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.
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-strings. 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;

<table>
<thead>
<tr>
<th>ASCII name</th>
<th>Multics Name</th>
<th>Octal value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL</td>
<td></td>
<td>000</td>
</tr>
<tr>
<td>SOH</td>
<td></td>
<td>001</td>
</tr>
<tr>
<td>STX</td>
<td></td>
<td>002</td>
</tr>
<tr>
<td>ASCII name</td>
<td>Multics Name</td>
<td>Octal value</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ETX</td>
<td>BEL</td>
<td>003</td>
</tr>
<tr>
<td>EOT</td>
<td>BS</td>
<td>004</td>
</tr>
<tr>
<td>ENQ</td>
<td>HT</td>
<td>005</td>
</tr>
<tr>
<td>ACK</td>
<td>LF</td>
<td>006</td>
</tr>
<tr>
<td>BEL</td>
<td>VT</td>
<td>007</td>
</tr>
<tr>
<td>BS</td>
<td>FF</td>
<td>010</td>
</tr>
<tr>
<td>HT</td>
<td>CR</td>
<td>011</td>
</tr>
<tr>
<td>LF</td>
<td>VT</td>
<td>012</td>
</tr>
<tr>
<td>VT</td>
<td>FF</td>
<td>013</td>
</tr>
<tr>
<td>FF</td>
<td>SO</td>
<td>014</td>
</tr>
<tr>
<td>CR</td>
<td>SI</td>
<td>015</td>
</tr>
<tr>
<td>SO</td>
<td>DLE</td>
<td>016</td>
</tr>
<tr>
<td>SI</td>
<td>BS*</td>
<td>017</td>
</tr>
<tr>
<td>BRS</td>
<td>018</td>
<td></td>
</tr>
<tr>
<td>019</td>
<td>020</td>
<td></td>
</tr>
<tr>
<td>021</td>
<td>022</td>
<td></td>
</tr>
<tr>
<td>023</td>
<td>024</td>
<td></td>
</tr>
<tr>
<td>025</td>
<td>026</td>
<td></td>
</tr>
<tr>
<td>027</td>
<td>028</td>
<td></td>
</tr>
<tr>
<td>029</td>
<td>030</td>
<td></td>
</tr>
<tr>
<td>031</td>
<td>032</td>
<td></td>
</tr>
<tr>
<td>033</td>
<td>034</td>
<td></td>
</tr>
<tr>
<td>035</td>
<td>036</td>
<td></td>
</tr>
<tr>
<td>037</td>
<td>177</td>
<td></td>
</tr>
</tbody>
</table>

Implementation

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

```plaintext
name ctl_char
...
segdef n1
...
n1: vfd 09/012 "New Line
...
end
```