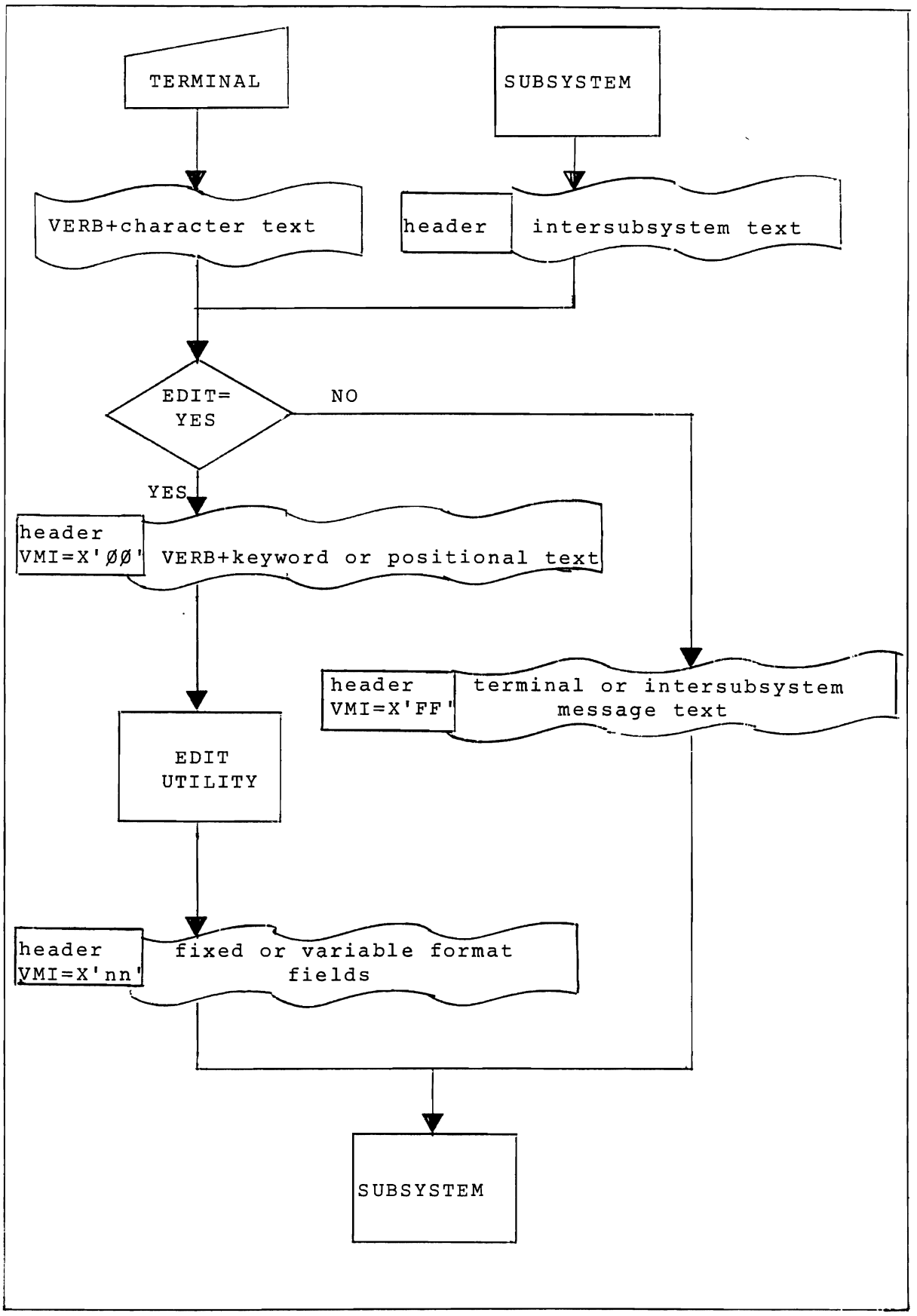


Figure 1. EDIT Format Options

```
TRNS Δ                          (Verb or transaction identifier)
CUS JOHN R. WILLIAMS, JR. Δ     (customer name)
ADR 727 E. 43rd St. Δ           (customer address)
C/S WEST HEMPSTEAD L.I. Δ       (customer address)
ACT 7432710 Δ                   (customer account number)
DBT $27.42 Δ                    (debit amount)
CRD $1.27 Δ                     (credit amount)
END O
```

The four character verb is the first field of each message. Each data element in the message is identified by a unique three character field identification. The field identification is immediately followed by the data for that field. (Any number of separating blanks can be inserted between the field ID and the actual data; these blanks are eliminated by the EDIT Utility.) The field ID (or keyword) must be unique within any one transaction type; it may be reused in other transaction types. For example, the letters CUS may be used in all transaction types that require a customer name to be entered.

For data elements that can be entered more than once in a particular input message, the terminal operator simply enters the given keyword more than once in the message; each use of the keyword is followed by the appropriate data. Thus, in the example above, if it were possible to have three debit amounts, the data entered would be as follows:

```
DBT $27.42 Δ                    (debit amount #1)
DBT $ 7.93 Δ                    (debit amount #2)
DBT $ 8.47 Δ                    (debit amount #3)
```

The terminal operator must be taught the proper keywords to use on input. Any field not applicable on the current entry can be omitted by not entering that keyword (and data) in the message. In addition, the fields may be input in any order.

An additional advantage of the keyword format is the capability of using the CANCEL and CORRECT options. The CANCEL option is intended to give a remote terminal operator the ability to cancel a message that has already been entered if it is determined that something is wrong with the fields entered. (This is particularly useful when using unbuffered devices where each character entered is transmitted immediately.) If an error is realized before the End of Message sequence (O) has been entered, the remote terminal operator can type the following:

```
   :
   :
END △
CANCEL △
END ○
```

The EDIT Utility will then cancel the message when it is called upon to Edit this transaction. (The use of CANCEL is only applicable if the transaction is to be Edited by the EDIT Utility.)

The CORRECT option is used in similar circumstances when the remote terminal operator wishes to correct one or more fields previously entered, rather than cancel the entire message. Suppose that the following message had been typed:

```
TRNS △
CUS JOHN R. WILLIAMS, JR. △
ADR 727 E. 43 ST. △
C/S WEST HEMPSTEAD, L.I. △
ACT 7432710 △
DBT $27.42 △
DBT $ 7.93 △
DBT $ 8.47 △
CRD $1.27 △
END △
CORRECT △
ACT 7432794 △
DBT (2, 9.74) △
END ○
```

The EDIT Utility will reformat this message as if it had been entered in the following form:

```
TRNS △
CUS JOHN R. WILLIAMS △
ADR 727 E. 43 ST. △
C/S WEST HEMPSTEAD L.I. △
ACT 7432794 △                (corrected field)
DBT $27.42 △
DBT $ 9.74 △                 (corrected field)
DBT $ 8.47 △
CRD $ 1.27 △
END ○
```

The first field entered after END and CORRECT was a correction to the customer account number; only the corrected number is recognized by Edit and given to the application program. The second field entered is a correction to the debit amount which in our example is a repetitive field. Thus it is necessary to specify which debit amount is to be corrected. This is indicated by using the following form:

kkk (n,data)

where:

kkk is the keyword to CORRECT;
n, is the occurence number of the keyword;
data is the corrected data to replace the original data entered.

The data entered in the example above corrects the second data field identified by the keyword DBT.

Positional Format

When entering data in the positional format, only data fields are entered; no field identifications are used by the remote terminal operator. The data fields are seperated by the system-wide field separator character (□). The example given in the section describing the keyword format would be entered in the following manner using positional format:

```
TRNS□ JOHN R. WILLIAMS □ 727 E. 43rd St. □ WEST HEMPSTEAD L.I. △
7432791 □ $27.42 □ $1.27 ○
```

As the example illustrates, the transaction identification (verb) is also required in this format. Every possible field for the particular message type must either be supplied by the terminal operator or indicated by the insertion of an extra separator character to indicate the absence of the field.

Blanks may be present within a data field (as in the name entered above); however, the first character of a data field may not be blank.

Positional by Line Format

The operator may also be instructed to enter positional data fields in a predefined line sequence, in which case separator characters are not required for omitted fields trailing at the end of each line. For example:

```
TRNS △
JOHN R. WILLIAMS □ 727 E. 43rd ST. □ WEST HEMPSTEAD, L.I. △
7432710 △                          (account number)
$27.42 △                           (Debit field(s) line)
$ 1.27 ○                           (Credit field(s) line)
```

The number of debit and credit fields is limited by the line width of the terminal.

Positional Within Keyword Input from a Remote Terminal

Positional Within Keyword Terminal Input combines the advantages of Keyword input and Positional input. The field identifications (Keywords) are followed by corresponding positional data items. Within any Positional Within Keyword Line, only data is entered; no individual field identifiers (Keywords) are used by the remote terminal operator. Note, however, that each keyword defines a new line format and all fields defined on a particular Positional Within Keyword Line must be given in the respective sequence within that line.

When data is to be entered by the remote terminal operator in Positional Within Keyword format, the message is entered in the following manner:

```
TRNS △
CUS JOHN R. WILLIAMS□7432710 △     (name, acct #)
ADR 727 E. 43rd ST.□WEST HEMPSTEAD, L.I.△ (address)
INF $27.42□$1.27 ○                  (debit, credit)
```

The four character Verb, as always, is the first data field. In Positional Within Keyword Format, the Keyword Identifier does not have specific fields associated with it. Instead, it is associated with a line of positional data. The keywords and the corresponding lines of positional data may be entered in any order. The data within any line, however, is positional and therefore must be entered in the predefined order for that line.

The data fields within the Positional Within Keyword Line are separated by the system separator character (□). Every possible field within a particular Positional Within Keyword Line must either be supplied by the terminal operator or marked by insertion of an extra separator character to indicate the absence of the field. Any omitted fields at the end of a line do not need separator characters to define their absence. The end of line sequence (△) defines the absence of trailing fields that have been omitted. If, in the above example, in the INF line, it was desired to only enter a credit amount, that line could be entered as follows:

```
INF□$27.42 △
```

Video Display Terminals With Screen Formatting

Certain Video Display Terminals have the facility to display "format" screens for the operator's convenience. For example:

```
: TRNS:
:.........................:  CUSTOMER NAME
:.............................:  ADDRESS
:.....................:  ACCOUNT NUMBER
:.............:  :...............:  :.........:  DEBITS
:.............:  :...............:  :.........:  CREDITS
```

where colons (or some other terminal dependent character) delimit those screen positions where an operator may enter data. Only the data within colons is transmitted as an input message.

In some instances the terminal itself will generate a tab character (or other terminal dependent character) at the end of each "formatted field" entered by the operator. In this instance, the tab character may be defined as the separator character (□) for the installation and the incoming message would correspond to standard Positional input to the EDIT Utility:

```
TRNS□ NAME □ ADDRESS □ DEBIT(S)□ CREDIT(S)○
```

However, special consideration must be given to the hardware operating considerations of each terminal type to ensure that this standard format is maintained.

See the section "Terminal Dependent Considerations" for those devices which the EDIT Utility treats in a special fashion due to their hardware characteristics.

EDIT PROCESSING

Edit Components

The Edit Utility consists of a major EDIT Control routine, specific EDIT Subroutines (INTERCOMM and user-supplied), and

an Edit Control Table (ECT) generated by INTERCOMM macros, and a Pad Character Table. Figure 2 depicts the functional components of the EDIT Utility.

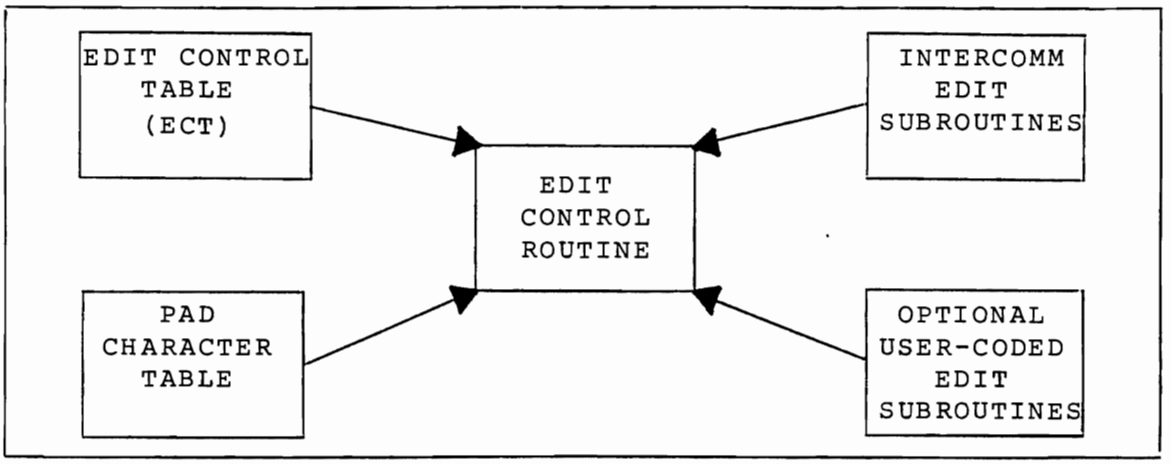


Figure 2. Functional Components of EDIT

The EDIT Control Routine provides total input message analysis, controls the calling of the individual EDIT subroutines based on the actual message and specifications in the EDIT Control Table, and produces the EDITed message.

The individual EDIT Subroutines called by the EDIT routine provide a field-by-field edit of the data items supplied by the remote terminal operator. The interface between the main EDIT Routine and any EDIT Subroutine is a standard CALL statement. The user's individual EDIT Subroutines, coded for special functions, can be written in either Basic Assembler Language (BAL) or any of the higher level languages, (COBOL, FORTRAN, PL/1).

The EDIT Control Table contains one entry for each incoming verb to be edited specifying general characteristics of the message, and detail requirements for field editing. Table entries may be core or disk resident.

The Pad Character Table defines pad (fill) characters to be used when edited fields are less than the length requirement of the EDIT Control Table.

Edit Control Routine Logic

When CALLed, the EDIT Control Routine (hereafter referred to as EDIT) is passed the address of the message input from the terminal. The verb in the message text is utilized to locate the specific EDIT Control Table (ECT) entry. EDIT logic then proceeds as illustrated in figure 3:

1. EDIT acquires a new area of core for the edited result.

2. EDIT copies the input message header to the edited result.
3. The Verb/Message Identifier (VMI) in the message header is set to a one-byte value according to the ECT specification, effectively replacing the VERB.
4. An individual data field is isolated, and EDIT finds the entry in the EDIT Control Table describing the field to be edited. From the table, EDIT determines which EDIT Subroutine is to do the edit checking and/or conversion of the data supplied; the EDIT Subroutine is then called. Upon return from the subroutine, the length of the edited data is compared to the length specified in the ECT for this field. If they are equal, no action is required. If the edited data length is larger, it is truncated on either the left or right as specified by the ECT (if truncation is allowed). If truncation is not allowed, the data item is rejected. If the edited data length is less than the maximum length for this field and the field is always to be given a fixed length (as specified by the ECT), the field is padded with the appropriate pad character from the Pad Character Table based on the EDIT Subroutine used. If the field is numeric, the field is padded on the left; otherwise it is padded on the right. If the field is not fixed length, the data is not padded at all.
5. The next field in the input message is then isolated and is processed as described above. Processing continues until all fields of the input message have been EDITed.

   The EDIT Control Routine strips the following field definition characters during the course of editing:

   - The system separator character (□), as defined in the System Parameter List (SPA);
   - New line characters (Δ);
   - Carriage Return or combined Carriage Return/ Line Feed (Δ);
   - End of Text, End of Message, End of Block, or End of Transmission (○);

   All other device control characters not translated or otherwise suppressed by the front end translation table for a particular device will be treated as text within a field.
6. When all input fields have been processed, EDIT adjusts the length field in the message header to reflect the actual length of the edited result, frees the remainder of storage not used and frees the original incoming message.

EDIT considers its processing successful if no required fields (as defined by the ECT) were omitted or given in error. The EDITed result is returned to the CALLing program as an address

in its original parameter list. Non-required fields not edited successfully are indicated by high-values (X'FF') in the appropriate field location in the EDITed result.

EDIT returns error messages to the originating terminal (via the Output Utility) for each required field omitted or in error. The CALLing program is notified of EDIT's rejection of the incoming message by a zero address in its original parameter list. The storage occupied by the EDITed result is freed. Similarly, if the transaction type (verb) entered is not defined in the EDIT Control Table, a zero returned as the address of the edited message indicates an unsuccessful EDIT.

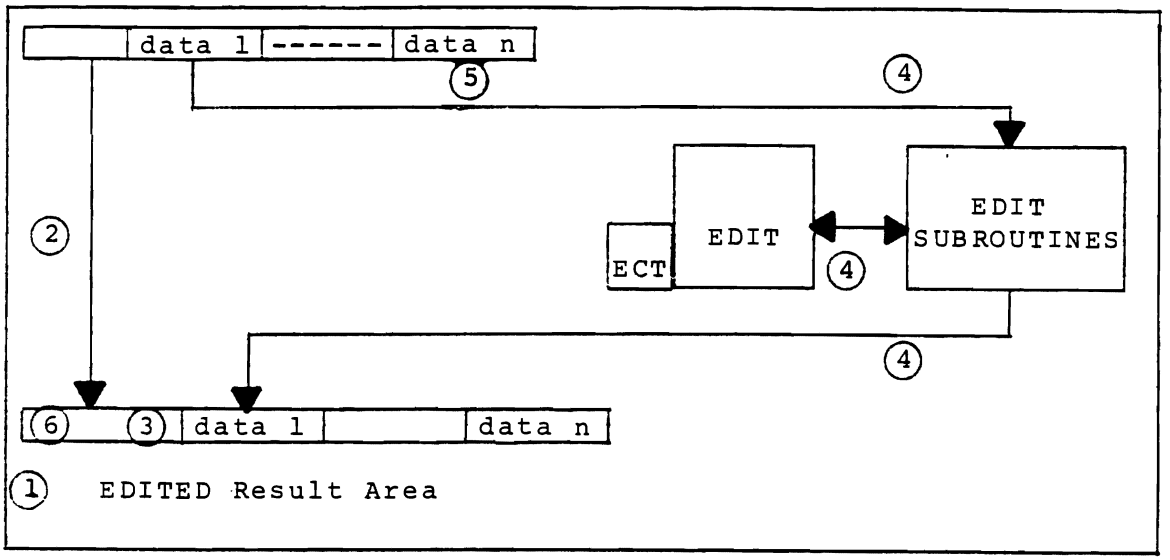


Figure 3. EDIT Control Routine Logic

EDIT Subroutines

The EDIT Subroutines supplied by INTERCOMM provide for basic editing capabilities necessary for any installation. They do numeric checking, convert data to packed decimal, convert data to binary, etc. In addition, there are unique routines for use in INTERCOMM applications. For example, one EDIT Subroutine validates an input field to check for valid terminal identification for the Output Utility.

The EDIT Subroutines are identified by number (nnn), where the subroutine entry point is coded as EDITnnn. EDIT Subroutines 000 thru 020 are reserved for INTERCOMM routines, 021 thru 255 may be utilized as identifiers of user specified EDIT Subroutines.

Each field in the incoming message is edited by one subroutine indicated in the EDIT Control Table entry by number. INTERCOMM-supplied subroutines are described in figure 4. The routine numbers in figure 4 are values to be given on the PARM macro as the EDIT Subroutine number (see Macros Manual). Techniques for coding user-supplied EDIT subroutines are described later in this section.

Routine Number	Description
0	Determines the length of a character field and places actual data into the output area for the field.
1	Determines the length of a character field, checks for valid numeric data, packs the data, and puts the packed data into the output field. Maximum length edited data can be is 15 bytes.
2	Acts like Routine 1, but converts data fields to binary. Maximum length edited data can be is 4 bytes.
3	Acts like Routine 1, for dollar amount fields, stripping dollar sign and decimal point and converting to packed decimal.
4	Reserved for future use.
5	Converts YES or NO fields. YES is converted to a Character 1; NO is converted to Character 0.
6	Validates that a terminal identification given as a character data field is contained in the OUTPUT Utility Station Table.
7	Acts like Routine Ø for numeric character data which is to remain as unpacked data.
8	Acts like Routine 1 for dollar amount fields, stripping dollar sign and decimal point and converting to binary data. (Maximum length edited data can be is 4 bytes.)
9-20	Reserved for future use.

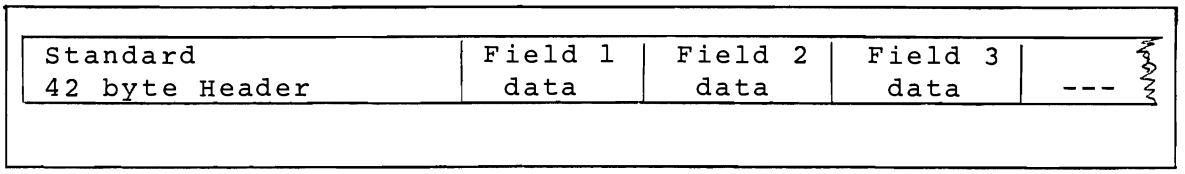
Figure 4. INTERCOMM Edit Subroutines

DATA FORMATS AFTER EDITING

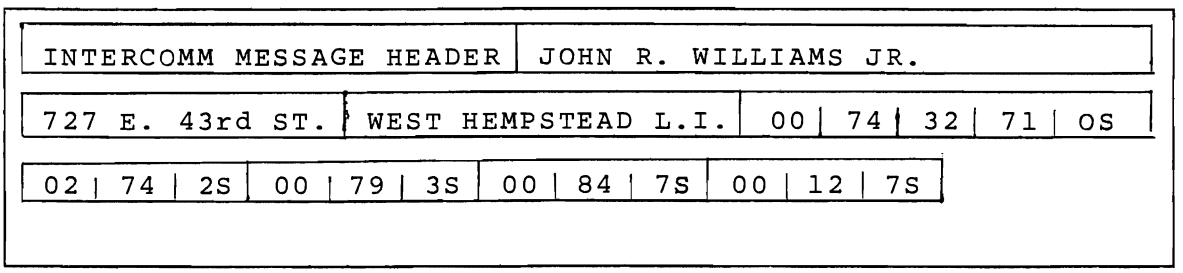
Once all fields of the input message have been edited successfully, a message formatted by the EDIT Control Routine is ready for processing by an INTERCOMM subsystem. The definition of required format results from EDIT processing is contained in the EDIT Control Table entry for each transaction type (verb). Thus, each transaction type can result in only one format. Different transaction types may result in different formats after EDITing.

Fixed Format

When called upon to produce data in a fixed format, the result of EDITing consists of data fields in a predefined sequence with each EDITed field a predefined fixed length. The user specifies the sequence of fields in the EDIT Control Table. Graphically, the message appears as:



The data given in the example earlier in this Section might be presented in the following format to the application program:



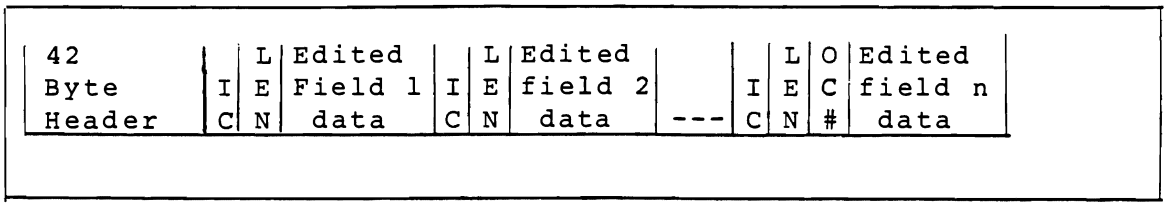
All fields have been formatted and edited as specified in the EDIT Control Table (Notice numeric fields were edited to packed decimal with S representing the sign character). It is important to note that regardless of the form of input supplied by the remote terminal operator, the field lengths and starting positions of each field in the message will be fixed for each EDITed message passed to the application program for processing.

If a data item was not supplied on input from the remote terminal, the pad character defined for the EDIT Subroutine which edited the field is used as a fill character for the entire field. This can be used as a basis for checking if a given field was omitted on input. If the data item was supplied erroneously on input, the field will be filled in with

high values (X'FF' in each byte of the field). This is only meaningful, of course, if the data item was specified as not required since if a required field is given in error the message is rejected by the EDIT Routine.

Variable Format

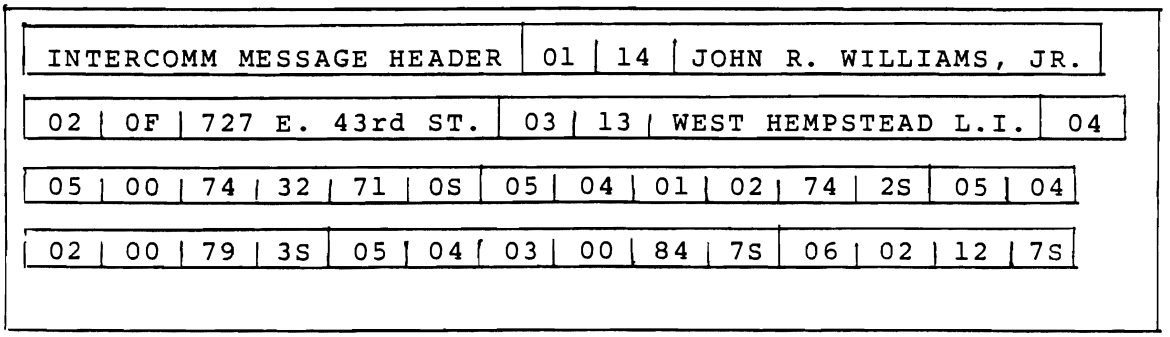
When using the variable format of presenting data to an application program, EDIT selects each field from the incoming message and places it in the message being produced with a 2 or 3-byte prefix based upon ECT specifications as follows:



where

- IC is the Item Code (defined by the ECT) to identify the field.
- LEN is the length of the data field (plus one if OC# exists)
- OC# is the occurence number for repetitive fields.

The data elements in the message produced are in the order entered by the operator. Application program logic is required to scan the message to locate particular data fields. In the example given in Section B, the data would be presented in the following form:



It should be noted that the same item code (05) is used for all of the debit amount fields given, the first time assigning an occurrence number of 01, the second time an occurrence number of 02, and the third time an occurrence of 03. Notice that the length field includes the length of the occurrence number byte for repetitive fields. Note also that in the variable format, the user has the option of always passing

the data with a fixed length, padding where necessary (as was done for the second and third debit amount fields above) or of passing only the data supplied with no unnecessary padding (as was done for the credit amount field).

Variable format result of EDITing allows an application program to access a string of Status Bytes showing which fields appeared in the incoming message. Status Bytes and their format are discussed later in this Section under the topic "Edit Error Analysis."

EDIT TABLES

The EDIT Control Table

The EDIT module is driven by the EDIT Control Table (ECT). This table contains all information about each message necessary to perform editing.

The ECT resides in core, in a separate CSECT labelled VERBTBL. (The address of the EDIT Control Table appears in the System Parameter List.) The ECT contains actual EDITing specifications or optionally may contain pointers to disk-resident entries. In general, frequently processed transaction types (verbs) utilize core-resident entries for fast retrieval of EDIT specifications; transactions with low volume utilize disk-resident entries.

The EDIT Control Table is a variable length table, and is created and maintained by the user. The following INTERCOMM macros are used to create the ECT:

- VERBGEN is a macro which must precede the table entries if any VERB macros contain the RBN= parameter to signify actual table entry is disk resident. (If no ECT entries are to be on disk, VERBGEN is omitted.)

- VERB is the macro defining general editing specifications (the transaction id or actual 4-character verb, input format, edit result format, etc.)

- PARM is the macro defining specific edit requirements for each data field (edit subroutine number, required field, truncation, maximum length, etc.)

- PMIELIN is the macro delimiting positional data fields by line.

- PMISTOP is the macro signifying end of the ECT.

Coding specifications for these macros are described in detail in Appendix B. Figure 5 illustrates the general structure of the EDIT Control Table.

```
VERBTBL         CSECT
                VERBGEN
VRB1            VERB            FIRST VERB
                PARM            FIRST PARAMETER FOR VRB1
                :
                :
                PARM            (N)th PARAMETER FOR VRB1
VRB2            VERB            SECOND VERB
                PARM            FIRST PARAMETER FOR VRB2
                PARM            SECOND PARAMETER
                PMIELIN         END OF POSITIONAL LINE
                :
                PARM            (N)th PARAMETER FOR VRB2
VRB3            VERB            RBN=1    VERB POINTER FOR DISK
                                          RESIDENT ENTRY
                :
                :
                :
VRBN            VERB            (N)th VERB
                :
                PARM            (N)th PARAMETER FOR
                                (N)th VERB
                PMISTOP
                END
```

Figure 5. EDIT Control Table Structure

Each individual table entry consists of one VERB macro and one or more PARM macros. Figure 6 illustrates allowable combinations of formats and VERB macro operand coding required.

INPUT FORMAT	EDITED FORMAT	VERB MACRO OPERANDS		
		FIX=	KEY=	LINE=
Keyword	Fixed	YES	YES	NO
	Variable	NO	YES	NO
Positional	Fixed	YES	NO	NO
	Variable	NO	NO	NO
Positional By-Line	Fixed	YES	NO	YES
	Variable	NO	NO	YES
Positional Within Keyword	Fixed	YES	YES	YES
	Variable	NO	YES	YES

Figure 6. VERB Macro and Message Formats

There is a distinct relationship between the order of the PARM macros and the VERB specifications for input message formats and resultant edited message formats.

The relationship between the PARM macros and message formats is as follows:

- For Keyword Input Format: No correlation exists between the order of the description of the fields in the EDIT Control Table via PARM macros and the order in which the actual data is entered.

- For Positional Input Format: The order in which the fields are entered by the remote terminal operator must correspond exactly to the order in which the fields are described by PARM macros in the EDIT Control Table.

- For Fixed Format EDIT Result: The order in which fields are sequenced in the edited result corresponds to the order of the PARM macros. If a message field appears on the input message more than the maximum number of times, specified by the PARM macro fields, the extra fields are rejected. The application programmer using fixed format editing must know the format of the EDIT Control Table; the position of a data field in the edited message is completely controlled by the coding of the EDIT Control Table when using fixed format.

- For Variable Format EDIT Result: There is no correlation between the order of the PARM macros in the Edit Control Table and the data presented to the application when using variable format. The data fields appear in the order entered in the input message.

Thus, the ordering of the PARM macros in the EDIT Control Table has the following implications:

1. If Keyword Format input is being used, the order of the PARM macros in the table is used only to define the order in which the data is to be arranged for the application program when fixed format results are specified.
2. If Positional Format input is being used, the order of the PARM macros in the table defines both the order in which the data fields are entered on the input device and also the order in which the data is to be arranged for the application program.

3. For positional by Line Input Format, the allocation of fields to a line is defined by the PMIELIN macro (coded after each set of PARM macros which define a line) and hence the order of the PARM macros defines the order in which the operator may enter data.
4. For positional Within Keyword Format: No correlation exists between the order of the description of the parameters in the Edit Control Table and the order in which the keyword lines are entered. However, for the positional data within any given line, the order in which the fields are entered by the remote terminal operator must correspond exactly to the order in which the PARM macros are sequenced for each keyword.

Creating and Maintaining the ECT

All INTERCOMM messages processed by the EDIT Utility share the same EDIT Control Table (both core and disk resident). Hence, maintenance of this table in the production (live) version of INTERCOMM should be the responsibility of the INTERCOMM System Manager.

There is one entry in the ECT for each verb to be edited. Core resident table entries are added to an existing ECT. Disk resident table entries are also contained in the core resident ECT to allow calculation of the actual disk resident entry length. The entire disk resident entry should be coded as though it were to be core resident; and is actually assembled as part of the resident table. After assembly, only the generation of the VERB macro remains in the resident table to specify the location of the disk table entry. The VERB and PARM macros for a disk resident table entry are assembled a second time to create a separate load module to be retrieved from disk when required. The Section "Disk Resident Tables for the Utilities" details procedures for loading disk table entries. For example, an ECT with core and disk resident entries might be:

```
VERBTBL          CSECT
* DISK RESIDENT ENTRIES INCLUDED
                 VERBGEN
* TRNB ECT ENTRY IN CORE
TRNB             VERB        TRNB,3,256,2,KEY=NO
                 PARM        NUM,1,1,8,1ØØØØ111     CUST.NUMBER
                 PARM        AMT,2,1,8,1ØØØØ111     DEBIT AMOUNT
* TRNS ECT ENTRY ON DISK
TRNS             VERB        TRNS,1,256,2,RBN=1
                 PARM        CST,1,Ø,3Ø,1ØØØØ111    CUST.NAME
                 PARM        ADR,2,Ø,3Ø,1ØØØØ111    CUST.ADDR.
* TRNA ECT ENTRY ON DISK
TRNA             VERB        TRNA,2,256,3,KEY=NO,LINE=YES,FIX=NO,
                             RBN=2
                 PARM        CST,1,Ø,3Ø,1ØØØØ111    CUST.NAME
                 PMIELIN                            END OF I/P LINE
                 PARM        ONM,2,1,8,1ØØØØ111     OLD CUST.NUMBER
                 PARM        NNM,3,1,8,1ØØØØ111     NEW CUST.NUMBER
* END OF TABLE
                 PMISTOP
                 END
```

The resident table CSECT name must be VERBTBL. The disk resident entries are each contained in a block (RBN) of the BDAM dataset with ddname VRBØØØ.

The ECT is searched sequentially; frequently accessed entries should appear in the beginning of the table. Since the disk resident table entries are assembled separately (without CSECT, PMISTOP and END macros), they may be maintained as members of symbolic libraries and COPYed into the resident CSECT for assembly; for example

```
VERBTBL          CSECT
                 VERBGEN
* CORE RESIDENT ENTRIES FOLLOW
TRNB             VERB          - - - - - -
                 PARM          - - - - - -
                 PARM          - - - - - -
                  .
                  .
                  .
TRNC             VERB          - - - - - -
                 PARM          - - - - - -
                  .
                  .
* COPY EACH INDIVIDUAL DISK RESIDENT ENTRY
                 COPY           TRNSTBL
                 COPY           TRNATBL
                 PMISTOP
                 END
```

The Pad Character Table

The Pad Character Table supplies the EDIT Control Routine with the character to be used for padding with each EDIT Subroutine.

The Pad Character Table is a resident CSECT named PADDTBLE. Each entry in the table is two bytes long. The first byte contains the number of the EDIT Subroutine with which the pad character is associated. The second byte contains the actual pad character. These entries are coded using the PADD macro or hexadecimal constants. The table must end with a PMISTOP macro.

If no pad character is supplied for a particular EDIT Subroutine a default of blank (X'40'), is assumed. A pad character of X'OC', X'OD', or X'OF' will pad a packed field with binary zeros, but when no data is supplied for the field EDIT will insert a sign in the rightmost byte (i.e., fill the field with a packed zero).

The following entries appear in the pad table as supplied in the INTERCOMM release:

```
         TITLE '*** PMI-INTERCOMM ***   PADD TABLE'
PADDTBLE CSECT
         PADD  0,BLANK          ROUTINE 0 - ALPHANUMERIC
         PADD  1,0F             ROUTINE 1 - PACKED
         PADD  2,BZER           ROUTINE 2 - BINARY
         PADD  3,0F             ROUTINE 3 - PACKED DOLLAR
         PADD  5,BZER           ROUTINE 5 - NO
         PADD  6,BZER           ROUTINE 6 - TPU
         PADD  7,F0             ROUTINE 7 - NUMERIC CHAR
         PADD  8,BZER           ROUTINE 8 - BINARY DOLLAR
         PMISTOP
         END
```

In the following sample Pad Character Table, a pad character of blank has been assigned to EDIT Subroutine 21 and a pad character of asterisk has been assigned to EDIT Subroutine 22.

```
PADDTBLE CSECT
         :          } standard INTERCOMM
         :          }      entries
         PADD          21,BLANK
         PADD          22,5C
         .
         PMISTOP
         END
```

SAMPLE TABLE ENTRIES

The examples in this section are included to assist the user in understanding the table entries specifying different options for input message formats and resulting EDIT data formats.

The examples illustrate:

- Keyword Input, Fixed Format Result (Figure 7)
- Keyword Input, Variable Format Result (Figure 8)
- Positional Input, Variable Format Result (Figure 9)
- Positional by Line Input, Fixed Format Result (Figure 10)
- Positional Within Keyword Input, Fixed Result (Figure 11)
- Verb only Result (Figure 12)

Each example illustrates input from terminal, ECT entry, EDITed result.

Message Input at Terminal:

```
PRCO△
PO# 174321△
PRD 745OR△
QTY 12△
UNT DOZ△
PRD 863PL△
QTY 1ØØ△
AGN X71△
END○
```

Message received by EDIT:

	VMI	
header	ØØ	PRCO△PO# 174321△PRD 745OR△QTY 12△UNT DOZ△

PRD 863PL△QTY 1ØØ△AGN X71△END○

EDIT Control Table entry:

```
PRCO  VERB  PRCO,ØA,256,11,FIX=YES
      PARM  PO#,Ø1,Ø1,Ø15,1ØØØØ111
      PARM  PRD,Ø2,ØØ,ØØ5,1ØØØ1111,REPT=3
      PARM  QTY,Ø3,Ø2,ØØ4,1ØØØ1111,REPT=3
      PARM  UNT,Ø4,ØØ,ØØ3,ØØØØ1111,REPT=3
      PARM  AGN,Ø5,ØØ,ØØ3,1ØØØØ111
```

EDITed result:

	VMI	
header	ØA	ØØØØØØØØØØØØØØØØØØØØØØØ174321F | F7F4F5D8D9

_ _	F8F6F3D7D3	4Ø4Ø4Ø4Ø4Ø	ØØØØØØØC	ØØØØØØ64	ØØØØØØØØ

_ _	C4D6E9	4Ø4Ø4Ø	4Ø4Ø4Ø	E7F7F1

Figure 7. Keyword Input - Fixed Format Result.

Message Input at terminal:

```
PRCO△
PO# 174321△
PRD 745QR△
QTY 12△
UNT DOZ△
PRD 863PL△
QTY 1ØØ△
AGN X71△
END ○
```

Message received by EDIT:

```
header  ØØ | PRCO△PO# 174321△PRD 745QR△QTY 12△UNT DOZ△
PRD 863PL△QTY 1ØØ△AGN X71△END○
```

EDIT Control Table Entry:

```
PRCO  VERB  PRCO,ØA,265,5
PARM  PARM  PO#,Ø1,Ø1,Ø15,1ØØØØØ11
      PARM  PRD,Ø2,ØØ,ØØ5,1ØØØ1111
      PARM  QTY,Ø3,Ø2,ØØ4,1ØØØ1Ø11
      PARM  UNT,Ø4,ØØ,ØØ3,ØØØØ1111
      PARM  AGN,Ø5,ØØ,ØØ3,1ØØØØ111
```

EDITed Result:

```
header  ØA | Ø1 | Ø4 | Ø174321F | Ø2 | Ø6 | Ø1 | F7F4F5D8D9
Ø3 | Ø2 | Ø1 | ØC | Ø4 | Ø4 | Ø1 | C4D6E9 | Ø2 | Ø6 | Ø2 | F8F6F3D7D3
Ø3 | Ø2 | Ø2 | 64 | Ø5 | Ø3 | E7F7F1
```

Figure 8. Key Word Input - Variable Format Result.

Message Input From Terminal:

INVC□1ØKR3□1Ø□15□□50○

Message Received by EDIT:

header ØØ	INVC□1ØKR3□1Ø□15□□50○

EDIT Control Table Entry:

```
INVC  VERB  INVC,ØB,256,7,KEY=NO
      PARM  ITN,Ø1,ØØ,ØØ5,1ØØØØ111
      PARM  ADD,Ø2,Ø2,ØØ4,ØØØØ1Ø11,REPT=3
      PARM  DEL,Ø3,Ø2,ØØ4,ØØØØ1Ø11,REPT=3
```

EDIT Result:

header OB	01	05	F1FØD2D9F3	02	05	01	ØØØØØØØA

02	05	02	ØØØØØØØF	03	05	01	ØØØØØØ32

Figure 9. Positional Input - Variable Format Result.

Message Input from Terminal:

```
INVC□1ØKR3△
1Ø□15△
5○
```

Message Received by EDIT:

```
| header ØØ| INVC□1ØKR3△1Ø□15△5○|
```

EDIT Control Table Entry:

```
INVC  VERB    INVC,ØB,256,7,FIX=YES,KEY=NO,LINE=YES
      PARM    ITN,Ø1,ØØ,ØØ5,1ØØØØ111
      PMIELIN
      PARM    ADD,Ø2,Ø2,ØØ4,ØØØØ1111,REPT=3
      PMIELIN
      PARM    DEL,Ø3,Ø2,ØØ4;ØØØØ1111,REPT=3
```

EDIT Result:

```
| header ØB | F1FØD2D9F3 | ØØØØØØØA | ØØØØØØØF | ØØØØØØØØ |

| ØØØØØØØ5 | ØØØØØØØØ | ØØØØØØØØ |
```

Figure 10. Positional by Line Input - Fixed Format Result.

Message Input from Terminal:

```
        TRNA △
        CUS  PETER JONES□ 10 ALLEN ST□ NY,NY△
        INF  176-424□□ $73.95○
```

Message Received by EDIT:

```
| header ØØ | TRNA△CUS PETER JONES□10 ALLEN ST□ NY,NY △
INF 176-424□□$73.95 ○ |
```

EDIT Control Table Entry:

```
        VERB  TRNA,C1,256,8,KEY=YES,LINE=YES,FIX=YES
        PARM  CUS,ØØ,Ø,Ø,ØØØØØØØØ          (CUS is a keyword)
        PARM  NAM,Ø1,Ø,25,ØØØØØ11Ø
        PARM  STR,Ø2,Ø,2Ø,ØØØØØ11Ø
        PARM  C/S,Ø3,Ø,25,ØØØØØ11Ø
        PMIELIN
        PARM  INF,ØØ,Ø,Ø,ØØØØØØØØ          (INF is a keyword)
        PARM  ACT,Ø4,23,7,ØØØØØ11Ø
        PARM  DEB,Ø5,24,7,ØØØØØ11Ø
        PARM  CDT,Ø6,25,7,ØØØØØ11Ø
```

EDIT Result:

```
| header C1|C'PETER JONESÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞ'|C'10 ALLEN STÞÞÞÞÞÞÞÞÞÞ'|
| C'NY,NYÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞÞ'|F1F7F6FFF4F2F4|ØØØØØØØØØØØØØØ |
| ØØØØØØØØ7395S |
```

Figure 11. Positional Within Keyword Input - Fixed Result.

An input transaction can be defined in the Edit Control Table with the parm parameter of the VERB macro equal to Ø. In this case, the edited message will consist of Header only. The 4-character input verb is indicated by the VMI value in the header. (No data may be entered at the terminal for this type of transaction.)

EDIT Control Table Entry:

```
      VERB  CUST,1,256,Ø,FIX=YES
```

EDIT Result:

```
header  Ø1
        VMI
```

Figure 12. VERB only RESULT.

EDIT ERROR ANALYSIS

Edit analyzes the results of editing by each individual subroutine and takes the following actions:

- REQUIRED Fields

  For errors and omissions of fields defined as REQUIRED in the PARM edit flags, error messages are sent back to the originator indicating the parm-name of the erring or missing field; all remaining fields are checked; non-Assembler subsystems will not receive any message to process.

- NOT REQUIRED Fields-Errors

  For errors in fields defined as NOT REQUIRED:

  1. Error messages are not sent to the originating terminal (unless Edit is reassembled with the global &EDDERS SETBd to 1 in SETGLOBE. Refer to the Operating Reference Manual, Section 8, for details.)

  2. For variable format Edit result, a string of status bytes included in the message text indicates field(s) in error. The field will not appear in the output message. If the field is a repeatable field, the "number of occurrences" in status byte B for the field does not include a count for the erred field.

  3. For fixed format Edit result, the corresponding output field will contain hexadecimals X'FF' (high value).

- NOT REQUIRED Fields-omissions

  Omissions of fields defined as NOT REQUIRED:

  1. For variable format Edit result, the status bytes for the omitted item code will indicate parameter not supplied.

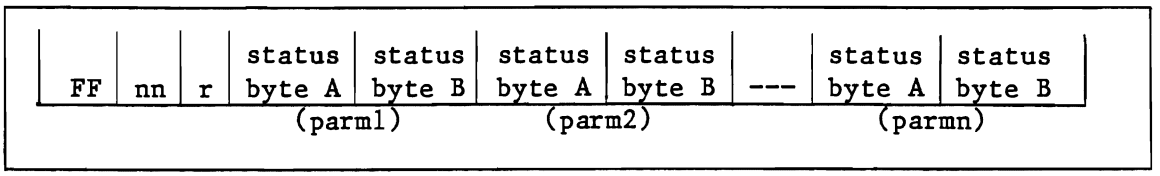
  2. For fixed format Edit result, the field corresponding to the omitted item will contain the specified pad characters. If there is no pad character defined, it will contain blanks (X'40').

Status Bytes (Variable Results only)

In addition to returning the edited message to the CALLing program, Edit returns (as part of the edited message) status indicators for each field which was or could have been supplied for the message when using the variable format result of editing (two bytes per PARM macro). Edit puts the address of the string of status bytes in the third word of the original parameter list passed to Edit. The status bytes physically appear in the edited message text, identified by an Item Code of X'FF'.

There are 2n+1 status bytes where n is the number of different parameters which can be given for the message. The first byte represents a status indicator byte for the entire message (currently not in use). The next 2n bytes represent two bytes for each PARM macro.

The status byte string is formatted as:



where:

FF is the item code indicating status byte string

nn indicates the length of the status bytes string if there are less than 127 parameters. Otherwise, this length is not used

r is the reserved byte

status byte A indicates:

Bit 0 - Parameter supplied
Bit 1 - Reserved
Bit 2 - Reserved
Bit 3 - Unused
Bit 4 - Parameter given but in error
Bit 5 - Unused
Bit 6 - Unused
Bit 7 - Reserved

Status byte B represents the number of times the parameter was given in the message (always one for non-repetitive parameters; otherwise it can be any number).

Standard Error Messages

Error messages reflecting problems encountered during editing of required fields are generated by the Edit Utility and queued for processing by the Output Utility. The messages are formatted according to Output Format Table entries. Each field found in error will cause an error message to be returned to the originating terminal.

Each error message explicitly defines the reason for rejecting the input data, for example:

```
NON-NUMERIC CHARACTER GIVEN ON xxx PARAMETER FOR xxxx VERB.
ALL CHARACTERS SHOULD BE NUMERIC.
MESSAGE NO._____ FROM TPU_____.
```

or:

```
REQUIRED PARAMETER xxx WAS OMITTED (OR GIVEN IN ERROR) ON
THE xxxx VERB.
MESSAGE NO._____ FROM TPU_____.
```

For a precise listing of Edit Utility error messages, see the Intercomm Messages and Codes.

CODING EDIT SUBROUTINES

Each installation can write up to 235 user-edit subroutines, as needed, to perform application-oriented editing of fields coming into the system from remote terminals. These Edit subroutines are CALLed in a standard manner with the parameter list described below. The Edit subroutine can be written in Assembler, COBOL, or PL/1. (A sample edit subroutine coded in COBOL is illustrated below.) Since the subroutine is passed only actual field data, the problem of teleprocessing control characters need not be considered.

The edit subroutines must have entry points named EDITxxx, where xxx may be any number from 021 to 255. It is inportant to note that edit subroutines may be generalized. For example, EDIT021 can be used to edit any number of transactions types by coding 21 as the subroutine number in the appropriate PARM macro of the Edit Control Table (ECT).

Note that the SPA CSECT must be reassembled to reflect the number of Edit subroutines in use (SPALIST macro, EDITRTN parameter). See Section 8 of the Operating Reference Manual for details.

The edit subroutine logic may be used to construct an error message to the input terminal and queue it for the Output Utility. However, in order to build an output message, the edit subroutine must have access to the original input message header. Register 9 on input to the edit subroutine points to a location which contains the address of the input message being processed, and consequently may be used to locate the input message header.

The parameter list passed to an edit subroutine is as follows:

1. Address of the unedited data field
2. Address of the System Parameter list (SPA)
3. Address of a one-word field containing the length of the unedited data
4. Address of the field in which the edited data is to be placed (field is maximum of 255 characters in length)
5. Address of a one-word field to place the length of the edited data
6. Address of the status bytes for this parameter
7. Address of the fullword return code field

The return code field is used to indicate whether or not a field has passed its edit. If the field edit is not successful, the edit subroutine should place a non-zero value (binary) in return code field, and turn on bit 4 of the first status byte. A zero return code indicates successful edit.

The following restrictions apply to edit subroutines coded in COBOL or PL/1:

1. The subroutine may not give up control (no CALLs to COBREENT or PMIPL1).
2. A COBOL-coded subroutine is not a subsystem; hence no "dynamic work space" is available.
3. The subroutine must reside in Intercomm's transient subroutine overlay region.

The following is a sample edit subroutine coded in COBOL:

```
ID DIVISION.
PROGRAM-ID.  EDIT 204.
AUTHOR.  PROGRAMMING METHODS.
DATE-COMPILED.
REMARKS.  INTERCOMM EDIT OF ONE-BYTE FIELD FOR A, C OR D.

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

LINKAGE SECTION.
01  IN-FIELD                          PIC X.
01  SPA                               PIC X.
01  IN-LENGTH                         PIC S9(7) COMP SYNC.
01  OUT-FIELD                         PIC X.
01  OUT-LENGTH                        PIC S9(4) COMP.
01  STATUS-BYTES                      PIC S9(7) COMP.
01  RC                                PIC S9(7) Comp.

PROCEDURE DIVISION
    USING IN-FIELD, SPA, IN-LENGTH, OUT-FIELD,
    OUT-LENGTH, STATUS-BYTES, RC.

    MOVE +0 TO STATUS-BYTES.
    IF  IN-LENGTH NOT = +1
        GO TO ERROR-EXIT.
    IF  IN-FIELD = 'A' OR 'C' OR 'D'
        MOVE IN-FIELD TO OUT-FIELD
      ELSE GO TO ERROR-EXIT.

VALID-EXIT
    MOVE +1 TO OUT-LENGTH
    MOVE +0 to RC.
    GOBACK.

ERROR-EXIT.
    ADD +2048 TO STATUS-BYTES.
    MOVE +1 TO RC.
    GOBACK.
```

Whether or not Editing is required for a message type is indicated in the Front-End Verb Table (BTVERB macro). EDIT is CALLed by INTERCOMM (if required) for high-level language subsystems. A A BAL subsystem CALLs the Edit Utility directly via a standard OS CALL statement. The message text to be edited may be in one of the basic formats:

Positional - data items in a pre-defined sequence
Keyword - data items in any sequence, each data item being prefaced by a 3 character identifier
Positional Within Keyword - positional data items within Keyword line

Editing proceeds field-by-field based upon the user-specified Edit Control Table. Data fields may be edited by INTERCOMM or user-coded Edit Subroutines. The result of processing by EDIT is a message with a standard 42-byte message header and data fields in one of the two basic formats:

Fixed Format: each edited field is of fixed length in a predefined sequence.

Variable Format: each edited field may vary in length and position in the edited result. Each edited field is prefixed with a one-byte identification code, one-byte length, and possibly a one-byte occurrence number for fields defined as repetitive in the ECT.

The EDIT Utility considers a message successfully edited if there are no required fields (as specified by the ECT) in error or omitted. Non-required fields omitted or in error are handled differently depending on the result format that EDIT is creating. Figure 12-1 details EDIT logic for the combinations of cases. In any event, EDIT frees the storage occupied by the unedited messages.

Field Type	Fixed Format Result	Variable Format
Non-Required Field Omitted	Field appears in edited result, filled with pad character associated with Edit Subroutine	Field does not appear in edited result
Non-Required Field in Error	Field appears in edited result filled with X'FF's	Field does not appear in edited result, the status byte A associated with the field is is set to X'Ø8'.

Figure 12-1 EDIT's processing of non-required fields omitted or in error.

In the case of unsuccessful editing, Edit sends error message(s) to the originating terminal for each required field omitted or in error. If none of the required fields are omitted or in error, it remains the responsibility of the application program to analyze the edited result and perform recovery logic for any non-required fields in error. A conversational design approach might be used to control an interactive dialog with the terminal to input new data for fields originally entered in error. Alternatively, the subsystem may return to the Subsystem Controller and the operator is responsible for entering the entire message again.

The Edit Utility is CALLed by an Assembler Language subsystem as follows:

[symbol]	CALL	EDITCTRL,(input-message,spa,0),VL,MF=(E,list)

where:

input-message is the address of the unedited message

spa is the address of the System Parameter List

0 reserves a third word in the parameter list for use by the Edit utility

list is the address of the parameter list for this CALL

On return from the Edit Utility, register 15 contains a return code (binary value) indicating the results of editing as illustrated in Figure 12-2. A zero return code indicates the message was edited successfully. The address of the successfully edited message is returned in the first word of the parameter list. For a non-zero return code, a zero address also indicates the input message was not successfully edited.

Code	Meaning
0	No errors or omissions of required fields
4	Transaction type (verb) not found in Edit Control Table (ECT).
8	Insufficient storage to carry out the Edit function
12	Message cancelled by remote terminal operator using CANCEL option
16	Message cancelled because required parameters were omitted or in error.
20	Unable to find end of keyword or more than 256 item codes

Figure 12-2. Edit Utility Return Codes

Figure 12-3 illustrates BAL program logic for EDITing an input message:

```
        LINKAGE   - - -,MSG=(R5),SPA=(R3),- - -
            .
            .
            .
TEST    CLI       MSGHVMI,X'ØØ'      EDIT REQUIRED?
        BNE       OKAY               NO
EDIT    CALL      EDITCTRL,((R5),(R3),),VL,MF=(E,LIST)
        LTR       R15,R15
        BZ        GOOD
        RTNLINK   - - -,RC=4         UNSUCCESSFUL
GOOD    L         R5,LIST            EDITED MESSAGE ADDRESS
OKAY    EQU       *
            .
            .
            .
```

Figure 12-3 EDITING an Input Message.

GENERAL DESCRIPTION

The OUTPUT utility provides simplified generation (and subsequent revision) of output formats to telecommunication devices; the complications of teleprocessing I/0 coding are transparent to the individual COBOL, FORTRAN, PL/1 or BAL application modules. Each application subsystem can create messages to be processed by the Output Utility containing only data fields to be inserted into a predefined format described by a table entry accessed by the Output Utility. OUTPUT will in turn perform the following:

1. Locate the actual format specification table entry.
2. Decide what data will go where in the format.
3. Format the output message(s) based on any device dependent constraints such as line or buffer size and line ending sequences.
4. Insert the necessary T/P control characters related to the transmission of the message.
5. Put the actual message(s) to the Telecommunications Interface Module (Front-End) for transmission to the designated terminal.

Additionally, Output will verify the operational status for each destination terminal and, if required, select an alternate terminal if specified.

The Output Utility executes as an INTERCOMM subsystem (message processing program) scheduled by the Subsystem Controller. It may be defined as a resident subsystem or as a subsystem residing in an overlay region or dynamically loaded as required. OUTPUT is a reentrant BAL Subsystem; concurrent message processing is defined by its Subsystem Control Table (SCT) entry.

Messages for transmission to terminals are created by application programs and queued for processing via INTERCOMM service routines. Hence, responsibility for terminal output is independent and asynchronous to application program message processing.

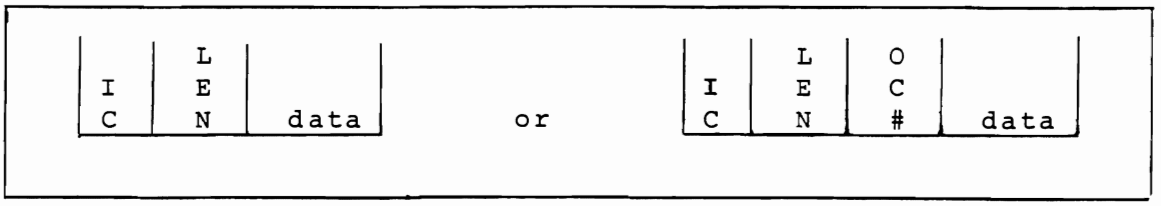
The actual functions of the Output Utility for a particular installation are specified by coding table entries using INTERCOMM macros. Format specification table entries may be core or disk resident at the User's option.

MESSAGE FORMATS PROCESSED BY OUTPUT

Application programs preparing messages to be processed by the OUTPUT Utility have a choice of message text formats as follows:

- Preformatted: The application program prepares a character string of message text ready for transmission to the terminal. Any terminal dependent control characters are the application programmer's responsibility. The OUTPUT Utility will perform Station Table checking to verify the terminal's availability, and assign an alternate terminal if required and defined.

- Variable Character Text Data for Formatting: The application program prepares message text as a series of character data fields for insertion into a format described by an Output Format Table Entry. Each data field in the message text is prefixed by 2 or 3 one-byte binary values as follows:



  where IC is item (or identification) code for the field
  LEN is length
  OC# is occurrence number for repetitive fields.

  The Output Format Table number (OFT#) is the first data item, in a variable message, following the message header. Data item code number 255 is reserved for the OFT# and is expected as a half-word binary value.

- Segmented Messages: The application program prepares and queues a series of messages of the Variable Character Text Data for Formatting type. This extension of the formatting capabilities of OUTPUT allows sequential transmission of several messages in varying formats to a particular terminal. This facility may be used to control uninterrupted transmission of "pages" of a multi-page report to one destination terminal.

- Control Terminal Messages: The OUTPUT Utility may be instructed to prefix particular error messages with a line of identifying information.

Each message type is identified by control fields in the INTERCOMM message header. Application programs may also prepare messages with Fixed Format Text, Formatting Required which are converted by the Change/Display to the Variable Text Format for the OUTPUT Utility. Figure 13 illustrates Format options.

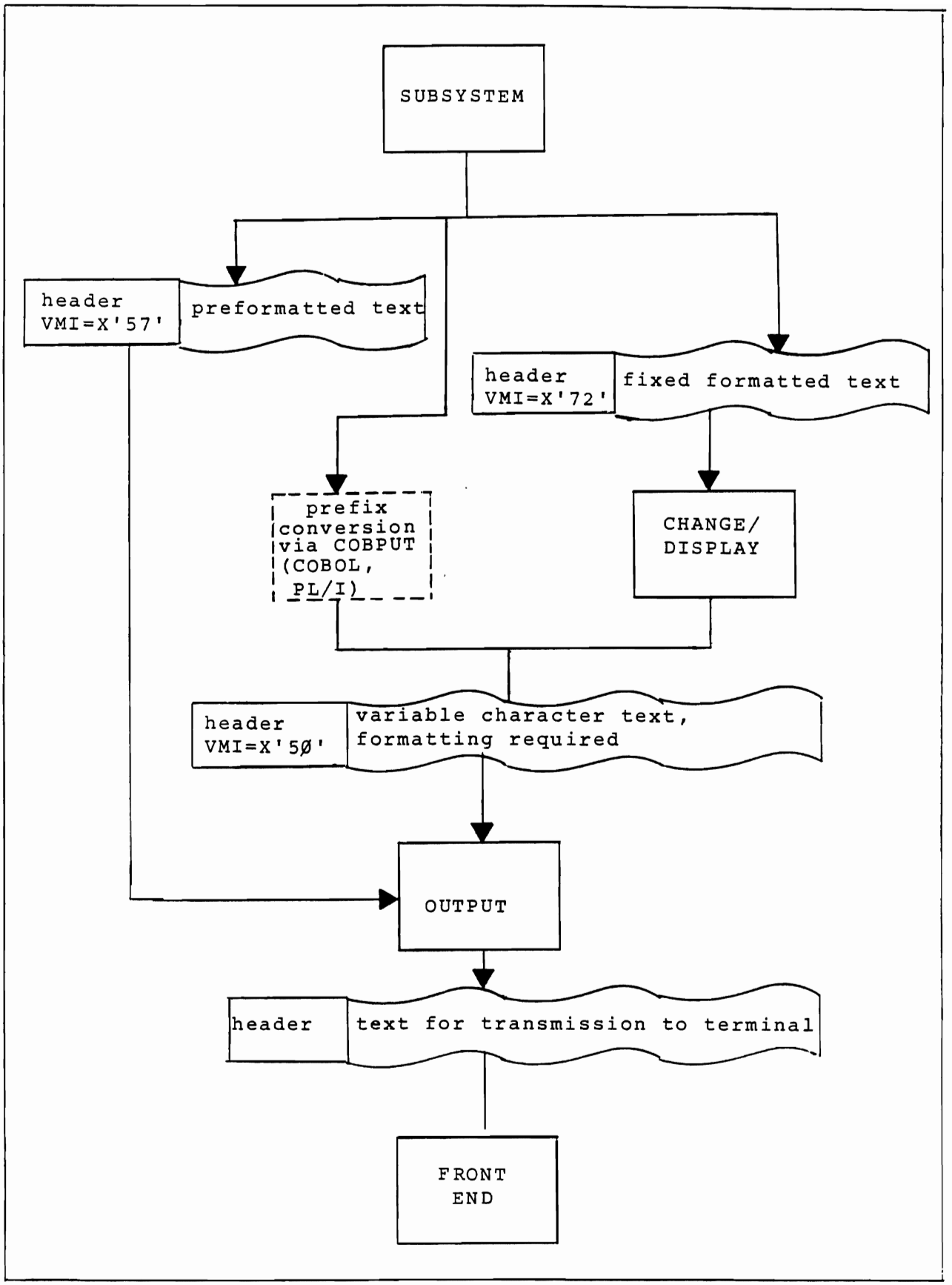


Figure 13. Format Options for messages to OUTPUT.

OUTPUT Components

The OUTPUT Utility is customized for a particular installation by coding the following tables using INTERCOMM macros:

- Output Format Table (OFT): one entry for each format required for conversion of messages from application subsystems to terminal-dependent formats. This Table is also used to define INTERCOMM error messages in generalized formats. Table entries may be core or disk resident depending on frequency of use.

- Station Table: one entry for each terminal. This table is used by OUTPUT to determine status of the device for which a message is intended. The table is used to determine if an alternate device is defined, and to select the appropriate alternate device if the destination terminal is not operational. If alternate devices are not defined, Front-End logic is used to queue messages for non-operational terminals or select a Front-End defined alternate device.

- Device Table: one entry for each terminal type defining hardware characteristics (line length, buffer size, etc.) and transmission control characters required (end-of-line, end of transmission, etc.).

The Output Format Table, Station Table, and Device Table are required, the following tables specify optional facilities of OUTPUT:

- Broadcast Table: defines groups of terminals with one Broadcast group name, allowing a message processed by OUTPUT with destination terminal identified (via MSGHTID in the message header) as a broadcast group to be routed to several different terminals. OUTPUT creates a separate message for each terminal in the Broadcast Group.

- Alternate Format Table: Optionally specifies alternate Output Format Table entries to be utilized for those instances where an alternate terminal is selected as the destination terminal.

- Batch Report Table: Indicates messages created by specific Output Format Table Entries are to be written to a sequential disk data set and subsequently printed off-line by an INTERCOMM batch utility program. This feature is designed for multi-page output reports, and is also helpful in Test Mode for analyzing usability of Output formats before actual terminals are installed.

- Format/Terminal Table: Specifies that particular Output Formats are to be directed to a particular terminal, regardless of the destination terminal specified by the message header.

OUTPUT Subsystem Logic

The OUTPUT Utility provides for simplified generation (and revision) of output formats to telecommunication devices; i.e. terminal and/or format independence for application subsystems. The individual application programmer determines only data necessary in the format and generates that appropriate data in the message text to be processed by OUTPUT. The application program's logic then queues its message for the OUTPUT Utility; OUTPUT executes as an INTERCOMM Subsystem. To process a message, OUTPUT performs the following steps, as illustrated by figure 14.

1. Determine if formatting is required. If not, do steps 4 and 7 only.
2. Locate the Output Format Table specification within message text.
3. Locate the OFT entry (core or disk-resident).
4. Verify the destination terminal (perhaps select alternate device).
5. Obtain storage for the formatted message and prepare the header.
6. Format the output message using both the corresponding Output Format Table and Variable Character Text data supplied in the message passed to the OUTPUT Module from an application program.
7. Pass the message to the Front-End for transmission.

Errors encountered during the processing of messages are reported by messages to the Control Terminal and/or OS Console.

Based on the Output Format Number specified in the message text being processed, OUTPUT searches the Output Format Table for the entry in core corresponding to the requested number. If the table entry is not found in the core-resident table, the Format Number is used as an RBN (relative block number) to access the Disk-resident Output Format Table entries.

The Station Table is then searched comparing the destination terminal identification specified in the message header (MSGHTID) until the correct terminal description is found. Once located, the STATION Table entry indicates the terminal's availability and alternate terminal identification.

- If the destination terminal is available, OUTPUT proceeds.

- If the destination terminal is not available, OUTPUT repeats its logic based upon the alternate terminal.

- If no alternate is specified, OUTPUT proceeds (Front-End logic must store and forward messages for non-operational terminals).

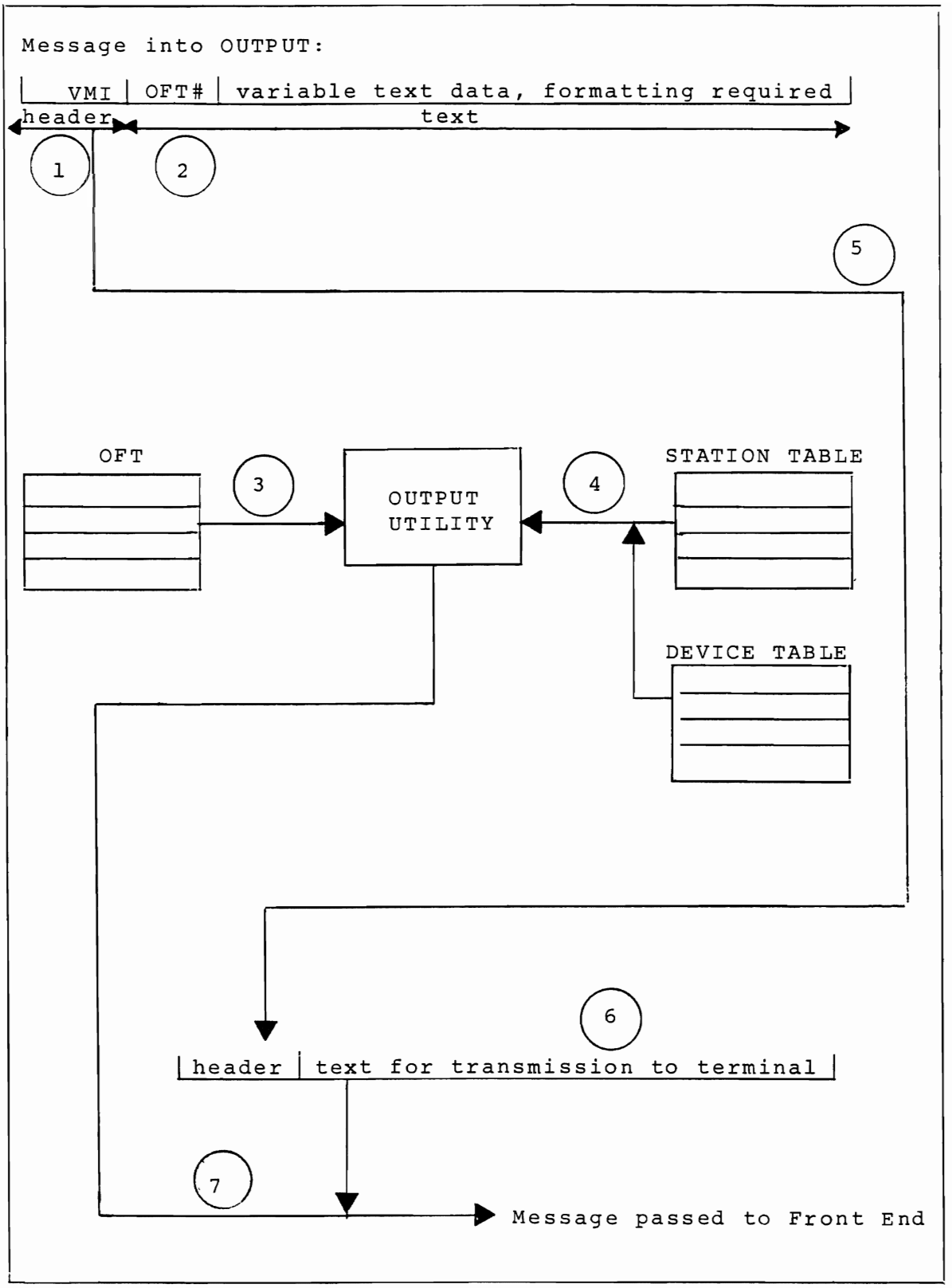


Figure 14. OUTPUT Processing Logic

The Station Table and Output Format Table are then compared to further verify the terminal's ability to receive the message being processed.

Once a terminal is selected, the type of I/O device is obtained from the Station Table and is used to index the Device Table to find the line length for this device. Based on the line length and the estimate of the number of lines for this format, OUTPUT obtains storage for creation of a message for transmission. The OFT entry is then used to produce the required format.

Each line format in the Output Format Table positions data items identified by item code. The program processes the item codes in the OFT one at a time. An item code of 255 indicates that data to be inserted in the line being prepared comes from the OFT itself (i.e., constants, titles, etc.) rather than from the application program's variable character text data. In this case, the data from the table entry is moved into the output line being prepared.

Output processes each specified line format individually. When a given line format is complete, OUTPUT goes on to produce the next line format. A line format may be specified as repetitive, in which case the format is repeated in the message being produced until no further data for that line is found in the incoming message. OUTPUT then proceeds to the next line format.

An item code other than 255 (or item codes reserved for terminal-dependent use), indicates that the data is to be inserted into the format from the message generated by the application program, where each text field is prefixed by Item Code and Length. The indicated item code in the OFT is located in the message developed by the application program, and OUTPUT moves the data into the line being prepared for transmission. If the line format being processed is a repetitive one, OUTPUT searches for Item Code, Length, Occurrence Number prefixes. The program assigns an Occurrence Number of one (1) to the first line prepared. After this line is finished, the same line format is reused, this time with an Occurrence Number of two (2). This process continues until no additional Item Codes are found in the incoming message text for this line format.

If the data found in the incoming message has a length greater than the field size allocated to this item in the line, OUTPUT saves the rest of the data and generates an overflow line (multiple overflows can occur).

The above procedure is followed for every item code in the table for each line. At the end of each line produced for the report, the appropriate control characters (carriage return/line feed or new line) are inserted to force a skip to the next line based on the device type. When all line

formats have been completely processed, OUTPUT passes the complete message to the appropriate Front-End Interface Module.

CONTROL TERMINAL MESSAGES

In order to output error messages to the supervisory control terminal, the individual programs prepare messages for OUTPUT as if they were normal output messages. However, since it is important to transmit these messages very quickly, and it is possible that the queue for the OUTPUT Module might be very large, there is a separate Subsystem Control Table (SCT) entry for messages destined to go to the system supervisory terminal through OUTPUT.

The subsystem code for this portion of the OUTPUT Module is in the System Parameter List in the field names SPAIDCNT. (The INTERCOMM default code is C'N'). This subsystem code should be placed in the MSGHRSC field of the message header. The sending application module should place the control terminal ID into the MSGHTID field of the message header. The message format type should be X'50' (placed in the MSGHVMI field of the message header). Following the header should be the item codes and data for all variable items to be put into the message. In addition, the error message number corresponding to the OFT# is identified by the item code 255 (X'FF') field in the message text.

If the format mask in the Output Format Table indicates that the message being output is an error (i.e., mask of X'8ØØØ), OUTPUT automatically adds a prefix line to the message consisting of the sending subsystem code followed by the format number (converted to character form).

CREATION AND MAINTENANCE OF TABLES USED BY OUTPUT

The following pages describe the seven major tables used by the OUTPUT Module and how to create them. These are:

1. Output Format Table (OFT).
2. Device Table.
3. Station Table.
4. Broadcast Table
5. Alternate Format Table
6. Format/Terminal Table.
7. Batch Report Table.

Output Format Table

The OUTPUT Utility's formatting functions are specified by the Output Format Table (OFT). This table contains all information required to position titles, column headings, special terminal dependent control characters, and to locate

and position variable data fields in a message being processed into a predefined field.

The OFT is core-resident, in a separate CSECT labelled PMIRCNTB. The address of the Output Format Table appears in the System Parameter List. Infrequently used OFT entries may be disk resident and dynamically loaded as required. The Table is an open ended CSECT, and is created and maintained by the user. The end of the table is indicated by four (4) bytes of hexadecimal 'FF' (generated by a PMISTOP macro).

The following INTERCOMM macros are used to create the OFT:

- REPORT: a macro which defines general format specifications (format number, number of line formats, etc.)

- LINE: a macro which signifies generation of a format line and its characteristics (i.e., repetitive or not, constants for page overflow, etc.)

- ITEM: a macro which defines constants or variable data item codes indicating the message being processed will be searched for data to position within the output format. ITEM specifies exact positioning details for each data field. Items referenced in the OFT but not appearing in the message processed will not appear in output. Items appearing in the message but not referenced in the OFT will not appear in output.

Coding specifications for all macros are described in detail in Appendix C. Figure 15 illustrates general structure of the Output Format Table.

```
PMIRCNTB CSECT
REP1     REPORT  . . .
         LINE    . . .
         ITEM    . . .
         ITEM    . . .
          .
          .
         LINE    . . .
         ITEM    . . .
          .
          .
REPn     REPORT  . . .
          .
          .
         PMISTOP
         END
```

Figure 15. OUTPUT Format Table Structure.

To add a new report to the table, just code the header and detail information for the new report (see REPORT, LINE, and ITEM macros) and insert the report in between the last report presently in the Table and the PMISTOP macro. Then reassemble and relinkedit and the new report is part of INTERCOMM. Procedures for loading table entries on disk are discussed in a subsequent section of this manual.

The OFT entry itself may be used to insert non-standard TP Control characters; however, this procedure may cause some OFT entries to be terminal dependent and should be used with caution. If TP control characters are to be included in an OFT entry for a buffered device, and if those control characters are not to count as part of the buffer length, they must be assigned an item code of 254 in the appropriate ITEM macro.

Certain terminals have been assigned exclusive use of other item codes; see the section "Terminal-Dependent Considerations".

Device Table

The Device Table is a core-resident CSECT named PMIDEVTB which specifies terminal dependent characteristics for each device in use at an installation. The Device Table is a system-oriented table, and as such is described in the Operating Reference Manual.

Station Table

The Station Table is a core-resident CSECT named PMISTATB which describes each terminal in the installation network. The terminal "logical name" (5 characters) is specified in this table, as well as an optional alternate terminal. The Station Table is a system-oriented table, and as such is described in the Operating Reference Manual.

The Broadcast Table

The Broadcast Table is a core-resident CSECT named BROADCST. Each entry in the table defines a group of terminals which can be referenced by a single broadcast group name (5 characters). The Broadcast Table is a system-oriented table, and as such is described in the Operating Reference Manual.

Alternate Format Table

The Alternate Format Table is core resident in a CSECT named PMIALTRP. The table is created and maintained by the user. Individual entries in the Table are created by the use of the PMIALTRN macro. The end of the Table must be indicated by the PMISTOP macro. The location in core of the PMIALTRP CSECT is pointed to by a V-type constant in the OUTPUT Module.

Station Table

The Station Table is core resident in a CSECT named PMISTATB. The Table is created and maintained by the user. Individual entries in the Table are created by use of the STATION macro (one for each terminal location). The end of the Table must be indicated by 4 bytes of hexadecimal 'FF', generated by the PMISTOP macro. The location in core of the PMISTAB CSECT is pointed to be a V-type address constant in the field SPASTATB of the System Parameter List.

The Station Table effectively creates 5-character "logical" names for each terminal in the system. The STATION Table Structure is as follows:

```
PMISTATB    CSECT
            STATION TPUS=n,....
            STATION .....
            STATION
            STATION .....
            STATION ....
              .
              .
              .
            PMISTOP
            END
```

Note that the number of station macros is defined by the first STATION macro, which generates a prefix to the table entries. In order to add a new terminal to the system, the Station Table must be modified by adding a STATION macro entry, and updating the first STATION macro to reflect the number of STATION macros in the table.

Broadcast Table

The Broadcast Table is core resident in a CSECT named BROADCST. The Table is created and maintained by the user. Each entry in the Broadcast Table represents one broadcast group. The end of this Tabel must be indicated by four bytes of hexadecimal 'FF', generated by the PMISTOP macro. The location in core of the BROADCST CSECT is pointed to by a V-type address constant in the OUTPUT Module.

The Broadcast Table is defined by the BCGROUP macro. The Broadcast group name (5 bytes) is followed by a specification of the terminals within the group. A message destined for a broadcast group (MSGHTID in the header) will cause a message to be passed to the Front-End for each terminal in the group.

In the following sample Broadcast Table, two broadcast-groups are defined.

```
BROADCST            CSECT
*BROADCAST GROUP GGGØ1
        BCGROUP     GROUP=GGGØ1,
                    TERMS=(NYCØ1,NYC02)

*BROADCAST GROUP GGGØ2
        BCGROUP     GROUP=GGGØ2,
                    TERMS=(LAXØ1,LAXØ2,LAXØ3)
            .
            .
            .
            .
            .
            .
            .
            .

        PMISTOP
        END
```

To add a new terminal to a broadcast group, simply insert the terminal-ID within the current group (preferably behind a terminal of the same device type). To add a new broadcast group to the Table, simply insert the new group behind the last group and prior to the PMISTOP macro. In either case, the CSECT must be reassembled and relinkedited.

Alternate Format Table

The Alternate Format Table is core resident in a CSECT named PMIALTRP. The table is created and maintained by the user. Individual entries in the Table are created by the use of the PMIALTRN macro. The end of the Table must be indicated by the PMISTOP macro. The location in core of the PMIALTRP CSECT is pointed to by a V-type address constant in the OUTPUT Module.

The following example shows how to code an Alternate Format Table:

```
*     Macro        Primary OFT, device type, (alternate OFT,
*                  device type,...)
ALT1  PMIALTRN     1ØØ,1,(1Ø1,2)
ALT2  PMIALTRN     2ØØ,2,(2Ø1,1,2Ø2,3,2Ø3,5)
ALT3  PMIALTRN     3ØØ,4,(3Ø1,2,3Ø2,1)
      PMISTOP
      END
```

The CSECT statement is generated automatically by the first PMIALTRN macro. To add a new entry to the table, code the PMIALTRN macro and insert it after the last entry in the table. Then reassemble and relinkedit the CSECT. The OUTPUT module then provides the ability to create different format layouts for the same message going to different device types through the use of the Alternate Format Table. The OFT number specified by the user application program in the message is the primary format number. If the terminal which is to receive the message is a different device type than the main device type as specified in the Alternate Format Table, then OUTPUT will use the table to find an alternate format which is suitable for the terminal. The output message will be formatted according to the appropriate OFT entry.

The Alternate Format Table is optional. Any combination of alternate formats is acceptable. If no Alternat Format Table is included, or if there is no entry for the specified format number, or no detail entry for the terminal device type, the primary OFT entry specified in the message being processed is used.

Format/Terminal Table

The Format/Terminal Table is used to associate a particular format with a given terminal only. Any OFT included in the table will generate a message to be sent to the terminal associated with it in the table, regardless of the destination terminal specified in the message header.

This table is constructed as a CSECT named PMIRPTAB. The CSECT is coded with BAL constants, and must end with a PMISTOP macro.

The following code outlines the construction of a Format/Terminal Table:

```
 PMIRPTAB     CSECT
* REQUIRED PREFIX CONSTANTS
              DC       X'ØØØ1'
              DC       AL2(ENDTAB-BEGTAB)
 BEGTAB       EQU      *
* OFT51 TO    NYC02    ONLY
              DC       HL2'51'
              DC       C'NYC02'
* OFT52 TO    NYC03    ONLY
              DC       HL2'52'
              DC       C'NYC03'
              .
              .
              .
 ENDTAB       PMISTOP
              END
```

Batch Report Table

Any report number entered in the Batch Report Table will be intercepted by the OUTPUT Module and written to a local sequential data set for off-line printing. This sequential data set must be given a ddname of RPTØØØ and must be defined appropriately to the system (i.e., a DD card must be in the job stream) as a variable length (blocked or unblocked) output sequential data set.

The table consists of a string of 2 byte OFT numbers. These are placed in a CSECT name REPTAPE. The module must end with a PMISTOP macro.

The following code outlines the construction of this table:

```
 REPTAPE      CSECT
              DC              H'51'
              DC              H'75'
              DC              H'93'
              .
              .
              .
              PMISTOP
              END
```

The INTERCOMM Utility, PRT1403, described in the Operating Reference Manual, may be used to print the sequential dataset produced.

SAMPLE TABLE ENTRIES

The examples in this section are included to assist the user in understanding the OFT entry options for Output Utility processing.

The examples are:

- Full messages, formatting only non-repetitive lines (Figure 16)
- Full message, formatting with repetitive lines (Figure 17)
- Segmented message formatting (Figure 18)

Each example illustrates input to the Output Utility, OFT entry, OUTPUT formatted result.

Message from Application Program:

```
| header,5Ø | FF | Ø2 | ØØ,34 | ØA,Ø4,(1Ø) | 14,Ø5,(2Ø) | 1E,ØF,(3Ø) |

| 28,Ø5,(4Ø) | 32,Ø2,(5Ø) |
```

Output Format Table Entry:

```
OFT#52   REPORT   NUM=52,LINES=7
         LINE     NUM=1,ITEMS=1
         ITEM     CCDE=255,FROM=5,TO=26,DATA='PARTS INVENTORY
                  REPORT'
         LINE     NUM=2,ITEMS=Ø
         LINE     NUM=3,ITEMS=4
         ITEM     CODE=255,FROM=1,TO=12,DATA='PART NUMBER:'
         ITEM     CODE=1Ø,FROM=15,TO=18
         ITEM     CODE=255,FROM=2Ø,TO=29,DATA='INVENTORY'
         ITEM     CODE=2Ø,FROM=31,TO=35
         LINE     NUM=4,ITEMS=Ø
         LINE     NUM=5,ITEMS=4
         ITEM     CODE=255,FROM=1,TO=12,DATA='DESCRIPTION:'
         ITEM     CODE=3Ø,FROM=15,TO=29
         ITEM     CODE=255,FROM=31,TO=35,DATA='COST:'
         ITEM     CODE=4Ø,FROM=37,TO=41
         LINE     NUM=6,ITEMS=Ø
         LINE     NUM=7,ITEMS=2
         ITEM     CODE=255,FROM=1,TO=15,DATA='RE-ORDER POINT:'
         ITEM     CODE=5Ø,FROM=17,TO=18
```

Result of Output Formatting:

```
                PARTS INVENTORY REPORT

PART NUMBER: (10)        INVENTORY: (20)

DESCRIPTION: (30)        COST:      (40)

RE-ORDER POINT: (50)
```

(n) indicates data item filled in from text of message processed by output.

Figure 16. Full Message Formatting with non-repetitive Lines.

Message from Application Program:

```
| header,5Ø| FF| Ø2| ØØ,33| 32,ØA,(50)| 37,Ø8,(55)| 1E,Ø8,Ø1,(30) |
| 14,ØD,Ø1,(20)| 5D,Ø8,Ø1,(93)| 1E,Ø8,Ø2,(30)| 14,ØD,Ø2,(20) |
| 5D,Ø8,Ø2,(93)| Ø7,ØC,(7)| Ø8,Ø7,(8) |
```

Output Format Table Entry:

```
OFTØ51   REPORT   NUM=51,LINES=9
         LINE     NUM=1,ITEMS=1
         ITEM     CODE=255,FROM=1Ø,TO=21,DATA='SALES REPORT'
         LINE     NUM=2,ITEMS=Ø
         LINE     NUM=3,ITEMS=4,REPET=6
         ITEM     CODE=255,FROM=1,TO=8,DATA='DIVISION'
         ITEM     CODE=5Ø,FROM=1Ø,TO=19
         ITEM     CODE=255,FROM=22,TO=25,DATA='DATE'
         ITEM     CODE=55,FROM=27,TO=33
         LINE     NUM=4,ITEMS=Ø
         LINE     NUM=5,ITEMS=3
         ITEM     CODE=255,FROM=1,TO=1Ø,DATA='SALESMAN''S  NAME'
         ITEM     CODE=255,FROM=14,TO=25,
                  DATA='TOTAL DOLLAR   SALES'
         ITEM     CODE=255,FROM=28,TO=33,DATA='PCT OFQUOTA'
         LINE     NUM=6,ITEMS=Ø
         LINE     NUM=7,ITEMS=3,REPET=8
         ITEM     CODE=3Ø,FROM=1,TO=1Ø
         ITEM     CODE=2Ø,FROM=14,TO=25
         ITEM     CODE=93,FROM=28,TO=34
         LINE     NUM=8,ITEMS=Ø
         LINE     NUM=9,ITEMS=3,REPET=6
         ITEM     CODE=255,FROM=1,TO=6,DATA='TOTALS'
         ITEM     CODE=7,FROM=14,TO=25
         ITEM     CODE=8,FROM=28,TO=34
```

Result of Output Formatting:

```
                    SALES REPORT

DIVISION ____(50)____   DATE ____(55)____

SALESMAN'S NAME    TOTAL DOLLAR SALES    PCT OF QUOTA

____(30)____       ____(20)____          ____(93)____
___________        ___________           ___________
___________        ___________           ___________

TOTALS             ____(7)____           ____(8)____
```

(n) indicates data item filled in from text of message processed by output.

Figure 17. Full Message Formatting with Repetitive Lines.

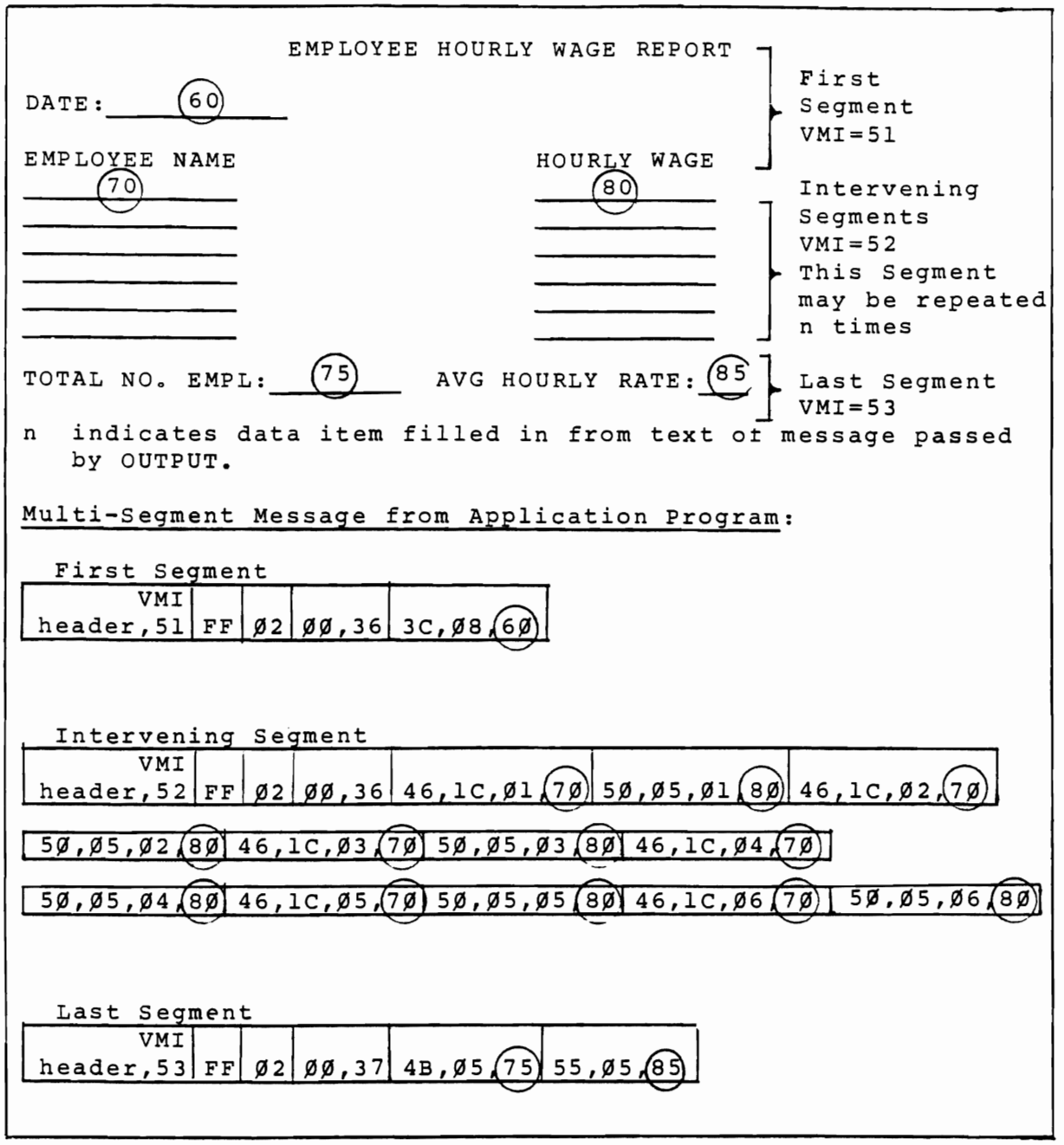


Figure 18. Full Message Formatting - Multi-Segmented Message. (part 1 of 2)

Output Format Table Entry:

```
OFT#54   REPORT   NUM=54,LINES=8
         LINE     NUM=1,ITEMS=1,REPET=5
         ITEM     CODE=255,FROM=5,TO=31,DATA='EMPLOYEE HOURLY
                  WAGE REPORT'
         LINE     NUM=2,ITEMS=Ø,REPET=5
         LINE     NUM=3,ITEMS=2,REPET=5
         ITEM     CODE=255,FROM=1,TO=4,DATA='DATE'
         ITEM     CODE=6Ø,FROM=6,TO=13
         LINE     NUM=4,ITEMS=Ø,REPET=5
         LINE     NUM=5,ITEMS=2,REPET=5
         ITEM     CODE=255,FROM=3,TO=15,DATA='EMPLOYEE NAME'
         ITEM     CODE=255,FROM=5Ø,TO=61,DATA='HOURLY WAGE'
         LINE     NUM=6,ITEMS=Ø,REPET=5
         LINE     NUM=7,ITEMS=2,REPET=1
         ITEM     CODE=7Ø,FROM=3,TO=27
         ITEM     CODE=8Ø,FROM=55,TO=58
         LINE     NUM=8,ITEMS=Ø,REPET=1
-------------------------------------------------------------------
OFT#55   REPORT   NUM=55,LINES=1
         LINE     NUM=1,ITEMS=4,REPET=6
         ITEM     CODE=255,FROM=1,TO=14,DATA='TOTAL NO. EMPL'
         ITEM     CODE=75,FROM=16,TO=2Ø
         ITEM     CODE=255,FROM=5Ø,TO=64,DATA='AVG HOURLY RATE'
         ITEM     CODE=85,FROM=66,TO=7Ø
```

Figure 18. Full Message Formatting - Multi-Segmented Message. (part 2 of 2)

ERROR MESSAGES FROM OUTPUT

Error messages reflecting problems encountered during message processing by the Output Utility are generated and queued for subsequent processing via the Output Utility. The messages are formatted according to OFT entries. Each error message is prefixed with identifying information:

SEQNO	(Sequence Number of message in error)
SSC	(Sending Subsystem Code)
RSC	(Receiving Subsystem Code)
TID	(Destination Terminal of message in error)

Each error message explicitly defines the reason for rejecting the message being processed, for example:

"THE FROM IS GREATER THAN THE TO FIELD"

"REPORT NUMBER NOT IN MESSAGE"

"RCTnnnn NOT FOUND" (OFT entry missing for nnnnn)

See Messages and Codes for a precise listing of Output Utility error messages.

OUTPUT USER EXIT

An optional user-coded exit, USROTEDT, is available in PMIOUTPT. If coded, control is passed to USROTEDT after a message has been queued for output. Options and coding techniques are documented in Section 8 of the Operating Reference Manual.

The OUTPUT Utility subsystem processes messages destined for terminals operating under control of INTERCOMM. It is responsible for completing any device-dependent formatting requirements in a message before passing it to the TP Interface for eventual transmission to the terminal device. It also checks the operational status of destination terminals. Should it find a destination terminal not operational, it will redirect messages to an alternate terminal, if an alternate terminal has been named in the Station Table entry for that particular destination terminal; otherwise the front-end will intercept a message to a non-operational terminal and queue it in the output queue assigned to that terminal to await its availability.

The OUTPUT Utility also may perform report and display formatting based upon specifications in the Output Format Table. This includes columnizing, titling, and positioning variable data. A message sent from an application subsystem to the OUTPUT Utility need consist of variable data characters only. Constants and spacing characters are generated from the table specifications and therefore are not necessary within the message itself.

The OUTPUT Utility will also facilitate the preparation of reports containing several different message formats, for example, a title page, detail pages, and a summary page. In this instance, an application subsystem creates multiple message segments where each segment may require formatting with a different Output Format Table entry. These message segments are sent one by one to the OUTPUT Utility, and it in turn will assure that all message segments are transmitted in the proper sequence and that no other messages are sent to the terminal until after the final segment has been sent. Multi-segment messages are discussed in a separate section of this chapter.

To use the OUTPUT Utility, a subsystem creates a message header and text and directs the message to the OUTPUT Utility. This is accomplished in a COBOL or PL/I subsystem by CALLing the INTERCOMM system program COBPUT, in a BAL subsystem by CALLing the INTERCOMM system program MSGCOL.

OUTPUT MESSAGE HEADER

The message header may be created by copying the input message header to the output message header area and adjusting the fields shown in Figure 18-1.

Field Name	Content
MSGHLEN	Message length: 42 plus the number of bytes of output message text.
MSGHRSCH, MSGHRSC	Receiving subsystem codes: The appropriate subsystem code (see figure 31) depending on the format of the message text.
MSGHSSCH, MSGHSSC	Sending subsystem codes: The code of the application subsystem (found in the receiving subsystem code fields of the input header).
MSGHTID	Terminal Identification: The 5-character name of the destination terminal. This field contains the name of the terminal originating the input message and need only be changed when sending a message to a different terminal, or to a broadcast group.
MSGHVMI	Verb/message identifier: The indicator defining the format of the message text (see figure 31).

Figure 18-1. Message header fields used by the OUTPUT Utility.

MESSAGE TEXT FORMATS FOR OUTPUT UTILITY

The OUTPUT Utility processes various output message formats in various ways as distinguishable from fields in the message header. The simplest is the preformatted message which requires no format processing. Formattable message types include two forms of variable field messages, those using character item and length codes (COBOL and PL/1) and those using binary codes (Assembler, COBOL, PL/1). There is also a fixed field formatting capability that uses the CHANGE/DISPLAY Utility.

Except for preformatted messages, all other formats can be either single-segment messages or multi-segment message groups. Figure 18-2 shows the message header fields describing the various message formats for single segment messages. Multi-segment messages are discussed separately (see Figure 18-13).

OUTPUT MESSAGE TEXT FORMAT	MESSAGE HEADER FIELDS*			CHANGE/ DISPLAY TEXT PREFIX BYTE 9
	MSGHRSCH	MSGHRSC	MSGHVMI	
Preformatted (Device-Dependent)	X'ØØ'	C'U'	X'57' or C'P'	N/A
Formatting Required, Variable Text - Character Format (COBOL and PL/1 only)	X'ØØ' or C'Ø'	C'U'	X'5Ø' or C'Ø'	N/A
Formatting Required, Variable Text - Binary Format (COBOL and PL/1)	C'U'	C'U'	X'5Ø'	N/A
Formatting Required, Variable Text - Binary Format (Assembler Language)	X'ØØ'	C'U'	X'5Ø'	N/A
Formatting Required, Fixed Text	X'ØØ'	C'H'	X'72' or C'S'	C'Ø'

*Note that whenever a character value is allowed for MSGHRSCH or MSGHVMI, it is the high-level language queuing routine COBPUT which performs conversion of the character value(s) to the hexadecimal value(s) expected by the Output Utility. Therefore, assembler subsystems must always use the hexadecimal value(s).

Figure 18-2. Message Header Specifications - Single Segment Messages.

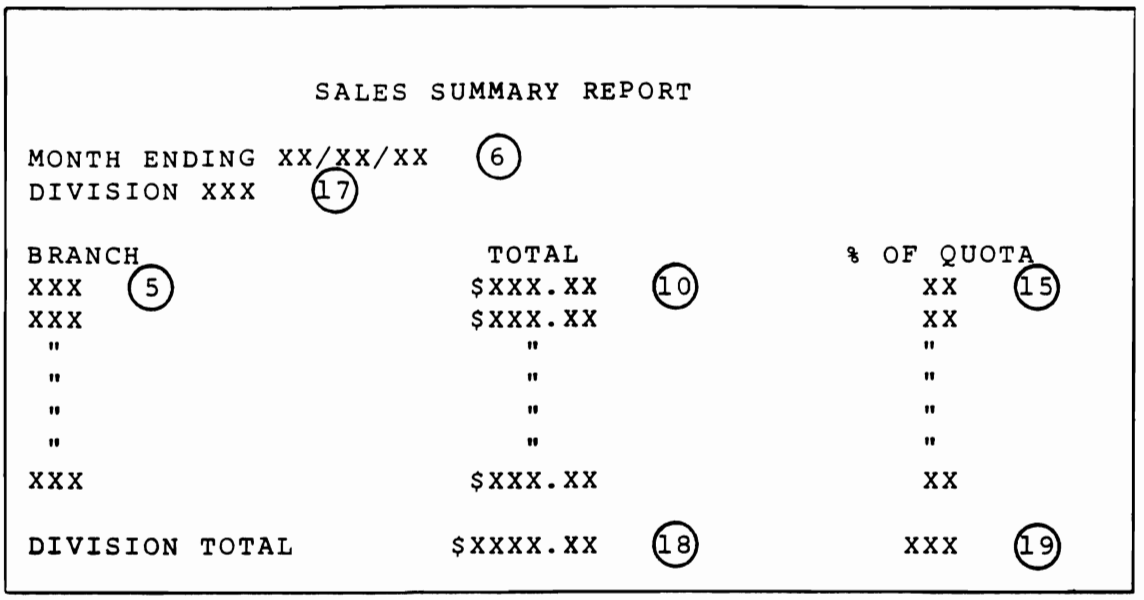


Figure 18-3. Sample display layout, showing typical item code numbers of variable data items.

BUILDING MESSAGE TEXT

Figure 18-3 depicts a sample screen or report layout showing the item code numbers assigned to the variable data fields. This layout corresponds to the sample format table and the sample output messages used in illustrating the various message types.

Preformatted Text

The message text consists of both text and device control characters ready for transmission to the terminal. All spacing and other format considerations (titles, column headings, etc.) are included in the message text. The OUTPUT Utility simply passes the message to the TP Interface. Figure 18-4 shows a sample pre-formatted message to generate a display or print-out conforming to the layout depicted in Figure 18-3.

Formatting Required, Variable Text

The message text consists of a string of data items to be inserted into a final message structure defined in a numbered Output Format Table (OFT). ITEM entries in the designated table define the position and content of all titles, headings, footings, etc., and the positioning specifications for text fields contained in the message. Figure 18-5 depicts the associated Output Format Table Entry where the REPORT, LINE and ITEM macros describe the display format and assigned data item code numbers shown in Figure 18-3.

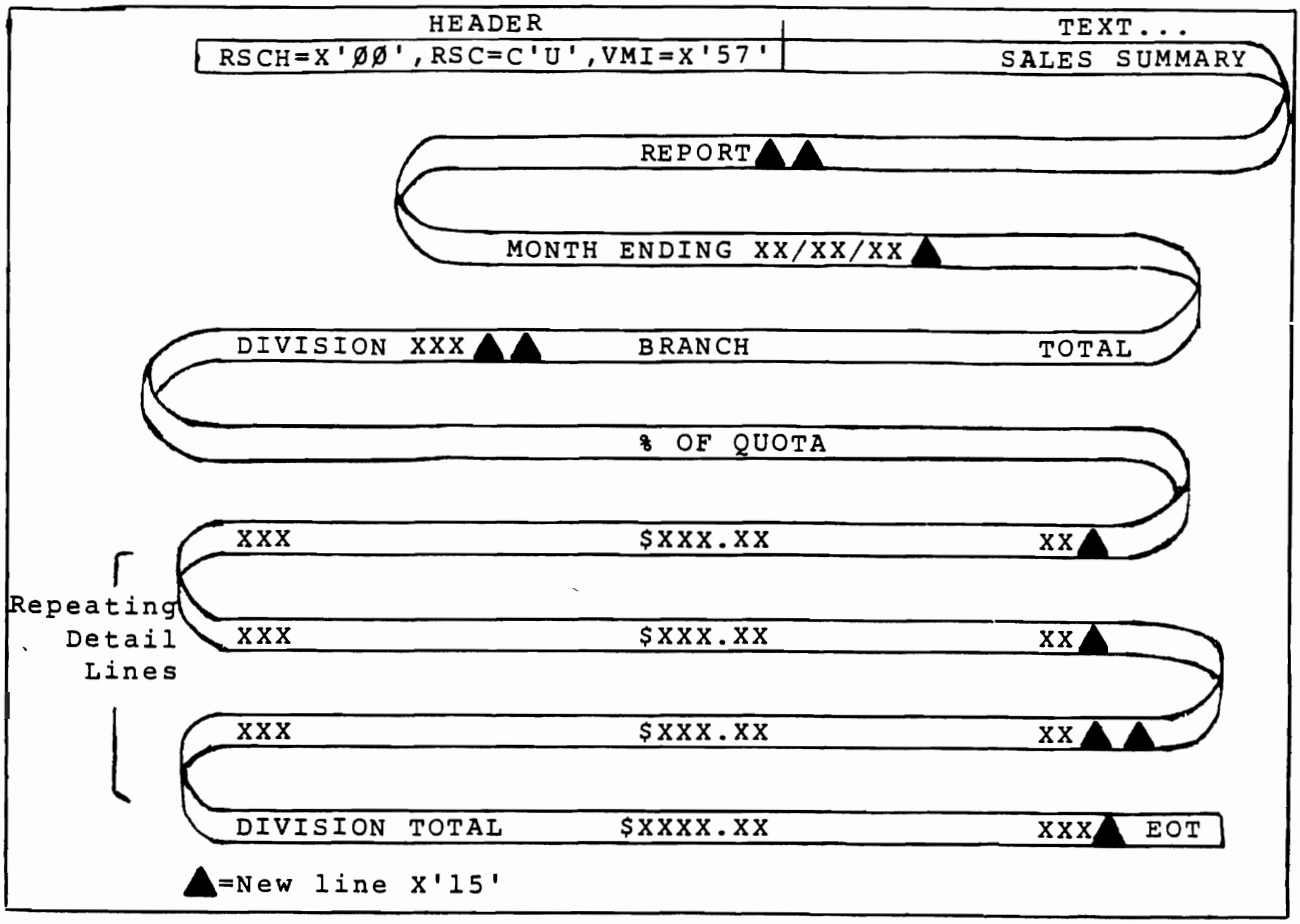


Figure 18-4. Sample pre-formatted output message.

Data items in the message text consist of an item code number, data field length count, and text field. Data items having the same item code number may appear more than once ina message. The brace in Figure 18-5 illustrates how in this case, a LINE entry in the designated OFT indicates that all ITEM code numbers within the domain of that prototype line are to be REPETitive. Whenever data items containing these so designated item code numbers appear in a message, an occurrence number must also appear within the corresponding text field immediately preceding the actual text. Data items defined in the OFT as repetitive require this occurrence number in the data field even if for some reason the data item itself were never to appear more than once in a message.

The first data item in a variable message to the OUTPUT Utility should designate the OFT# to be used for formatting the message. Data item code number 255 is reserved for this use, and the OFT# is expected as a halfword binary value. Otherwise, data items may appear in any convenient sequence, because corresponding ITEM entries in the Output Format Table control the field

```
RPTØØØ99        REPORT          NUM=99,LINES=9
                LINE            NUM=1,ITEMS=1
                ITEM            CODE=255,DATA='SALES
                                 SUMMARY REPORT',
                                 FROM=1Ø,TO=29
                LINE            NUM=2,ITEMS=Ø
                LINE            NUM=3,ITEMS=2
                ITEM            CODE=255,DATA='MONTH
                                 ENDING', FROM=1,
                                 TO=12
                ITEM            CODE=6,FROM=15,TO=22
                LINE            NUM=4,ITEMS=2
                ITEM            CODE=255,DATA='DIVISION',
                                 FROM=1,TO=8
                ITEM            CODE=17,FROM=1Ø,TO=12
                LINE            NUM=5,ITEMS=Ø
                LINE            NUM=6,ITEMS=3
                ITEM            CODE=255,DATA='BRANCH',
                                 FROM=1,TO=6
                ITEM            CODE=255,DATA='TOTAL',
                                 FROM=1Ø,TO=15
                ITEM            CODE=255,DATA='% OF
                                 QUOTA',FROM=19,
                                 TO=28
                LINE            NUM=7,ITEMS=3,REPET=1
                ITEM            CODE=5,FROM=1,TO=3
                ITEM            CODE=1Ø,FROM=1Ø,TO=16
                ITEM            CODE=15,FROM=19,TO=2Ø
                LINE            NUM=8,ITEMS=Ø
                LINE            NUM=9,ITEMS=3
                ITEM            CODE=255,FROM=1,TO=14,
                                 DATA='DIVISION TOTAL'
                ITEM            CODE=18,FROM=1Ø,TO=17
                ITEM            CODE=19,FROM=19,TO=21
```

Figure 18-5. Sample Output Format Table Coding (OFT #ØØØ99).

location and sequencing in the output message. Hence, application subsystems require no modification when screen or report format changes are required. Also several different application subsystems may utilize various common OFT's. When doing this, the application programmer must be aware of the item code numbers assigned to each text field defined in the OFT's to be used.

There are two forms of variable text messages; character and binary. The variable character message format varies from the variable binary message format in that the binary message format utilizes 8-bit binary item code numbers, data field length counts, and occurrence numbers, while the character format uses 3-character integer fields instead, along with a special N or R repetition character to indicate whether or not an occurrence number is to be expected within the text field. The OUTPUT Utility, however, accepts only the one-byte binary format, and it is the COBPUT high level language queuing routine that converts the 3-character values to one-byte values and eliminates the repetition character.

Variable Character Text (COBOL or PL/1 only)

As shown in Figure 18-6, each data item consists of a 3-integer item code number, 3-integer data field length count, and data field. The data field is further broken down to include a repetition character, and if the field is repetitive, an occurrence or line number. Figure 18-2 indicates the applicable message header codes for this message format.

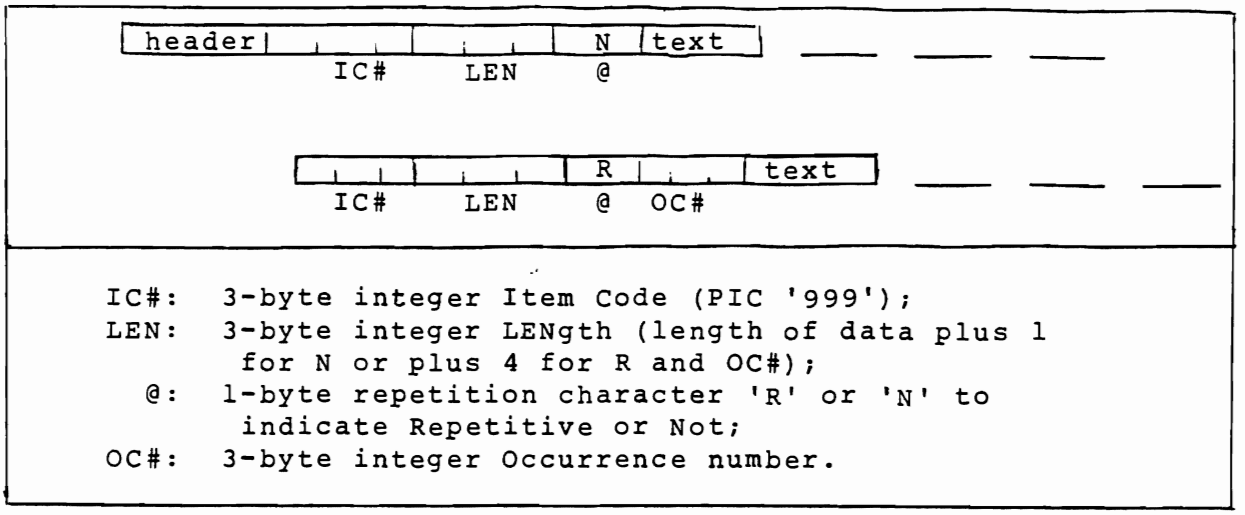


Figure 18-6. Illustration of variable character message format.

As shown in figure 18-7, the first data item in a variable character message to the OUTPUT Utility should consist of:

1. Item code of 255;
2. Length (003);
3. N;
4. Fixed binary (15) Output Format Table entry number (OFT#).

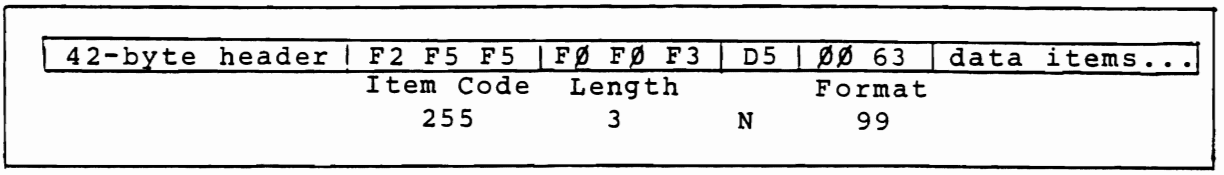


Figure 18-7. Illustration of variable character message showing OFT# field.

Figure 18-8 shows a sample variable character format message to generate a formatted message conforming to the layout depicted in Figure 18-3. Remember that the data items may appear in any sequence convenient to the programmer; they need not be grouped or sequenced as shown.

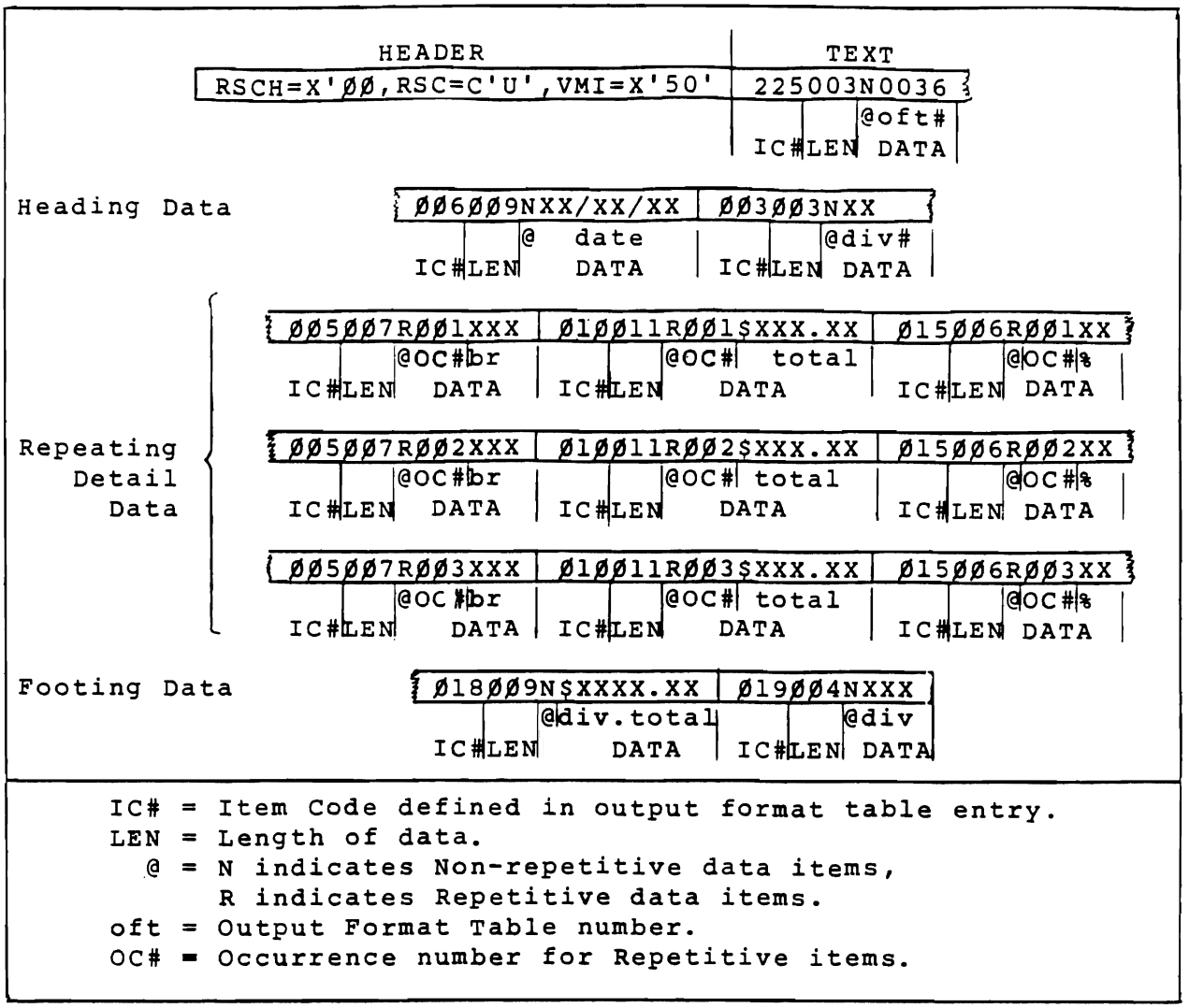


Figure 18-8. Sample variable character format output message.

Variable Binary Text (COBOL, PL/1, Assembler)

As shown in Figure 18-9, each data item consists of a one-byte binary item code number, a one-byte binary data field length count, and data field. In the case of repetitive data items, the data field contains a one-byte binary occurrence number immediately preceding the text. Figure 18-2 indicates the applicable message header codes for this message format. As with variable character text, one of the data items in the message text must be item code:255, length:2, data:OFT#.

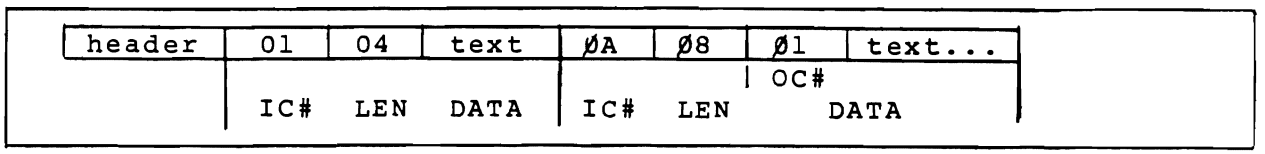


Figure 18-9. Illustration of variable binary message format.

Formatting Required, Fixed Field Text

The message text consists of adjacent fixed length text fields similar to the layout of a logical record in a data set. The fields may be character, binary, or packed decimal. The message is operated upon by the CHANGE/DISPLAY Utility to convert it to a variable binary format message as described above. A Format Description Record must be created for this fixed format in order for the CHANGE/DISPLAY Utility to build a corresponding item code, length and data, variable binary format message and pass it on to the OUTPUT Utility.

The message text is prefixed by 12 characters:

1-8: 1 to 8 alphanumeric characters, left adjusted. The Change Table will be searched for a match on this field to find the appropriate Format Description Record.

9: Ø if this data is to become a single segment message for OUTPUT.

1Ø-12: OFT number to override FDR default value.

Figure 18-10 illustrates the fixed field message format. The length field in the header must include the length of the prefix. Figure 18-2 indicates the applicable message header codes for this message format.

42-byte header	FRMTREC1ØØ73	FIELDAFIELDBETC.
	PREFIX	DATA

FRMTREC1	The Fixed Format Name associated with the Format Description Record as listed in the Change Table.
Ø	Indicates a single segment message.
Ø73	The Output Format Table entry number (to override RPTNO in FDHDR).

Figure 18-10. Fixed field message illustration showing prefix field.

Figure 18-11 shows a sample fixed field message to generate a formatted message conforming to the layout depicted in Figure 18-3.

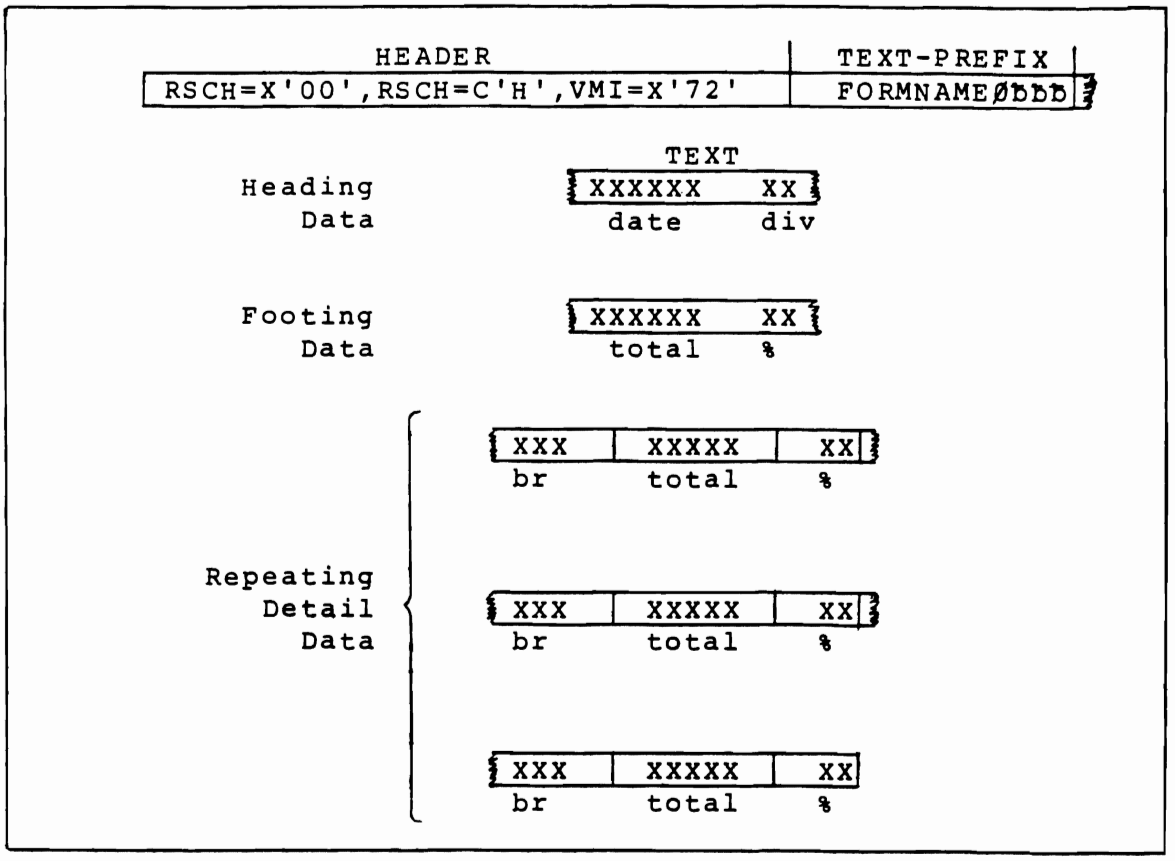


Figure 18-11. Sample fixed format output message.

The first eight characters of the 12-character text-prefix identifies the name of the Format Description Record in which each of the individual text fields is defined. This includes defining every item code, length, and data type. It is used by the CHANGE Utility in converting the fixed format text fields to variable binary format data items for the OUTPUT Utility. The remaining characters indicate the message segment number and overriding Output Format Table entry number (OFT#), if any.

Notice in Figure 18-11 that the repeating detail data appears at the end of the message, instead of between the heading and footing lines. This is because the CHANGE Utility requires that all repeating data items must appear as a group at the end of the message or record to be converted for the OUTPUT Utility. Figure 18-11 shows the sample Format Description Record (FDR) and Change Table entry needed by the CHANGE Utility to convert the sample fixed format message shown in Figure 18-11.

```
DESØØØØ1    CSECT
FORMNAME    FDHDR       NAME=FORMNAME,RPTNO=99,FIELDS=7,
                        REPSZ=14
DATE        FDETL       OFSET=Ø,LEN=8,CODE=6,NAME=DATEX
DIVISION    FDETL       OFSET=8,LEN=3,CODE=17,NAME=DIVSN
DVTOTDOL    FDETL       OFSET=11,LEN=8,CODE=18,NAME=TOTDL
DVTOTPCN    FDETL       OFSET-19,LEN=3,CODE=19,NAME=TOTPC
BRNCHNUM    FDETL       OFSET=22,LEN=4,CODE=5,NAME=BRNCH,
                        FLD=REPET
BRNCHTOT    FDETL       OFSET=26,LEN=7,CODE=1Ø,NAME=BTOTX,
                        FLD=REPET
BRNCHQTA    FDETL       OFSET=33,LEN=3,CODE=15,NAME=BQTAX,
                        FLD=REPET
            END
```

```
            DC          CL8'FORMNAME'   NAME corresponding
                                        to DESØØØØ1
            DC          A(Ø)            RBN Ø on DESØØØ=
                                        DESØØØØ1
```

Figure 18-12. Sample Format Description Record Coding (DESØØØØ1) and Change Table Entry.

MULTI-SEGMENTED MESSAGES

When a subsystem produces a report with many different detail lines, it may be constructed as a multi-segment message. It requires formatting and can be either variable character or fixed length text. A multi-segment message, then, consists of two or more logically related INTERCOMM messages (header followed by text) directed to a single terminal without interreuption.

The responsibility of the OUTPUT Utility here is more complex. It must provide formatting of each segment of a message transmitted to a single terminal under exclusive control and upon completion of transmission, release the terminal exclusive use status. Again to determine the message format OUTPUT analyzes the MSGHRSCH, MSGHRSC and MSGHVMI fields of the message header. Each message segment has its own 42-byte message header. The contents of the fields to be adjusted by the application subsystem are diagrammed in Figure 18-13. Since an OFT# is specified for each segment of the message, the text formatting can be controlled by different OFT entries.

OUTPUT MESSAGE TEXT FORMAT	MESSAGE HEADER FIELDS*			CHANGE/ DISPLAY TEXT PREFIX BYTE 9
	MSGHRSCH	MSGHRSC	MSGHVMI (see note below)	
Variable Text - Character Format (COBOL and PL/1)	X'ØØ' or C'Ø'	C'V'	X'51' or C'1' X'52' or C'2' X'53' or C'3' X'5C' or C'4'	N/A
Variable Text - Binary Format (COBOL and PL/1)	C'V'	C'V'	X'51' X'52' X'53' X'5C'	N/A
Variable Text - Binary Format (Assembler)	X'ØØ'	C'V'	X'51' X'52' X'53' X'5C'	N/A
Fixed Text	C'H'	C'H'	X'72' or C'S'	C'1' C'2' C'3' C'4'

*Character values for MSGHRSCH or MSGHVMI are converted by COBPUT to the expected hexadecimal values. Assembler subsystems must use the hexadecimal values.

The VMI value of Change/Display Prefix Byte 9 indicates segment type:

X'51'/C'1' indicates header segment and causes the OUTPUT Utility to select only non-REPETitive items, as coded in the corresponding OFT, from the message text;

X'52'/C'2' indicates intermediate segment and selects only REPETitive items from the message text; and

X'5C'/C'4' indicates intermediate segment and selects only non-REPETitive items from the message text.

X'53'/C'3' indicates final segment and selects only non-REPETitive items from the message text.

Figure 18-13. Message Header Specifications - Multi-segment Message.

The OUTPUT Utility accomplishes the processing of a multi-segment message in conjunction with the DVASN service routine.

When the COBPUT queuing routine first recognizes a multi-segment message for OUTPUT (VMI=X'51'), the INTERCOMM system program DVASN is called to assign the destination terminal named in the message header to a "multi-segmented-message-transmission-in-progress" condition. Once this assignment is made, the OUTPUT Utility will accept only intermediate segments (VMI=X'52' or X'5C') for that device until a final message segment (VMI=X'53') for that service is processed. For this reason, the COBPUT routine requires that the name of the field containing the binary OFT# in the first message of a multi-segment message group be named in the otherwise optional third parameter, OFT#.

A BAL application subsystem must CALL the DVASN subroutine to obtain exclusive use of a terminal for a variable character text multi-segment message, before queuing the message for transmission to OUTPUT via MSGCOL. DVASN will assign the destination terminal indicated in the message header, to a "multi-segmented-message-transmission-in-progress" condition.

When Fixed Format Text is used, the CHANGE/DISPLAY Utility assumes responsibility for the CALL to the DVASN routine to "assign" the terminal for the duration of segmented message transmission. The CALL to COBPUT, then, is made with only two parameters as usual.

SEGMENTED MESSAGE OUTPUT TERMINAL ASSIGNMENT (DVASN)

The DVASN message processing service routine has been referenced previously in this text in conjunction with the OUTPUT Utility. DVASN is called by a subsystem to obtain exclusive use of a terminal for the purpose of transmitting a multi-segment message without interruption. The DVASN subroutine must be CALLed before queuing the first segment of a multi-segment message via MSGCOL for formatting by OUTPUT.

The coding format for calling DVASN is:

[symbol]	CALL	DVASN, (cmp,spa,term,oft,ret), MF=(E,list)

Where:

cmp is the address of a field containing the number of the company or division being serviced. (2 byte, binary).
spa is the address of the System Parameter List.
term is the address of a field containing character MSGHTID destination terminal name field in the message header.
oft is the address of a field containing the OFT number of the format about to be started. (2 byte, binary).
ret is the address of a five byte field in which to return the terminal ID (CCCNN)

As a result of this call, DVASN will locate and assign a terminal to the subsystem and designate it as a "multi-segmented-message-transmission-in-progress" condition in its respective entry in OUTPUT's Station Table. This action thus prevents other messages from being transmitted to the designated terminal until its busy status is freed by OUTPUT.

THE CHANGE/DISPLAY UTILITY

GENERAL DESCRIPTION

The CHANGE/DISPLAY utility provides predefined transactions to access on-line files from remote locations. A terminal operator can trigger the retrieval of one record at a time from a user file. All or part of the information accessed may be displayed back on the remote device (DSPL verb). DISPLAY data will appear in character format according to an Output Utility OFT. Alternatively, one or more of the data fields in an accessed record can be altered (CHNG verb). The record as CHANGEd is then written back to the user file.

For the processing of CHNG and DSPL verbs, CHANGE/DISPLAY supports BDAM, ISAM, and VSAM files. In each case, records must be in fixed format. BDAM files are accessed by relative block number (RBN); ISAM files are accessed by key. VSAM files are accessed by key, via the File Handler logic for VSAM compatibility with ISAM. Individual fields within a record may contain binary, packed, or character data.

Terminal input transactions are entered in key word format, and processed by the EDIT Utility. A file ddname and a key (or RBN) are entered, and the specified record is retrieved. If DSPL was entered, the data accessed is formatted as a message for the OUTPUT Utility. In addition to file ddname and key, a CHNG transaction specifies fields to be altered. Replacement data is indicated for each field entered. The record in core is altered as specified and then rewritten. A message designating completion of processing, successful or unsuccessful is formatted for OUTPUT in this case.

The CHANGE/DISPLAY Utility also processes fixed format data passed from an application subsystem. (No user data base is accessed in this case.) Such intersubsystem messages are reformatted as new messages for the OUTPUT utility. This application program technique is referred to as Fixed Format Text, Formatting Required in the Section "Using the Output Utility" in the Programmer's Guides. The application programmer is not concerned with item code/length prefixes for data fields; thus the potential for program errors is reduced. The text of such an intersubsystem message simulates a fixed-format file record. Processing bypasses access of a user file and continues as though a DSPL transaction had been entered.

The CHANGE/DISPLAY utility is entirely table driven. The following table entries (illustrated in Figure 19) are required:

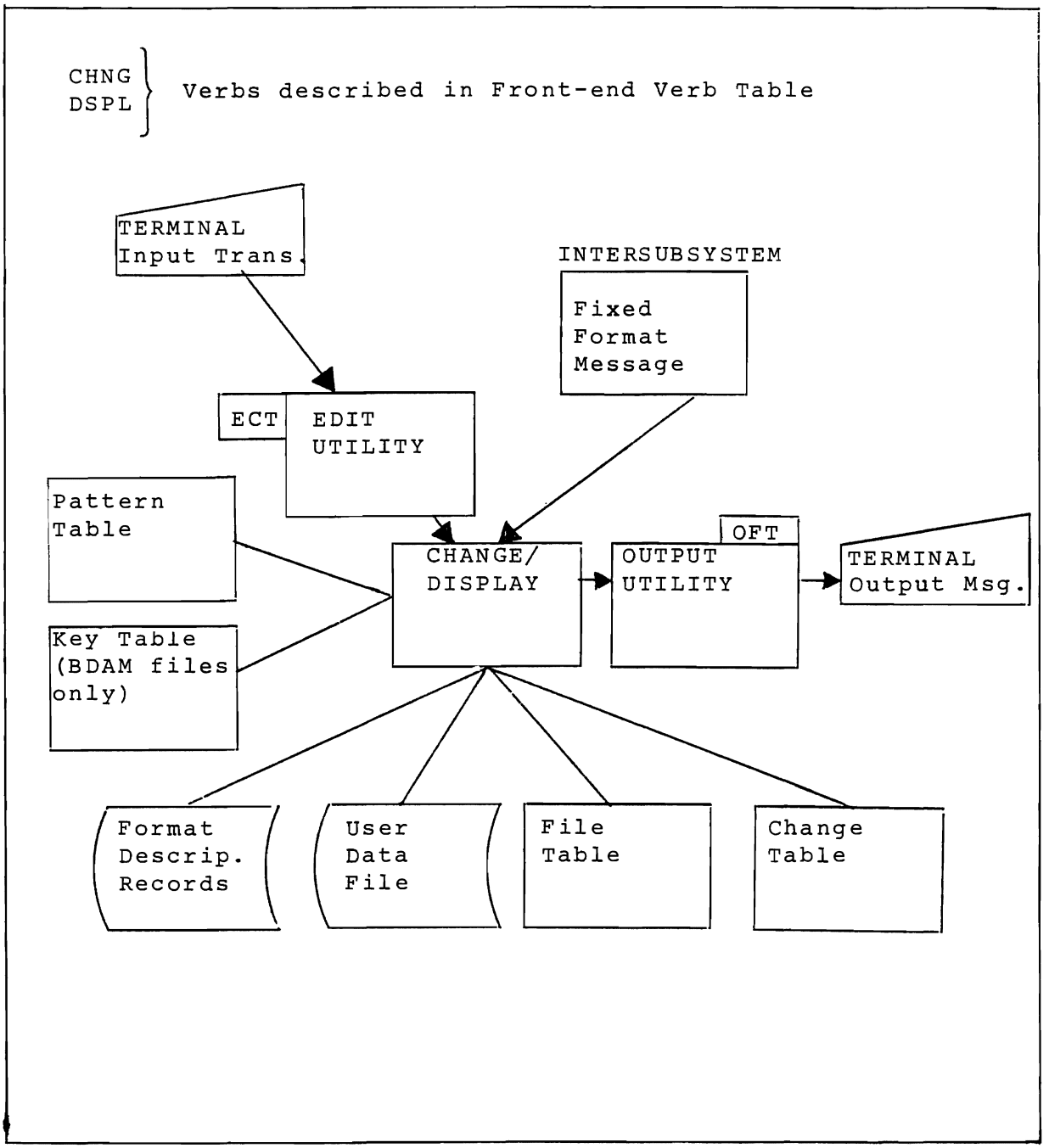


Figure 19. Functional Components of CHANGE/DISPLAY.

1. Entries for CHNG and DSPL in the Front-End Verb Table.
2. Entries for CHNG and DSPL in the EDIT Control Table.
3. Entries in the File Table for each on-line file to be accessed.
4. A Format Description Record (FDR), a disk resident table entry for each user file to processed; and/or each format of intersubsystem message to be processed for OUTPUT.
5. An Output Format Table entry for each display format to be produced.
6. A Change Table entry for each format of intersubsystem message to be processed for OUTPUT.
7. A Key Table entry for each user-coded key editing routine.
8. A Pattern Table entry for each editing pattern specified in the Format Description Records.

Some installations may not wish to exercise all CHANGE/DISPLAY options. In such cases, not all of the above table entries are required. For instance, CHNG transactions may not be desired; CHNG entries would then be omitted from the Front-End Verb Table and the EDIT Control Table. Another installation might not wish to process intersubsystem messages for OUTPUT; in such cases, the FDR and Change Table would be omitted. In other words, the table entries supplied will determine which CHANGE/DISPLAY options are available.

CHANGE/DISPLAY executes as an INTERCOMM subsystem and may be defined as resident, overlay region, or dynamically loaded.

TERMINAL INPUT MESSAGE FORMATS

For a DISPLAY transaction, a DSPL verb is entered at the terminal. For a CHANGE transaction, a CHNG verb is entered. The explicit formats of these verbs are detailed below.

DSPL Verb

```
DSPL Δ                          (Display verb)
FLN  XXXXXXXX Δ                 (File ddname)
KEY XXXX.........Δ              (Key or record to be displayed)
RPT XXX Δ                       (Output format number)
END O                           (End of message)
```

The file name entered following FLN must:

1. Be defined to INTERCOMM by a DD card in the execution JCL.
2. Correspond to an entry in the File Table.
3. Be defined by a Format Description Record.

Data entered following KEY will be one of the following:

1. Actual key for an ISAM file.
2. Relative block number (RBN) for a BDAM file.

The EDIT utility passes these keyword fields through Edit Subroutine Ø: field length is determined; no conversion of the data. CHANGE/DISPLAY will pass the key field through its own edit processing.

The RPT parameter is optional. If included, the number supplied will override the default OFT number in the FDR. This option might be used to allow certain terminals to display more information than others. For instance, a customer file might contain names, addresses, and credit information. All terminals in the system are to have access to names and addresses. Only terminals in the credit office are to have access to credit information. The standard OFT number specified in the FDR would format only name and address for OUTPUT. Operators of credit-office terminals would be told an overriding number to enter. This second OFT number would format name, address, and credit information for OUTPUT. Of course, every format number (whether from FDR or RPT data) must correspond to an entry in the Output Format Table.

CHNG Verb

To update a record, the following transaction is keyed in at the remote location:

```
CHNG Δ                      (Change verb)
FLN XXXXXXXX Δ              (File ddname)
KEY XXXX.... Δ              (Key of record to change)
FDN XXXXX Δ                 (Name of field to change)
SKY XXXX.... Δ              (Secondary key)
VRY XXXX.... Δ              (Verify data)
DTA XXXX.... Δ              (Change data)
  :
  .
END O                       (End of message)
```

The FLN and KEY parameters are subject to the same restrictions as are the corresponding DSPL parameters.

In the FDN parameter, the operator enters a field identifier of up to five characters, indicating the name of the field to be changed.

The SKY parameter is optional. It is used to supply a secondary key for use in searching groups of repetitive fields to locate a particular field to CHANGE. When the SKY data matches the repetitive field group secondary key, the associated FDN is CHANGEd.

The VRY parameter should contain the data actually in this field in the record. The FDR may specify that verification is optional; in such a case, the parameter may be omitted from input. If VRY is entered, CHANGE/DISPLAY will verify the existing data before proceeding.

Actual replacement data is entered following the DTA keyword. (Both DTA and VRY data are edited in accordance with FDR specifications; only after editing are they applied against the actual file record.) For VRY and DTA, the user may wish to enter blanks from the terminal. However, blanks will be eliminated from the message during processing. If verification of a blank field is desired, the operator should enter:

```
VRY PMIBLNK
```

To blank out an existing character field, enter:

```
DTA PMIBLNK
```

The parameters FDN, VRY, and DTA are repetitive. Thus, as many fields in a record as desired can be altered via one CHNG transaction. Note, however, that there are special considerations when SKY or VRY parameters are omitted. EDIT, in processing the transaction, will assign occurrence numbers to repetitive input items. The first appearance of a repetitive item will be assigned occurrence number 1; the second, 2; and so forth. CHANGE/DISPLAY requires identical occurrence numbers for corresponding FDN, SKY, VRY, and DTA parameters. VRY or SKY may be omitted for one FDN entry but included in a subsequent one. In such a case, correspondence of occurrence numbers would be lost; to overcome this difficulty, a special input configuration is provided. To force an out-of-sequence occurrence number, the following may be entered:

VRY(n,data) or SKY (n,secondary key)
where n is the desired occurrence number. This facility is illustrated in the following CHNG transaction:

```
CHNG △
FLN FILEA △
KEY 25 △
FDN ADDR △                    (occurrence 1)
DTA 475 BROADWAY △
FDN CITY △                    (occurrence 2)
VRY (2,NEWARK) △              (force occurrence of 2)
DTA NEW YORK △
FDN CREDT △                   (occurrence 3)
SKY (3,TYPEA) △               (force occurrence of 3)
VRY (3,5ØØØØ) △               (force occurrence of 3)
DTA 1ØØØØØ △
FDN MANGR △                   (occurrence 4)
DTA PMIBLNK △                 (blank out MANGR field)
END ○
```

RECORD FORMATS

Record formats processed by CHANGE/DISPLAY refer to either actual data records on a user BDAM or ISAM file or "pseudo records" representing fixed format fields of message text from an application subsystem to be converted to the appropriate "variable character text, formatting required" for the OUTPUT Utility. In either case the following conventions are required:

1. records are fixed length
2. each data field may be unique in character, packed decimal, or binary form, or
3. the record may consist of a fixed number of unique header fields followed by a variable number of repetitive groups of fields. If the number of repetitive groups is not maximum for the record, the trailing portion contains high-value (X'FF') fill.

Figure 20 illustrates record formats; note that intersubsystem messages also include an additional 54 bytes prefixed to the record format illustrated for the standard 42 byte INTERCOMM message header plus 12 bytes of information required for CHANGE/DISPLAY processing.

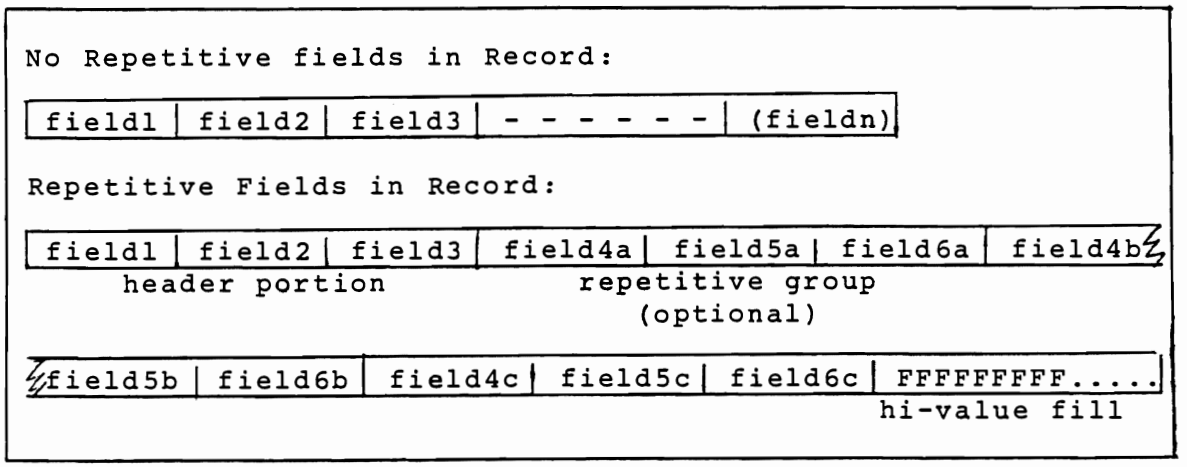
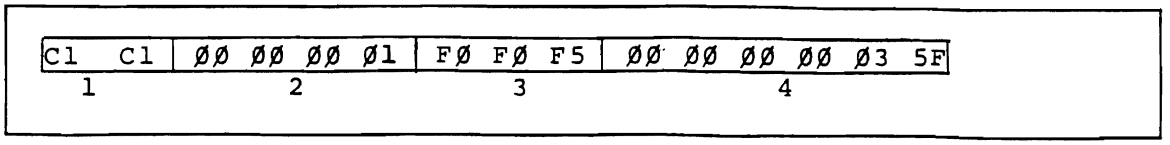


Figure 20. Record Formats supported by CHANGE/DISPLAY

For ISAM files, the key will be converted according to the field type specification in the FDR. If a key consists of several fields, they must be entered in the same order as the FDR description of key fields. Thus, an ISAM key might consist of:

1. A two byte alphanumeric field.
2. A four byte binary field.
3. A three byte numeric field.
4. A six byte packed field.

Assume that a particular key on the file appeared as follows:



The corresponding field descriptions in the FDR must appear in the identical sequence. To display a record with the above key, an operator would enter:

```
DSPL △
FLN ddname △
KEY AA/1/5/35 △
END ○
```

The RBN for a BDAM file, of course, would not be defined in the FDR. The supplied key is simply converted to binary, unless the user supplies a conversion routine. The (/) slash character must be part of the key supplied to CHANGE/DISPLAY and therefore cannot be used as the Edit Separator Character.

CHANGE/DISPLAY PROCESSING

All messages for CHANGE/DISPLAY enter the module at the same point. A check is made to determine whether the message is a CHNG or DSPL verb. If it is, the processing described under "Common Processing for CHNG and DSPL Verbs" is performed. Otherwise, the processing described under "Processing of Fixed Format Messages for OUTPUT" is performed.

Common Processing for CHNG and DSPL Verbs

The first program step for a CHNG or DSPL verb is a CALL to EDIT. The EDIT Control module is driven by the entries in the EDIT Control Table. If EDIT rejects the incoming message, the sending terminal is notified and processing is discontinued.

Next, the File Table is searched for the file name entered at the terminal. If no entry is found, the remote location is notified and processing is terminated. If there is a file table entry, it will contain:

1. Record number of Format Description Record (FDR) for this file. (The FDR describes the detail characteristics of every data field in the file record.)
2. Type of user file to be accessed (BDAM or ISAM).
3. Amount of dynamic core required for retrieval of one block from the user file.

The FDR is next retrieved from disk. This is accomplished via the File Handler; a BDAM READ is issued with an RBN based on the record number in the File Table. An I/O error or record-not-found condition will generate an error message and processing will be terminated.

The file ddname given as input is now compared to that appearing in the FDR. A not equal condition indicates an error in either the File Table or the FDR'; processing is terminated.

The FDR is now checked to determine that editing is required on the incoming key. Standard editing options are:

- Editing of key elements based on FDR detail specifications (ISAM keys).
- Conversion of entered data to binary (BDAM RBN's). The user also has the option of supplying his own BDAM key editing routine.

The edited key is returned to CHANGE/DISPLAY.

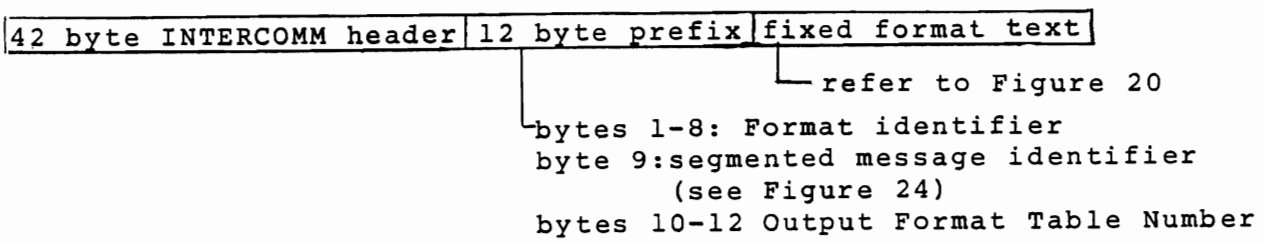
The next step is to retrieve the User File record specified by the input message. The amount of storage indicated in the File Table is obtained. The type of file is checked, and an appropriate CALL to the File Handler is issued. An I/O error or record-not-found condition will produce a message and cause termination of CHANGE/DISPLAY processing.

If a record is retrieved successfully, a check is made whether DSPL or CHNG WAS ENTERED. From this point on, processing differs. If DSPL, the action described under "Creating a Display Message" is taken. If CHNG, the action described under "Update Processing for CHNG" is taken.

Processing of Fixed Format Messages From Application Programs

Intersubsystem messages with fixed format text to be processed for OUTPUT bypass the CALL to EDIT, as identified by VMI of X'72' in the message header. The 12-byte CHANGE DISPLAY prefix provides information required to convert the fixed format text to the appropriate message for the OUTPUT Utility.

The message format, then is:



The Change Table contains a record number of an FDR corresponding to this particular message format as identified by the format identifier. The table itself consists of a series of associated identifiers and FDR record numbers.

The FDR is next retrieved, as it would have been for a DSPL or CHNG transaction. The "file name" in this FDR will actually be the identifier supplied in the intersubsystem message; no user file is accessed. From this point on, processing continuses as it would for a DSPL verb. This procedure is described immediately below.

Creating a Message for the OUTPUT Utility

At this point, CHANGE/DISPLAY has a fixed-format record in core. This may be either an actual file record or data sent from an application subsystem. There is also a Format Description Record in core that describes the fixed format record. The FDR indicates how each data field in the record is to be edited for OUTPUT.

For either a DSPL verb or an intersubsystem message, core for a message for OUTPUT is obtained. In this area, the user file record (or data from an intersubsystem message) is formatted. This processing follows:

1. A Message Header for OUTPUT is constructed.
2. An OFT number (item code 255 in message for OUTPUT) is assigned. This number is specified by the FDR, unless a substitute number has been indicated with a DSPL verb via the optional RPT parameter. Or, for intersubsystem messages, overriding report numbers appear in the message prior to the actual data to be displayed; in this case, blank or zero indicates that the FDR OFT number is to be used.
3. Data fields are now processed one at a time in the following manner:
   a) Binary and packed fields are converted to character.
   b) Editing specified in the FDR is performed (dollar, date, etc.).
   c) Necessary padding or truncation (if allowed by FDR) is effected.
   d) Item code and length is prefixed to the data item.
   e) For repetitive fields, an occurrence number is also prefixed to the data.
   f) If invalid data is found in any field being edited, the output field will contain the word "INVALID".
4. The new message is queued, via Message Collection, for OUTPUT.
5. Standard INTERCOMM housekeeping (i.e., freeing core, RELEASE of files, etc.) is performed. CHANGE/DISPLAY then returns to the Subsystem Controller.

Update Processing

If the incoming message has been a CHNG verb, the following steps would be taken at this point:

1. FDR detail description for first field to be altered is located. This search compares field identifiers from the input message with those coded in the FDR. A no-match generates an error message. Steps 2

through 4 will be performed only if verify data is supplied. The FDR specifies whether verification is required. If it is required, failure to provide verify data will cause an error. If it is not required, steps 2 through 4 are taken only if the VRY keyword was entered for this field.

2. Verify data of the input message is translated to the format indicated for this field in the FDR.
3. Translated verify data is compared to the field in the file record.
4. If (3) compares not equal, "VERIFY REJECT" message is sent and processing terminates.
5. Change data is translated to the format specified in the FDR detail segment.
6. Translated change data is placed in the file record, overlaying existing data.
7. A check is made to determine whether more data fields are to be changed. If so, processing repeats from (1) above.
8. When all change fields have been processed:
   a) A test is made to determine whether a User Change Exit has been included in the system. (Details below.) If so, control is passed to it.
   b) The record is updated via the File Handler.
9. A message stating whether the change had been successful or not is formatted for OUTPUT. Required housekeeping is performed. Return to the Subsystem Controller is effected.

CHANGE/DISPLAY TABLES

Tables for the CHANGE/DISPLAY Utilities are generated via INTERCOMM macros. The table entries required for CHANGE/DISPLAY processing are:

- Entries for CHNG and DSPL in Front-End Verb Table: Omission of either entry will effectively prevent processing of the omitted verb.

- Entries for CHNG and DSPL in the EDIT Control Table: Omission of either entry will prevent EDIT processing of the transaction type. If verb is left out of the Front-End Verb Table, leave it out here as well. The coding of ECT entries for DSPL and CHNG is detailed later.

- Entry in File Table for each on-line file to be accessed: In addition to entries for CHANGE/DISPLAY user files, there must be an entry for the Format Description File itself; and an entry for the Output Format Table File, if any Output Format Table entries reside on disk; and an entry for the EDIT Control Table File, if any ECT entries reside on Disk. Construction of the File Table is discussed fully below.

- Format Description Record (on disk) for: Each user file to be processed by CHANGE/DISPLAY, and/or each format of intersubsystem message to be processed for OUTPUT. The method of constructing individual FDR's and of loading them onto disk is described later.

- Output Format Table Entries: For each message to be produced by CHANGE/DISPLAY, there must be an entry in the Output Format Table. In other words, there must be an OFT entry for each format number specified in the FDR's; and/or each format number to be entered as a RPT parameter of a DSPL verb; and/or each format number to be specified in a fixed format message for OUTPUT. particular note should be made of the following facts:
  1. CHANGE/DISPLAY will create a message for OUTPUT containing every data item from the processed record. When item codes supplied in the message to OUTPUT are not found in the OFT entry, they will be ignored. Hence, the design of the OFT will determine what portion of the supplied data is to be displayed.
  2. Records with repetitive field groups will be formatted for OUTPUT with occurrence numbers corresponding to the sequential position of each repetitive item. To output such items, repetitive lines must be defined in the OFT entry.

- Change Table entry: for each fixed format message to be processed for OUTPUT. Creation of this table is fully documented later in this text.

- Key Table entry: for each user-coded key editing routine. Specifications for construction of this table are included in the discussion below.

- Pattern Table entry: for each editing pattern specified in the Format Description Records. This table is described below.

Edit Control Table Entries

If the DSPL verb is to be used, the following entry must appear in the EDIT Control Table:

```
DSPL        VERB        DSPL,71,256,3
            PARM        FLN,Ø1,ØØ,ØØ8,1ØØØØ111
            PARM        KEY,Ø2,ØØ,255,1ØØØØØ11
            PARM        RPT,Ø3,Ø2,ØØ2,ØØØØØ111
```

Note should be taken of the following characteristics of the DSPL entry in the ECT:

- The VMI code (2nd parameter of VERB macro) must be 71. After EDIT has been CALLed, this VMI will be the only tag identifying the message. (Note that CHNG has a VMI of 7Ø, a fixed format message for OUTPUT has a VMI of 72. Choices of program paths within the CHANGE/DISPLAY utility are based on tests of the VMI.)

- The FLN and KEY parameters are passed through EDIT subroutine Ø. This subroutine will assign item code and length but will not perform any conversion of incoming data. Note that a fixed length of 8 is forced for the FLN field. (Bit 5 of the bit string of the PARM macro is on.) Search of the File Table will be based on the 8 character file ddname returned from EDIT. A file ddname of less than 8 characters will be padded on the right with blanks. Conversion of data supplied with KEY will be done within CHANGE/DISPLAY. A fixed length is not forced for KEY. (Bit 5 of bit string is off.)

- The RPT parameters must be indicated as optional. (Bit Ø of the bit string is off.) This incoming paramter is passed through EDIT subroutine 2 (if RPT is entered). Subroutine 2 will convert the supplied parameter to binary; a length of 2 is forced. Thus, EDIT puts the format number, if supplied, into the form required for the message to OUTPUT.

- Truncation is not allowed for any of the incoming parameters. (Bits 6 and 7 are both on in the bit string.)

- Item codes 1, 2, and 3 must be assigned, respectively, to the FLN, KEY and RPT parameters. (Do not code a 255 for RPT: CHANGE/DISPLAY expects a 3 and will replace it with 255 when building the message for OUTPUT.)

If the CHNG verb is to be used, the following entry must appear in the EDIT Control Table:

```
CHNG        VERB        CHNG,7Ø,256,6
            PARM        FLN,Ø1,ØØ,ØØ8,1ØØØØ111
            PARM        KEY,Ø2,ØØ,255,1ØØØØØ11
            PARM        FDN,Ø3,ØØ,ØØ5,1ØØØ1111
            PARM        SKY,Ø4,ØØ,255,ØØØØ1Ø11
            PARM        DTA,Ø5,ØØ,255,1ØØØ1Ø11
            PARM        VRY,Ø6,ØØ,255,ØØØØ1Ø11
```

The following characteristics of the ECT entry for CHNG should be noted carefully:

- The VMI code must be 7Ø.

- The FLN and KEY parameters are identical to the corresponding parameters for DSPL. Comments on these parameters following the DSPL verb ECT apply here as well.

- FDN and DTA are required (bit Ø of the bit string is on); SKY and VRY are optional (bit Ø of the bit string is off). Note, however, that VRY will be logically required if so flagged in the FDR. FDN, SKY, DTA, and VRY must all be tagged as repetitive. (Bit 4 of the bit string is on.) FDN must be forced to a fixed length of 5. (Bit 5 is on.) The search of FDR detail segments will be based on the data entered following this keyword.

- Truncation is not allowed for any of the incoming parameters.

- All parameters are passed through EDIT subroutine Ø. DTA, VRY, and SKY information will be converted later by CHANGE/DISPLAY. This conversion will be based on the description of the particular field in the FDR. Normally, the above ECT entries would be included during the initial installation of INTERCOMM. In some cases, the CHANGE/DISPLAY utility may be added to an existing installation.

The File Table

The File Table must contain an entry for each user file to be accessed by CHANGE/DISPLAY. It must also contain an entry for the Format Description File (DESØØØ). (Creation of Format Description Records is discussed later in this section.) In addition, the Output Format Table may have some or all of its entries on disk (file ddname RCTØØØ). Similarly, the EDIT Control Table may have all or some of its entries on disk (file ddname VRBØØØ). Loading Utility Table Entries on Disk is detailed in a later section. These files, if included in the system, must also have entries in the File Table.

User files that will not require CHANGE/DISPLAY processing should not be included in the File Table. Every data set referenced in the File Table must, of course, have a corresponding DD card in the execute deck.

The File Table is constructed by coding GENFTBLE macros. See Appendix D. One GENFTBLE macro must be coded for each file to be included in the table. The macros are contained in a CSECT named PMIFILET. Following the last GENFTBLE macro, a PMISTOP macro must be coded. The general structure of the File Table is:

```
PMIFILET CSECT
         ENTRY PMIFILTB
PMIFILTB EQU   *
         GENFTBLE . . .
         GENFTBLE . . .
           .
         PMISTOP
         END
```

The complete File Table, then would look as follows:

```
PMIFILET    CSECT
            ENTRY       PMIFILTB
PMIFILTB    EQU         *
            GENFTBLE    FNAME=DESØØØ,BLKSIZE=75Ø,TYPE=BDAM
            GENFTBLE    FNAME=RCTØØØ,BLKSIZE=1ØØØ,TYPE=BDAM
            GENFTBLE    FNAME=VRBØØØ,BLKSIZE=75Ø,TYPE=BDAM
            GENFTBLE    FNAME=USERFILT,BLKSIZE=xxxx,TYPE=ISAM,
                        DESNUM=7
* BLKSIZE FOR DESØØØ,RCTØØØ,VRBØØØ CORRESPONDS TO INTERCOMM
* RELEASE SPECIFICATIONS.  USER MUST CHANGE FOR LARGER TABLE
* ENTRIES.  INSERT ADDITIONAL ENTRIES FOR USER FILES HERE.
*
            PMISTOP
            END
```

For a user file, TYPE= may be either BDAM or ISAM. FNAME=, of course, is the ddname of the user file. The DESNUM= parameter contains the record number of the FDR for this user file.

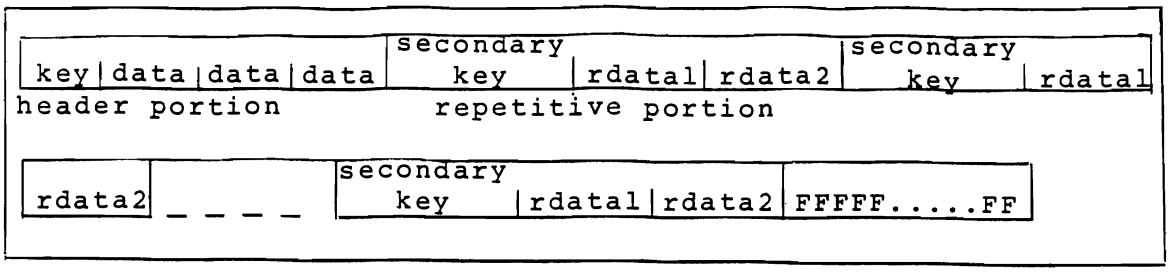
Format Description Records

Each file to be accessed by the CHANGE/DISPLAY utility must be described by a Format Description Record. Each format of intersubsystem message to be processed for OUTPUT must also be described by an FDR. Format Description Records are assembled and then loaded on to the Format Description File (DESØØØ). Retrieval of an FDR from this file is based on the record number supplied in the File Table or Change Table. The FDR defines exactly how CHANGE/DISPLAY is to edit and process each field in a record.

The individual FDR consists of a header portion followed by detail segments. The header portion supplies information relevant to all records on the file. Each detail segment describes a particular data field within a record. The header portion is generated by an FDHDR macro instruction; detail segments are generated by FDETL macros. (These macro instructions are described in detail in Appendix D; examples of their use are included below.)

CHANGE/DISPLAY allows one group of repetitive fields to be included at the end of a record being processed. When CHNG transactions are entered for repetitive fields, each field must be identified by a secondary key. The entire length of the repetitive portion is indicated in the FDR header portion; each field in the repetitive group is defined by one FDETL macro. That item which is the secondary key must be so flagged in the detail segment of the FDR. (On CHNG transactions, the information entered with the SKY Parameter will be used to search the repetitive portion of the record.)

A record format as follows would be described by one FDHDR macro and seven FDETL macros.



Certain fields may require editing other than simple conversion to character format for transmission to the terminal. Describing editing for DISPLAY is accomplished by specifications in the FDETL macro; for example:

A field has been defined by the following FDETL macro:

```
FDETL NAME=FLDØ1,OFSET=Ø,FRMAT=PACK,LEN=8,CODE=1,
    PADD=LEFT,PDCHR=BLANK,EDIT=NUM
```

Actual data on the file is:

ØØ	ØØ	Ø4	56	78	91	23	6F

Output at the terminal would be formatted as follows:

ØØØØØØØ4,567,891,236 (pad with blanks on left)

If padding had not been specified, data at the terminal would appear as follows:

4,567,891,236ØØØØØØØ (default pad with blanks on right)

Note that in either case the displayed length is 19 bytes.

If editing is requested on a character field, the field must be the maximum length as defined, <u>not</u> padded with blanks. Blank padding will lead to invalid data for packing. The edited length of data items must be considered in the OFT positioning specifications. The following table shows the edited length of data items after processing by CHANGE/DISPLAY. To use the table:

1. Look at the FRMAT column for the type of data on the file.
2. Look at the row in that column with the length of the field to be displayed
3. Scan across to the correct EDIT column to find the displayed length of the field.

LEN = (IN FDR)			LENGTH OF DISPLAYED FIELD		
FRMAT = BIN	FRMAT = PACK	FRMAT = CHAR	EDIT=DOLLAR	EDIT=NUM	EDIT=DATE
	1	1	4	1	8
	2	2	4	2	8
1	2	3	5	3	8
		4	6	5	8
2	3	5	7	6	8
		6	9	7	8
	4	7	10	9	N.A.
3		8	11	10	N.A.
	5	9	13	11	N.A.
4		10	14	13	N.A.
	6	11	15	14	N.A.
		12	17	15	N.A.
	7	13	18	17	N.A.
		14	19	18	N.A.
	8	15	21	19	N.A.
		N.A.	22	21	N.A.
	9	N.A.	23	22	N.A.
		N.A.	25	23	N.A.
	10	N.A.	26	25	N.A.
		N.A.	27	26	N.A.
	11	N.A.	29	27	N.A.
		N.A.	30	29	N.A.
	12	N.A.	31	30	N.A.
		N.A.	33	31	N.A.
	13	N.A.	34	33	N.A.
		N.A.	35	34	N.A.
	14	N.A.	37	35	N.A.
		N.A.	38	37	N.A.
	15	N.A.	39	38	N.A.
		N.A.	41	39	N.A.
	16	N.A.	42	41	N.A.

Change Table

The Change Table associates identifiers of fixed format messages for OUTPUT with Format Description Records. It must be built as a separated CSECT named CHNGTB. Entries are two lines each, the first line is a DC of CL8 containing the format identifier. The second line is a DC containing the RBN of the FDR to be accessed. The following sample shows a table containing 3 entries.

```
CHNGTB      CSECT
            DC          CL8'MSGDSPL'
            DC          A(6)
            DC          CL8'MSGDSP2'
            DC          A(1)
            DC          CL8'REPORTC'
            DC          A(2)
            PMISTOP
            END
```

The message identified by the name MSGDSPL will be processed using the FDR at RBN 6 on the DESØØØ file. The message MSGDSP2 will be formatted for OUTPUT in accordance with the FDR at RBN 1. Message REPORTC will be processed using FDR at RBN 2.

Key Table

The Key Table consists of entries associating particular key conversion routine numbers with specific addresses. The CSECT name must be KEYTABLE. The following example shows a table with a single entry for key routine 11. Entry point for this routine is named KEYRT11. The File Description Record specifies the key conversion routine for each file.

```
KEYTABLE   CSECT
           DC        A(11),V(KEYRT11)
           PMISTOP
           END
```

Coding Key Conversion Routines is discussed in a later section.

Pattern Table

The Pattern Table is resident in a CSECT named PTRNTBLE. It is created by coding a series of PATRN macros. (See Appendix D). For using EDIT= parameter coded in an FDETL macro (Format Description Record detail segment), there must be a corresponding entry in the Pattern Table. Following is an example of a Pattern Table:

```
PTRNTBLE    CSECT
            PATRN    NUMBER=DATE,PATTRN=DATE,MAXSIZE=8
            PATRN    NUMBER=DOLLAR,PATTRN=DOLLAR
                     FLOAT=$,MAXSIZE=42
            PATRN    NUMBER=NUMERIC,PATTRN=NUMERIC
                     MAXSIZE=41
*
* USER SUPPLIED PATTERNS GO HERE
*
            PMISTOP
            END
```

Any packed or binary fields which the user wishes to convert to character format require a user pattern table entry unless covered by the PATRN macros in the supplied PTRNTBLE CSECT.

Sample Table Entries

The examples in this section are included to assist the user in understanding the Table entries specifying record format descriptions.

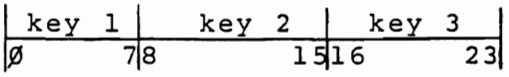
The examples illustrate:

- ISAM File Record, no repetitive group (Figure 21)
- BDAM File Record, no repetitive group (Figure 22)
- ISAM File Record, repetitive group (Figure 23)
- Intersubsystem Fixed Format message (Figure 24)

Each example illustrates the record format and associated FDR.

There exists an indexed sequential file with a ddname ISFILE. The key to each record in the file is eight characters. The record consists of keys to other ISAM records in the file.

Each record is 24 bytes long and consists of 3 record keys. The second key is the key to this record. The format, then, is as follows:

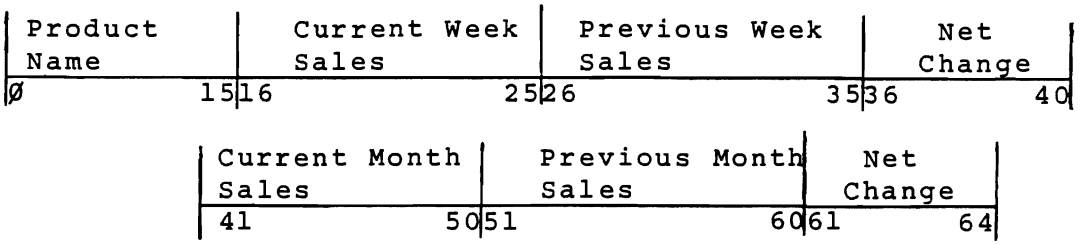


An Output Format Table entry with number 51 will be built to display this record. Item codes in this table for the three keys will be 1, 2, and 3. The FDR would be coded as follows:

```
FDHDR  NAME=ISFILE,FIELDS=3,RPTNO=51
FDETL  NAME=KEYØ1,OFSET=Ø,FRMAT=CHAR,LEN=8,CODE=1,
       PADD=LEFT,PDCHR=BLANK
FDETL  NAME=KEYØ2,OFSET=8,FRMAT=CHAR,LEN=8,CODE=2,
       PADD=LEFT,PDCHR=BLANK,KEY=YES
FDETL  NAME=KEYØ3,OFSET=16,FRMAT=CHAR,LEN=8,CODE=3,
       PADD=LEFT,PDCHR=BLANK.
```

Figure 21. ISAM File Record, no repetitive group.

There exists a BDAM file with a ddname BDFILE. The file contains sales information for each product the company manufactures. The record contains:



Assuming that an Output Format Table with report number 52 is to be built, the FDR would be coded as follows:

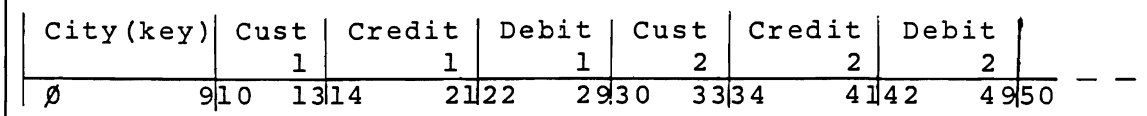
```
FDHDR  NAME=BDFILE,FIELDS=7,RPTNO=52,KEYRT=2
FDETL  NAME=PRODT,OFSET=Ø,FRMAT=CHAR,LEN=16,
       CODE=1,PADD=LEFT,PDCHR=BLANK
FDETL  NAME=CUWIC,OFSET=16,FRMAT=PACK,LEN=1Ø,
       CODE=2,PADD=LEFT,PDCHR=BLANK
FDETL  NAME=PVWKS,OFSET=26,FRMAT=PACK,LEN=1Ø,
       CODE=3,PADD=LEFT,PDCHR=BLANK
FDETL  NAME=PERWK,OFSET=36,FRMAT=PACK,LEN=5,
       CODE=4,PADD=LEFT,PDCHR=BLANK
FDETL  NAME=CUMOS,OFSET=41,FRMAT=PACK,LEN=1Ø,
       CODE=5,PADD=LEFT,PDCHR=BLANK
FDETL  NAME=PVMOS,OFSET=51,FRMAT=PACK,LEN=1Ø,
       CODE=6,PADD=LEFT,PDCHR=BLANK
FDETL  NAME=PERMO,OFSET=61,FRMAT=PACK,LEN=5,
       CODE=7,PADD=LEFT,PDCHR=BLANK
```

Figure 22. BDAM File Record, no repetitive group.

There exists an ISAM file by the name of MASTER. Records contain a key (city) followed by a repetitive group consisting of:

1. Customer ID (secondary key)
2. Credit
3. Debit

The repetitive group repeats up to 1ØØ times. The record, then, looks as follows:



Assuming the output report number for this record is 53 and that a user key routine number 11 is required, the FDR would be coded as follows:

```
FDHDR  NAME=MASTER,FIELDS=4,RPTNO=53,KEYRT=11,
       REPSZ=2Ø
FDETL  NAME=CITY,OFSET=Ø,FRMAT=CHAR,LEN=1Ø,CODE=1,
       KEY=YES
FDETL  NAME=CUST,OFSET=1Ø,FRMAT=BIN,LEN=4,CODE=2,
       FLD=REPET,PADD=LEFT,PDCHR=ZERO,EDIT=NUM,
       SUBKY=YES
FDETL  NAME=CREDT,OFSET=14,FRMAT=PACK,LEN=8,CODE=3,
       FLD=REPET,PADD=LEFT,PDCHR=BLANK,EDIT=DOLL
FDETL  NAME=DEBIT,OFSET=22,FRMAT=PACK,LEN=8,CODE=4
       FLD=REPET,PADD=LEFT,PDCHR=BLANK,EDIT=DOLL
```

Figure 23. ISAM File Record, repetitive group.

Fixed Format Message:

Message Header:

nnnn	X'ØØ'	C'H'		c c c c c		X'72'
MSGHLEN		MSGHRSC		MSGHTID		MSGHVMI

Message Text:

aaaaaaaa	i	nnn	data fields in fixed format

- nnn — overriding output format number
- i — segmented message indicator*
- aaaaaaaa — format identifier

Format Description Record:

```
FDHDR  NAME=aaaaaaaa,FIELDS=x,RPTNO=xx
FDETL  ┐
FDETL  │
  .    │  description of message text fields
  .    │   (no FDETL macros for the 12-byte prefix)
  .    │
FDETL  ┘
```

* The segmented message indicator is used to place the appropriate value in the VMI field of the message header to be routed to the OUTPUT Utility. The correspondence is:

Segmented Message Identifier (character)	Message Type	OUTPUT Message VMI Value
Ø	Full Message	X'5Ø'
1	Header Segment	X'51'
2	Detail Segment with Repetitive Data	X'52'
3	Final Segment	X'53'
4	Detail Segment with no Repetitive data	X'5C'

Figure 24. Intersubsystem Fixed Format Message

USER EXITS FROM CHANGE/DISPLAY

The CHANGE/DISPLAY module provides two user exits. Through these, the user has the following options:

1. To perform special editing on a key supplied via a CHNG or DSPL verb with a key conversion routine.
2. To examine and further alter a record about to be re-written during processing of a CHNG transaction.

Coding Key Conversion Routines

Normal processing of keys by CHANGE/DISPLAY is based on the FDR for ISAM keys. BDAM RBN's are simply converted to binary. (Note that for normal processing of ISAM Keys, KEYRT= is not coded in the FDHDR macro. For normal processing of BDAM RBN's the FDHDR macro should specify KEYRT=2.)

If key processing other than that just described is desired, the user must code his own routine. The numbers assigned to these routines must be 5 or greater. This number is coded in the KEYRT= parameter of the FDHDR macro; and the key routine itself is identified by the Key Table entry described above.

The CHANGE/DISPLAY subsystem will pass a five word parameter list to the user-written routine:

1. Address of the key to be manipulated. The first byte is the length of the key in binary, followed by the key as entered at the terminal.
2. Address of SPALIST.
3. Address where manipulated key value is to be placed when routine is finished. Maximum length of manipulated key is 255.
4. Address of Format Description Record for this file.
5. Address of File Table Entry for this file.

The user must also supply a return code in register (15) to indicate whether processing was successful or not. Return codes are:

Ø - Successful key processing

1 - Unsuccessful, no CHANGE/DISPLAY action is taken. Instead, an error message is passed to the terminal.

CHANGE User Exit (CHNGEXIT)

The CHANGE User exit allows the user to modify a CHANGEd record before the file has been updated. The user must code the routine to effect this, and, it must have a CSECT name or entry point name of CHNGEXIT. CHANGE will CALL CHNGEXIT if

such a module is linkedited with INTERCOMM. The parameter list passed to CHNGEXIT consists of three words:

1. Address of edited input message.
2. Address of System Parameter List.
3. Address of record that was changed.

When control returns to the CHANGE subsystem, the record is updated.

CHANGE/DISPLAY ERROR MESSAGES

Error messages reflecting problems encountered during CHANGE/DISPLAY processing are generated and queued for processing via the OUTPUT Utility. The messages are formatted according to OFT entries. Each message contains the input message FLM data (ddname) or fixed format name and other identifying information to be returned to the originating terminal.

Each error message explicitly defines the reason for rejecting the CHANGE/DISPLAY transaction, for example:

"NO FDR FOUND FOR FILE (ddname)"

"KEY WAS NOT DEFINED IN FDR FOR (ddname)"

"CHANGE TABLE NOT FOUND OR NO ENTRY IN
CHANGE TABLE FOR (format name)"

"A RECORD DOES NOT EXIST WITH THE KEY_____
ON FILE (ddname)."

For a precise listing of CHANGE/DISPLAY Utility error messages, the OFT entries on the INTERCOMM Release Library should be assembled (or printed) by the INTERCOMM System Manager.

GENERAL CONSIDERATIONS

Table entries for EDIT and OUTPUT are optionally disk resident; Format Description Records for CHANGE/DISPLAY are always disk resident. The System Manager(s) for each INTERCOMM installation should maintain and control assignment of table entries to disk. Individual table entries on disk are maintained by the user on symbolic and load module libraries (partitioned data sets) and loaded to a BDAM dataset for retrieval at execution time. An INTERCOMM batch utility is used to accomplish the file load process.

Each INTERCOMM Utility uses a separate BDAM dataset for its table entries as follows:

- EDIT (ddname=VRBØØØ): Each record represents one EDIT Control Table (ECT) entry giving specifications for EDITing one verb.

- OUTPUT (ddname=RCTØØØ): Each record represents one OUTPUT Format Table (OFT) entry giving specifications for formatting one message. INTERCOMM error messages generated via OFT entries are disk-resident.

- CHANGE/DISPLAY (ddname=DESØØØ): Each record represents one Format Description Record (FDR) giving record formats for data files accessed through CHNG/DSPL verbs or record formats for fixed format messages from an application subsystem to be converted to variable character text and passed to the OUTPUT utility.

All table entries on disk are fixed length records.

Each disk resident table is described in the CHANGE/DISPLAY File Table (PMIFILET CSECT). Initial table entries for VRBØØØ, RCTØØØ, DESØØØ are provided on the installed INTERCOMM libraries. The maximum blocksize specified by these entries is 750 bytes. Care must be taken to ensure this table is modified if and when an installation's own disk resident table entries are longer than this specified value.

Each of the previous sections of this document have described coding conventions for the individual table entries for each utility. Figure 25 summarizes coding conventions, naming and dataset conventions (detailed later in this section) for each Utility.

UTILITY / REQUIREMENTS:	EDIT	OUTPUT	CHANGE/ DISPLAY
ddname of disk resident table entries in INTER-COMM execution JCL	VRBØØØ	RCTØØØ	DESØØØ
PMIFILET blocksize specification at installation time	75Ø	1ØØØ	75Ø
Symbolic Table Entry Library (created at installation time)	PMI.SYMVRB	PMI.SYMRCT	PMI.SYMDES
Load Module Table Entry Library (created at installation time)	PMI.MODVRB	PMI.MODRCT	PMI.MODDES
Table Entry Library member name convention	VRBnnnnn	RPTnnnnn	DESnnnnn
Coding convention within disk resident entry	VERB macro, RBN=nnnnn	REPORT macro, NUM=nnnnn	none
Core-resident table requirements.	VERBTBL CSECT: VERBGEN macro plus in-line assembly of disk resident entries	PMIRCNTB CSECT: None (OFT#-1 is used for RCTØ00rbn)	PMIFILET CSECT: GENFTBLE macro, DESNUM=DESØØØrbn or CHNGTB CSECT: DC A(DESØØØrbn)

Figure 25. Summary Requirements for Disk-Resident Table Entries.

The File Load Program

The File Load Program is a generalized batch utility supplied with INTERCOMM which transfers members of a Partitioned Data set to relative blocks in a BDAM data set. Given that member naming conventions are followed, it can be used to convert any PDS to BDAM.

The File Load Program is linkedited and used at INTERCOMM installation time to initially load all error messages generated via OUTPUT Utility OFT entries to disk. The following JCL may be used to linkedit:

```
//                EXEC LKEDP,Q=LIB,LMOD=PMIEXLD
//LKED.SYSIN      DD *
     INCLUDE      SYSLIB(BATCHPAK)
     INCLUDE      SYSLIB(PMIDCB)
     INCLUDE      SYSLIB(PMIFILET)
     INCLUDE      SYSLIB(PMISERC2)
     INCLUDE      SYSLIB(IXFABEND) (not required for VI.1)
     INCLUDE      SYSLIB(IJKDSPØ1)
     INCLUDE      SYSLIB(IXFHNDØØ)
     INCLUDE      SYSLIB(IXFHNDØ1)
     INCLUDE      SYSLIB (PMILOAD)
     ENTRY        PMILOAD
     NAME         PMIEXLD (R)
```

At file load execution time, a SYSIN control card specifies VRB, RCT, or DES or a user-assigned file identifier to define program logic, represented by XXX in the following naming conventions. A partitioned data set specified by ddname XXXLOAD is searched for member names XXXnnnnn. Members are loaded sequentially to the BDAM data set specified by a ddname XXXØØØ with the RBN being one less than nnnnn. Members are loaded until a member on the PDS is "not found". Hence, member names must ascend sequentially from XXXØØØØ1 to XXXnnnnn, incremented by 1 for each new member added to the PDS.

With INTERCOMM VI.1 and subsequent versions, the File Load Program has been modified allowing the SYSIN data set to specify replacement of one existing BDAM data set table entry or creation of the BDAM data set from a given range of PDS member names regardless of the existence of member names in ascending sequential order or not. See "INTERCOMM VI.1 Enhancements" at the end of this chapter.

The actual JCL for loading the disk-resident table for each of the Utilities is detailed below. Each Utility's table entries may be loaded as a separate job step, or the JCL may be combined to load all three data sets (VRBØØØ, RCTØØØ, and DESØØØ) in one step.

Loading ECT Entries to Disk

1. Create a symbolic library (PMI.SYMVRB) and a load module library (PMI.MODVRB) for the EDIT Control Table (ECT) entries on disk. These libraries are normally allocated and catalogued at INTERCOMM installation time.
2. Add the appropriate ECT to the symbolic library (PMI.SYMVRB). On the ADD card, code NAME=VRBXXXXX where XXXXX corresponds to the RBN operand of the VERB macro for this particular entry. Disk resident entries must be named VRBØØØØ1 to VRBnnnnn, numbered sequentially. The following JCL might be used:

```
// EXEC LIBE,Q=VRB
./ ADD NAME=VRBØØØØ1
*ECT ENTRY TO BE LOADED TO DISK
   VERB - - -,RBN=Ø1
   PARM
    :
   PARM
/*
```

Note the absence of CSECT, PMISTOP, and END cards. These members might also be COPYed during the assembly of the core-resident ECT. The symbolic form of the table entry is now VRBØØØØ1 on PMI.SYMVRB.

3. Assemble and link-edit the ECT into the load module library (PMI.MODVRB). The following JCL might be used:

```
//            EXEC    ASMPCL,Q=VRB,LMOD=VRB0001,NAME=VRBØØØØ1
/*
```

Ignore the assembly errors for lack of CSECT and END card. The load module form of the table entry is now member VRBØØØØ1 on PMI.MODVRB.

4. Load all disk-resident ECT's (VRBØØØØ1 to VRBnnnnn) to the EDIT Control File, ddname=VRBØØØ. The following JCL might be used:

```
//            EXEC    PGM=PMIEXLD,PARM='NOCHECK'
//STEPLIB     DD      DSN=PMI,MODLIB,DISP=SHR
//VRBØØØ      DD      DSN=VRBØØØ,DISP=(NEW,KEEP),
//                    SPACE=(    ),
//                    DCB=(DSORG=PS,BLKSIZE=nnn,RECFM=F),
//                    UNIT=SYSDA,VOL=SER=nnnnnn
//VRBLOAD     DD      DSN=PMI.MODVRB,DISP=SHR
//SYSPRINT    DD      SYSOUT=A
//SYSIN       DD      *
VRBØØØ
/*
```

The members from the partitioned dataset VRBLOAD are now blocks on the BDAM dataset VRBØØØ, The actual RBN is one less than tne number XXXXX described above.

Loading OFT Entries to Disk

To create the Output Format Table, the following procedures must be followed:

1. Create a symbolic library (PMI.SYMRCT) and a load module library (PMI.MODRCT) for the Output Format Table (OFT) entries on disk. These libraries are normally allocated and catalogued at INTERCOMM installation time.
2. Add the appropriate OFT to the symbolic library (PMI.SYMRCT). On the ADD card, code NAME=RPTXXXXX where XXXXX corresponds to OFT number for this particular entry. INTERCOMM error messages generated from disk resident OFT entries are contained on PMI.SYMREL and PMI.MODREL with member names RPTØØØØ1 to RPTØØØ5Ø. User OFT entries to be loaded to disk must be named RPTØØØ51 and up, numbered sequentially. The following JCL could be used:

```
//             EXEC          LIBE,Q=RCT
//LIB.SYSIN DD               *
./             ADD           NAME=RPTØØØ51
* OFT NUMBER 51- DISK RESIDENT
               REPORT        NUM=51
               .
               .
               .
/*
```

Note the absence of CSECT, PMISTOP, and END cards. The symbolic form of the table entry is now RPTØØØ51 on PMI.SYMRCT.

3. Assemble and linkedit the OFT into the load module library (PMI.MODRCT). The following JCL might be used:

```
//    EXEC ASMPCL,Q=RCT,NAME=RPTØØØ51,LMOD=RPTØØØ51
/*
```

Ignore the assembly errors for lack of CSECT and END cards. The load module form of the table entry is now member RPTØØØ51 on PMI.MODRCT.

4. Load all disk-resident OFT's (RPT00001 to RPTnnnnn) to the Output Format File, ddname-RCT000. The following JCL might be used:

```
//          EXEC     PGM=PMIEXLD,PARM='NOCHECK'
//STEPLIB   DD       DSN=PMI,MODLIB,DISP=SHR
//RCT000    DD       DSN=RCT000,DISP=(NEW,KEEP),SPACE=(     ),
//                   DCB=(DSORG=PS,BLKSIZE=nnn,RECFM=F),
//                   UNIT=SYSDA,VOL=SER=nnnnnn
//RCTLOAD   DD       DSN=PMI.MODRCT,DISP=SHR
//          DD       DSN=PMI.MODREL,DISP=SHR
//SYSPRINT  DD       SYSOUT=A
//SYSIN     DD       *
RCT000
/*
```

The members from the partitioned data set RCTLOAD are now blocks in the BDAM dataset RCT000. The actual RBN that the OFT entry will occupy is one less than the number XXXXX (defined above).

NOTE: To change the Output Format Table block size from the Intercomm release, the following steps should be taken:

1. The File Table (PMIFILET CSECT) entry created via the GENFILE macro for RCT000 must be changed from BLKSIZE-1000 to BLKSIZE=nnnn, where nnnn is the new block size. PMIFILET must be reassembled.

2. The File Load program (PMIEXLD) must be relinkedited to contain the new block size.

3. Execute the File Load program (PMIEXLD) in RCT000 with the JCL shown above.

Loading FDR Entries to Disk

To create the Format Description File, the following procedures must be followed:

1. Create a symbolic library (PMI.SYMDES) and a load module library (PMI.MODDES) for the Format Description Records. These libraries are normally allocated and catalogued at Intercomm installation time.

2. Add the appropriate FDR to the symbolic library (PMI.SYMDES). On the ADD card, code NAME=DESxxxxx where xxxxx is one greater than the resulting RBN after the file

load program is executed. The user must relate the corresponding ddname and FDR by the DESNUM parameter of the GENFTABLE macro for user files, or relate the fixed format identifier and FDR by the number indicated in the CHANGE table for fixed format message text descriptions. The first record should be DES00001; the second, DES00002; etc. No number should be skipped. The following JCL can be used:

```
//       EXEC          LIBE,Q=DES
//LIB.SYSIN   DD       *
./       ADD           NAME=DES00001
* FORMAT DESCRIPTION RECORD 1
DES00001  CSECT
          FDHDR
          FDETL
          FDETL
          END
/*
```

The symbolic form of the table entry is now member DESØØØØ1 on the library PMI.SYMDES.

3. Assemble and link-edit the FDR into the load module library (PMI.MODDES).

   The following JCL might be used:

```
//      EXEC      ASMPCL,NAME=DESØØØØ1,LMOD=DESØØØØ1,
                  Q=DES
/*
```

   The load module form of the table entry is now member DESØØØØ1 on the library PMI.MODDES.

4. Load all FDR's (DESØØØØ1 to DESnnnnn) to the Format Description File, ddname=DESØØØ.

   The following JCL might be used:

```
//      EXEC      PGM=PMIEXLD,PARM='NOCHECK'
//STEPLIB         DD   DSNAME=PMI.MODLIB,DISP=SHR
//DESØØØ          DD   DSNAME=PMI.DESØØØ,
//                     DISP=(,KEEP),SPACE=(TRK,(1,1)),
//                     UNIT=XXXX,VOL=SER=XXXXXX,
//                     DCB=(DSORG=PS,BLKSIZE=XXXX,
//                     RECFM=F)
//DESLOAD         DD   DSN=PMI.MODDES,DISP=OLD
//SYSPRINT        DD   SYSOUT=A
//SYSIN           DD   *
DESØØØ
/*
```

   The members from the partitioned dataset DESLOAD are now blocks on the BDAM dataset DESØØØ. The actual RBN that the FDR will occupy is <u>one</u> LESS than the number XXXXX (defined above).

INTERCOMM VI.1 Enhancements

With INTERCOMM VI.1 (April, 1974) and subsequent versions, the File Load Program has been modified allowing the SYSIN data set to specify replacement of a specific member of the partitioned data set or loading all members within a specified range of member names.

- Example 1: To copy PDS member name XXX00007 to the existing BDAM data set XXXØØØ; the SYSIN data set specifies:

  ```
  //SYSIN DD *
  XXX00007
  ```

- Example 2: to create the BDAM data set XXXØØØ from PDS member names XXX00001 to XXXØnnnn irrespective of the number of actual members on the PDS. The SYSIN data set specifies:

  ```
  //SYSIN DD *
  XXX000-nnnn
  ```

  The File Load Program will copy PDS members in ascending sequence to the BDAM data set XXXØØØ beginning with XXX00001. When a "member not found" condition arises, the file load does not terminate (as was previously the case) but the last member found will be copied to the BDAM data set until the next "member found" occurs, or the "upper limit" member XXXØnnnn is encountered. To illustrate, assume members RPT00001 to RPT00050, RPT00100 to RPT00106 exist on the library PMI.MODRCT. Specification to the File Load Program as follows;

  ```
  //SYSIN DD *
  RCTØØØ-0110
  ```

  will cause creation of a BDAM data set (RCTØØØ) with 110 RBN's. The member RPT00050 will be duplicated in RBN's 49 to 98 (once as the actual table entry RBN 49, repeated until RPT00100 is found and loaded to RBN 99). The member RPT00106 will be duplicated in RBN's 105 to 109.

There is no limit to the number of control cards input via the SYSIN data set. Further, given the proper JCL as discussed earlier, several BDAM data sets may be updated or recreated in one execution of the File Load Program.

This section provides information pertaining to special considerations for terminals with non-standard operating characteristics. Some of the data covered in this section is as follows:

- Special operating features such as function keys, and field formatting

- Restrictions which may be peculiar to a specific terminal.

It is assumed the reader has knowledge of the actual hardware operating specifications of each terminal type as described by the manufacturer's publications.

This section may also reference required specifications for Front-End Tables as appropriate (also documented in the INTERCOMM Operating Reference Manual).

THE IBM 3270

All IBM 3270 Display System components are supported by the INTERCOMM BTAM Front-End including the following:

IBM 3271 control unit, models 1 and 2

IBM 3272 control unit, models 1 and 2

IBM 3275 display station, models 1 and 2

IBM 3277 display station, models 1 and 2

IBM 3284 printer, models 1 and 2

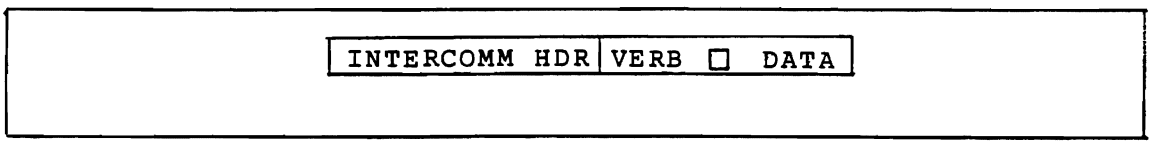
Selector pen

Operator identification card reader

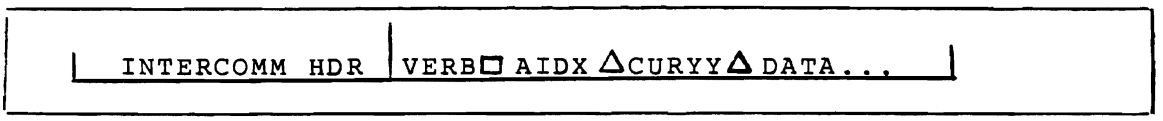
Both the IBM 3270 Remote and the IBM 3270 local display systems are supported by the INTERCOMM BTAM Front-End. Other IBM BISYNC devices are also supported on the same line as the IBM 3270 display system. The question of whether or not the 3270 display system is local or remote is transparent to the application program.

INPUT Message Formats

The format of the input to the application (or the EDIT Utility) depends on whether or not the user intends to use the aid or the cursor address. If the user desires neither, he should either code HDR3270=NO or omit this parameter from the BTVERB macro used to define his verb in the Front-End Verb Table (BTVRBTB). The following is an example of an input transaction without a 3270 header.



If the user desires either the aid or the cursor address, he must code HDR3270=YES in his verb definition (BTVERB macro) in the BTVERBTB. The following is an example of an input message with the aid and the cursor address header (3270 header):



where

X=the aid byte; YY=the cursor position bytes. If the 3270 header is desired and the aid is that of an input that does not contain a cursor address, a dummy cursor address of "home" will be provided.

The format of DATA depends on whether or not the terminal is operating with "formatted" screen capabilities as shown in figure 26 and 27.

Bypassing the Edit Utility

If the application programmer desires, the input message may be processed without the use of the EDIT Utility. Under this method, either formatted or unformatted input is acceptable; however, all editing must be done by the application program. If the input screen is unformatted then VERB and DATA is what was entered on the screen. If the input screen is formatted VERB is the contents of the first "modified" field on the screen prefixed by their SBA sequences. In either case, the verb may be "locked", or part (or all) of the transaction may have come from the table used for program attention keys.

Using the EDIT Utility - Unformatted Input

If the input screen is unformatted and the aid and cursor position are not desired, the input to the EDIT Utility can be of any standard form currently accepted by the EDIT Utility (positional or keyword). If it is desired to use the 3270 in this mode, no additional facilities of edit are needed, no existing application programs or tables need be changed. The output of the EDIT Utility will be either standard form desired (fixed, variable).

If the input screen is unformatted and the aid or the cursor position is desired, the EDIT Control Table VERB definition must be modified to accept this additional data as parameters of the verb. If the input is keyword, an AID parm of 1 byte and a CUR parm of 2 bytes must be defined anywhere in the verb definition.

```
VERB1   VERB   VERB,Ø1,156,5,FIX=YES,KEY=YES
        PARM   CUS,Ø1,Ø,25,ØØØØØØ11
        PARM   ADR,Ø2,Ø,2Ø,ØØØØØØ11
        PARM   C/S,Ø3,Ø,25,ØØØØØØ11
        PARM   AID,Ø4,Ø,1,1ØØØØ111
        PARM   CUR,Ø5,Ø,2,1ØØØØ111
```

VERB*DATA1*DATA2

Unformatted screen
(e.g., positional)

Message Input - No Aid or cursor address:

INTERCOMM HDR	VERB*DATA1*DATA2	ETX

Message Input - With aid and cursor address:

INTERCOMM HDR	VERB*AIDX	NL	CURYY	NL	DATA1*DATA2	ETX

Figure 26. 3270 Input Message Format - Unformatted Screen

□VERB□ NAME□XXX
ADDR□YYY

VERB,XXX,YYY fields have modified data tags on

Formatted screen

Message Input - No aid or cursor address:

INTERCOMM HDR	VERB*	SBA seq XXX	SBA seq YYY	ETX

Message Input - With aid and cursor address:

INTERCOMM HDR	VERB*	AIDX	NL	CURYY	NL	SBA seq XXX SBA seq YYY	ETX

Figure 27. 3270 Input Message Format - Formatted Screen

If the input is positional, an AID parm of 4 bytes and a CUR parm of 5 bytes must be defined as the first 2 parms of that verb. (If LINE=YES is specified PMIELIN macros must be coded):

```
VERB2   VERB    VERB,Ø2,256,5,FIX=YES,KEY=NO
        PARM    AID,Ø4,Ø,4,1ØØØØ111
        PARM    CUR,Ø5,Ø,5,1ØØØØ111
        PARM    CUS,Ø1,Ø,25,ØØØØØ11Ø
        PARM    ADR,Ø2,Ø,2Ø,ØØØØØ11Ø
        PARM    C/S,Ø3,Ø,25,ØØØØØ11Ø
```

For positional input the letters AID and CUR will be passed to the application as data along with the actual aid and cursor position. (A special Edit Routine may be written to strip this off.) If either PARM is coded with a length of zero, that parameter will be omitted from the output of the EDIT Utility.

Using the EDIT Utility - Formatted Input

A facility has been added to the EDIT Utility to allow the processing of 3270 formatted input. With this facility, the screen data will be treated as positional data. Instead of a separator indicating the position of the field, a buffer address is used. It is required that the user code the buffer position of each field as the CHAR-ID of that field in the PARM macro when defining that verb in the EDIT Control Table. (For standard positional input, the CHAR-ID is used only for error reporting identification.)

If the AID or CURSOR address is desired, an AID parm of 4 bytes and a CUR parm of 5 bytes must be coded as the first 2 parms of that verb. The output from the EDIT utility will be either of the standard formats (i.e., fixed, variable).

Figure 28 shows a formatted screen appearing at the terminal allowing the operator to enter customer name, salary, title, and phone #, an input message processed by EDIT, the ECT, and EDIT results.

Alternatively, the user may code PREONLY=YES in the Edit Control Table VERB macro. In this case the SBA sequence in the incoming message will be replaced by the system separator character. The individual data fields will not be edited. This technique effects conversion of a 3270 formatted input message to standard positional input as if the message forwarded to the subsystem had come from an IBM 2260-type terminal.

Formatted Screen

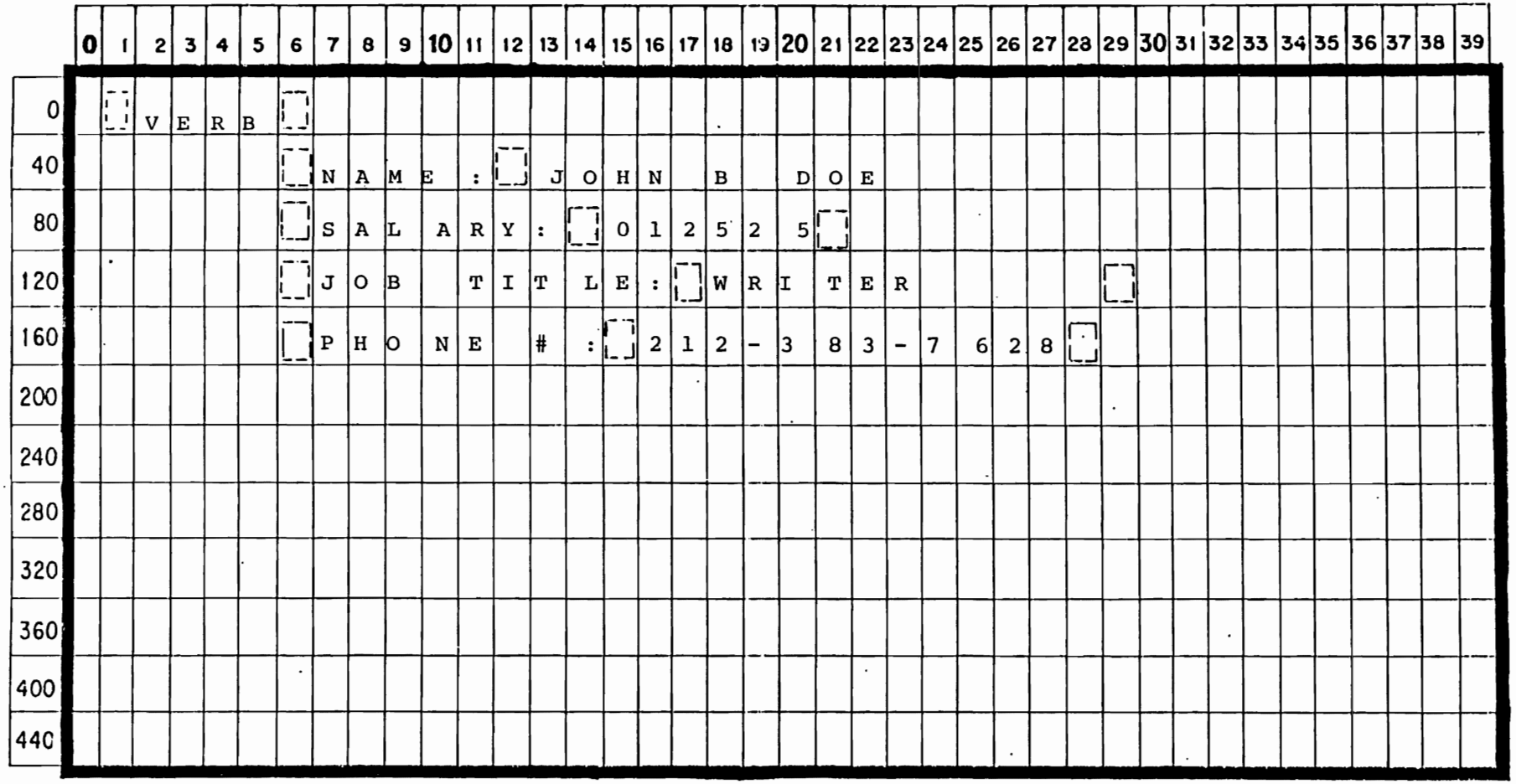


Figure 28. EDIT example - Formatted Screen (Sheet 1 of 2)

INPUT TO EDIT UTILITY

Screen position*

MSG HEADER	VERB	*	AID	X	NL	CUR	C2	7C	NL	SBA	40F5	JOHN B DOE	SBA	C15F	012525

IN BTVERB
HDR3270=YES

SBA	C24A	WRITER	SBA	C2F0	212-383-7628	ETX

*All SBA addresses really are in buffer address form

EDIT Control Table Entry:

```
VERB    VERB    VERB,01,256,6,FIX=YES,KEY=NO,LINE=NO
        PARM    AID,05,0,4,10000111
        PARM    CUR,06,0,5,10000111
        PARM    40F5,01,0,22,10000111
        PARM    C15F,02,0,6,10000111
        PARM    C24A,03,0,11,10000111
        PARM    C2F0,04,0,12,10000111
```

RESULT OF EDIT PROCESSING (Input to Application Program):

MSG HEADER 01	AID	X	CUR	C27C	JOHN B DOE	012525	WRITER	212-383-7628

Figure 28. EDIT example - Formatted Screen (Sheet 2 of 2)

IPN:031 2/1/74

"ATTENTION" Selector Pen Input

If the input is generated by an operator touching a selector pen "ATTENTION" field, the input buffer will consist of SBA sequences only, no data will be sent in from any "modified" fields read. Therefore, the user is advised not to define data entry fields in any format screen that contains an "attention field". (Data entry fields may be defined with "selection" fields and "selection" fields may be defined with "attention" fields.)

Since no data is received from the screen, a verb will have to be appended to the contents of the input buffer. In order for the verb to be appended, either the terminal should be "locked" to a verb or data with a verb should be specified in the Front-End table entry defined for this AID.

For all fields read, the EDIT Utility will generate an "X" as the data read. In other words, an "attention" screen will be processed by the EDIT Utility as though each field read contained an "X". In the verb definition in the EDIT Control Table each field should be defined as a one byte field. If a field is read, the output from the EDIT Utility will have an X as the contents of the field.

If the EDIT Utility is not used, the appended prefix to the screen will be followed by only SBA sequences (from the screen) and an ETX.

Output Message Formats

All messages going to any 3270 must begin with a valid command (i.e., write: C'1', erase write: C'5', erase all unprotected: C'?'), a WCC, and an SBA sequence. In addition all messages going to a remote 3270 must be prefixed by an STX and an ESC.

The INTERCOMM BTAM front-end will prefix all output messages for a given device type (see BDEVICE STCHAR=). The remote 3270 BDEVICE macro must specify STCHAR=0227 (STX,ESC); the local 3270 may omit the STCHAR parameter from the BDEVICE macro.

The INTERCOMM back-end message OFT entries and INTERCOMM BTAM front-end messages do not contain any command, WCC or SBA sequence. A valid command, WCC and SBA sequence must be specified for a given device type (see BDEVICE CTCHAR=). For all BTAM front-end messages and for all INTERCOMM back-end messages that do not contain a valid command, this additional information will be taken from the CTCHAR= parameters specified on the BDEVICE macro and placed at the beginning of the message.

Bypassing Formatting by the OUTPUT Utility

The application program may build the entire output message including the command, WCC, SBA sequence and any orders and data needed. This message can be sent to OUTPUT using a VMI of X'57'; the OUTPUT Utility will simply forward that message to the front-end.

All messages written to a 3270 display system must contain a prefix containing a command sequence and an SBA sequence. As previously noted, the INTERCOMM Front-End will optionally prefix all output messages going to a specific device type. Using this option it is unnecessary to change either the application program or the OUTPUT Format Table. However, all messages to the same device type will have the same prefix.

If the application has OFT entries or messages already operational and these messages contain characters acceptable to the 3270, no program change or table change is necessary. NL characters do not function on a 3270 display. To allow NL characters in a 3270 screen an additional facility of the OUTPUT Utility translates all NL characters to SBA sequences.

Using the Output Utility - Minimal user modification

In order to have greater flexibility of type of command and SBA sequence, the user may include the prefix in the beginning of each output message. In order to utilize Output formatting capabilities for 3270 messages without using the additional formatting of the OUTPUT Utility, the application may define all control bytes in the message and thus eliminate NL characters.

Using the Output Utility-Extended Facilities for the 3270

The User may choose to implement the following OUTPUT Utility Features to fully utilize the hardware characteristics of the 3270.

Generating a Format Screen or Print Buffer:

In order to facilitate the use of the format generating features of the Output Utility, operands have been added to the REPORT and ITEM macros.

The COMM and WCC operands of the ITEM macro may be used to define a command and WCC to prefix a specific display. These keyword operands must be placed in the first ITEM macro coded; causing a command and a WCC to be generated in a 254 ITEM. (An SBA sequence to address zero will also be generated.) Since the command, the WCC and the SBA sequence take up 5 positions, the first ITEM macro must be coded with an additional 5 positions in the TO=parameter.

The ATT operand of the ITEM macro may be used to define a field. If this parameter is used, an SF order and an attribute byte will be generated to precede the item defined. If the ATT operand is omitted no SF and ATT byte is generated. The SF and the attribute take up 2 positions in the buffer used to create the report. Therefore, any ITEM macro that specifies the ATT= parameter must be coded with an additional two positions in the TO= parameter.

In order to fully utilize the features of the IBM 3270; if more than 4 contiguous blanks are to be transmitted, or more than 4 of the same non-blank characters, the "repeat to address" sequence will be generated by the OUTPUT Utility. If it is desired to define more than 4 contiguous blanks or more than 4 of the same non-blank characters, these positions in the OFT should be defined as unique items; a following data item starts a new field. Any order that changes the current buffer address (SBA,RTA,EVA,TAB) should not be defined within data items unless that item is the last item of the message and the data begins with an SBA order.

Any NL orders can be inserted by either the application or the OUTPUT Utility. Any Insert Cursor or EM orders should be specified in regular data items. Any order that takes up a buffer position should be coded as an item code 255 item. Any order that does not take up a buffer position should be coded as an item code 254 item. If the report is to go to a screen (WCC≠PRINT) any NL orders will be replaced by an SBA sequence by the Output Utility.

As previously noted, an SBA order to buffer position ZERO will be generated by the first ITEM macro. If it is desired to have a display start at other than buffer position ZERO, specify as many NL characters as needed to position the report at the desired line location.

Figure 29 illustrates a sample OFT for Screen Generation, resulting display at the 3270; and an input message to OUTPUT requesting the OFT screen generating entry, and the actual message produced by OUTPUT.

Filling in a Screen Format:

In order to facilitate the filling in of unprotected items of a screen format (already displayed) by an application program, a VMI of X'56' has been added to the Output Utility message types. With this facility the application program can pass to the Output Utility the report number of the screen format along with the data to be filled in. The report number and the data are passed in the variable format (e.g., IC, LEN, DATA). The data items passed are to be displayed in fields defined as unprotected. The Output Utility will generate any SBA and TABS needed and put an insert cursor order after the first tab. Any field shorter than its length on the screen will be padded with nulls on the right; any unprotected field not passed data will be skipped.

OUTPUT FORMAT TABLE ENTRY:

```
RPT00051    CSECT
RPT51       REPORT    NUM=51,LINES=5
LINE1       LINE      NUM=01,ITEMS=3
            ITEM      CODE=255,FROM=1,TO=6,DATA='Ø',COMM=ERASE,WCC=(RESET,RESTORE)
            ITEM      CODE=255,FROM=7,TO=12,ATT=(PRO,SKIP,HIGH,MOD),DATA='VERB'
            ITEM      CODE=255,FROM=13,TO=15,ATT=(PRO,SKIP),DATA='Ø'
LINE2       LINE      NUM=2,ITEMS=4,REPET=6
            ITEM      CODE=255,FROM=6,TO=12,ATT=(PRO,SKIP),DATA='NAME:'
            ITEM      CODE=255,FROM=13,TO=15,ATT=YES,DATA=(X'13') INSERT CURSOR
            ITEM      CODE=Ø1,FROM=16,TO=38
            ITEM      CODE=255,FROM=39,TO=41,DATA='Ø',ATT=(PRO,SKIP)
LINE3       LINE      NUM=03,ITEMS=3,REPET=6
            ITEM      CODE=255,FROM=7,TO=15,ATT=(PRO,SKIP),DATA='SALARY:'
            ITEM      CODE=02,FROM=16,TO=23,ATT=NUM
            ITEM      CODE=255,FROM=24,TO=26,ATT=(PRO,SKIP),DATA='Ø'
LINE4       LINE      NUM=04,ITEMS=3,REPET=6
            ITEM      CODE=255,FROM=7,TO=18,ATT=(PRO,SKIP),DATA='JOB TITLE:'
            ITEM      CODE=03,FROM=19,TO=31,ATT=YES
            ITEM      CODE=255,FROM=32,TO=34,ATT=(PRO,SKIP),DATA='Ø'
LINE5       LINE      NUM=05,ITEMS=3,REPET=6
            ITEM      CODE=255,FROM=7,TO=15,ATT=(PRO,SKIP),DATA='PHONE #:'
            ITEM      CODE=04,FROM=16,TO=29,ATT=YES
            ITEM      CODE=255,FROM=30,TO=32,ATT=(PRO,SKIP),DATA='Ø'
```

Figure 29. Sample 3270 Screen Generation. (Sheet 1 of 3)

IPN: 079 12/31/74

INPUT TO OUTPUT UTILITY

HEADER	FF020033

OUTPUT FROM OUTPUT UTILITY (including OUT327Ø)

ERASE WRITE	RESET RESTORE	SBA	00	Ø	SF	PRO SKIP HIGH MOD	VERB	SF	PRO SKIP	SBA	40

RTA	46	Ø	SF	PRO SKIP	NAME:	SF	NO BITS ON	IC	RTA	75	Ø	SF	PRO SKIP	SBA	80

RTA	86	Ø	SF	PRO SKIP	SALARY:	SF	NUM	RTA	101	Ø	SF	PRO SKIP	SBA	120

RTA	126	Ø	SF	PRO SKIP	JOB TITLE:	SF	NO BITS	RTA	149	Ø	SF	PRO SKIP	SBA	160

RTA	166	Ø	SF	PRO SKIP	PHONE #:	SF	NO BITS	RTA	188	Ø	SF	PRO SKIP	ETX

Figure 29. Sample 3270 Screen Generation. (Sheet 2 of 3)

IPN: 079 12/31/74

DISPLAY AT 3270:

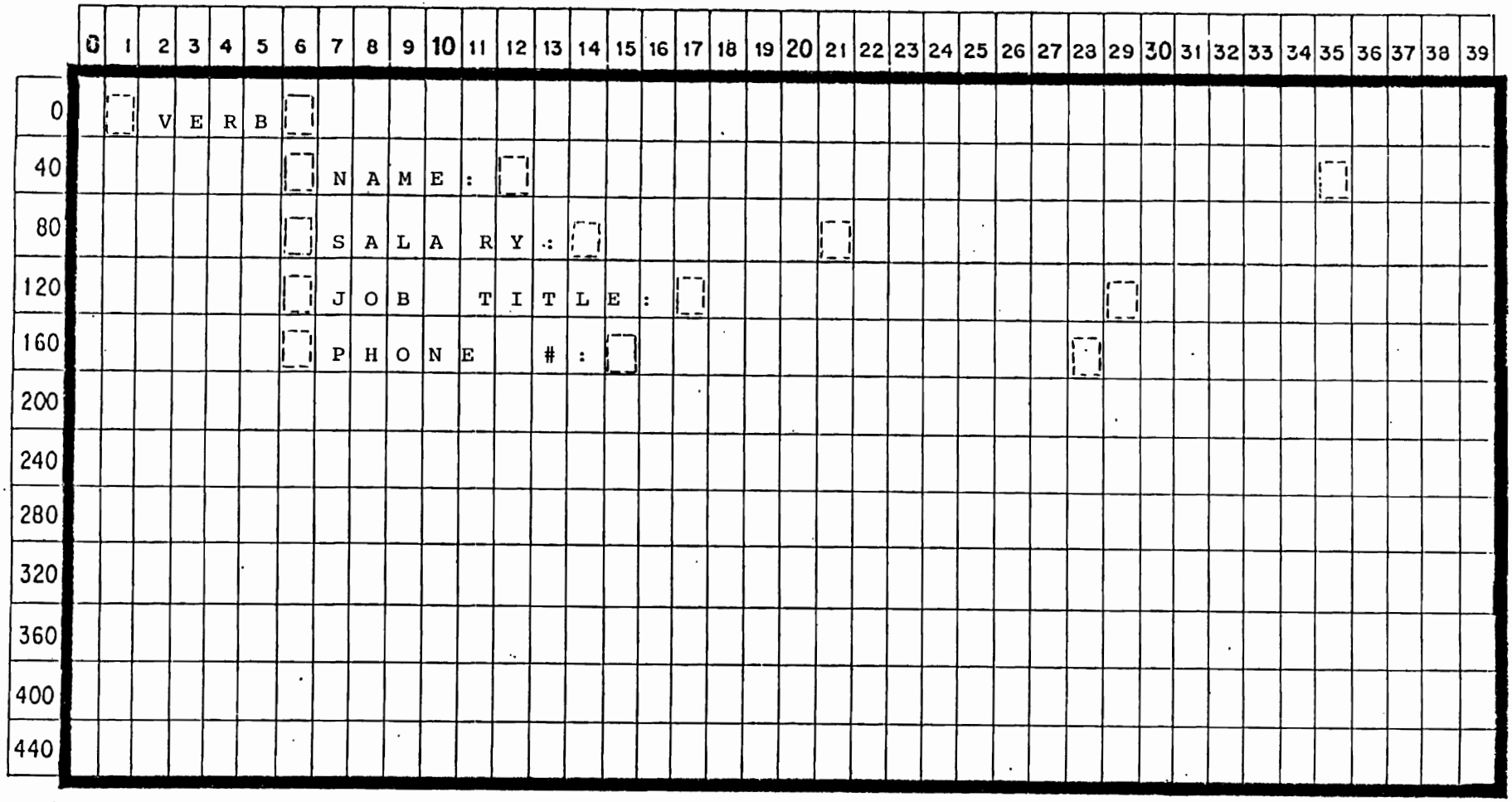


Figure 29. Sample 3270 Screen Generation. (Sheet 3 of 3)

The command, WCC, and SBA sequence will be inserted by the Output Utility. The default command for all X'56' VMI output messages is a Write command. The default WCC is restore keyboard and not reset modified data tags. The default SBA sequence is an SBA to zero. In order to erase all unprotected data and reset modified data tags, when filling in a screen format, the ERASE operand may be specified in the REPORT macro of the OFT entry. Since an EAU command cannot contain data, an erase unprotected to address order will be used instead.

If the application desires other than the default command, WCC and SBA sequence (or ERASE=YES), he should send the command sequence in an item code 251 item, passed to the Output Utility in the VMI of X'56' message. The 251 ITEM does not get coded in the report. If the data following ITEM CODE 251 in the message text contains 1 byte that byte will become the command; if it contains 2 bytes those bytes will become the command, WCC sequence; if it contains 5 bytes those bytes will become the command, WCC, SBA sequence.

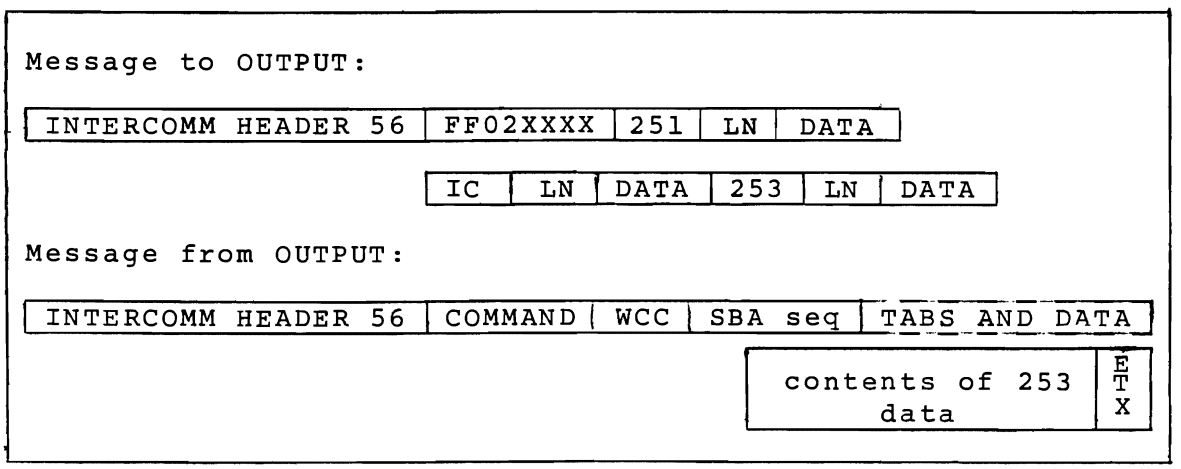
If an erase all unprotected (EAU) command is placed in a 251 item and there is also a data item passed, an Erase Unprotected to Address order will be used to erase the unprotected fields.

Figure 30 illustrates an existing screen format, a message to OUTPUT, and the resulting message to the terminal.

Modifying Screen Format:

It may be necessary for the application program to modify definition of fields or field data (other than unprotected) in a format already on the screen.

In order to facilitate this, a special 253 item has been added. The 253 item does not get coded in the report. When the application program passes the report number to the output utility (via a VMI of X'56') he can pass all orders or data needed along with a 253 item. The contents of the item will be placed at the end of the output message before the ETX is added by the Output Utility. For example:



AN EXISTING SCREEN FORMAT

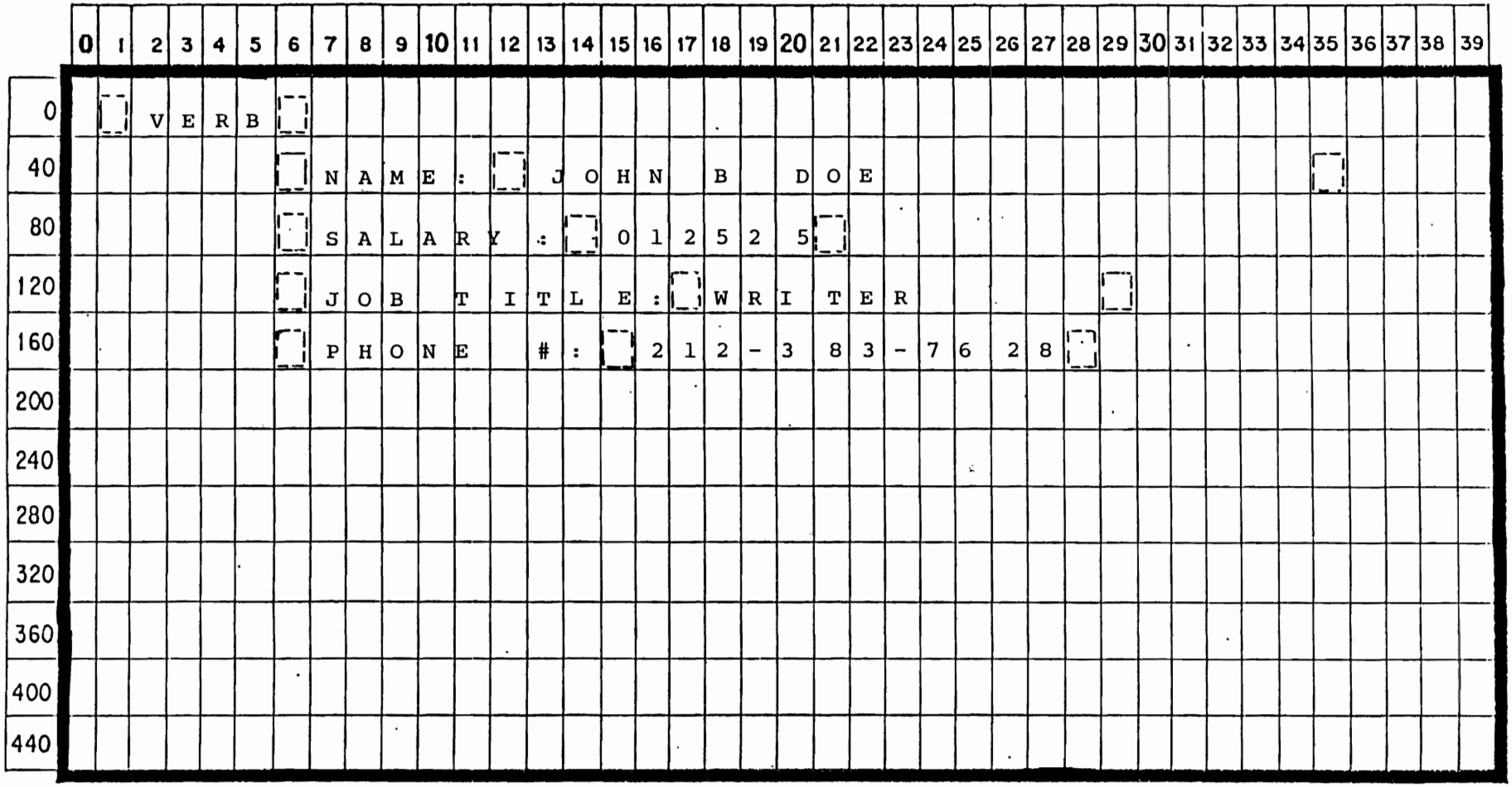


Figure 30. Filling in a 3270 Screen Format (Unprotected Data) (Sheet 1 of 2)

INPUT TO OUTPUT UTILITY

MESSAGE HEADER 56 VMI	FF	02	0033	01	0A	JOHN B DOE	02	06	012525

03	06	WRITER	04	OC	212-383-7628

OUTPUT FROM OUTPUT UTILITY

WRITE	RESTORE RESET	SBA	00	TAB	JOHN B DOE	TAB	012525	TAB	WRITER	TAB	212-383-7628	ETX

Figure 30. Filling in a 3270 Screen Format (Unprotected Data) (Sheet 2 of 2)

IPN: 079 12/31/74

Adding to an Existing OFT Dynamically:

To modify an existing OFT (more than just data), the application passes all necessary orders and data in a special 253 item, along with other items, in the (VMI of X'56') message. (The 253 ITEM does not get coded as an item in the report.)

This special item must begin with an SBA order. The contents of the item will be placed at the end of the output message before an ETX is added by the OUTPUT Utility.

INTERCOMM VI.1 Enhancements

With INTERCOMM VI.1 (April 1974), a new operand was added to the REPORT macro to simplify specification of FROM and TO positions of data in ITEM macros. The optional operand DEV=3270 specifies that the receiving terminal is an IBM 3270 Model 2 (Remote or Local) and that the OUTPUT Utility is to be responsible for generating all hardware dependent control characters. Note that use of this feature results in terminal-dependent Output Format Table entries which must not be referenced when a message is destined for a terminal other than the IBM 3270. The Output Format Table may contain a combination of entries of which some specify the DEV=3270 operand on their REPORT macros.

If DEV=327Ø is specified, the FROM and TO values specified on subsequent ITEM macros indicate only those characters in the data item which occupy a hardware buffer position. That is, the FROM and TO may reference the actual positions in a line of 327Ø display. Additionally, SBA sequences will be used for positioning to the start of a new field. LINE macros may not define repetitive lines (REPET=1, 4 or 8 not allowed).

If DEV=327Ø is not specified, the generation of the report will be exactly as it was before the DEV= operand was available. That is, all characters generated for a given field must be reflected in the FROM and TO specification, including characters such as SF or IC which do not occupy a 327Ø buffer position. Positioning to a new field is done by padding with blanks.

The following table entry illustrates an Output Format Table Coded with the DEV= 3270 operand.

```
RPT00060 CSECT
RPT0060  REPORT NUM=60,LINES=3,DEV=3270
         LINE  NUM=1,ITEMS=3
         ITEM  CODE=255,FROM=1,TO=5,ATT=(PRO,MOD),
               COMM=WRITE,WCC=RESTORE,DATA='VERB'
         ITEM  CODE=255,FROM=6,TO=6,ATT=PRO
         ITEM  CODE=01,FROM=16,TO=25,ATT=YES
         LINE  NUM=2,ITEMS=3
         ITEM  CODE=255,FROM=9,TO=20,ATT=HIGH,
               DATA='3270 LINE 2'
         ITEM  CODE=255,FROM=21,TO=21,ATT=PRO
         ITEM  CODE=02,FROM=27,TO=42,ATT=YES
         LINE  NUM=3,ITEMS=2
         ITEM  CODE=3,FROM=42,TO=45,ATT=YES
         ITEM  CODE=255,FROM=50,TO=61,ATT=PRO,
               DATA='3270 LINE 3'
               END
```

Teletype Dataspeed 40/1 and 2 (DS40)

The Teletype Dataspeed 40 Models 1 and 2 are supported for the Edit and Output Utilities, as descirbed below.

Edit Utility

If TYPE=DS40 is coded in the DEVICE and STATION macros for the DS40 terminal, Edit Utility processing accepts:

- The Horizontal Tab (HT) X'05' (required for input from formatted screens)
- The New Line (NL) X'15'

as field separator characters in addition to the system separator character. When editing positinal data, an HT followed by a second HT or NL, or an NL followed by an HT indicates absence of a field. However, an HT following the system separator after a verb assumes input from a formatted screen, not absence of the first positional field.

Output Utility

If DEV=DS40 is coded in the REPORT macro, the corresponding ITEM macro ESC parameter allows specification of ESC sequences and other control characters for output report formatting. Thus, formatted screens with protected and unprotected fields may be defined.

New Line as a line ending control character is automatically generated in Output Utility processing. This does not affect the current protected/unprotected status of the screen area.

Further details on message control characters, terminal operation considerations, and format design considerations are discussed in the Operating Reference Manual, Section 5.

NOTE: The Teletype Dataspeed 40 Model 4 terminal is IBM 3270 hardware compatible and is supported under Intercomm as such.

Leased 2780 Considerations

If TYPE=2 (and FIRST=YES) are coded on the DEVICE macro, the Output Utility assumes the terminal is a leased 2780. All messages processed by the Output Utility will start with 'X022761'. Each line will end with X'1F2761' instead of a CRLF value.

INTERCOMM TABLES FOR THE UTILITIES

Basic Tables are included on the INTERCOMM release library and must be modified for each installation. An asterisk indicates optional tables which may be generated individually at each installation according to application program requirements.

TABLE or CSECT Name	Description	Created by	SYMREL & MODREL Member Name
BROADCST	*OUTPUT Broadcast Table	DC's	PMIBROAD
CHNGTB	*CHANGE/DISPLAY Table for Fixed Format Identifiers	DC's	none
FDR(s)	*Format Description Records	FDHDR, FDETL macros	none
KEYTABLE	*CHANGE/DISPLAY Key Conversion Routine Table	DC's	none
PADDTBLE	EDIT Pad Character Table	PADD macro	PADDTBLE
PMIALTRP	*OUTPUT Alternate Format Table	PMIALTRN macro	none
PMIDEVTB	OUTPUT Device Table	DEVICE macro	PMIDEVTB
PMIFILET	CHANGE/DISPLAY File Table	GENFTBLE	PMIFILET
PMIRCNTB	OUTPUT Format Table	CSECT	PMIRCNTB
		REPORT, LINE, ITEM macros	
		PMISTOP	PMIRCEND
PMIRPTAB	*OUTPUT Format Terminal Table	DC's	none

INTERCOMM TABLES
FOR THE UTILITIES

(Continued)

TABLE or CSECT Name	Description	Created by	SYMREL & MODREL Member Name
PMISTATB	OUTPUT Station Table	STATION macro	PMISTATB
PTRNTBLE	*CHANGE/DISPLAY Field Editing Pattern Table	PATRN macro	none
REPTAPE	*OUTPUT Batch Report Table	DC's	none
VERBTBL	EDIT Control Table	VERBGEN, VERB, PARM, PMIELIN macros	PMIVERBS

VERB -- Define Edit Control Table Entry

The VERB macro is used in conjunction with the PARM, PMIELIN and PMISTOP macro instructions to generate an Edit Control Table for processing by Intercomm's Edit Utility. When used, the VERB macro subordinates one or more PARM macros, and if used, one or more PMIELIN macros. Each VERB macro instruction defines a unique Edit Control Table entry.

The form of the VERB macro instruction is as follows:

```
[symbol]  VERB   transaction-id,hex-id,{coresize},
                                      {256     }
                 {#-of-parm-instructions}
                 {0                     }
                 [,FIX={YES}]
                       {NO }
                 [,KEY={NO }]
                       {YES}
                 [,LINE={YES}]
                        {NO }
                 [,RBN=vrb000-rbn]
                 [,PREONLY={YES}]
                           {NO }
```

coresize
: specifies the length, in bytes, of storage required after editing, for the longest possible message associated with the transaction entry being defined. Note that the length of the message after editing is equal to the combined length of the edited fields plus the message header, and any applicable status bytes. Any "coresize" residue is freed. Code as a decimal value, 256 to 4095. The default is 256.

FIX=
: indicates the kind of format the Edit Utility is to produce. Coding YES indicates that the data is to be placed in a fixed format. Coding NO indicates that the data is to be placed in a variable itemcode/length/data format. The default is NO.

hex-id
: provides a hexadecimal transaction identification code. Code as a hexadecimal value, 1 to FF. A program calling the Edit Utility can expect to have this code returned in the one byte MSGHVMI field of the message header, if the value coded for #-of-parm-instructions is non-zero.

KEY=
: indicates whether the transaction format is keyword or positional. Coding YES indicates that it is either wholly in keyword transaction format, or in positional-within-keyword transaction format. Coding NO indicates that it is wholly in positional transaction format. The default is YES.

LINE=
: indicates whether PMIELIN macros are used within the transaction entry being defined. Coding YES indicates that they are used. Coding NO indicates that they are not used. The default is NO.

#-of-parm-instructions
: specifies the number of PARM instructions associated with the transaction entry being defined. Note that this is not necessarily the same as the number of PARM instructions actually coded. Refer to the PARM macro, REPT parameter. If the hex-id operand value is to be moved to MSGHVMI, a non-zero value must be specified for #-of-parm-instructions.

PREONLY=
: indicates whether the message is to be processed only by the pre-Edit facility which converts IBM 3270 input (SBA sequences) into standard unedited positional format. When coded YES (if input is from IBM 3270, with preformated screen), the above statement applies. When coded NO the message is edited in the usual way.

RBN=
: provides the relative block number of the VRB000 file at which the corresponding transaction format entry can be located. If the entry being defined is to be resident, this parameter is meaningless. However, if the entry is to be located within the VRB000 file, then the RBN value should reflect the last five digits of the member-name used to place the entry in the file. For example, if the member-name is VRB00006, then code RBN=6. Code as a decimal value, 1 to 8388607.

transaction-id
: provides the unique transaction character code, the verb name, for the transaction entry being defined. Code as a fixed length character string of four characters.

PARM -- Define Data Unit

The PARM macro instruction is used in conjunction with the VERB, PMIELIN, and PMISTOP macro instructions. These four instructions are employed together to furnish the Edit Control Tables required by Intercomm's Edit utility. When employed, the PARM macro instruction is subordinated to a VERB macro instruction, and in function provides that utility on a data unit basis with the information it requires to process it.

The form of the PARM macro instruction is as follows:

[symbol]	PARM	char-id,itemcode,editrout-#,max-length, editflags [,REPT={repetition-# 1}]

char-id
: provides the character symbol to be associated with the data unit upon which editing is to be performed. If "KEY=NO" has been coded on the associated VERB instruction, "char-id" is used by Intercomm only for error reporting identification. Code as a fixed length 3-character string. If the IBM 3270 pre-formatted screen input is being used, the parameter name must be 4 hex digits. These identify the parameter by 3270 buffer address.

itemcode
: provides the numerical identification to be associated with the data unit upon which editing is to be performed. If "FIX=YES" is coded on the associated VERB instruction, this identification is used only in processing internal to the Edit utility. Code as a decimal value, 0 to 254. Note that if the text represented by the PARM instruction is to be completely ignored (as might be desired for template screens), "itemcode" must be coded as a 0.

editrout-#
: provides the number of the subroutine to be used to perform editing upon the data unit. Note that this number designates the editing subroutine whose three-digit csect-name suffix is identical in value i.e. a code of 8 designates subroutine EDIT008. Code as a decimal value, 0 to 255.

max-length
specifies the maximum permissible length, in bytes, within which the data unit, after editing, is permitted to be contained. Note that if "FIX=YES" is coded on the associated VERB instruction, "max-length" is always allocated regardless of actual data length. Code as a decimal value, 0 to 255.

editflags
provides specification and procedural information. Code as a fixed length bit string of 8 bits whose bit signification is as follows:

Bit 0	Code 1 if the data unit is a required item within the transaction format under definition. Code 0 if the data unit is not a required item.
1-3	Reserved. Code 000.
4	Code 1 if the data unit is permitted to be present more than once within a single transaction. Code 0 if the data unit is not permitted to be present more than once.
5	Code 1 if, after editing, the data unit is always to be contained within the length specified by the "max-length" parameter. Note that if "FIX=YES" is specified on the associated VERB macro instruction this bit will default to 1. Code 0 if the data unit is permitted to be assigned a variable length.
6-7	If a given data unit exceeds the "max-length" parameter value these two bits denote the action to be taken. Code 11 if no truncation is permitted. Code 10 if right truncation is permitted. Code 01 if left truncation is permitted. Code 00 is not meaningful.

REPT=
specifies to the macro processor the number of times the PARM macro instruction is to be generated. Note that the "#-of-'parm'-instructions" parameter on the associated VERB macro instruction must include this value. Note also that if the VERB instruction specifies "KEY=YES" and "FIX=NO", the default must be taken. If editflags bit Ø is on, then at least one data unit is required. Code as a decimal value, 1 to 100. The default code is 1.

VERBGEN -- Conditionally Assemble Edit Control Table PARM Macros

The VERBGEN macro instruction informs the internal processor of each PARM macro instruction that it is not to generate the code for that instruction if it is associated with a VERB macro instruction specifying an "RBN=" value. The VERBGEN macro instruction is used when creating the core-resident edit control and is meaningful only when one or more table entries are located within the VRB000 file.

The form of the VERBGEN macro instruction is as follows:

(blank)	VERBGEN	(blank)

PMIELIN -- Designate End of Line (Positional Transaction Format)

The PMIELIN macro instruction is used in conjunction with the VERB, PARM, and PMISTOP macro instructions. These four instructions are used together to create edit control tables for INTERCOMM's Edit utility. The PMIELIN macro instruction is used to delineate a line only when a positional transaction format is under definition. The generated code - C'$$$$' - signifies to the utility the legitimate end of expected data for a given input line and functions as a tab setting to which a skip is made when an end-of-line character (NL or CR/LF) is detected as having signified a permissible premature termination of that line.

The form of the PMIELIN macro instruction is as follows:

(blank)	PMIELIN	(blank)

PADD -- Define Padding Character

The PADD macro instruction supplies the EDIT Control Routine with the pad character to be used with each EDIT subroutine.

The form of the PADD macro instruction is as follows:

[symbol]	PADD	editrout-#,pad-char

editrout-#
provides the number of the subroutine to be used to perform editing upon the data unit. Note that this number designates the editing subroutine whose three-digit csectname suffix is identical in value i.e. a code of 8 designates subroutine EDIT008. Code as a decimal value, 0 to 255.

pad-char
specifies the character that is to be generated as a pad character to be used with each EDIT subroutine. The permissable codes are BLANK, BZER, ZERO, Char. Only one may be selected.

- BLANK - specifies that a blank (X'40') is to be generated as a pad character.
- BZER - specifies that a binary zero (X'00') is to be generated as a pad character.
- ZERO - specifies that a character zero (C'0') is to be generated as a pad character.
- Char - any user specified hexadecimal value. Code as a 2 digit hexadecimal value i.e. 3C.

REPORT -- Identify Reporting Format

The REPORT macro instruction is used in conjunction with the LINE and ITEM macro instructions to define a reporting format for the Output Utility. The REPORT macro subordinates one or more LINE macros. Each REPORT instruction identifies and defines a unique reporting format while the subordinated LINE instructions detail, line by line, the contents of that format.

The form of the REPORT macro instruction is as follows:

[symbol]	REPORT	NUM=report-number [,LINES={#-of-line-instructions / 1}] [,MASK={hex-mask / 0000}] IBM 3270, GTE IS 7800, and DS40 specifications: [,ERASE={YES / NO}] [,DEV={3270 / IS7800 / DS40}]

DEV=

DEV=3270 signifies an IBM 3270 Model 2 or a GTE IS7800 operating in 3270 mode. If DEV=3270 is specified, the FROM and TO values on subsequent ITEM macros include only those control and/or data characters which occupy a hardware buffer position. That is, the FROM and TO may reference the actual positions in a line of 3270 display. Additionally, an SBA sequence will be used for positioning to the start of a new field (attribute value). The LINE macros associated with this REPORT may not specify REPET=1, 4, or 8.

If DEV=3270 is not specified for a 3270, then all characters generated for a given field must be reflected in the FROM and TO specification, including characters such as SF or IC which do not occupy a 3270 buffer position. Positioning to a new field is done by padding with blanks.

DEV=IS7800 signifies that the REPORT is for a GTE IS7800 and that it may contain requests for blink, underlined, or inverted characters, and double width characters.

DEV=DS40 signifies a Teletype (Dataspeed) Model 40/1 or 2 terminal; this specification allows coding of the ESC parameter on the ITEM macro.

ERASE=

specifies whether or not an ERASE-unprotected-to-address order will be generated for messages processed by the Output Utility with header field MSGHVMI=X'56'. YES causes the message to contain a WRITE command, WCC of restore the key-board and reset modified data tags, and the ERASE order. The default is NO.

LINES=

specifies the number of LINE macro instructions used in the format being defined. Code as a decimal value, 1 to 255. The default code is 1.

MASK=

specifies by its bit configuration the terminal recipient class to which the report belongs. Since all classification is founded upon the relation noted below, each STATION macro MASK parameter must be co-ordinatively set in conjunction with this REPORT macro MASK parameter. The code is to be thought of as a 32 bit string written as a one to four hexadecimal character string with all codes not four characters being right adjusted then left padded with zeros. The following codes should be noted:

0000 - signifies that the associated report belongs to no unique terminal recipient class and is free to be received by any terminal within the system. This is the default code.

8000 - represents a class of error reports for which Intercomm's OUTPUT Utility will, in addition to forwarding the report, construct and attach a one-line prefix consisting (if printable) of the low order sending subsystem code followed by the report number.

The classification relation is as follows: If the STATION macro MASK code is ANDed into the REPORT macro MASK code and the latter's code remains unchanged, then the terminal assigned the former is considered eligible to receive the report assigned the latter.

NUM=

provides the logical number assigned to identify the report being defined. Code as a decimal value, 51 to 32767. Codes 1 to 50 are reserved.

NOTE: If the reporting format is to be placed on disk via a PMILOAD execution then this number must be embedded within the five-digit suffix of the format's member name. For example report number 62 provided with a member name of RPT00062 will be accessed via RBN 62.

symbol

If a label is provided it should not exceed seven characters in length. If its length exceeds seven characters, only the first seven will be recognized.

LINE -- Define Line

The LINE macro instruction is used in conjunction with the REPORT and ITEM macro instructions. These three macro instructions are employed together to define prescribed reporting formats for INTERCOMM's Output utility. When employed, the LINE macro instruction subordinates one or more ITEM macro instructions, and in function claims its domain over all those ITEM instructions associated with it. In principle, the LINE instruction defines a unique line within the reporting format under definition while the subordinated ITEM instructions detail segment by segment the constitution of that line.

The form of the LINE macro instruction is as follows:

[symbol]	LINE	NUM={line-number / 1} ,ITEMS={#-of-'item'-instructions / 1} ,REPET={line-control-code / 0}

NUM=
assignes a logical number to the line. This number must be one unit higher in value than that assigned to the preceding LINE macro instruction within the reporting format under definition. If there are no preceding LINE instructions this value must be 1. "line-number" is to be coded as a decimal value, 1 to 255. The default code is 1.

ITEMS=
specifies the number of ITEM macro instructions that are to be associated with the LINE macro instruction. The default code is 1. Code as a decimal value, 0 to 24. A code of 0 will provide one blank line.

NOTE: The maximum number of coded ITEMs per line is 24 for 3270 or IS7800 users who have coded DEV=3270 or IS7800 in the REPORT macro because as many as three ITEMs can be generated for each coded ITEM. For all non 3270 users, the maximum ITEMs per line is 72.

REPET=

provides the line control code for the line under definition. The permissable character codes are 0,1,4,5,6,8. Only one may be selected.

0 - designates a non-repetitive line.
1 - designates a repetitive line.
2 - is no longer in use.
3 - is no longer in use.
4 - designates a repetitive line and requests that the line's constant data is always to be included in the report even if no variable data exists for the line.
5 - designates a non-repetitive line and is to be used only with respect to segmented messages. It specifies that the line is to be included in the report only when a message whose MSGHVMI field is X'51' is received. This does not apply to messages whose MSGHVMI field is X'5C'.
6 - designates a non-repetitive line and requests that the line's constant data is always to be included in the report even if no variable data exists for the line.
8 - designates a repetitive line and requests that if a buffer or screen overflow occurs, the preceeding non-repetitive line(s) already contained within the buffer or screen are to be repeated within the subsequent buffer or screen. The default code is 0.

Repetitive lines may not be used in a table entry where the REPORT macro specifies DEV=3270 or IS7800.

ITEM -- Define Line Segment

The ITEM macro instruction is used in conjunction with the REPORT and LINE macro instructions to define reporting formats for Intercomm's Output Utility. The ITEM macro is subordinate to a LINE macro. Its primary function is to define a segment of the line and assign a unit of text for it. The ITEM instruction directly or indirectly designates a set of characters that ultimately become part of the output character stream; the ITEM macro instruction may therefore have a secondary function of inserting non-texted (for example, device control) characters into that stream. Refer particularly to the CODE parameter.

The forms of the ITEM macro instruction is as follows:

[symbol]	ITEM	FROM=column,TO=column,CODE=itemcode [,DATA={'constant-data' / (con-el,con-el,...)}] IBM 3270 and GTE IS7800 specifications: [,COMM={X'nn' / write / ERASE}] [,WCC={X'nn' / (subparameters)}] [,ATT={X'nn' / YES / (subparameters)}] GTE IS7800 specification: [,ORDER=SC] Teletype Model 40/1 or 2 specification: [,ESC={esc-code / (esccode1,esccode2,...,)}]

ATT=

(for IBM 3270 and GTE IS7800 only)
Use of this parameter causes an IBM 3270 field prefix to be generated consisting of a SF order and an attribute byte. If DEV=3270 or IS7800 has been specified on the REPORT macro, the FROM and TO parameters on the ITEM macro refer to the screen positions occupied by the attribute byte and the data; if DEV=3270 or IS7800 has not been specified, then the FROM and TO parameters must refer to the relative 3270 buffer addresses (relative to the beginning of the line), including the SF, the attribute byte and the data.

Designates that an SF order and an attribute byte are to be generated preceding the DATA item defined.

X'nn' - specifies hexadecimal attribute byte

YES indicates the attribute byte is generated for all default attributes: unprotected, no autoskip, alphanumeric, nondetectible, nonmodified.

Subparameters is a keyword list to indicate attributes.

PRO	protected
SKIP	autoskip
NUM	numeric
PEN	normal intensity, detectable
HIGH	high intensity, detectable
NON	non-display,non-print,non-detectable
MOD	field data tagged as modified

Additionally, if DEV=IS7800 is specified on the REPORT macro, then the following subparameters are allowed:

BLNK	blink
INVT	inverted characters
DW	double width characters
ULNE	underline

The subparameters are not positional. If any subparameter is omitted, the default is taken in the corresponding order of the YES attributes. All IS7800 subparameters must be specified in order to be used. For valid combination of IS7800 ATTs and IBM 3270 ATTs see the IS7800 manual. If the ATT operand is omitted, no SF and ATT bytes are generated.

CODE=

specifies the numerical identification associated with the set of characters that is to be placed in the output stream. This set of characters may originate from within the text of the message initiating the report, or the reporting format table itself. If the set of characters is to be extracted from the message text, code the associated user-assigned itemcode, 1 to 249. If, however, the set of characters is to be extracted from the reporting format itself, the code (250 to 255) is to specify the type of constant data that is being extracted.

250-253 - Reserved codes. Code 252 is used by the Page facility for the page number.

254 - specifies that the constant(s) supplied by the DATA parameter are one or more device control characters.

255 - specifies that the constant(s) supplied by the DATA parameter are one or more text characters.

When DEV=3270, codes 250, 251, 253, 254 and 255 require coding of either DATA or ATT. For non-3270 devices, codes 254 and 255 require the DATA parameter.

NOTE: The distinction to be made between codes 254 and 255 is that the constant data denoted by a 254 code is not considered to occupy any space within the device's buffer. However, because the FROM and TO parameters must always be coded to position characters relative to others, a line containing a 254 constant is internally expanded by the length of that constant without altering the relative positions of the printed text. The FROM and TO parameters may therefore specify 'column' positions not actually available on the device. For example, if, on setting up an 80-character line, a 6-letter non-constant text unit is to be placed in positions 72 through 77 and three control characters are then to be sent before an asterisk is inserted at position 80, the pertinent successive specifications would be as follows:

(CODE=user-assigned itemcode) - FROM=72,TO=77;then

(CODE=254) - FROM=78,TO=80;then

(CODE=255) - FROM=83,TO=83.

COMM=

designates a command to prefix the message text created for an IBM 3270 terminal. X'nn' specifies the actual command; WRITE specifies the Write Command; ERASE specifies the Write Erase Command. ERASE is the default. This parameter is used in conjunction with the WCC parameter.

DATA=

specifies constants to be generated in the output stream. Constant-data is coded as a character string. Con-el specifies any expression that can be used as an Assembler DC instruction operand, such as, DATA=(4XL1'15',C'LINE''S OF DATA'). This parameter is meaningful only if a code of 254 or 255 has been assigned to the CODE parameter.

ESC=

specifies the control character to be prefixed with an escape (ESC) character (X'27'), or, a special control character.

Code as a single code or list of codes. Valid codes are any single code valid after an escape character, e.g., H,X,R, etc.; or one of the following special codes:

Code	Meaning
FF	form-feed
DC2	start print
DC4	stop print
BEL	bell
HT	horizontal tab

There is no default code. This parameter has meaning only when DEV=DS40 is coded on the corresponding REPORT macro. The parameter may be coded alone when CODE=254, or in addition to initial value data (CODE=255), or with a data field code number. See the Switched Teletype (Dataspeed) Model 40/1-2 support in the BTAM Terminal Support Guide, for format design considerations.

FROM

specifies, by character column relative to 1, the location within the line of the left-most boundary of the line segment being defined, i.e., the position at and from which character insertion is to begin. The code, in value, must be equal to or less than that assigned to the TO parameter. Code as a decimal value, 1 to 255. See note under the CODE parameter.

ORDER=SC

For GTE IS7800 only. Designates an SC order if an attribute byte ATT=X'nn' is specified. Requires DEV=IS7800 to be coded on the REPORT macro. (Does not apply if DEV=3270 is coded on the REPORT macro.)

TO

specifies, by character column relative to 1, the location within the line of the right-most boundary of the line segment under definition, i.e., the position at which a character can be inserted but beyond which no overflow characters are to be permitted.

NOTE: If there is insufficient room to include all data within the segment of the line assigned to it, the Output utility will generate a sufficient number of lines of the same format to contain it. The code, in value, must be equal to or greater than that assigned to the FROM parameter. Code as a decimal value, 1 to 255. See note under CODE parameter.

WCC

designates a WCC to prefix the message text created for an IBM 3270 terminal. X'nn' specifies the actual WCC; subparameters are (linesize, PRINT, ALARM, RESTORE, RESET) and specify WCC options. The subparameters are not positional. If they are omitted, a default value is taken. The default value of the subparameter is its opposite (that is, default for PRINT is NO PRINT).

PMIALTRN--Specify Alternate Reporting Formats

The PMIALTRN macro is used to specify to Intercomm's Output Utility the alternative reporting formats that are to be employed when an alternate destination terminal is a device different in kind from the primary destination terminal.

The form of the PMIALTRN macro is as follows:

```
{symbol}    PMIALTRN    format-#,devcode,(alt-format-#,alt-
                        devocde {,alt-format-#,alt-devcode},...)
```

format-#
: specifies the reporting format identification for which one or more alternative reporting formats are being supplied by the "alt-format-#" parameter. Code as a decimal value, 1 to 32767. Refer to the REPORT macro, NUM parameter.

devocde
: specifies the type code of the device directly associated with the reporting format designated by the format-# parameter. Code as a hexadecimal value, 0 to F. Refer to the DEVICE macro, TYPE parameter.

alt-format-#
: specifies the identification of an altervative reporting format that is to be used in place of format-# when the destination terminal is a device of the type designated by the immediately following "alt-devcode" parameter. Code as a decimal value, 1 to 32767.

alt-devcode
: specifies the type code of the device to be associated with the reporting format designated by the immediately preceding "alt=format-#" parameter. Code as a hexadecimal value, 0 to F.

NOTE: Alternate report formats cannot be used from terminals whose DEVICE macro specify MMV designation (IBM 3270, Dataspeed 40, etc.) for the TYPE parameter.

FDHDR -- Define Change-Display Information Table

The FDHDR macro instruction is used in conjunction with the FDETL macro instruction. These two macro instructions are employed together to furnish the tables required by INTER-COMM's Change and Display utilities. When employed, the FDHDR macro instruction subordinates the FDETL macro instruction, and provides information unique to the data structure partially or wholly detailed by its associated set of FDETL macros.

The form of the FDHDR macro instruction is as follows:

[symbol]	FDHDR	NAME= {chngtbl-identifier / ddname} , FIELDS=#-of-'fdetl'-instructions, RPTNO=format-number [,KEYRT= {number-id / 0}] [,REPSZ= {number-of-bytes / 0}]

NAME=
: provides the unique name that will identify the table under generation. This name, coded as a variable length character string of 1 to 8 characters, will be either an identifier located within the CHNGTB table or a ddname of a file.

FIELDS=
: provides the number of associated FDETL macro instructions used to detail some or all of the fields within the data structure. Code as a decimal value, 1 to 4095.

RPTNO=
: supplies the format number that will, by default, be used to format selectively or otherwise, data found within the fields specified within the table under definition. "format-number" is to be coded as a decimal value, 51 to 32767. (Refer to the REPORT macro "NUM=" parameter.)

KEYRT=

provides the numerical identification assigned to that routine whose function it is to ultimately provide a valid key for file data retrieval. If the "NAME=chngtbl-identifier" prescription is used, this parameter is not meaningful. If the table under generation is being applied to the record format of either a BDAM or ISAM file, it is recommended that for BDAM a 2 be coded, and for ISAM a 0 be coded. "Number-id" is to be coded as a decimal value, 0 to 255. The default code is 0. Note that user coded routines are numerically identified via the KEYTABLE table.

REPSZ=

provides the length, in bytes, of the repetitive substructure if one is present. Note that the total length of all fields described by the associated FDETL macros specifying "FLD=REPET" will never exceed the value of this code. Code as a decimal value, 0 to 4095. The default code of 0 signifies that no substructure is present.

FDETL -- Define Data Structure Field

The FDETL macro instruction is used in conjunction with the FDHDR macro instruction. These two instructions are employed together to furnish the tables required by INTERCOMM's Change and Display utilities. When employed, the FDETL macro instruction is subordinated to a FDHDR macro instruction and functions to provide on a field basis, the following five classes of information

- specificative information.
- logically associative information.
- data descriptive information.
- procedural information if the data contained within the field is to be CHANGE'd.
- procedural information if the data contained within the field is to be DISPLAY'ed.

The form of the FDETL macro instruction is as follows:

[symbol]	FDETL	specificative parameters: [OFSET=byte-offset,]LEN=length, [FLD={HDR / REPET},] logically associative parameters: NAME=field-id,CODE=itemcode [,KEY={YES / NO}][,SUBKY={YES / NO}] data descriptive parameter: [,FRMAT={BIN / PACK / CHAR}] 'Change' utility parameters: [,CHNG={YES / NO}][,VRY={YES / NO}] continued

continued

'Change' and 'Display parameters:

```
[,PADD= {RIGHT}  ,PDCHR= {BLANK}]
        {LEFT }          {ZERO }
```

'Display' utility parameters:

```
[,CHAR= {EDIT}]
        {NO  }
[,EDIT= {pattern-id}  [,TRUNC= {RIGHT}]]
        {DATE      }           {LEFT }
        {DOLLAR    }
        {NUMBER    }
```

OFSET=
specifies the displacement, in bytes, from data structure orgin to the beginning of the field. If, on the associated FDHDR macro instruction, the "NAME=ddname" parameter description is used, data structure origin will be record origin. If "NAME=chngtb=identifier" is used, data structure origin will be the first byte immediately following the 12-byte control block. If the field is embedded within a repetitive substructure "byte-offset" is to be measured to the first occurrence of the field. Code as a decimal value, 0 to 32767. This parameter may be omitted if the field is contiguous to, and directly follows, the field defined by the prior FDETL macro.

LEN=
specifies the length, in bytes, of the field under specification. If "FRMAT=BIN" is specified, code as a decimal value within the range 1 to 4. If "FRMAT=PACK", 1 to 16; if "FRMAT=CHAR", 1 to 255.

FLD=
specifies whether the field is located within the fixed portion of the data structure or, if one is used, within the repetitive portion of the data structure. Code HDR for the fixed portion, REPET for the repetitive portion. The default code is HDR.

NAME=
provides the character identification for the field. Code as a 1 to 5 character string. (Note that this is the name that is to be supplied on the FDN keyword of INTERCOMM's 'CHNG' transaction.)

CODE=
provides the numerical identification for the field. Note that this itemcode becomes the agent through which data contained within the field is ultimately assigned a

location within any report. Code as a decimal value, 1 to 249. (Refer to the ITEM macro "CODE=" parameter.)

KEY=
denotes whether or not the field is, in whole or in part, the key of a record. If "NAME=chngtbl-identifier" has been specified on the associated FDHDR macro instruction, this parameter is not meaningful. If "NAME=ddname" has instead been specified and the associated file uses an embedded key, and if, furthermore, that key incorporates the field under consideration, YES must be coded. The default code is NO. Note that "FLD=HDR" must also be coded. (Note further that the data supplied via the KEY keyword of both INTERCOMM's CHNG and DSPL request messages will accordingly be wholly or partially compared against this field.)

SUBKY=
denotes whether or not the field is, in whole or in part, the INTERCOMM subkey of a record. If "NAME-chngtbl-identifier" has been specified on the associated FDHDR macro instruction, this parameter is not meaningful. If "NAME=ddname" has instead been specified and the associated file's record format contains a repetitive substructure, and if, furthermore, the INTERCOMM subkey incorporates the field under consideration, YES must be coded. The default code is NO. Note that "FLD=REPET" must also be coded. (Note further that the data supplied via the SKY keyword of INTERCOMM's CHNG request message will accordingly be wholly or partially compared against this field.)

FRMAT=
describes the kind of data that is to be expected within the field. Code BIN for binary data, PACK for packed decimal data, or CHAR for character data. The default code is CHAR.

CHNG=
indicates whether or not it is permissable for the field to be altered by a CHNG request. Used only by INTERCOMM's Change utility, this parameter is meaningful only if "NAME=ddname" has been specified on the associated FDHDR macro. Code NO if such changing is not permitted. Code YES if it is. The default code is YES.

VRY=
indicates whether or not the field must be verified before it is permitted to be changed. Used only by INTERCOMM's Change utility, this parameter, is meaningful only if "NAME=ddname" has been specified on the associated FDHDR macro. Code NO for verification not required. Code YES for verification required. The default code is YES.

PADD=

provides a request for either left or right padding. The absence of this parameter indicates the absence of any request. In this particular instance the following will occur: If, after the calculation of the maximum number of characters required for the displaying of the whole field (a function of field length, data type, and edit characters), it is found that the data (stripped of leading blanks or zeros though including editing characters) possesses a lesser length, this lesser length will be the data unit considered expected by the outputting format through which this field is to be DISPLAY'ed. The presence of this parameter with either of the key-codes LEFT or RIGHT, indicates the presence of a request. In this instance, data of lesser length is expanded to the maximum length. A code of RIGHT is regarded as a request to first left-justify the data, then pad on the right. A code of LEFT is regarded as a request to first right-justify the data, then pad on the left. Note that the "PDCHR=" parameter must be coded to provide the padding character.

PDCHR=

specifies which one of two characters (blank or zero) is to be used when the padding function is requested. This parameter must be coded if the "PADD=" parameter is used. Code BLANK for a blank pad character, ZERO for a zero character.

CHAR=

indicates whether or not editing is to be performed upon a character field that is expected to contain solely numerical characters. This parameter is meaningless unless "FRMAT=CHAR" has been coded. If such editing is requested, code EDIT and assign the appropriate "pattern number" to the "EDIT=" parameter; if such editing is not requested, code NO. The default code is NO.

EDIT=

provides the editing pattern identification denoting the pattern to be used to edit the data before DISPLAY'ing it. The code assigned to this parameter must be identical to that assigned to the "NUMBER=" parameter of the PATRN macro instruction that supplies the editing pattern. Code "pattern-id", DATE, DOLLAR, NUMBER accordingly. The default code is NUMBER.

TRUNC=

provides a request for either left or right truncation. The absence of this parameter indicates the absence of any request. The presence of this parameter indicates the presence of a request; however, such will be honored only wnen the "MAXSIZE=" parameter on the associated PATRN macro is not zero. Refer to "EDIT=". Data longer than the value assigned to "MAXSIZE=" may be truncated either on the left or right and truncation will always be to "MAXSIZE=". Code RIGHT for right truncation, LEFT for left truncation. Note that this parameter applies only to data that is to be edited.

GENFTBLE -- Define File

The GENFTBLE macro instruction is used to create the PMIFILET table entries that furnish certain INTERCOMM subsystems with the file information they require.

The form of the GENFTBLE macro instruction is as follows:

[symbol]	GENFTBLE	FNAME=ddname, TYPE={BDAM / ISAM / SAM},BLKSIZE=blksize [,DESNUM='des000'-file-rbn]

FNAME=
provides the ddname of the file under specifications. Code as a character string of 1 to 8 characters in length. Any code not eight characters will be padded on the right to eight characters.

TYPE=
specifies the type of file under specification. Code one of the following symbols:

BDAM - if the file is a BDAM file.
ISAM - if the file is an ISAM file.
SAM - if the file is either a BSAM or QSAM file.

There is no default code.

BLKSIZE=
specifies, in bytes, the maximum length of the physical blocks of the file under specification. Code as a decimal value, 1 to $2^{15}-1$.

DESNUM=
specifies the RBN value of the data structure description entry within the DES000 file that is to be used in the CHANGE'ing and DISPLAY'ing of the file under definition. This value is to be one less than the value of the last five digits of the member-name used to place the entry in the file, e.g., if the member-name is DES00013, code DESNUM=12. Code as a decimal value, 0 to $2^{23}-1$.

PATRN -- Specify Edit Instruction Pattern

The PATRN macro instruction is used in conjunction with the FDETL macro instruction "EDIT=" parameter. It is employed only when this parameter is utilized and serves to provide the editing pattern for the EDIT instruction performed upon all fields so associated with it.

The form of the PATRN macro instruction is as follows:

```
symbol    PATRN    NUMBER= {user-id},
                          {DATE   }
                          {DOLLAR }
                          {NUMERIC}
                   PATTRN= {(user-pattern)}
                           {DATE          }
                           {DOLLAR        }
                           {NUMERIC       }
                  [,FLOAT= {character}]
                           {blank    }
                  [,MAXSIZE= {field-length}]
                             {0           }
```

symbol
code any symbol valid in the Assembler language. This symbol must be coded.

NUMBER=
specifies the identification of the pattern under definition. If one of the three keywords, DATE, DOLLAR, NUMERIC has been specified on the "PATTRN=" parameter, the identical keyword must be assigned to this parameter. If none of these keywords have been employed, specify a decimal value, 1 to 3, 5 to 7, 9 to 15. Codes 0, 4, and 8 are reserved.

PATTRN=
specifies the editing pattern under definition. If a standard date, dollar, or numeric pattern is desired, code DATE, DOLLAR, or NUMERIC accordingly. If a user-written pattern is under definition, it must contain a total of thirty-one digit-select and significance starter characters and this pattern may be any combination of digit select (X'20'), significance starter (X'21'), field separator (X'22'), and message characters.

The standard DATE, DOLLAR, and NUMERIC patterns are as follows:

DATE - X'40',24X'20',X'212020',2X'612020'
resulting in the form: MM/DD/YY

DOLLAR - X'40',X'20206B',8X'2020206B',X'2020214B2020'
resulting possibly in:
99,999,999,999,999,999,999,999,999.99

NUMERIC - X'4020',9X'6B202020',X'6B202021'
resulting possibly in:
9,999,999,999,999,999,999,999,999,999

Note: A user-written pattern must be enclosed within parenthesis.

FLOAT=
specifies the character to be inserted in front of the data after editing. Code as a single character not enclosed within quotes. The default code is a blank character.

MAXSIZE=
specifies, in bytes, the maximum permitted post-edit length of a field edited by this pattern when that field's associated FDETL macro instruction "TRUNC=" parameter specifies either a code of LEFT or RIGHT, i.e., the length down to which a field is to be reduced after the editing operation has been performed. Code as a decimal value, 1 to 31. The default code is 0. A request for truncation is meaningless when 0 is specified.