

TO: MSPM Distribution  
FROM: K. J. Martin  
SUBJECT: BY.15.01  
DATE: 11/17/67

The description of `unique_chars` has been updated to more fully explain its use. The user may pass `unique_chars` a bit string of 70 or fewer bits. `unique_chars` converts the string to a 70-bit string padded with zeros on the left. This is useful for the user who wishes to convert a process id (36 bits) to characters. He simply calls `unique_chars` with a 36-bit argument.

Published: 11/17/67  
(Supersedes: BY.15.01, 07/07/67,  
BY.15.01, 03/31/67)

## Identification

Generating Unique Identifiers  
unique\_bits, unique\_chars, when\_created  
L. B. Ratcliff

## Purpose

The procedure unique\_bits provides the user with a source of identifiers (bit-string or character-string) guaranteed to differ from all other identifiers generated by these procedures. The procedure unique\_chars provides a character representation of a unique bit string. The procedure, when\_created, is useful to the user who wants to determine the time at which a specific identifier (character string) was created.

## Discussion

A unique 70-bit string is assured by concatenating the 18-bit processor serial number with the low-order 52 bits of the Multics calendar clock obtained by executing the rccl (read calendar clock) instruction. The resulting identifier will remain unique for 140 years. Uniqueness is guaranteed with a single system if two processors cannot access one clock simultaneously, and between systems because the processor serial number is unique. The planned implementation of the calendar clock requires that two processors not access the clock simultaneously. A change in implementation permitting simultaneous access would require that the procedure unique\_bits be inhibited between accessing the clock and obtaining the processor serial number.

## Usage

A unique bit string is obtained by executing the statement:

```
bit_string = unique_bits;
```

with the declaration:

```
dcl bit_string bit (70), unique_bits ext entry bit (70);
```

A unique character string is obtained by executing the statement:

```
char_string = unique_chars(bits);
```

with the declaration:

```
dcl char_string char (15), unique_chars ext entry
char (15), bits bit(N);
```

where  $1 \leq N \leq 70$ . If  $N < 70$ , `unique_chars` pads `bits` with zeros on the left to produce a 70-bit string. If `bits` equals zero, `unique_chars` calls `unique_bits` to obtain a unique bit string. Note that if `bits` is supplied (non-zero) and is not a unique bit string, the character string returned by `unique_chars` cannot be guaranteed to be unique.

The first character in the character string produced is always `!` (exclamation point) to identify the string as a unique identifier. The remaining 14, forming the unique identifier, are alphanumeric.

To obtain the processor serial number and the time of creation of a unique character string, the user executes the statement:

```
call when_created (char_string, time, processor);
```

where `char_string` is described above, `time` is a 71-bit fixed binary integer and `processor` is an 18-bit string. If `when_created` ascertains (by checking to see if `char_string` is of the form and composition described below) that the character string was not created by `unique_char`, it signals an error. Otherwise, it returns with `time` containing a calendar clock time and `processor` containing the 18-bit serial number of the processor used to create the identifier.

### Implementation Notes

Procedure `unique_bits` calls the PL/I built-in abnormal function `"clock_π"` (see BP.0.03) to obtain the current clock time, and obtains the processor serial number from `pds$processor_number` (the processor data segment, `pds`, is described in BK.1.02). `Unique_bits` returns the 70-bit string, `bit_string`, containing

```
processor_number||clock time
```

Procedure `unique_chars` checks the argument `bits`. If `bits` is less than 70 bits long, `unique_chars` places `bits` into the right most N bits of a 70-bit variable. If `bits` is `"0"`, `unique_chars` calls `unique_bits`. `Unique_chars` then creates the corresponding character string.

The character string has the form

$$\{C_1 C_2 \dots C_{14}\}$$

Each character,  $C_i$  ( $i = 1, \dots, 14$ ) is determined by the value ( $j$ ) of the  $i$ -th 5-bit byte of the 70-bit string. The character  $C_i$  is the  $j$ -th entry in a table of 32 characters which are

upper case alphabet except vowels, R, S, T, V and Y

lower case alphabet except vowels, r, s, t, v and y

in the order listed Vowels are eliminated to avoid profanity; V, v, Y, and y are eliminated because they suggest U, u, I, and i. The other six consonants are eliminated because of their frequent occurrence in normal identifiers. Procedure `when_created` is related to `unique_chars` only in that it is aware of the algorithm used in creating the character string. It reverses the algorithm to obtain the original 70-bit string, then extracts the 52-bit clock time and processor serial number.

The three procedures are slave with slave access.