There are two solutions: (1) Go back into
the input files and change all conflicting font
designators—this can get very messy; or (2) use
a version of \TeX in which no fonts have been
preloaded; such a version, commonly known as
"\textsc{VirginTex}," will start up much more slowly
than a preloaded version, owing to the greater number of
font metric files that must be loaded at run time.
The following convention has been adopted at many
installations: Preloaded fonts use no capital letters.
Thus you are always safe if you introduce a new font
called A, B, \ldots, Z. (Actually, the AMS requires an
extended set of fonts, including a full complement
of cyrillic fonts in 6 sizes; these are called A, \ldots, F,
but G through Z remain open for special use.)

Barbara Beeton

* * * * * * * * * * *

MACRO COLUMN

Send Submissions to:
Lynne A. Price
TUG Macro Coordinator
Calma R&D
212 Gibraltar Dr.
Sunnyvale, CA 94086

The macro column is a new regular feature of
TUGboat. It is a forum where \TeX users can ex-
change formatting problems (with or without solu-
tic is), questions about writing macros, comments
on macros published in earlier issues of TUGboat,
etc.

* * *

Discussion of macros at the \TeX Implementors’
Workshop in May included some simple suggestions
for increasing portability of macros across \TeX sites.
First, the excellent suggestion was made that ASCII
sites attempt to standardize the characters chosen
to replace SAIL delimiters. The \textsc{AMS-\TeX} conven-
tions are recommended: ampersand \& for the tab
character, underscore \_ for the subscript indicator,
and caret \^ for the superscript delimiter. Second,
macro packages typically include several font decla-
rations. Incompatible assignment of font codes
makes it difficult for users to select an assortment
of macros from different packages. If font codes as-
signed in a macro file do not correspond to the fonts
preloaded by some versions of \TeX, strange results
can be difficult to explain. There is no total solution
to this problem, but it can be minimized. Macro
packages should come with documentation describ-
ing the fonts and font codes used. When sending files
to another installation, users should remember that
preloaded fonts differ from site to site. A helpful
convention in assigning font codes is to reserve up-
percase letters for user declarations and to let stan-
dard macro packages use other characters. Patrick
Milligan's DefineFont macro described below can be
used to automatically assign available font codes.

* * * * * * * * * * *

Macros on Microfiche

Editor's note: In an effort to hold down expenses,
some of the more extensive macro packages in future
issues of TUGboat will be published on microfiche, with
a summary or introduction to each package included in
this column. Authors of macro packages who submit
their work for publication here are requested to supply
such an introduction along with the camera copy of the
package. Because microfiche is not as easy to use as paper,
an attempt will be made to arrange for the collection
and distribution of these macro packages in machine-
readable form (probably on magnetic tape); details will
be published as soon as they are known. Microfiche will
conform to the following specifications: negative image
(white characters on black), 105mm x 148mm, 24-to-1
reduction ratio, containing 98 frames per microfiche.

* * * * * * * * * * *

ERRATUM:
NOFILL PROGRAM

Patrick Milligan
BNR Inc.

There was one subtle error in the program listing
of both the SAIL and Pascal versions of the NOFILL
program that appeared in TUGboat Vol 2, No. 1. In
both programs, the definitions of macros \backslash and \textbackslash
were reversed (see pages 90 and 96). As printed, the
definitions are correct, but the program source was
incorrect. Since the program source was run through
NOFILL for publication, the incorrect definitions
became correct, but all other uses of \textbackslash (acute accent)
and \backslash (grave accent) were incorrect.

Also, there was some confusion about the table
of contents entry on page 136 entitled NOFILL
Program with Pascal Source. When the two pro-
grams were submitted to TUGboat, it was not clear
if the SAIL version would be printed, or the Pascal
version, or both. The introduction to the SAIL ver-
sion was appropriate to both versions, but no intro-
duction was prepared for the Pascal code.
READFONINFO

This is an integer function that has the following parameters:

<table>
<thead>
<tr>
<th>Var.</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var</td>
<td>FYL</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Var</td>
<td>FONTINFO</td>
<td>FNTINFOARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>FMEM</td>
<td>FMEMARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>WDBASE</td>
<td>FBASEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>HTBASE</td>
<td>FBASEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>DPBASE</td>
<td>FBASEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>ICBASE</td>
<td>FBASEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>LGBASE</td>
<td>FBASEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>KRBASE</td>
<td>FBASEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>EEXTBASE</td>
<td>FBASEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>PARBASE</td>
<td>FBASEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>FCKSUM</td>
<td>FBASEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>FFPB</td>
<td>FBASEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>FSIZE</td>
<td>FSIZEARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>FFP1</td>
<td>FPIARRAY</td>
</tr>
<tr>
<td>Var</td>
<td>FMEMPTR</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Var</td>
<td>PSIZE</td>
<td>REAL</td>
</tr>
<tr>
<td>Var</td>
<td>ATCLAUSE</td>
<td>BOOLEAN</td>
</tr>
</tbody>
</table>

Reads font information from file FONTFIL. The integer FYL is used as an index in the various array parameters to establish the destination of this information.

RELEASE

Procedure with one parameter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FYL</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>

The integer FYL must be in the range [1..6]. It selects one of ICHAN1 through ICHAN6 and executes RESET(ICHANx) followed by FILPTR:=FILPTR-1.

This closes and releases the indicated file and frees the entry in FILENAME.

RSETFILE

Procedure with four parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>FNAME</td>
<td>CHAR8</td>
</tr>
<tr>
<td>FDIRECTORY</td>
<td>INTEGER</td>
</tr>
<tr>
<td>FDEVICE</td>
<td>CHAR8</td>
</tr>
</tbody>
</table>

The integer ID must be in the range [1..6]. It selects one of ICHAN1 through ICHAN6 to be opened for input and associates it with FNAME, FDIRECTORY, and FDEVICE.
Examples:

\DefineFont{cmtt}{\tt} % typewriter font
\DefineFont{mflogo}{\mflgo} % METAfont logo font

\def\DefineFont#1#2{
  \if !\UserFonts!
    \xdef#2{}\send9{Error: No font codes available for font #1})
  \else{
    \Apply {\First} to {\UserFonts!} -> {\Fontcode} % Get font code
    \font\Fontcode=#1 % Load font
    \xdef#2{\current\Fontcode} % Define macro to use font
    \Apply {\Rest} to {\UserFonts!} -> {\UserFonts} % Remove code from list
  \}
}

% The \Apply macro is used to apply a macro to its argument, when the
% argument is a macro also. The trick is to fool \TeX into expanding
% the argument before the macro is applied. If a better way exists to
% perform this feat, please send your solution to TUGBoat.
% Usage:
% \Apply {<function>} to {<argument>} -> {<result>}
% where:
% <function> is the macro to apply
% <argument> is the macro containing the argument to <function>
% <result> is the macro used to save the result

\def\Apply #1 to #2 -> #3{
  \let\Func=\let % Setup dummy function
  \xdef\Eval{\xdef#3{\Func #2}} % Expand argument
  \let\Func=#1 % Redefine function to use macro
  \Eval % Apply macro to its argument
}

% The \First and \Rest macros are used to manipulate strings terminated with
% an exclamation mark (!).
\def\First#1#2!{#1} % Returns first character of string
\def\Rest#1#2!{#2} % Returns string with first character removed

% The macro \UserFonts describes the set of font codes available to
% \DefineFont. The list of font codes should not contain an exclamation
% mark (!) since this is used to terminate strings passed to the \First
% and \Rest macros (and it isn't a valid font code anyway). A reasonable
% convention for font codes is to have all upper case letters available
% for user fonts:
\UserFonts{ABCDEFGHIJKLMNOPQRSTUVWXYZ}

% If \DefineFont is used to allocate all fonts used (including those in
% BASIC), then all 64 possible font codes should declared.