

SUBJECT: Service Routines for Debugging  
Multics - Case 27131-41

DATE: March 2, 1967

FROM: L. D. Whitehead

WORKING NOTES

Introduction

Several service routines which are of great use in debugging parts of Multics exist on a special library. No documentation of these routines exists at present. This memorandum was produced to provide a bare minimum of information on how to use the routines. Additional notes will be produced as the new routines become available.

Use of Special Library

To use the special library, one needs to effect the steps below. This will cause the special library (which also contains the regular library) to be used automatically.

1. Unlink MRGEDT (TB08)
2. Link MRGEDT (TB08) T234 CMFLO3

Include in your GECOS file a new MRGEDT command, LIMITS,  
e.g., LIMITS 048000 (for 635 or 645)

LIMITS 071000 (for 645 only)

LIMITS 105000 (for 645 only)

Use of Library Routines

These routines are in the special library. The author has used most of them and they work as stated.

1. call `messag (text);`

This routine writes a message in your error file.

- text - can be
1. name of a character string
  2. character string literal
  3. a function call which returns a character string
  4. the concatenation of any of the above (1-3)

NOTE: See notes at the end of this section for examples and comments on the use of `messag`.

2. call `makeseg (name,delta,ptr,error_return,error_code);`

This routine creates segments at run time. It uses the system routines "newseg" and "grow". The created segment is set to zeros.

name - (character string literal or name of a string)  
segment name

delta - (fixed bin (17)) size of segment in words

ptr - (pointer) receives pointer to the created segment

error\_return - (label) error return as from "newseg" and "grow"

error\_code - (fixed bin (17)) error codes same as for "newseg" and "grow"

See MSPM.8.03 for "newseg" and "grow".

3. call `dump_(ptr,ct);`

This routine produces an octal dump in your error file. It starts at ptr and dumps ct words.

ptr - (pointer) pointer to aligned data

ct - (fixed bin (17)) word count

4. `char = bconv(integer,0);`

A function which converts integers to octal characters. Returns char (6).

`integer` - fixed bin (17)

0 - the decimal literal zero. (The routine has other uses.)

5. `char = bin_oct(bits);`

A function which converts a bit string into octal characters. It returns char (12).

`bits` - (bit (36)) a bit string

6. `char = bin_dec(integer);`

A function which converts an integer to a character string. It returns char (12).

`integer` - fixed bin (17) or (35)

7. `char = conv(bits);`

A function which converts 3 bits to one octal character. It returns char (1).

`bits` - bit (3)

NOTE: BCONV and CONV generate and use the following entries in `stat_`. This will not hurt you unless you use the same names in `stat_`.

`stat_$flag`

`stat_$bflag`

the arrays B and C

8. Two functions for dumping the current stack frame:

`stack_frame$ptr` - returns a pointer (pointer) to the current stack frame

`stack_frame$size` - returns the size of the current stack frame (fixed bin (17))

To dump the current stack frame, use the following statement.

```
call dump_((stack_frame$ptr),(stack_frame$size));
```

Comments on Use of MESSAG

1. Example:

```
dcl t1 fixed bin (17),
    t2 bit (36),
bin_dec entry char (12),
bin_oct entry char (12);
t1 = 76;
t2 = "00---1111"b;
call messag("t1 = " || bin_dec(t1) ||
            " t2 = " || bin_oct(t2));
```

2. EPL will take a maximum of 36 characters in a literal as used here. Several strings can be concatenated to get around this. Ex., "NOW IS " || " THE TIME".
3. There is a maximum on the number of things that can be concatenated in a call. This maximum is currently 13. Several calls to messag will get around this; however, each call creates a new line.
4. Messag as of February 27, 1967, will not accept varying strings, but it is to be modified shortly to permit them.

Some Routines Not on the Library

Two routines of general interest have been written by the author. They will be put on the library at an early date. At present the links needed are:

OCT	TEXT	T269	8046
OCT	LINK	T269	8046
DEC	TEXT	T269	8046
DEC	LINK	T269	8046

1. char = oct(bits,ct);

This function converts a bit string to octal and return varying char (12).

bits (bit (36)) - input bit string

ct (fixed bin (17)) - count of the number of characters to be returned. The input is converted to 12 characters and truncated on the left

2. char = dec(integer,ct);

This function converts an integer to decimal and returns varying character (12).

integer is fixed bin (17) or (35)

ct (fixed bin (17)) - count of the number of characters to be returned. The input is converted to 12 characters and truncated on the left.

These routines are particularly useful when dumping data bases into your error file for return to a 1050. Uninteresting leading zeros (for octal) or blanks (for decimal) can be deleted thereby reducing CTSS record size, error file size, and 1050 typing time.

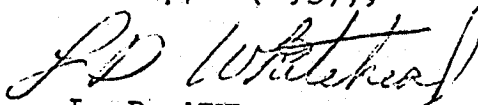
Examples:

1. Until "messag" is modified to take varying character strings, converted data must be placed into fixed length character strings before "messag" is called.

e.g., dcl dec entry varying (12),  
cl character (3),  
i = 134;  
cl = dec(i,3);  
call messag("i = "||cl);

2. When "messag" takes varying strings the same output can be effected by

i = 134;  
call messag("i = "||dec(i,3));

  
L. D. WHITEHEAD

WH-4235-LDW-AMG

Copy to  
See next page

Copy to  
Messrs. T. H. Crowley - MH  
P. G. Neumann - MH  
J. C. Noll - HO  
J. F. Ossanna - MH  
R. K. Rathbun - WH  
W. C. Ridgway, III - WH  
J. D. Van Hausen - WH  
V. A. Vyssotsky - MH  
Miss S. M. Wolf - WH  
Multics Distribution Office - MH