TO: MSPM Distribution

FROM: Charles Garman

DATE: November 25, 1966

SUBJ: Symbolic Reference to Single Character Literals in EPL-PL/I

The two enclosed MSPM sections (BY.8.01 - 8.02) describe a reference mechanism for non-graphic or unavailable ASCII characters, and conform to the latest editions of BC.2.01-2.04.

NOTE: These segments are available on the Multics Segment Library for use in 6.36 and 64.5 simulations; to use include the following line in the GECOS file for the MRGEDT command:

```
LIBE        X       (DATA, SLVACC)
```

where X is CTL.CHAR, UPPER.CASE.CHAR, OR PUNCTUATION.CHAR, as described within.

Published: 11/25/66
(Supersedes: BY.8.01, 08/18/66)

Identification

Symbolic reference to non-graphic character constants
ctl_char
Charles Garman

Purpose

This section describes a library data segment, ctl_char, which provides the user with facilities for symbolic reference to single non-graphic characters of the ASCII data-character-set.

Background

In EPL or PL/I programs which work with non-graphic characters, (such as form-feed), the visual appearance of the programs suffers if these characters are embedded within character-string constants. For example, the form-feed character (ASCII 014), if embedded in a literal, would present certain confusing aspects to a person reading a program, either from the blank lines on the paper or its appearance in its escape representation. A more extreme case is that the backspace character is barred from a single-character literal by the particular definition of canonical-form.

Note that the space character is representable as " ", and thus is not included in this category.

Usage

For each character of this type which a program needed, the following declaration would appear:

```
dcl ctl_char$character_name char(1)ext;
```

where character_name is the lower-case counterpart of the ASCII or Multics name of the character, as defined in the following table (see also section BC.2.01); a reference might then be in a statement such as this:

```
message = ctl_char$rrs ||"type"|| ctl_char$brs;
```

ASCII name	Multics Name	Octal value
NUL		000
SOH		001
STX		002

ASCII name	Multics Name	Octal value
ETX		003
EOT		004
ENQ		005
ACK		006
BEL	BEL	007
BS	BS	010
HT	HT	011
LF	NL	012
VT	VT	013
FF	FF	014
CR		015
SO	RRS	016
SI	BRS	017
DLE		020
DC1		021
DC2	HLF	022
DC3		023
DC4	HLR	024
NAK		025
SYN		026
ETB		027
CAN		030
EM		031
SUB		032
ESC	MC	033
FS		034
GS		035
RS		036
US		037
DEL		177

Implementation

This segment is coded in EPLBSA; the following text shows the coding for a sample entry.

```
                    name ctl_char
                    ...
                    segdef nl
                    ...
nl:                 vfd o9/012      "New Line
                    ...
                    end
```