Published: 05/03/67

Identification

Formatted Time Conversion
get_calendar, put_calendar
L. B. Ratcliff

Purpose

The procedures get_calendar and put_calendar return or accept external calendar times as character strings in specific formats.

Get_calendar has two entries, one providing all the information that can be obtained from calendar_output but as a character string and the other providing a brief but normally-sufficient amount of time data as a character string.

Put_calendar accepts calendar time in character string form and under format control determines the input arguments for calendar_input. It then calls calendar_input to obtain the internal form of the calendar time.

Procedures calendar_output and calendar_input are described in Section BY.15.02.

Usage: get_calendar

The procedure get_calendar has two entries. To obtain all information obtainable from calendar_output, but as a character string,

```
call get_calendar$full (clock_time, full_string);
```

with the declaration

```
dcl clock_time fixed bin (71),
    full_string char (43);
```

The resultant string has the form

```
dd mmm xxxx.xx zzz www yyyy hh:nn:ss.uuuuuu
```

The time components begin in the character positions indicated below and are of length specified. Leading zeros are replaced by blanks in the day field as in the example below.

component	character position	field size	representation
day	1	2	dd
month	4	3	mmm
military time	8	7	xxxx.xx
time zone	16	3	zzz
day of week	20	3	www
year	24	4	yyyy
exact time	29	15	hh:nn:ss.uuuuuu

The following abbreviations are used for the months and weekdays:

Jan	Sun
Feb	Mon
Mar	Tue
Apr	Wed
May	Thu
Jun	Fri
Jul	Sat
Aug	
Sep	
Oct	
Nov	
Dec	

Example:

```
2 Jan 1435.23 EST Mon 1967 14:35:13.008291
```

For most users and for most purposes, date and time are sufficiently pinpointed by the call

```
call get_calendar$brief(clock_time, brief_string);
```

with the declaration

```
dcl clock_time fixed bin(17);
    brief_string char (24);
```

The resultant string has the form

mm/dd/yy xxxx.x zzz www

The time components begin in the character positions indicated and are of the length specified.

component	character position	field size	representation
date	1	8	mm/dd/yy
military time	11	6	xxxx.x
time zone	18	3	zzz
day of week	22	3	www

The time specified in the example above would appear as

01/02/67 1435.2 EST Mon

Usage: put calendar

The task of converting a string of characters which supposedly represents some calendar time into an internal calendar time is performed by the procedure put_calendar for certain formats. With any format specified there are three basic restrictions:

1. The order in which components appear is fixed.

2. A component may be null (missing) only if all following components are null.

3. Component delimiters consist of <SP>, | or groups of these.
<delim>::=<SP>|/|<delim><SP>|<delim>/

The input string is interpreted according to the setting of the option, calendar_format. If the option is on, its specification is the format and may contain a sequence from the following character set:

character	component	component consists of
d	day of month	1 or 2 digits
m	month	1 or 2 digits, or 3 alph.chars.
y	year	1, 2 or 4 digits
x	military time	4 to 12 digits
h	hour	max. 6 digits
n	minute	max. 6 digits
s	seconds	max. 6 digits
z	time zone	3 alphabetic chars.
u	microsecond	max. 6 digits
w	day of week	anything but delimiters. (field is ignored)

If the option calendar_format is off, the format "mdyxz" is assumed. This would accept, for example

```
Jan 2 1967 2432.5 EST
01/02/67 2432.5
1 2 67 2432.50/EST
```

and any string produced by get_calendar$brief.

If time zone is omitted, then the zone used is obtained from the time_conversion_table. Time of day is expressed either as military time (x) or as hour, minutes, seconds and microseconds. The conflicts presented by a format such as "h n x s u" are resolved (or ignored) by using the value of the last component (when scanned left to right) which specifies a specific portion of the time of day where more than one such component is specified. In this example, the integral part of military time ("x") is used (overriding "h" and "n"); the fractional part of "x" is overriden by "s" and "u".

Default values for missing or unspecified components are as follow:

```
y    "this year"
m    "this month"
d    "today"
h    0
n    0
s    0
u    0
z    "as determined by time_conversion_table"
```

Example:

The calendar_format option specification is "h n s d y". All times input using this format will be in an integral number of seconds (u = 0), in the current month of the year specified. The time zone will be determined by the time_conversion_table. If the year is not specified, the current year is used. If day and year are not specified, the current day and year are used.

Input conversion is performed by the statement

```
call put_calendar (input_string, calendar_time);
```

with the declarations

```
dcl input_string char (*),
          calendar_time fixed bin(71);
```

Implementation

A. get_calendar

The task of building the return string is straightforward. Get_calendar calls calendar_output and gets back a binary representation of the time. The functions bin_ascii which converts a binary integer into an ascii string of decimal digits is used to convert components of the time to a more readable form. The components are placed in the string provided by the user and get_calendar returns.

B. put_calendar

The option, calendar_format, is read. If "on" the format in the specification is used. If "off", the standard format is used. The current year, month and day are determined by a call to calendar_output. Default values are set with zone = " <SP> <SP> <SP>". The input string is searched. Each element is converted according to its corresponding specification in the format and then stored into its specific component or components. The search is terminated when either the format or the input string is exhausted. These components are then passed in a call to calendar_input which converts the external component representation to an internal clock time. On the return of calendar_input, put_calendar returns.

Some checks are made on the input components. Month must be either an integer between 1 and 12 or the first three characters of the name of a month. Year must be positive but no such restriction is placed on day, hour, minute, second and microsecond. In case of unacceptable data, put_calendar calls seterr (BY.11.01) to record the error, then signals condition (put_calendar_err).

C. Utility functions

The following external functions are required by get_calendar and put_calendar.

bin_ascii

ascii_bin

The function of converting an integer of maximum precision 35 to an ascii string of decimal digits is provided by "bin_ascii" for use in procedure get_calendar. The resultant character string is the decimal value (signed, if negative) right adjusted and <SP> filled. Its usage is as follows:

```
dcl bin_ascii entry ext char (12),
    i fixed bin;
string = bin_ascii (i);
```

An ascii string of digits is converted to an integer by the function ascii_bin which is described as

```
dcl ascii_bin entry ext fixed bin (35),
    c char (*);
value = ascii_bin (c);
```

This function is used by the procedure put_calendar.