Published: 06/18/68

Identification

BCPL-MULTICS library routines
R. H. Canaday

Purpose

This document describes new library routines, or new versions of old routines, which have been written to support BCPL programs and the BCPL compiler on MULTICS. Note that these routines will not execute properly on 6.36.

New versions of BCPL I/O (reference (1)) and six new routines are contained in segment <BCLIB1>, filed in `BCLIB1 (EPL, TEXT, LINK)' in T269 CMFL03. These routines can be called from BCPL through the global vector. They cannot be called from EPL. They are initialized by the EPL statement:

```
call BCLIB1;
```

which also initializes segment <BCPLGL>, which is filed in T269 CMFL03.

Segment <BCPLGL> contains runtime support routines, including a routine `BCPLGL$SETAP', callable from EPL, which is used in BCPL stack management as described in MSPM BZ.6.04.

The library routines are accessible through the declaration:

```
global $( Save:   4;     Restore:  5
          Open:   6;     Close:    7
          ITS:    8;     B to M string:  9
          Getadr:   13;  Call:     14
          Readch:   15;  Writech:  16     $)
```

I/Ø Routines

The routines `InitializeIØ', `Readch', and `Writech' are as described in reference (1). Routines `Open' and `Close' replace the four routines:

`Createoutput', `Findinput', `Endread', and `Endwrite'.

The call `Close(F)' is equivalent to `Endread(F)' or `Endwrite(F)'.

The call

```
F := Open(Ioname, Code)
```

is used to initialize the I/O routines for input or output on stream `Ioname'.

The arguments are:

Ioname - a string which is an ioname which has previously been attached. Note that `Open' does not issue an attach call. Standard ionames are:

"user-input" = typewriter keyboard
"user-output" = typewriter printer

Code - = 1 for an input file (to be read)
= 2 for an output file (to be written)

Each file which is open will require 300 words from the I-O buffer area, provided by InitializeIØ.

The return value, `F', is the file descriptor word to be used in subsequent references to this file by Readch, Writech, and Close.

New Routines

The six new routines are `Getadr', `Call', `Slave', `Restore', `ITS', and `BtoMstring'. Each of these is described below.

Function BtoMstring

This function takes a BCPL string and creates dope and specifier for it, so that the string will correspond to the EPL declaration `char(*)'. The call is

```
let  v = vec 6

     x := BtoMstring(string, v)
```

Dope and specifier are created and stored in v. The specifier starts at x[0]. The value of x is v or v + 1, (on return it will be even).

Routines `Save' and `Restore'

These routines save and restore the global vector in a pushdown stack. They are useful primarily for avoiding global vector usage conflicts when executing independent BCPL programs in the same process. Their usage, together with the EPL-callable routine `Setap', is described more fully in MSPM BZ.6.04.

Function `Getadr'

BCPL call:

```
A := Getadr(x,y)
```

Purpose: to return the (BCPL) address of x$y. The address is created by the MULTICS routine generate-ptr. (cf. MSPM BY.13.02)

Arguments: x is a BCPL string which is the segment name.
y is a BCPL string which is a symbol in the segment, or entry name, or null (cf. generate-ptr.)

Routine `Call'

BCPL call:

```
Call(Procedure,A1,A2,...,An)
```

Purpose: To call an EPL procedure from BCPL

Arguments:

Procedure - the address of the EPL procedure or entry point. This address will normally be generated by `Getadr' (which see).

Aj - the address of the j^{th} argument.

Example:

The routine Writeout (MSPM BY.4.02) is called from EPL as

```
call Writeout("Some Message", 0)
```

from BCPL it would be called as:

```
let x = 0
let y = vec 6

Call(Getadr("Writeout", "Writeout"), BtoMstring("Some
     Message", y), lv x)
```

Alternate BCPL code to Call Writeout is:

```
let x, w = 0, "Writeout"
let y = vec 6
let a = Getadr(w,w)
y := BtoMstring("Some Message", y)
Call (a, y, lv x)
```

Reference

1. M. Richards, The BCPL Reference Manual, Multics Repository Document M0099, 15 May 1968.