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 "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, ..., Z. (Actually, the AMS requires an extended set of fonts, including a full complement of cyrillic fonts in 6 sizes; these are called A, ..., 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 exchange formatting problems (with or without solutions), 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 AMS-TEX conventions 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 declarations. Incompatible assignment of font codes makes it difficult for users to select an assortment of macros from different packages. If font codes assigned 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 describing 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 uppercase letters for user declarations and to let standard 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 fiche 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. Fiche will conform to the following specifications: negative image (white characters on black), 105mm × 148mm, 24-to-1 reduction ratio, containing 98 frames per fiche.

* * * * * * * * * * *

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 \´ and \` 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 ´ (acute accent) and ` (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 programs 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 version was appropriate to both versions, but no introduction was prepared for the Pascal code.

READFONTINFO

This is an integer function that has the following parameters:

Var.?	*Name*	*Type*
	FYL	INTEGER
Var	FONTINFO	FNTINFOARRAY
Var	FMEM	FMEMARRAY
Var	WDBASE	FBASEARRAY
Var	HTBASE	FBASEARRAY
Var	DPBASE	FBASEARRAY
Var	ICBASE	FBASEARRAY
Var	LGBASE	FBASEARRAY
Var	KRBASE	FBASEARRAY
Var	EXTBASE	FBASEARRAY
Var	PARBASE	FBASEARRAY
Var	FCKSUM	FBASEARRAY
Var	FPFB	FBASEARRAY
Var	FSIZE	FSIZEARRAY
Var	FPFI	FPIARRAY
Var	FMEMPTR	INTEGER
Var	PSIZE	REAL
Var	ATCLAUSE	BOOLEAN

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

RELEASE

Procedure with one parameter.

Name	*Type*
FYL	INTEGER

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.

Name	*Type*
ID	INTEGER
FNAME	CHAR9
FDIRECTORY	INTEGER
FDEVICE	CHAR6

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}{\mflogo}    % 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{\curfont \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:

\def\UserFonts{ABCDEFGHIJKLMNOPQRSTUVWXYZ}

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