To: Distribution

From: Paul Green

Date: 01/04/77

Subject: Influence of APL on the Multics Character Set

With the release of Version 2 APL (now scheduled for MR 6.0), we expect an increase in the usage of APL at many Multics sites. This memo discusses some of the implications that this increased usage will have on the rest of the system.

The APL Character Set

APL does not use the standard ASCII (or even EBCDIC) character set. It has its own set of graphics, which include italicized alphabetic letters, Greek letters, and mathematical symbols. There is no standard mapping of the APL symbols onto any existing code. Each APL interpreter has chosen an internal representation that best suits its own interests. Multics APL is no exception; we have mapped the APL symbols into the ASCII character set in those instances where the graphics were the same, and we have extended the ASCII character set from 128 codes to 196 codes to accomodate those graphics that had no ASCII equivalent.

Implications for Standard Software

Because the Multics APL user will be able to create text segments with characters outside of the normal ASCII range, it will no longer be possible for any standard Multics software to assume that only 7 bits of a 9 bit byte are used. Any command or subroutine that make such an assumption will fail to work when processing segments created by APL. If we are to offer a useful APL product, we must ensure that the whole system supports segments containing APL characters. Generally, only APL will "understand" the meaning of the characters whose codes are greater than 177. Programs that are not a part of the APL interpreter should not become confused by these codes, however.

Impact on Coding Styles

There are several commonly-used PL/I constructs that will not work with non-ASCII data, unless care is exercised by the programmer. By listing them here, and publishing this list far ahead of the official release of APL, I hope that we can correct existing code before the customers discover any problems.

The next release of the Multics PL/I compiler (Release 22, now running in >exl>o) will contain new implementations of the search, translate, and verify builtin functions that will work properly for arguments that contain non-ASCII characters. It will also contain two new builtins, collate9 and high9, that will return the 9-bit collating sequence, and the highest character in that sequence. The existing collate and high builtins will remain unchanged. Therefore, the easiest way to correct programs that use search, translate, or verify is to recompile them with Release 22 of PL/I.

List of Troublesome Constructs

CONSTRUCT:	Indexed goto, or indexed array reference, based on the ASCII value of a character.
PROBLEM:	Can't assume biggest character has a value of 127.
SOLUTION:	Either assume a limit of 511, or special-case characters whose value is over 127.

CONSTRUCT:	translate, search, or verify builtins.
PROBLEM:	Prior to Release 22 of PL/I, these builtins assume that their input arguments contain only ASCII characters. If given non-ASCII characters, they perform erratically.
SOLUTION:	Recompile with Release 22 of PL/I.

Known Problems

The following programs are known to fail when processing non-ASCII data:

PROGRAM	STATUS
abs_io_	being fixed by PG
qedx	fixed, being submitted for MR5.0
any_to_any_	fixed in EXL
edm	not fixed
teco	not fixed
PL/I I/O	not fixed

Unresolved Issues

Should the system be changed to permit non-ASCII characters to appear in entrynames? Presently, the system does not permit names with non-ASCII characters, and the salvager will delete any such names that it finds.