


Burroughs 

B 1800/B 1700
Generalized Message
Control System
(GEMCOS)

FORMATTING GUIDE

PRICED ITEM

Burroughs 

B 1800/B 1700
Generalized Message
Control System
(GEMCOS)

FORMATTING GUIDE

Copyright © 1978 Burroughs Corporation, Detroit, Michigan 48232

PRICED ITEM

Burroughs believes that the application package described in this manual is accurate and reliable, and much care has been taken in its preparation. However, no responsibility, financial or otherwise, can be accepted for any consequences arising out of the use of this material, including loss of profit, indirect, special, or consequential damages. There are no warranties which extend beyond the program specification.

The Customer should exercise care to assure that use of the application package will be in full compliance with laws, rules and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change. Revisions may be issued from time to time to advise of changes and/or additions.

Table of Contents

Section	Title	Page
	INTRODUCTION	v
1	OVERVIEW OF FORMATTING	1-1
2	BASIC GEMCOS FORMATTING	2-1
	General	2-1
	Defining Field Types	2-1
	Adding Data	2-2
	Manipulating Fields	2-3
	Ignoring Data and Inserting Blanks	2-4
	Rearranging Data	2-5
	Summary	2-7
3	ADVANCED GEMCOS FORMATTING	3-1
	General	3-1
	Minimizing System Overhead	3-1
	Handling Variable-Length Fields	3-1
	B and J Fields	3-1
	Delimited A and I Fields	3-2
	Translating Data	3-3
	Repeating Multi-element Variables	3-4
INDEX	one

List of Illustrations

Figure	Title	Page
2-1	Use of Editing Strings	2-3
2-2	Operation of Pointers	2-4
2-3	Use of X Item and Location Specifiers in Formats	2-6
2-4	Pointer Manipulation	2-7
2-5	Output Formatting	2-8
2-6	Input Formatting	2-9
3-1	Variable Repeats	3-6

INTRODUCTION

Over the years, computers have developed a reputation for being inflexible, unyielding machines that do not process data unless it has been entered in a very precise manner and only after very rigid rules have been meticulously followed. Data control clerks and key-punch operators had to be concerned with the length of each data item, taking care to insert leading zeroes and trailing spaces so that the next data item could be recognized by the system. These people, of course, were subjected to lengthy training and could be considered specialists in their respective fields.

The advent of data communications, however, put more nonspecialists in close proximity to the computer. It was not feasible to subject bank tellers and department store clerks to the same type of training that was provided to data control clerks. It became obvious that a man-machine interface was required in order to make data entry more palatable to the average person.

Systems analysts and programmers rose to the task, and "human engineering" quickly became the buzzword of the day. Terminal operators were treated to formatted displays, free-form input, and an array of features designed to make their jobs both efficient and pleasant. The problem was solved, but not for long; new terminal devices were designed, networks changed and expanded, and the data being presented to the computer became more and more volatile. As a result, analysts and programmers were deluged with requests to constantly modify the man-machine interface. The original problem had to be repeatedly solved.

It soon became clear that if a man-machine interface were to be a viable entity, it must be free from any dependence upon terminal devices. Additionally, it must be designed in such a way as to be transparent to developers of application-oriented programs. If possible, it must also be capable of adapting to changing requirements without the necessity for reprogramming. It should allow a user to take advantage of new terminals and features with little or no impact upon his operation. It was with these requirements in mind that the format function of GEMCOS was developed.

The purpose of this document is to highlight formatting capabilities as implemented in the Generalized Message Control System, Advanced Version (Style ID B1800 MCA and B1700 MCA). For detailed and complete syntax and semantics of formatting, refer to the B 1800/B 1700 Generalized Message Control System (GEMCOS) User's Reference Manual, form 1093499.

SECTION 1

OVERVIEW OF FORMATTING

Formatting defines how fields within a message are to be manipulated (by GEMCOS) prior to their delivery to a program (input messages) or a terminal (output messages). A field of data is that portion of a message (a character or contiguous group of characters) comprising a logical entity, e.g., social security number, clock number, or gross pay. Fields in a record can be rearranged, deleted, expanded, compressed (via the addition or deletion of leading zeroes or trailing spaces), or translated (e.g., DOZ could become 12 or vice versa). Fixed information can be inserted between fields. Numeric fields can be checked to be sure they contain only numbers. Certain esoteric hardware features, such as highlighting, blinking, blanking, and reverse video, can be utilized, as well as the more common ones, such as tab stops and forms feed.

Formatting with GEMCOS can be rather simple or quite complex, depending upon the problem to be solved. In its most complex form, however, GEMCOS formatting remains transparent to the application programmer and adaptive to the requirements of the terminal operator. A programmer can define data fields to be processed with no knowledge of the device or devices which will be sending and receiving that data. Once the devices have been identified, the format of the data as it appears on those devices can be decided upon according to the preferences of the people who use those devices. In most cases, the programmer need do nothing to the application software.

A GEMCOS message format provides the man-machine interface, manipulating the data such that it is compatible with the requirements of the terminal and the program. A message format can be prepared for input and output messages; in some cases, a format may be used on input and output.

The person preparing the message formats must have the desired record layouts available for each type of message. The first layout should describe the message as it appears to the application program. Subsequent layouts should describe the message as it appears at the terminal devices. The format writer must also be familiar with terminal device characteristics, such as buffer size, screen size, and control codes necessary to perform special functions. (Refer to the B 1800/B 1700 Series GEMCOS User's Reference Manual for a discussion of device classes.)

The format applied to a message depends on two things. First on the transaction involved, since different transactions are comprised of different fields in different orders and secondly it depends on the station involved since the same message sent to different stations may require different control codes, buffer size, screen size etc. Therefore GEMCOS provides a means of defining formats as well as allowing the user to specify which formats are to be applied to which station/transaction combinations.

GEMCCS formatting accommodates a wide range of applications. In many cases, however, the formatter need be familiar with only the simple procedures described in section 2. More advanced formatting techniques are discussed in section 3.

SECTION 2
BASIC GEMCOS FORMATTING

GENERAL.

A GEMCOS message format is described as follows:

FORMAT <format name> (<format description>).

The <format name> is a unique name for the <format description>. The first character must be a letter, but the remainder may be numbers or letters in any combination. Spaces may not be used within the name.

The following four specifications, part of the <format description>, are those most frequently utilized in GEMCOS formatting:

- a. Defining field types.
- b. Adding data to a message.
- c. Ignoring data or inserting blanks in a message.
- d. Rearranging data within a message.

All items specified within the parentheses delimiting the <format description> are separated by commas.

DEFINING FIELD TYPES.

It is necessary to define the type of each field in the record when formatting an input or output message. There are two primary kinds of data fields in a format description (although variations of each exist). Alphanumeric (alpha) fields may contain any combination of characters; numeric (integer) fields may contain only numbers, or integers. Alpha fields are identified by the letter A, and integer fields are identified by the letter I. All integer fields are edited, as subsequently described.

The length of the field immediately follows the letter designation. Thus a 10-character alpha field would be described as A10, while a 15-digit integer field would be described as I15. A message consisting of a 50-character alpha field followed by a 6-digit integer field would be described:

FORMAT F1 (A50, I6).

Fields may be described so as to divide a group of characters into logical groups. For example, 123456789012035 may be described as A15 or I15 or may be subdivided as I9, I6 or as A9, I6.

A shorthand notation may be employed when consecutive identical fields are specified. This notation is called a repeat part. For example, three consecutive I6 fields may be described as 3I6. When the LEVEL OF consecutive groups of identical fields are to be repeated, an additional parentheses is employed. For example, three consecutive A3, I6 fields may be described as 3(A3, I6).

Any integer field definition invokes editing. Editing of the A50, I6 field works as follows: On input (from terminal to system), the first 50 characters are moved from the raw message to the formatted message. The next six characters are then examined. If leading or trailing blanks are present, the field is right-justified and leading zeros are inserted. The field is then added to the formatted message. If the field is not entirely numeric, a flag is set to inform the program that a field failed the edit test. On output, the first 50 characters are moved from the raw message to the formatted message. The next six characters are checked for numeric content, then moved intact to the formatted message. If the data is incorrect, the control station is notified, but the message is delivered to the station in either event. If no editing is desired, the field may be defined as A56 instead of A50, I6.

ADDING DATA.

Once fields within a record have been defined, the next step is to format those fields so they can be easily read by the terminal operator. One method is the insertion of editing strings. Editing strings are nothing more than simple strings inserted into the format description wherever they are to appear in the output message. Editing strings are also used to create blank screen formats for operator input.

Editing strings are declared by using quotation marks. For example, if a 10-character field were to be identified as LAST NAME and made accessible to the operator, the following phrases could be used:

```
"LAST NAME [", A10, "]" "
```

Hexadecimal strings can be used to define characters for which no graphics are available. Hex strings are identified by placing a 4 in front of the string. For example, the DC1 character is expressed as 4"11".

Refer to figure 2-1 for a more comprehensive example of editing strings. When the GEMCOS output message format is applied to the message from the program, the screen message shown results. The same message format yields the blank screen shown in response to a forms request from the terminal.

OUTPUT MESSAGE FORMAT

FORMAT FMTOUT (4"OC", % FORMS FEED
"LAST NAME [",A10,"]"",4"OD", % CARRIAGE RETURN
"FIRST NAME [",A6,"]"",4"OD",
"SSAN [",A9,"]"",4"OD",
"AMOUNT [",16,"]"",4"12"). % PUT IN FORMS MODE

MESSAGE FROM PROGRAM

SMITH__JOHN_123456789004960
A10 A6 A9 I6

MESSAGE ON SCREEN

LAST NAME [SMITH]
FIRST NAME [JOHN]
SSAN [123456789]
AMOUNT [004960]

THE RESULT OF A FORMS REQUEST

LAST NAME[]
FIRST NAME []
SSAN []
AMOUNT []

Figure 2-1. Use of Editing Strings

MANIPULATING_FIELDS.

Message fields may be manipulated so that data is ignored, blanks are inserted, or data is rearranged. Formatting with GEMCOS to manipulate fields in these ways is clearest once the user understands how the GEMCOS formatter works.

The GEMCCS formatter has two areas for each message. The first area contains the message as it looks to the terminal; this is the external message. The second area, or common area, contains the message as the program sees it; this is the internal message.

Each area has associated with it a pointer, the external pointer and the internal pointer, respectively. As a field is processed, both pointers are advanced as the field is moved from one area to the other. Both pointers are initially set to 1; and after processing a field, both advance to the position immediately following the last character moved. For example, if the first field is A5, five characters are moved from one area to the other, and both pointers advance to position 6. Figure 2-2 illustrates how pointers work.

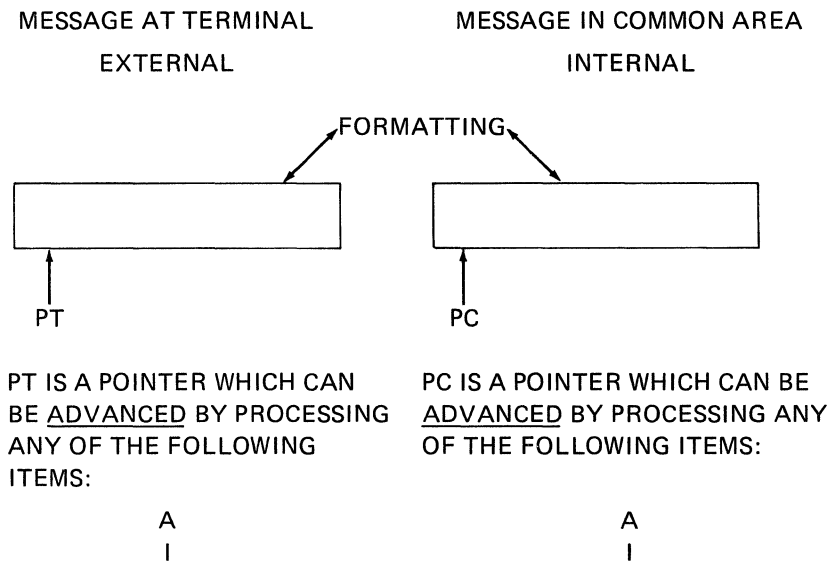


Figure 2-2. Operation of Pointers

The subsequent discussions illustrate how to manipulate GEMCDS external and internal pointers to ignore data, insert blanks, and rearrange data.

IGNORING DATA AND INSERTING BLANKS.

It may sometimes be desirable to ignore certain data received from a station or insert blanks into fields going to a station. These blank are inserted by using an X item phrase, which advances the external pointer only.

For example, a message sent to a station contains both name and social security number for the operator's convenience. After altering some data, the operator transmits the message back to the program for update, but the program requires only the social security number. The name field can be edited out by means of an X item phrase:

FORMAT F2 (X10, I9, I6).

When this format is applied to an input message which contains

```
SMITH_____123456789123456
```

the following is presented to the program:

```
123456789123456
```

The X item phrase may be used to insert blanks into an output message. Consider the following format:

```
FORMAT F3 (A6, X4, I3).
```

When this format is applied to an output message which contains:

```
WIDGET123
```

the following is displayed at the terminal:

```
WIDGET___123
```

REARRANGING DATA.

Data may be rearranged by manipulating the internal pointer. Unlike the external pointer, which can only be advanced (regardless of message direction), the internal pointer can be moved in either direction. To adjust the setting of the internal pointer, a location specifier (@) is used with an unsigned or signed integer.

When an unsigned integer is used, the internal pointer is adjusted to the absolute position indicated by the integer. When a signed integer is used, the internal pointer is adjusted in the direction of the sign relative to its present position by the number of positions indicated by the integer.

For example, an input message of ABC123DEF456GHI789 is described as A3, @10, I3, @4, A3, @13, I3, @7, A3, @+6, I3. The following sequence of events occurs:

- a. Initially, both pointers are set to 1.
- b. The A3 causes ABC to be moved to the internal message area, and both pointers are set to 4.
- c. @10 sets the internal pointer to 10.
- d. I3 causes 123 to be moved to positions 10 thru 12.
- e. @4 sets the internal pointer to 4.
- f. A3 moves DEF to positions 4 thru 6.

- g. @13 sets the internal pointer to 13.
- h. I3 moves 456 to positions 13 thru 15.
- i. @7 sets the internal pointer to 7.
- j. A3 moves GHI to positions 7 thru 9.
- k. @+6 sets the internal pointer to 16.
- l. I3 moves 789 to positions 16 thru 18.

The message delivered to the program is ABCDEFGHI123456789.

Figure 2-3 shows the usage of location specifiers with unsigned integers in combination with X items. Figure 2-4 illustrates pointer manipulation for strings, X items, and location specifiers.

INPUT MESSAGE FORMAT

FORMAT FMTIN (A10,X6,@21,A9,@11,I6)

MESSAGE ON SCREEN

LAST NAME [SMITH]
FIRST NAME [JOHN]
SSAN [123456789]
AMOUNT [4960]

MESSAGE AS TRANSMITTED

SMITH__JOHN__123456789_4960__
 A10 X6 A9 I6

MESSAGE AFTER FORMATTING

SMITH__004960__123456789
 1 1 2
 1 1

Figure 2-3. Use of X Item and Location Specifiers in Formats

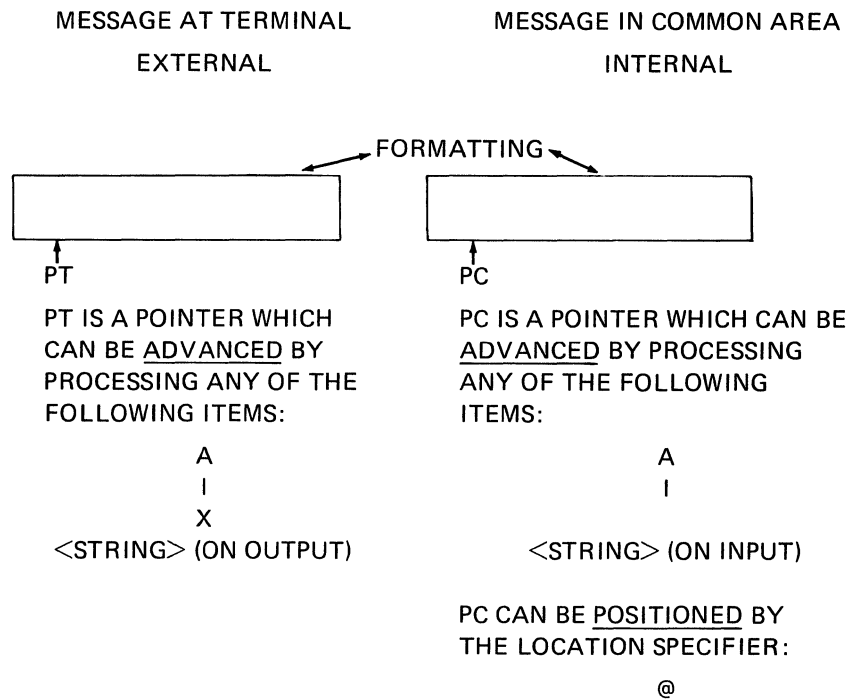


Figure 2-4. Pointer Manipulation

SUMMARY.

Based upon what has been written thus far, it is possible to prepare GEMCOS message formats which handle an extensive variety of formatting situations. In many cases it may not be necessary to utilize the GEMCOS advanced formatting options in section 3. The following summarizes what has been discussed:

- a. A format is declared as follows:
FORMAT <format name> (<format description>).
- b. Alphanumeric fields are declared as:
A <length of field>.
- c. Numeric (integer) fields are declared as:
I <length of field>.
- d. Items within the parentheses are separated by commas, as:
(A9, I16, A7).
- e. Consecutive identical fields or groups of fields can be described with a repeat part, as:
3A50 or 3(A50, I6, A3).

- f. External data, in the form of strings, can be inserted anywhere within a message, as:
A15, "XYZ", I5, 4"OC", A20.
- g. Input fields can be ignored by use of the X item phrase, as:
A5, X20, A5.
- h. The order in which fields appear can be rearranged via location specifiers, as:
X15, @30, X10, @16, A14, or X15, @30, A10, @-45, I5.
- i. If no editing is required, numeric (I) fields can be expressed as alphanumeric (A) fields.

Figures 2-5 and 2-6 summarize the results of applying a GEMCOS message format to an output message and an input message, respectively, for one transaction.

MESSAGE FORMAT

```

FORMAT FMTOUT (4"OC", % FORMS FEED
"LAST NAME [",A10,"]"",4"OD", % CARRIAGE RETURN
"FIRST NAME [",A6,"]"",4"OD",
"SSAN [",A9,"]"",4"OD",
"AMOUNT [",I6,"]"",4"12"). % PUT IN FORMS MODE

```

MESSAGE FROM PROGRAM

```

SMITH__JOHN_123456789004960
  A10   A6   A9   I6

```

MESSAGE ON SCREEN

LAST NAME [SMITH]
FIRST NAME [JOHN]
SSAN [123456789]
AMOUNT [004960]

THE RESULT OF A FORMS REQUEST

LAST NAME []
FIRST NAME []
SSAN []
AMOUNT []

Figure 2-5. Output Formatting

MESSAGE FORMAT

FORMAT FMTIN (A10,X6,@21,A3,@11,I5,10).

MESSAGE ON SCREEN

LAST NAME [SMITH
FIRST NAME [JOHN
SSAN [123456789]
AMOUNT [4960]

MESSAGE TRANSMITTED

SMITH__JOHN_123456789_4960
A10 X6 A3 I5

MESSAGE AFTER FORMATTING

SMITH__004960__123456789
1 1 2
1 1

Figure 2-6. Input Formatting

SECTION 3

ADVANCED GEMCOS FORMATTING

GENERAL.

In some cases it may be convenient to utilize advanced GEMCOS formatting features to accomplish the following:

- a. Minimizing system overhead.
- b. Handling variable-length fields.
- c. Translating data.
- d. Repeating multi-element variables.

MINIMIZING SYSTEM OVERHEAD.

The format writer can minimize Disk I/O overhead by declaring certain formats as RESIDENT. This step permits the message format to reside in memory rather than on disk. This mechanism is best employed with small, frequently used formats. The RESIDENT declaration is optional, however, since format writers, who are familiar with an application, are in the best position to decide where to store message formats.

To declare a message format as resident, the word RESIDENT is declared in brackets following the format name:

```
FORMAT F4 [RESIDENT] (A7, X4, Q12, I6).
```

HANDLING VARIABLE-LENGTH FIELDS.

When input fields are variable in length, their message formats may be declared with delimiters to ensure that the program always receives fixed-length fields without operator entry of leading zeroes or trailing blanks. Two types of variable-length fields may be accommodated by GEMCOS format declarations:

- a. Variable-length fields which may or may not be terminated by a delimiter. These fields are identified as B or J fields.
- b. Variable-length fields which are always terminated by a delimiter. These fields are delimited A and I fields.

B AND J FIELDS.

When the presence of delimiters is optional (tabbed fields for example) alpha and integer items are declared as B and J items, respectively, and the field lengths declared are the maximum lengths.

When the standard horizontal tab character (4"05") is used as a delimiter, B and J items are declared as follows:

```
B10, J9
```

When a different delimiter is used, the delimiter code and maximum field length are declared in parentheses following the B or the J. The field length declared does not include the delimiter code. For example, if the delimiter character is (4"11"), the following would be declared for the B10 field:

```
B(4"11", 10).
```

The shorthand repeat part may be used with B and J items as described previously.

GEMCOS implements these item phrases as follows. In a B10 field, characters are moved until either 10 have been moved or until a delimiter character is encountered. When a delimiter character is recognized, trailing spaces are inserted to fill the field to 10 characters. In a J field, leading zeroes are inserted to fill a field terminated early by the delimiter character. If the field is completely filled, the delimiter code is not present and the pointer advances automatically to the next input field. On output messages, B and J fields are treated as if they were A and I fields, respectively.

The following message format describes a record consisting of a 3-character field followed by a 4-digit field, a variable-length tabbed field having a maximum of six characters, two variable-length tabbed numeric fields (each having a maximum of four digits), and a 4-character field at the end of the message which is ignored:

```
FORMAT F5 (A3, I4, B6, 2J4, X4).
```

The message is received from the station in the following format (note that the tab code is not present with the first numeric field because the 4-digit field was completely filled):

```
      T      T
XYZ1234XYZa1234567aSNVF
      b      b
```

After the application of the format, the following message is presented to the program:

```
XYZ1234XYZ__12340567
```

DELIMITED S AND I FIELDS.

When delimiters are always present, A and I are used with the delimiter codes and maximum field lengths; for example:

```
A (".", 5)
I ("p", 8)
```

These item phrases are implemented by GEMCOS as described for B and J fields. The only difference is that for these fields, the delimiters are mandatory, even when a field is entirely filled.

The shorthand repeat part may also be used with these delimiter fields. For example, if three contiguous variable-length fields have a maximum of 10 characters each and are always delimited by an asterisk (*):

```
3(I(4"*", 10))
```

The following example illustrates how data in a variable-length field in the input message may be ignored:

```
FORMAT F6 (A5, X("W"), I6).
```

When applied to an input message of ABCDEPQRSTUVWXYZ123456, this message format results in ABCDE123456 being delivered to the program.

TRANSLATING DATA.

The translation of program-compatible fields to terminal operator-compatible fields and vice versa is readily accommodated in GEMCOS. Abbreviations such as SUN, MON, JAN at the terminal can appear to the program as 1, 2, 01, respectively. Translation is accomplished in two steps. A function declaration is prepared and referenced in input and output message formats.

The function declaration identifies the terminal and program equivalents. These equivalents are declared in the form of strings, which must be no longer than six characters. An external string, which declares how the field appears to the terminal, is specified first; then the internal string, which declares how the field appears to the program, is specified:

```
FUNCTION SEX ("MALE": "1", "FEMALE": "2").
```

The function declaration is then referenced as follows in the input and output formats: The letter T is declared to indicate translation, followed by the name of the function, the item phrase describing the external string, and an integer specifying the length of the internal string:

```
T(SEX, A6, 1).
```

The following declarations, for example, would be used to translate FEB 1975 to 2 1975:

```
FUNCTION F ("JAN": "1", "FEB": "2", "MAR": "3").  
FORMAT F10 (T(F, A3, 1), X1, " ", I4).
```


The preceding are examples of unedited translation specifications. An unedited string of less than six characters in length is right-justified within a 6-character word with leading nulls (4"00"). As long as all internal strings are the same length and all external strings are the same length, an unedited function specification works well. If strings vary in length, however, the use of unedited function specifications can cause confusion. For example, suppose a function is declared as follows:

```
FUNCTION TEST ("FEMALE": "11", "MALE": "1").
```

On output, an internal string of 1 matches 11, because GEMCOS looks only at the right-hand character. Similarly, on input, an external string of MALE matches FEMALE.

To avoid this confusion, edited translation specifications are introduced. The example just given would be:

```
FUNCTION TEST [EXTERNAL: ALPHA, INTERNAL: INTEGER]  
("FEMALE": "11", "MALE": "1").
```

An edited integer string of less than six characters is right-justified with leading zeroes. An edited alpha string of less than six characters is left-justified with trailing blanks. Now when GEMCOS searches for 1, it actually searches for 00001. When GEMCOS searches for MALE, it searches for MALE__.

REPEATING MULTI-ELEMENT VARIABLES.

Some output messages have a variable number of fields of repeated data, as in tables with columns of values. These messages can provide counters which specify the number of elements present in these fields. The counters can be used in the repeat parts to declare a GEMCOS message format.

If a counter field is used, the key word VARIABLE must be specified as the first declaration in the format, followed by one of six variables (V1 thru V6) which accepts the contents of the counter. A location specifier may be used to indicate where in the raw message the counter field resides. The length of the counter is then specified as an integer following the key word FOR:

```
VARIABLE V1 FOR 1;  
VARIABLE V3 @+16 FOR 5;
```

The variable in the internal message must be in EBCDIC numerals and must not be greater than 255 in value. The use of the location specifier in the variable declaration alters the position of the pointer within the internal message, and the format must position the pointer to the data if necessary.

The repeat part in the output message format is then constructed by using this VARIABLE declaration, plus an expression consisting of the assigned variable designation and a maximum repeat indicator:

```
FORMAT F11 (VARIABLE V1 FOR 2; V1 OR 6A6, I2).
```

The number of times the GEMCOS formatter employs the repeat part depends on which is less, the variable repeat part or the maximum repeat part. In the following output message, for example, four groups of A6 fields are processed after the 04 is loaded into variable V1; the 67 is processed as a 2-digit integer field:

```
04ABCDEF123456GHIJKL78901267.
```

Variable repeat parts may be nested if the situation requires. In this way a variable number of groups having multi-element variables may be declared in one message format through proper use of parentheses (refer to the B 1800/B 1700 GEMCOS User's Reference Manual for syntax details).

An optional update variable may be used while a multi-element variable is being processed. In the following case, for example, V2 is the variable repeat part and V1 is the update variable:

```
V1:V2 OR 6(A16,4"0D")
```

Assume V2 has been initially set to 10. Since the maximum repeat part is 6, only six fields of the message are processed by the format (leaving four fields unprocessed). At the completion of the phrase, the update variable, V1, contains the value 4. V1 could then be used in subsequent phrases within the format. All variables used in a format must be declared and given an initial value. The update variable may be the same as the variable repeat part. Figure 3-1 is an example of variable repeat specifiers used.

MESSAGE FORMAT

FORMAT F12 (VARIABLE V1 FOR 2;
4"OC","+ PART NO",X4,
"QUANTITY",4"OD",X3
V1:V1 OR 5(I4,X9,I2,4"OD").

MESSAGE FROM PROGRAM

O4123401567812901216345606
FIELDS OF 14,12

MESSAGE ON SCREEN

+ PART NO	QUANTITY
1234	01
5678	12
9012	16
3456	06

Figure 3-1. Variable Repeats

2" BINDER

B 1800/B 1700 GEMCOS
FORMATTING GUIDE

1106531

Printed in U.S.A.

1" BINDER

1½" BINDER