

REPGEN

**Hydrologic Engineering Center
Report Generator**

User's Manual

**Version 3.2
March 1995**

**Hydrologic Engineering Center
U.S. Army Corps of Engineers
609 Second Street
Davis, California 95616-4687
(916) 756-1104**

REPGEN

Table of Contents

Chapter	Page
1. Introduction	1
1.1 Purpose	1
1.2 Acknowledgements	1
1.3 Differences From Previous Versions	1
2. Description	3
3. Program Use	7
4. Input Description	9
4.1 Input Format	9
4.2 Report Form	11
4.3 Variable Definitions	11
4.3.1 Time Specification	12
4.3.1.1 Time Expressions	13
4.3.2 Predefined Variables	13
4.3.3 Time Series Variables from DSS	14
4.3.4 Variables from Text Files	15
4.3.5 Variables From Mathematical Formulae and Functions	16
4.3.6 Mathematical Functions	17
4.3.7 Time-Related Functions	19
4.3.8 Grouping of Variables	21
4.3.9 Subscripted Time Series Variables	21
4.3.10 Missing Values	23
4.3.11 Display Formats for Variables	23
4.3.12 Row and Column Displays	27
4.3.13 Conditional Messages	28
4.4 Function Characters	28
5. Examples	29
6. Obsolete Features	33

FIGURES

	Page
1. Example Report	3
2. Input File	4
3. Example Gage Record	30

Chapter 1

Introduction

1.1 Purpose

The report generator, REPGEN, can be used to simplify and automate the production of routine reports in a variety of settings. REPGEN provides for the retrieval and presentation of data from a Data Storage System (DSS) or text file in a pre-specified, user-designed, fixed format. The format is the equivalent of a blank form onto which variable information is entered in designated locations. REPGEN could be used, for example, in a water control operations setting to automatically produce routine reports showing the current stage and flow at selected locations in a river basin.

1.2 Acknowledgements

Development of computer program REPGEN was supported by the Rock Island and St. Louis Districts, Corps of Engineers, in support of their water management activities. The basis for and structure of the program were conceived by Art Pabst. The program was designed and written by Paul Ely, Consultant. Substantial testing of the program was performed by the Sacramento District, which resulted in suggestions for a number of program improvements. Many of these improvements are incorporated in the present version of the program. The User's Manual was written by Paul Ely with contributions from Art Pabst, Dennis Huff, Al Montalvo, Carl Franke and John Peters. Bill Eichert was Director of the Hydrologic Engineering Center during development of the initial version of the program; Darryl Davis is the current Director.

1.3 Differences From Previous Versions

Arithmetic operations can be performed using multivalued-time-series variables.

Functions to accumulate time-series values and compute differences between successive values of a time series have been added.

Functions to round values to number of significant digits or to a designated place have been added.

Each moment in time is a single entity. Date and time-of-day are treated as a unit. Time can be adjusted using multi-term expressions. Time expressions can be used wherever time is specified. Functions to get/set specified components of time have been added.

A function to set the date for end-of-month for a given date has been added.

Function DATATIME returns a list of times for the designated variable. This list can be displayed in rows or columns like a time series.

New PICTURE characters are used to indicate which components of time will be displayed in the report.

When a variable which appears in the report is redefined, the last value will be displayed in the report. In previous versions of REPGEN the first-defined value was displayed.

An instruction to display intermediate variable values has been added to aid in debugging reports.

The method for computing a time window using DURATION and STIME or ETIME has been changed.

Chapter 2

Description

REPGEN essentially fills in designated locations in a user-defined form with variable text or numeric information that is retrieved from an external source, such as an operational, hydrologic database. In particular, data values can be entries from a DSS time series, numeric values from a text file, or blocks of text from a text file. REPGEN also provides for conditional text messages, such as missing data flags, that depend on data values.

Figure 1 is an example of a report created with REPGEN.

```
*****  
  
Date: 01 JUNE 1985           Time: 0800  
  
Gage           Stage       Discharge  
Rescue         115.20      26450  
  
                WARNING: ABOVE FLOOD STAGE  
  
*****
```

FIGURE 1. Example Report

Figure 2 shows input for creating the report in Figure 1. Program input includes the report form, data definitions and data display formats. The input data set is divided into 'form' and 'definitions' sections by the lines #FORM, #ENDFORM, #DEF, and #ENDDEF.

Locations where data is to be inserted are indicated by variable labels beginning with '%'. In Figure 2 the labels are %BASDATE, %BTM, %STG, %FLOW, and %MESSAGE.

```

#FORM
*****

Date: %BASDATE           Time: %BTM

Gage                     Stage           Discharge

Rescue                   %STG           %FLOW

%MSG

*****

#ENDFORM
#DEF
%BASDATE
PICTURE = DDBAAAByyyy
%BTM = TIME (0800)
PICTURE = ZZZT
%STG
FILE = GAGEDATA TYPE = DSS A=Muddy B=RESCUE
C=STAGE E=1HOUR F=OBS
TIME = %BTM
PICTURE = NNN.ZZ
%FLOW
C = FLOW PICTURE = NNNNN
#IF %STG > 100
%MSG = "WARNING: ABOVE FLOOD STAGE"
#ELSE
%MSG = " "
#ENDIF
#ENDDEF

```

FIGURE 2. Input File

A variable definition starts with the label as the first item in the line and includes all lines up to the next line beginning with a label. In Figure 2 the definition of %STG is:

```

%STG
FILE = GAGEDATA TYPE = DSS A=Muddy B=RESCUE
C=STAGE E=1HOUR F=OBS
TIME = %BTM
PICTURE = NNN.ZZ

```

FILE is the file where stage data is located. TYPE shows the file to be a DSS file, and A, B, C, E and F set parts of the DSS pathname. TIME sets the time associated with the data retrieved from the DSS record.

PICTURE establishes the display format of the data. There is a one-to-one correspondence between PICTURE characters and characters placed in the report. The first character of the displayed data (which may be a blank) will be placed in the report where the % of

the corresponding variable label is located.

The report time is specified by variables %BASDATE and %BTM. Values for these variables can be set in the program execution command, or they will default to the current date and time when the program is run.

REPGEN provides capabilities for retrieving time series data from DSS. An element of the time series can be referenced individually or as a group of contiguous elements referenced collectively.

(This page intentionally left blank)

Chapter 3

Program Use

REPGEN was designed to use parameters on the execution line. Files and report time can be set using parameters in the program execution command as follows:

```
REPGEN [-option] [parameter]...
```

"parameter" has the form keyword=value

When option character D is used, values of variables named on #SHOW instructions will be written to the output file.

File names and program parameter values used by the program are:

<u>Parameter</u>	<u>Meaning</u>	<u>Default</u>
IN	Input filename	Standard input
OUT	Output filename	Standard output
REPORT	Report filename	report
FUNFILE	PREAD functions filename	repfun
MACFILE	PREAD macro filename	repmac
DATE	Date for intrinsic time variables %BASDATE and %BTM	today's date
TIME	Time for intrinsic time variables %BASDATE and %BTM	current time

DATE establishes the date for the variables %BASDATE and %BTM. It is defined with a 9-character date of the form DDMMMYYYY, where DD is the day of month, MMM are the first three letters of the month name, and YYYY is the year; e.g., 01MAY1985. Similarly, TIME establishes the time-of-day for the variables %BASDATE and %BTM. It is defined with a 4-character time from a 24-hour clock of the form HHMM, where HH is hour and MM is minutes, e.g., 0800. %BASDATE and %BTM are described in a following section.

Error messages are written to the terminal if the job is interactive; otherwise, they are written to batch job output. Several work files not listed here are also used; a complete list can be obtained with the execution command "REPGEN ?". Other files are used by REPGEN when they are named in the variable definitions.

On an IBM PC, the filenames and parameter values are:

<u>Parameter</u>	<u>Meaning</u>	<u>Default</u>
IN	Input filename	CON
OUT	Output filename	CON
REPORT	Report filename	REPORT
FUNFILE	PREAD functions	REPFUN
MACFILE	PREAD macros	REPMAC
DATE	Date for intrinsic time variables %BASDATE and %BTM	today's date
TIME	Time for intrinsic time variables %BASDATE and %BTM	current time

Chapter 4

Input Description

4.1 Input Format

The words 'line' and 'card' in this input description are used interchangeably to refer to a line in an input file. In parameter definitions, items enclosed in angle brackets, $\langle \rangle$, are generic names; items enclosed in brackets, $[\]$, are optional; and a bar, $|$, is used to separate items where one or the other is to be used.

Lines beginning with # are used to divide the input file into the report form and variable definitions and to control processing. The '#' character must be in column one. Data is read from the input file in the following sequence:

```
Optional Comments
#FORM
Body of Form (no comments allowed)
#ENDFORM
Optional Comments
#DEF
Variable Definitions
#ENDDEF
Optional Comments
```

#Preview

If a #PREVIEW line appears before the #FORM line, the report will show the PICTURE strings for each variable. Each line of the form will be printed twice: first the original line; then the same line with the variable label replaced by its PICTURE.

```
#SHOW <variable> [PICTURE = <string>]
```

The #SHOW instruction can be used to display intermediate values of a variable. Place the #SHOW instruction after the variable definition and the value of the variable will be written to the output file. Subscripts can be used to display selected elements of a time-series variable. An optional picture string can be used for numeric values. Note that values are not written to output unless option character D is used on the program execution command.

#FORM and #ENDFORM

#FORM and #ENDFORM lines define the limits of the report form. Comments cannot be included in the report form.

#DEF and #ENDDF

#DEF and #ENDDF lines define the limits of the variable definitions. However, comments may be used within this portion of the input file.

#IF-#ELSEIF-#ELSE-#ENDIF

#IF, #ELSEIF, #ELSE, and #ENDIF are used to skip lines in the variable definitions depending on the condition on a #IF or #ELSEIF line.

```
#IF <condition>
    variable definitions
#ELSEIF <condition>
    variable definitions
#ELSE
    variable definitions
#ENDIF
```

#ELSEIF and #ELSE lines are optional, but #IF and #ENDIF lines must be paired.

<condition> is a logical expression which can be evaluated to be true or false. A logical expression is two arithmetic expressions combined by a comparison operator.

Allowable comparison operators are:

=	equal
<>	not equal
<	less than
>	greater than
<= or =<	less than or equal
>= or =>	greater than or equal

Logical expressions may be joined or negated by logical operators. Allowable logical operators are:

NOT	negation
AND	logical and
OR	logical or

Example:

```
#IF MONOFYR(%date) >= 5 AND MONOFYR(%date) <= 10
%LEVEL = 11.7
#ELSE
%LEVEL = 12.9
#ENDIF
```

Comment

Lines with a # in the first column and a blank in the second column are ignored for report generation and can be used for comments.

Comments may not be included in the report form. Further, any line occurring before the #FORM line, between the #ENDFORM and #DEF lines, or after the #ENDDEF line will be ignored, and can therefore be used for comments.

4.2 Report Form

The report form is a literal image of the report, except that locations where data are to be inserted are indicated by variable labels. The number of characters occupied by a data value and the position of the field relative to the label position are specified in the display field definitions. If the specified field width is less than the width of the variable label, the line will be compressed to compensate for the difference in the number of characters in the variable label and the field width. If the field width is greater than the width of the variable label, adjacent characters will be overwritten.

4.3 Variable Definitions

Variable definitions describe the assignment of values represented by labels and their display formats. The value of a variable may be defined by: reference to a file; a direct assignment to another variable; assignment to the result of a mathematical expression; or assignment to a function.

Six types of variables can be used: numbers, times, alphanumeric strings, text, list, and paired-data. Numbers are read from DSS files or text files, or result from evaluating a mathematical expression. Numbers can be used in mathematical or logical expressions. Time consists of date and time-of-day. Alphanumeric strings are missing-data strings or other strings assigned in a statement. Text is read from a file. A single variable may consist of a block of one or more lines of text. List variables are used as parameters in math functions and cannot be displayed. Paired-data are read from DSS, and are used to interpolate values for number variables. Paired-data cannot be used in a report, but the values can be displayed using #SHOW.

A variable definition is entered in free form on one or more lines. The definition begins with a label line which has a label as the first item in the line. A variable label is two to eight characters beginning with a '%'. Only numerals (0, 1, ..., 9) and letters (A, B, ..., Z) may be used besides the initial '%'. No spaces are allowed in variable labels. A '%' in the report form may be used to indicate percent when it is followed by a space. The variable definition continues until a % line is entered for the next variable or an #IF, #ELSEIF, #ELSE or #ENDIF statement is encountered. Comments can be included in a variable definition by placing a '#' in the first column of the comment line. Any line within the variable definition portion of the input file which does not have a # or % as the first non-blank character is assumed to contain parameters defining the variable. Blank lines are ignored.

Variables defined more than once always have the value last defined at any point in the input sequence. Variables whose value is displayed in a report should not be defined more than once, but if they are the last-defined value will appear in the report. A warning message will be printed when a report variable is redefined. If too many report variables are redefined, REPGEN may run out of memory.

There is no distinction between upper and lower case characters in the variable definitions. Lower case characters will be interpreted as upper case, except for characters enclosed within double quotes, ", or the single character following an up arrow, ^.

4.3.1 Time Specification

Time consists of two parts, date and time-of-day. These two values are used together to indicate a single moment in time.

Time is specified as <date> <time_of_day> where

<date> is a 9-character date of the form ddmmmyyyy (using 2 digits for year implies year in the 1900's), e.g., 21MAR1982, 03JAN88;

<time-of-day> is a 4-character time in the form hhmm based on a 24-hour clock, e.g., 1630, 0400.

4.3.1.1 Time Expressions

Time expressions can be used wherever time is specified in REPGEN.

A time expression consists of

```
<time> [+/- <increment>] [+/- <increment>] ...
```

where <time> can be expressed as
a time variable, or
<date>[<time_of_day>], or
[<date>]<time_of_day>;

+/- means plus or minus;

<increment> is expressed as an integer, n, combined with a unit of time: nMINUTE, nHOUR, nDAY, nMONTH, nYEAR. HOUR, DAY, MONTH, and YEAR can be abbreviated using their first letter. MINUTE can be abbreviated as MIN.

4.3.2 Predefined Variables

There are five variables which are defined when REPGEN begins: %CURDATE, %CTM, %BASDATE, %BTM, and %MISSDTA.

%CURDATE, %CTM, %BASDATE and %BTM are reference times. They may be used as arguments for time in a function or may be displayed in the report.

%CURDATE and %CTM are initially set to the same time. Both variables are defined to maintain compatibility with older reports.

%BASDATE and %BTM are initially set to the same time. Both variables are defined to maintain compatibility with older reports.

%BASDATE - time set using execution line parameters DATE and TIME. If DATE or TIME is not defined, the current date and/or time-of-day are used.

%BTM - same as %BASDATE.

%CURDATE - time when REPGEN begins running.

%CTM - same as %CURDATE.

%MISSDTA is a numeric value which is used to indicate missing values in a time series and may be defined by the user. The string set by parameter MISSTR will be displayed in the report for values equal to %MISSDTA.

4.3.3 Time Series Variables from DSS

Time series variables are read from DSS. A variable can be a single value or a series of values occurring within a time window.

Parameter TIME is used to read a single value from a time series. Parameters STIME, ETIME, and DURATION are used to set a time window to read a series of values. Up to 1000 values may be included in the window. Once the window is defined, any element or series of elements in the series may be referenced by a subscript or by the ELEMENT function. If a report uses several values from a time series, it is most efficient to define a time window which includes those values and then use subscripts or functions to identify the subsets or individual elements of the series in the definitions immediately following in the input sequence.

Parameters defining a time series variable are:

FILE = <file name>

TYPE = DSS

A = <pathname part A> F = <pathname part F>

Used to specify parts of the DSS pathname. If a blank is to be included in a pathname, the pathname part must be enclosed in double quotes, e.g., A="WHITE RIVER". Pathname parts remain in effect for all variable definitions until they are changed.

STIME = <time_expression>

Sets start time for a time series to be read from DSS.

ETIME = <time_expression>

Sets end time for a time series to be read from DSS.

DURATION = <duration>

Sets the duration of a time window. Duration is used with STIME or ETIME. <duration> can be nHOUR, nDAY, nWEEK, nMONTH, or nYEAR, where n is an integer. For example:

```
DURATION = 1MONTH
DURATION = 15DAY
```

REPGEN computes STIME or ETIME using the following formulas:

```
ETIME = STIME + DURATION - 1 minute
STIME = ETIME - DURATION + 1 minute
```

TIME = <time_expression>

Sets time for locating a data value in a DSS time series.

REPGEN gets the last entry at or before this time. The actual time of an entry can be obtained with the function DATETIME.

VALUE = <value type>

Sets allowable type of data value read from DSS. Default value is MISSOK. When VALUE = MISSOK, values representing missing data are 'acceptable'. That is, if the last entry at or before the time set by TIME represents a missing value, that value will be displayed as MISSTR. If the entry is a valid number, it will be displayed as defined by PICTURE.

If VALUE = NOMISS, the last entry, at or before the time set by TIME, not representing a missing value will be returned from DSS. The function DATETIME may be used to obtain the actual time of the data value.

If VALUE = EXACT, only a value which occurs at the time set by TIME will be returned from DSS. If no value is recorded at this time, the value will be undefined, and the string set for UNDEF will be displayed in the report.

VALUE is ignored when STIME and ETIME are used to define a time window.

4.3.4 Variables from Text Files

Text or numbers can be read from a text file. A text or number variable can be defined by setting parameters which specify the location of a piece of data in a text file. This file reference is defined by setting several parameters. Once a parameter is set, its value will be used for all future variables until it is redefined.

Parameters used for a text file reference are:

FILE = <file name>

TYPE = TEXT

START = <start string>

<start string> is an optional character string. A text file will be searched until <start string> is found. The line containing this string will be line zero. START should be null ("" or blank) if absolute line numbers are to be used.

END = <end string>

<end string> is an optional character string. Text lines will be copied until this string is encountered. The line containing this string will not be copied. END should be null ("" or blank) if absolute line numbers are to be used.

LINE = <first line> [- [<last line>]]

This parameter sets the lines of a text file which will be copied to the report file. <first line> is the number of the first line to be copied. <last line> is the number of the last line to be copied. If START is not null, line numbers will be relative to the line containing <start string>.

Examples:

LINE = 10	will copy line 10
LINE = 10-30	will copy lines 10 through 30.
LINE = 10-	will copy from line 10 to end-of-file (or line containing <end string> if <end string> is not null).

COL = <first column> [- [<last column>]]

This parameter sets the columns of a text file which will be copied to the report file.

Examples:

COL = 10	will copy column 10
COL = 10-30	will copy columns 10 through 30
COL = 10-	will copy columns 10 to end-of-line

A text block (from a text file) is printed as it appears in the text file. The first line and column of the text block is printed where the first character (%) of the variable label appears in the report form. If the text block contains more than one line, additional lines will be added to the report as required by the text.

When a text-block variable appears on a report line, no other variables can be placed to the right of that variable on the line.

A number may be read from a text file only if these conditions are met:

- 1) Only one line is read from the file, and
- 2) the characters in the specified columns can be converted to a number.

4.3.5 Variables From Mathematical Formulae and Functions

A formula is a mathematical expression beginning with an equals sign, '='. The arithmetic operations (and operators) are addition (+), subtraction (-), multiplication (*), and division (/). Multiplication and division are performed before addition and subtraction. Parentheses may be used to modify the order in which operations are performed. Spaces in a mathematical expression are ignored.

If any term or factor in a mathematical formula has a value indicating missing data, the result of the formula will be a value indicating missing data.

Example: %CHANGE = %NEW - %OLD
 %A = (1.2*%P1 + 1.4*%P2 + 1.1*%P3) / 3

If one of the variables to the right of the equals sign is a missing data value, the variable to the left of the equals sign will be set to %MISSDTA.

A time series can be used in a mathematical expression. In that case the designated operation is carried out on each element of the time series. When two time series are combined using an arithmetic operation, the operation is performed using corresponding elements from each time series. Both time series must have the same number of elements and the corresponding values must occur at the same time. Each calculation is treated individually, i.e., if one time series contains a missing value, the result will have a missing value at the corresponding time, but values at other times will not be affected by the missing value. Two irregular-interval time series cannot be combined with an arithmetic operation.

4.3.6 Mathematical Functions

The following functions can be used in a mathematical expression:

SUM (<treat>,<list>) - summation of parameter list

MAX (<treat>,<list>) - find maximum value in parameter list

MIN (<treat>,<list>) - find minimum value in parameter list

AVERAGE (<treat>,<list>) - compute average of values in parameter list

COUNT (<list>) - number of valid data values in list

<list> consists of a one or more variables separated by commas. Spaces are ignored. Subscripted variables (see Section 4.3.9) can be used in a list.

<treat> indicates treatment of missing data:

- | | | |
|--------|---|--|
| ZERO | - | use zeros for missing data; |
| IGNORE | - | do not use missing data and decrease number of values used to compute average; |
| MISS | - | if missing data occurs in list, return missing data value for function. |

For example:

Assume -901 indicates missing data, and let %a = 10, %b = -901, %c = 20, then MIN (ZERO, %a, %b, %c) is 0, and MIN (IGNORE, %a, %b, %c) is 10, and MIN (MISS, %a, %b, %c) is -901.

ACCUM (<treat>,<variable>) - generates a time series consisting of the cumulative amount from the beginning of the time series, <variable>, to the current value.

<treat> indicates treatment of missing data:

- ZERO - use zeros for missing data;
- IGNORE - do not use missing data. Corresponding value is missing, but remaining values are not affected;
- MISS - if missing data occurs in time series, remaining cumulative values are missing.

DIFF (<treat>,<variable>) - generates a time series consisting of the differences between the current value and the preceding value. The difference for the first value is undefined.

<treat> indicates treatment of missing data:

- ZERO - use zeros for missing data;
- IGNORE - do not use missing data. Corresponding difference is undefined;
- MISS - if data is missing, resulting difference is missing.

RNDDIG (<variable>,<ndig>) - makes a copy of <variable> with the values rounded to <ndig> significant digits.

RNDPOS (<variable>,<place>) - makes a copy of <variable> with the values rounded to <place> where <place> designates a position as a power of 10, e.g., -1 specifies rounding to the nearest tenth (0.1).

INTERP (<ts-variable>,<pd-variable>,<n>) - uses linear interpolation to compute a value for each value in a time series. Result is a time series.

<ts-variable> is the time-series variable,
<pd-variable> is a variable containing paired-data values, and
<n> is 1 or 2, indicating that the first or second value in the data pairs is the dependent value.

For example: if %STG contains stage data, and %STGFLO contains a stage-flow rating table, with stage first in each pair, then INTERP (%STG,%STGFLO,2) will compute flows corresponding to the stages in %STG.

Subscripted variables can be used in the preceding functions.

4.3.7 Time-Related Functions

These functions are used to define the value of variables which refer to time.

<time_string> a string including time-of-day or date or both separated by a space

<time_variable> a variable which points to a date and time-of-day

<time_expression> an expression which begins with a <time_string> or <time_variable> which can be followed by modifying increments of time. (See paragraph on time expressions above.)

DATETIME () - returns the actual time for the last data value read from a DSS time series. Assign result to a time variable. (Obsolete: use following form of DATETIME.)

DATETIME (<time_series>) - returns a time list for <time_series>. Assign result to a time variable.

DAY (<time_expression>) - returns two-digit day of month as a number.

DAYOFWK (<time_expression>) - returns name for day of week. Day of week is left-justified in a 9-character string. Use PICTURE to mask characters to be displayed.

DAYOFYR (<time_expression>) - returns day of year as a number.

DMY2DATE (day, month, year) - gets date from integers for day, month, and year. Time-of-day is assumed to be 2400 hours. Assign result to a time variable.

EOM (<time-expression>) - returns date of end-of-month for given date. Time-of-day is the time-of-day from the time expression, or 2400 if time-of-day cannot be determined from the expression.

HOUR (<time_expression>) - returns hour from time_expression as a number.

LASTHOUR (<time-expression>) - time-of-day is truncated to the last whole hour. Date is the same as the date from the time expression, or undefined if date cannot be determined from the expression.

MINUTE (<time_expression>) - returns minute from time_expression as a number.

MONOFYR (<time_expression>) - returns month of year as a number.

MONTH (<time_expression>) - returns month name. Month name is left-justified in a 9-character string. Use PICTURE to mask characters to be displayed.

NDAYS (<time_expression>) - returns the number of days in the month given by time expression.

NEARHOUR (<time_expression>) - time-of-day is rounded to nearest whole hour. See **LASTHOUR**.

NEXTHOUR (<time_expression>) - time-of-day is set to next whole hour when minutes is greater than zero. See **LASTHOUR**.

SETTIME (<time_expression>, <component>, <value>) - begin with time from <time_expression>, then change the designated component of time to the given value. When <component> is YEAR, MONTH, DAY, HOUR, or MINUTE, <value> is a number or a variable with a numeric value. If <component> is DATE or TIME, <value> is a time_expression, and date or time-of-day is changed to the value from the <value> time_expression. Assign result to a time_variable.

TIME (<time_string>) - identifies a string as a time string. Assign result to a time variable, e.g.,
%btm = TIME (0400 12JAN88)

TIMEWHEN (<time_series>, <op>, <value>) - finds the time when a time series reaches a specified value. <time_series> is a time-series variable, <op> is one of the operators:

LT	-	less than
LE	-	less than or equal
EQ	-	equal
GE	-	greater than or equal
GT	-	greater than

<value> is a number or a variable.

YEAR (<time_expression>) - returns year as a four-digit number.

YEAR2 (<time_expression>) - returns year as a two-digit number.

YMDHM2T (<year>, <month>, <day>, <hour>, <minute>) - gets a time (date and time-of-day) from integers for year, month, day, hour, and minute. Assign result to a time variable.

SELECT (FIRST | CENTER | LAST, <time_series>, <start_time>, <end_time>)

The **SELECT** function selects one value from a time series for a time window. **FIRST**, **CENTER**, **LAST** indicate which value to return. **FIRST** means the result will be the first value encountered in the time window. **CENTER** indicates the value nearest the center of the time window, and **LAST** means the last value in the time window. If there are no data in the time window the result is a missing value. <time_series> is a time-series variable. The resulting value is selected from this time series. <start_time> is a time expression defining the beginning of the time window. <end_time> is a time expression setting the

end of the time window.

`SELECT (NEAREST, <time>, <time_series>, <start_time>, <end_time>)`
NEAREST indicates that the values nearest <time> will be returned. <time> is a time expression. <time_series> is a time-series variable. The resulting value is selected from this time series. <start_time> is a time expression defining the beginning of the time window. <end_time> is a time expression setting the end of the time window.

`SNAP (<time_series>, <interval>, <offset>)` The SNAP function creates a regular-interval time series from a time series. SNAP may be used to convert irregular-interval data to regular intervals, or to convert regular-interval data to another time interval. <time_series> is a time-series variable. <interval> is a time interval specified as nMINUTE, nHOUR, or nDAY. <offset> is a time string indicating the offset from a series of intervals beginning at midnight. For example, 6HOUR data is normally reported at 0600, 1200, 1800, and 2400; but setting <offset> to 0700 moves the times to 0700, 1300, 1900 and 0100.

4.3.8 Grouping of Variables

`GROUP (list)` - can be used to assign a label to a list of variables for use in the mathematical functions described previously.

`IGROUP (<index string>, n1, n2, <index label>)` - used to define a group using an index to reference the variables. n1 and n2 are integers; n2 > n1; the index string takes on the values of n1 through n2 and replaces that string in <index label>. Example: `IGROUP (xx,1,31,%flowxx)` will create a group of variable labels consisting of %flow1, %flow2, ..., %flow31.

Example:

The summation of variables can be computed by:

`%SUM = %15 + %42 + %23 + %85`

or `%SUM = SUM (MISS,%15,%42,%23,%85)`

or `%A = GROUP (%15,%42,%23,%85)`
`%SUM = SUM (MISS,%A)`

4.3.9 Subscripted Time Series Variables

A numeric variable may be defined from individual elements or a subset of a time series defined within a time window either by assignment or through a mathematical expression or function. The subset is similar to a group. The element or subset may be referenced by position

or by time using subscript notation or using the ELEMENT function.

Subscript notation is:

`%var(<elem>)`

where <elem> is:

- | | | |
|----------|---|--|
| position | - | a number indicating position in the time series |
| | - | START, indicating the first element of the time series |
| | - | END or LAST, indicating the last element of the time series |
| time | - | indicating the element occurring at time
where time is a time expression. |

For example, if a time series, %X, begins at 02JUL1987,0400 and ends at 06JUL1987,0300 with time interval of 1HOUR, then

<code>%X(5)</code>	is the element occurring at 02JUL1987,0800
<code>%X(START)</code>	is the element occurring at 02JUL1987,0400
<code>%X(END)</code>	is the element occurring at 06JUL1987,0300

If %DAT1 is 02JUL1987 and %TIM1 is 0400, then

<code>%X(%DAT1,%TIM1)</code>	is the element occurring at 02JUL1987,0400
<code>%X(%DAT1,%TIM1+24H)</code>	is the element occurring at 03JUL1987,0400
<code>%X(04JUL1987,1200)</code>	is the element occurring at 04JUL1987,1200

A subset of a time series can be referenced by position, times for the beginning and end of the subset, or by a block using the following notation:

`%VAR(<range>)` where range is

- | | | |
|----------------------|---|-------------------------------------|
| position1, position2 | - | beginning position and end position |
| time1,time2 | - | beginning time and end time |
| block | - | a month or year |

Examples of block specification are:

- | | | |
|-------|---|---|
| JAN87 | - | refers to a time series for January 1987 |
| JAN | - | refers to a time series for January; the year is based on the
parent time series |
| MONTH | - | uses month and year from %BASDATE |
| YEAR | - | uses year from %BASDATE |

For example, if a time series, %Y, starts on 01OCT1986 and ends on 30SEP1987 with time interval of 1DAY, then

%Y(NOV) - refers to the time series for November 1986

The ELEMENT function provides reference to an element in a time series:

ELEMENT (<time_series>, BEFORE | AFTER | AT, <time>, <missval>)

<time_series> is a time-series variable,

<time> is a time_expression, and

<missval> indicates the action to take when the nearest data element is a missing value. If <missval> is MISSOK the missing value will be returned. If <missval> is NOMISS the nearest non-missing value will be returned if one exists in the time series.

BEFORE and AFTER indicate the direction from time to look for the nearest data element. When AT is specified, the resulting value will be undefined if there is not a value at the given time.

4.3.10 Missing Values

MISSVAL (<list>) - used to define the numeric values with which missing data is indicated. Up to 10 numbers can be included in <list>. Use: %MISSDTA = MISSVAL (-901,-902). The variable %MISSDTA is set to the first value in the list.

%MISSDTA - set to the first missing data value listed in the MISSVAL function.

%MISSDTA is initially set to -901. If a missing data value is encountered while a mathematical formula is being evaluated, the result of the formula will be set to %MISSDTA. A special string, set by the MISSTR parameter, will be displayed in the report when a variable is equal to one of the missing data values.

KNOWN (<variable>) - used in a logical expression to determine if the value of <variable> is known. If the value is missing or undefined, KNOWN is false.

4.3.11 Display Formats for Variables

Display parameters determine how the value of a variable will be displayed in the report. Once a parameter is set, its value will be used for subsequent variables until the parameter is redefined.

DIGITS = <number of significant digits>

Digits sets the number of significant digits (n) to be displayed for a variable. The displayed number will be rounded to n significant digits. Non-significant digits will be shown as zeroes or blanks. All digits will be considered significant if n = 0.

PICTURE = <picture>

The appearance of the display field is shown in a picture. The picture consists of one or more of the characters described below. The number of characters in the picture is the maximum number of characters in the display field. If the number of digits in a number is greater than the field width, the field will be filled with asterisks (*). If an alphanumeric value has more characters than the field width, the left-most characters will be shown, and the right-most characters will be truncated.

A numeric display is defined by PICTURE characters S, N, Z, B, comma, and period (see below). Numeric fields are assumed to have an implied decimal point to the right of the field if a decimal point does not appear in the picture.

Time displays are defined by picture characters A, B, D, M, Y, T, and Z. Characters D, M, Y, and T select the component of time to be displayed. An A is used to indicate that a month name is to be displayed instead of the month number.

Alphanumeric values will be left-justified in the field. Day of week, and month of year use PICTURE to determine how they will be displayed. Other strings such as set by MISSTR and IF-THEN-ELSE statements appear exactly as they are given. Only picture characters A and B have special meaning for displaying alphanumeric fields.

PICTURE is not used to format a text block.

PICTURE is initialized to be "?PICTUR?", so it will be necessary for the user to define a picture for all variables.

- S - Forces the sign of a number to be displayed. This must be the first character of a picture if it is used.
- N - A numeric digit appears in the display field. Non-significant zeroes are printed as blanks, except where the number of significant digits requires trailing zeroes to be shown.
- Z - A non-significant zero is to be displayed. This will override the number of significant digits.
- . - A decimal point (.) is displayed. Only one decimal point may be used in a picture.
- , - A comma (,) is displayed. More than one comma may be used. Commas preceding the first significant digit will appear as blanks.
- A - An alphanumeric character will be displayed.

- B - A blank will be displayed.
- D - A numeric digit from the day component of time will be displayed.
- M - A numeric digit from the month component of time will be displayed.
- Y - A numeric digit from the year component of time will be displayed.
- T - A numeric digit from the time-of-day component of time will be displayed.
- F - A flag character will be displayed. Flag characters will be based on flag values from DSS. Note: This feature does not yet exist in DSS. Until it does, a blank will be printed for the letter F.
- H - A numeric digit from the hour portion of time-of-day will be displayed.

Any other character which does not have a special meaning defined above may be used in a picture. This character will appear in its corresponding position in the display field.

Examples:

Value	Picture	Display
25.40	NNN.NN	24.4
25.40	SNNN.NN	+25.4
25.40	NNN.ZZ	25.40
25.40	ZZZ.ZZ	025.40
07JAN1985	AABAAAABAAAA	07 JAN 1985
abcde	AABABAA	ab c de
100000	N,NNN,NNN	100,000

Assume DIGITS = 3:

1.20	NNNBNNN.NN	1.20
12.04	NNNBNNN.NN	12.0
120.45	NNNBNNN.NN	120.
1204.5	NNNBNNN.NN	1 200.
12045	NNNBNNN.NN	12 000.
1234567	NNNBNNN.NN	*****
12.34	NNNBNNN	12

MISSTR = <missing data string>

This parameter sets a string which will be displayed in the data field whenever a variable has a value equal to a value which represents missing data. MISSTR is initialized to be "?MISSTR?".

UNDEF = <undefined variable string>

This parameter sets a string which will be displayed for elements of a time series which are undefined. When a time series has more positions in the report form than values in the time series, this string will be displayed for the extra positions.

UNDEF defaults to the variable label.

JUSTIFY = L | C | R

This parameter sets how alphabetic and time strings will be justified within the display field. The string can be left justified (L), centered (C), or right-justified (R).

Alphabetic strings are normally left-justified.

Time is displayed using the PICTURE characters D, M, Y, H and T for day, month, year, hour, and time-of-day respectively. These characters can be preceded by Z's where leading zeros are to be displayed, but the last character of each component must be a D, M, Y, H or T. The letter A can be used to indicate that the alphabetic month is to be displayed instead of a numeric month. The month name will be right-justified in the space reserved by for month name. If the month name is longer than the reserved space, the right-most characters will be truncated.

Examples:

Picture	Display
DDbAAAbYYYY	4 JUL 1987
ZDbAAAbYYYY	04 JUL 1987
MM/DD	7/4
MM/DD/YYbbHH	7/4/87 8
ZM/ZD/YYbbHH	07/04/87 08
DDbAAAbYY	4 JUL 87
ZDbAAAbYY	04 JUL 87
AAAbDD,bYYYY	JUL 4, 1987
AAAAAAAAAbDD,bYYYY	JULY 4, 1987
ZZZT	0830
ZZ:ZT	08:30
ZT	30
TZ:ZT	8:30
HH	8
ZH	08
AAAAAAAAAbDD,bYYYYb@bZZZT	JULY 4, 1987 @ 0830

Time strings are normally right-justified within the picture. The JUSTIFY parameter can be used to left-justify or center the string.

4.3.12 Row and Column Displays

Time series data can be printed in rows or columns in the report. The variable label is placed wherever a time series value is to appear. As each line of the report form is processed, the line is scanned from left to right, and the labels are replaced by the corresponding data value. That is, the first appearance of a label is replaced by the first value of the time series, the second appearance is replaced by the second value, etc. So, if a label is repeated in a row, the time series will be printed in a row. If the label is repeated in a column, the time series will be printed in a column. A label must be repeated for the number of values in the time series, otherwise all values will not be printed in the report. The UNDEF parameter, previously described, is used to define a string to replace values that may not exist under some circumstances, such as daily data for February 29. See Figure 6 for an example of input for a column-oriented report.

4.3.13 Conditional Messages

The #IF-#ELSEIF-#ELSE-#ENDIF statements may be used to display a message or flag when a condition is met by the data. For example:

```
#IF %stage > 1000
  %warning = "Flood Warning"
#ELSE
  %warning = " "
#ENDIF
```

and

```
#IF %b >= 100 AND %b < 110
  %range = "A"
#ELSEIF %b >= 110 AND %b < 125
  %range = "B"
#ELSEIF %b >= 125 AND %b < 150
  %range = "C"
#ELSEIF %b >= 150
  %range = "D"
#ELSEIF %b = %missdta
  %range = " "
#ENDIF
```

4.4 Function Characters

REPGEN uses PREAD to read and expand special characters which have been defined in the function file referenced by execution line parameter FUNFILE. Function characters are only expanded within the variable definitions. They are not expanded in the report form. Function expansion can be turned off or on by including PREAD instructions in the input file. Function character expansion is initially "on" when REPGEN begins execution. See the PREAD manual included in this document for details.

Chapter 5

Examples

Figure 3 shows an example input file that could be used to produce an annual streamflow report. This example shows how time series can be printed in columns.

#FORM

NATURAL FLOWS AT HAWLY
%YEAR WATER YEAR

	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP
1	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
2	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
3	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
4	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
5	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
6	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
7	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
8	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
9	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
10	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
11	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
12	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
13	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
14	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
15	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
16	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
17	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
18	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
19	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
20	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
21	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
22	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
23	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
24	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
25	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
26	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
27	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
28	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
29	%OCT	%NOV	%DEC	%JAN	%FEB	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
30	%OCT	%NOV	%DEC	%JAN	---	%MAR	%APR	%MAY	%JUN	%JUL	%AUG	%SEP
31	%OCT	---	%DEC	%JAN	---	%MAR	---	%MAY	---	%JUL	%AUG	---
AVG	%A10	%A11	%A12	%A01	%A02	%A03	%A04	%A05	%A06	%A07	%A08	%A09
MAX	%X10	%X11	%X12	%X01	%X02	%X03	%X04	%X05	%X06	%X07	%X08	%X09
MIN	%N10	%N11	%N12	%N01	%N02	%N03	%N04	%N05	%N06	%N07	%N08	%N09

#ENDFORM

Figure 3. Example Input for Tabulating Gage Records

```

#DEF

# Set %DATE to 30SEPyearch using year from %BASDATE

%Y = YEAR(%BASDATE)
%DATE = DMY2DATE (30, 9, %Y)
%YEAR = YEAR(%DATE)    PICTURE = NNNN

%FLOW    FILE=DSSDLY  TYPE=DSS
        A=LACKAWAXEN B=HAWLY C=FLOW-NAT E=1DAY F=CMP
        ETIME=%DATE  DURATION=1YEAR

%JAN = %FLOW(JAN) PICTURE = NNNN DIGITS = 3 MISSTR = " ---"
%FEB = %FLOW(FEB) UNDEF = " ---"
%MAR = %FLOW(MAR)
%APR = %FLOW(APR)
%MAY = %FLOW(MAY)
%JUN = %FLOW(JUN)
%JUL = %FLOW(JUL)
%AUG = %FLOW(AUG)
%SEP = %FLOW(SEP)
%OCT = %FLOW(OCT)
%NOV = %FLOW(NOV)
%DEC = %FLOW(DEC)

%A01 = AVERAGE("MISS",%JAN)
%A02 = AVERAGE("MISS",%FEB)
%A03 = AVERAGE("MISS",%MAR)
%A04 = AVERAGE("MISS",%APR)
%A05 = AVERAGE("MISS",%MAY)
%A06 = AVERAGE("MISS",%JUN)
%A07 = AVERAGE("MISS",%JUL)
%A08 = AVERAGE("MISS",%AUG)
%A09 = AVERAGE("MISS",%SEP)
%A10 = AVERAGE("MISS",%OCT)
%A11 = AVERAGE("MISS",%NOV)
%A12 = AVERAGE("MISS",%DEC)

%X01 = MAX("MISS",%JAN)
%X02 = MAX("MISS",%FEB)
%X03 = MAX("MISS",%MAR)
%X04 = MAX("MISS",%APR)

%X05 = MAX("MISS",%MAY)
%X06 = MAX("MISS",%JUN)
%X07 = MAX("MISS",%JUL)
%X08 = MAX("MISS",%AUG)
%X09 = MAX("MISS",%SEP)
%X10 = MAX("MISS",%OCT)
%X11 = MAX("MISS",%NOV)
%X12 = MAX("MISS",%DEC)

%N01 = MIN("MISS",%JAN)
%N02 = MIN("MISS",%FEB)
%N03 = MIN("MISS",%MAR)
%N04 = MIN("MISS",%APR)
%N05 = MIN("MISS",%MAY)
%N06 = MIN("MISS",%JUN)
%N07 = MIN("MISS",%JUL)
%N08 = MIN("MISS",%AUG)
%N09 = MIN("MISS",%SEP)
%N10 = MIN("MISS",%OCT)
%N11 = MIN("MISS",%NOV)
%N12 = MIN("MISS",%DEC)
#ENDDEF

```

Figure 3 Example Input for Tabulating Gage Records (cont.)

NATURAL FLOWS AT HAWLY
1943 WATER YEAR

	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP
1	634	620	682	2840	29	728	742	816	61	7	97	59
2	514	565	3360	154	280	576	69	55	719	71	75	57
3	428	64	190	1200	261	47	675	589	565	80	64	50
4	37	621	1060	952	260	415	525	577	457	78	56	50
5	378	498	842	712	271	380	553	487	377	97	55	55
6	1360	452	705	661	287	370	529	419	310	128	55	55
7	767	420	589	536	331	400	409	402	347	90	49	54
8	535	389	503	502	318	360	442	397	530	73	47	55
9	438	378	457	438	300	330	424	433	385	65	49	5
10	381	369	410	433	291	349	385	640	366	128	4	50
11	343	442	389	415	30	510	343	1070	34	15	49	45
12	304	385	381	37	360	2740	32	164	294	118	46	45
13	277	35	36	392	341	225	366	1830	258	89	44	45
14	26	335	318	385	330	1220	498	1040	240	163	50	42
15	261	290	357	335	311	1310	438	750	223	124	49	42
16	268	290	321	318	300	2060	397	621	201	87	44	43
17	449	284	307	365	290	4100	434	577	181	71	40	45
18	1040	402	300	425	300	3460	750	712	174	62	42	45
19	648	521	290	629	310	2540	1010	1180	159	56	41	4
20	487	397	281	743	350	3990	2290	1700	144	51	4	42
21	462	369	270	720	150	2480	1790	2650	15	11	41	40
22	655	345	301	64	3110	1480	161	358	126	178	40	38
23	577	31	29	570	3890	104	1130	1840	116	247	40	37
24	48	381	281	438	3620	842	878	1200	105	135	39	35
25	415	3110	270	334	3240	793	720	913	97	97	40	34
26	1510	3470	270	343	1600	758	675	1740	90	80	38	32
27	3690	1610	311	321	1160	758	559	1660	90	72	55	30
28	1600	1070	753	289	888	661	577	1070	82	63	202	29
29	1070	816	1860	309	---	530	530	869	82	70	124	2
30	800	800	6800	300	---	541	565	675	82	258	7	28
31	690	---	7140	290	---	962	---	571	---	15	65	---
AVG	680	659	970	531	829	1160	630	903	230	93	53	40
MAX	3690	3470	7140	2840	3890	4100	2290	2650	719	258	202	59
MIN	26	31	29	37	29	47	32	55	15	7	4	2

Figure 3 Example Output Gage Record

Chapter 6

Obsolete Features

This chapter lists features which were used in previous versions of REPGEN. Program changes attempted to minimize impact on current report forms. Most of these features continue to work. But they should not be used in new reports.

* Time window for reading data from DSS is specified using time expressions rather than specifying date and time-of-day separately. SDATE, EDATE, and DATE are no longer used.

SDATE = <date>

Sets start date for a time series to be read from DSS. <date> is a 9-character military date, e.g., 21SEP1987.

EDATE = <date>

Sets end date for a time series to be read from DSS. <date> is a 9-character military date, e.g., 21SEP1987.

DATE = <date>

Sets date for locating a data value in a DSS time series. <date> is 9-character military date, e.g. 03MAR1985.

REPGEN gets the last entry at or before this date. The actual date of an entry can be obtained with the function DATADATE.

* Obsolete Functions:

CLOCK - Use TIME function

DATADATE - Use DATATIME function

DATEWHEN - Use TIMEWHEN function

* Subscripts FIRST and START should not be used with time series variables. Use the number 1 instead.

* Using PICTURE characters A and N for dates and times:

Picture	Display
AABAAABAAAA	04 JUL 1987
NN/NN	7/4
NN/NN/NN	7/4/87
ZN/ZN/NN	07/04/87
NNBAAABNN	4 JUL 87
ZNBAAABNN	04 JUL 87
NNBAAABNNNN	4 JUL 1987
ZNBAAABNNNN	04 JUL 1987
AAABNN , BNNNN	JUL 4, 1987
AAAA	0830
AA:AA	08:30
ZZNN	0830
ZZ:NN	08:30
NN:NN	8:30