The `amsfonts` package

David M. Jones*

Version 3.01, 2013/01/14

1 Introduction

The `amsfonts` package provides access to a limited selection of features from the AMSFonts package. For other features, see the following packages: `amssymb`, `eucal`, `eufrak`, and `euscript`. (For wholesale substitution of Euler math, see Frank Jensen's `euler` package.)

This file also contains the external font definitions (`.fd` files) needed to use the AMS font collection with LaTeX 2$_\varepsilon$.

1.1 The `psamsfonts` option

In early versions of AMSFonts, the 6, 8 and 9 point sizes of many fonts were implemented only in METAFONT and were not available as separate Type 1 fonts. Because of this, many AMS-LaTeX classes and packages were given a `psamsfonts` option that would adjust the font definitions so that only the TFM files corresponding to design sizes available as Type 1 fonts would be used. This relieved DVI-to-PostScript processors of the responsibility of doing an appropriate font substitution, which was of particular value in those early days when support for Type 1 fonts was still experimental and highly variable.

However, the `psamsfonts` option had a second, undocumented, effect: It changed the scaling behavior of the Euler and AMS Symbol fonts. Without the `psamsfonts` option, the fonts were made available at the same set of "quantized" sizes used by default for the standard LaTeX fonts. For example, if you asked for `msbm` at 13 pt, you would instead get `msbm` at 12 pt, with a warning about the substitution. With the `psamsfonts` option, the AMS fonts were continuously scalable, so if you asked for `msbm` at 13 pt, you would get a 13 pt version of the font. This means that the `psamsfonts` could change TeX's formatting decisions, including which linebreaks were chosen.

As of version 3 of the AMSFonts distribution, there is no longer a discrepancy between the METAFONT implementations and the Type 1 implementations, which means that we can use a single set of font definitions for both. The question then becomes which approach we should adopt. Should we provide the

*Previous versions of this package were written by Frank Mittelbach, Rainer Schöpf and Michael Downes. Most of the code remains unchanged and the credit should continue accruing to them. However, blame for the commentary and decisions regarding the `psamfonts` option and the handling of font scaling belong to the current author alone.

fonts only in the standard LaTeX quantized sizes or should we allow for full and continuous scaling?

It is our position that the quantized font sizes are not desirable for their own merits; they are a legacy of the era when METAFONT was virtually the only source of fonts for use with TeX and when generating a new size of a font was onerous and resulted in a potentially long-term use of precious disk space. In the modern TeX environment, the cost of allowing arbitrary scaling of fonts is so low that most users will never even be aware of them. Furthermore, any user who sticks to the standard LaTeX document classes and the font sizes they provide will never stray outside of the officially sanctioned sizes anyway, even when using the AMSFont package. Any users who do request nonstandard sizes presumably know what they are doing and should get the font size they ask for. Although they will have to take special measures to achieve that for the standard LaTeX fonts, we can make their tasks easier by providing the AMS fonts in a fully scalable form from the beginning.

1.2 The cmex fonts

The situation with `cmex` is even more stark. In standard LaTeX, `cmex` is only supplied at 10 pt. That is, no matter what size you ask for, you get `cmex` at 10 pt. The `amsfonts` package redefines the `cmex` font family so that it is also continuously scalable. This means that *merely loading the* `amsfonts` *package can change line breaks.*

2 Customization

You should *not* change any of the files generated from this module. If you really want to change the font shape groups preloaded you should copy the relevant portions to another file (having a different name) and edit the other file according to your needs.

2.1 The docstrip modules

The following modules are used in an installation procedure to direct `docstrip` in generating external files:

`sty`	generate `amsfonts.sty`
`eur`	generate `fd` for Euler Roman
`eus`	generate `fd` for Euler Script
`euf`	generate `fd` for Euler Fraktur
`euex`	generate `fd` for Euler extra symbols
`msa`	generate `fd` for AMS symbols (A)
`msb`	generate `fd` for AMS symbols (B)

3 The implementation

Require LaTeX 2_ε.

```
⟨*sty⟩
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
⟨/sty⟩
```

Identify the file and version.

```
⟨sty⟩\ProvidesPackage{amsfonts}[2013/01/14 v3.01 Basic AMSFonts support]
⟨eur⟩\ProvidesFile{ueur.fd}    [2013/01/14 v3.01 Euler Roman]
⟨eus⟩\ProvidesFile{ueus.fd}    [2013/01/14 v3.01 Euler Script]
⟨euf⟩\ProvidesFile{ueuf.fd}    [2013/01/14 v3.01 Euler Fraktur]
⟨euex⟩\ProvidesFile{ueuex.fd}  [2013/01/14 v3.01 Euler extra symbols]
⟨msa⟩\ProvidesFile{umsa.fd}    [2013/01/14 v3.01 AMS symbols A]
⟨msb⟩\ProvidesFile{umsb.fd}    [2013/01/14 v3.01 AMS symbols B]
```

4 Package options

```
⟨*sty⟩
\DeclareOption{psamsfonts}{%
    \PackageWarningNoLine{amsfonts}{The 'psamsfonts' option is
      obsolete in AMSFonts v3}
}
```

Process the options for this package.

```
\ProcessOptions\relax
```

4.1 Font definitions for cmex

The `cmex10` option of `amsmath` is now obsolete, but it will be some time before we release an updated version of `amsmath`. In the meantime, when version 3 of `amsfonts` is used with older versions of the `amsmath` package, we want to disable the `cmex10` option. So, we do two things: (a) in `amsfonts`, we install the new `cmex` definitions unconditionally, and (b) we set the `\cmex@opt` flag to a value that will cause `amsmath` to skip processing of the `cmex10` option.

```
\DeclareFontShape{OMX}{cmex}{m}{n}{%
       <-7.5>cmex7%
    <7.5-8.5>cmex8%
    <8.5-9.5>cmex9%
    <9.5->cmex10%
}{}%

\expandafter\let\csname OMX/cmex/m/n/10\endcsname\relax

\def\cmex@opt{10}
```

5 Miscellaneous

`\ams@newcommand` Where `stix` and `amsfonts` define the same control sequences, we want to avoid inadvertently overriding `stix`'s definitions. If `stix` is loaded before `amsfonts`, the following conditional and a few more like it later in the code take care of the problem. There is similar code in the `stix` package in case `amsfonts` is loaded first.

```
\@ifpackageloaded{stix}{%
    \def\ams@DeclareMathDelimiter#1#2#3#4#5#6{}%
    \def\ams@DeclareMathSymbol#1#2#3#4{}%
}{%
```

```
    \let\ams@DeclareMathDelimiter\DeclareMathDelimiter
    \def\ams@DeclareMathSymbol#1#2#3#4{%
        \global\let#1\undefined
        \DeclareMathSymbol{#1}{#2}{#3}{#4}%
    }%
}
```

In case the `amsfonts` package is used apart from the `amsmath` package, we need to define the following functions.

```
\providecommand*{\@mathmeasure}[3]{%
  \setbox#1\hbox{\frozen@everymath\@emptytoks\m@th$#2#3$}}
```

`\newtoks` was still outer in early releases of LaTeX 2_ε, so we need to take a little extra care here.

```
\@ifundefined{@emptytoks}{\csname newtoks\endcsname\@emptytoks}{}
```

6 Preparing to use the extra math fonts

We declare the encoding schemes for two AMS math symbol fonts. `U` encoding with `\noaccents@` for math is already provided in base NFSS.

The AMS symbol fonts will be assigned via `\DeclareSymbolFont` since most of their characters are accessed with the `\mathchardef` primitive. This means that these fonts are loaded for the particular size whenever a size or version switch is requested by the user. At the present time bold forms of these fonts don't exist. So there are no overriding `\SetSymbolFont` commands for math-version 'bold'.

```
\DeclareSymbolFont{AMSa}{U}{msa}{m}{n}
\DeclareSymbolFont{AMSb}{U}{msb}{m}{n}
```

The next constructs are symbols that work the same in text or math. They are built the hard way using `\hexnumber@\sym...`. We can't use `\DeclareTextSymbol` since it only specifies the font by its encoding, which, being `U` for both `msam` `msbm`, is ambiguous.

```
\@ifundefined{yen}{%
  \edef\yen{\noexpand\mathhexbox{\hexnumber@\symAMSa}55}
}{}
\@ifundefined{checkmark}{%
  \edef\checkmark{\noexpand\mathhexbox{\hexnumber@\symAMSa}58}
}{}
\@ifundefined{circledR}{%
  \edef\circledR{\noexpand\mathhexbox{\hexnumber@\symAMSa}72}
}{}
\@ifundefined{maltese}{%
  \edef\maltese{\noexpand\mathhexbox{\hexnumber@\symAMSa}7A}
}{}
```

We now define a few symbols which reside in the `msam` and `msbm` fonts. The `\catcode` change is to ensure that the double-quote character is not active (which at one time was a problem when something like `german.sty` was used).

```
\begingroup \catcode`\"=12
\ams@DeclareMathDelimiter{\ulcorner}{\mathopen} {AMSa}{"70}{AMSa}{"70}
\ams@DeclareMathDelimiter{\urcorner}{\mathclose}{AMSa}{"71}{AMSa}{"71}
\ams@DeclareMathDelimiter{\llcorner}{\mathopen} {AMSa}{"78}{AMSa}{"78}
\ams@DeclareMathDelimiter{\lrcorner}{\mathclose}{AMSa}{"79}{AMSa}{"79}
```

The next two definitions redefine the `\widehat` and `\widetilde` command to use a special accent if their argument is suitably wide. (In plain TEX these commands can produce three different accents depending on the size of the argument.) The current implementation will tend to give wrong results (tilde or hat symbol too wide) if these accents are used in script or scriptscript math style. But making that part work properly is too much effort given the limitations of TEX 3.x.

```
\@ifpackageloaded{stix}{}{%
    \xdef\widehat#1{\noexpand\@mathmeasure\z@\textstyle{#1}%
      \noexpand\ifdim\noexpand\wd\z@>\tw@ em%
      \mathaccent"0\hexnumber@\symAMSb 5B{#1}%
      \noexpand\else\mathaccent"0362{#1}\noexpand\fi}
    \xdef\widetilde#1{\noexpand\@mathmeasure\z@\textstyle{#1}%
      \noexpand\ifdim\noexpand\wd\z@>\tw@ em%
      \mathaccent"0\hexnumber@\symAMSb 5D{#1}%
      \noexpand\else\mathaccent"0365{#1}\noexpand\fi}
}
```

Now we define two special arrows which are built with special characters from the first symbol font.

```
\@ifpackageloaded{stix}{}{%
    \DeclareMathSymbol{\dabar@}{\mathord}{AMSa}{"39}
    \xdef\dashrightarrow{\mathrel{\dabar@\dabar@
                            \mathchar"0\hexnumber@\symAMSa 4B}}%
    \xdef\dashleftarrow{\mathrel{\mathchar"0\hexnumber@\symAMSa 4C\dabar@
                            \dabar@}}%
    \global\let\dasharrow\dashrightarrow
}
```

To avoid using too many control sequence names by defining all new symbols provided with the two fonts, we defined so far only symbols which are not assigned via `\mathchardef`. The majority however will be defined only if the user loads the `amssymb` package or explicitly defines symbols using the `\DeclareMathSymbol` macro.

Finally we test the `\DeclareMathSymbol` command by redefining the PLAIN TEX symbols which were made up from different characters (and thus could not change sizes properly) but are now available as real characters. Note that we have to make them undefined first, otherwise `\DeclareMathSymbol` will complain that they are already defined.

```
\ams@DeclareMathSymbol{\rightleftharpoons}{\mathrel}{AMSa}{"0A}
\ams@DeclareMathSymbol{\angle}            {\mathord}{AMSa}{"5C}
\ams@DeclareMathSymbol{\hbar}             {\mathord}{AMSb}{"7E}
```

Include a few common symbols which are both in `msam` or `msbm` and also in

lasy, which might have been loaded already, so it is a good idea to make them undefined.

```
\ams@DeclareMathSymbol{\sqsubset}          {\mathrel}{AMSa}{"40}
\ams@DeclareMathSymbol{\sqsupset}          {\mathrel}{AMSa}{"41}
\ams@DeclareMathSymbol{\mho}               {\mathord}{AMSb}{"66}
```

Now we close the group so that " will get its old \catcode back.

```
\endgroup
```

7 Defining ⟨*math alphabet identifiers*⟩

\mathfrak The Fraktur alphabet will be accessed by the command \mathfrak inside of math mode.

```
\@ifundefined{mathfrak}{%
    \DeclareMathAlphabet{\mathfrak}{U}{euf}{m}{n}
    \SetMathAlphabet{\mathfrak}{bold}{U}{euf}{b}{n}
}{}
```

\mathbb The AMS symbol font B contains the blackboard bold math alphabet. There is only a single weight of this alphabet so it is used in all math versions.

```
\@ifundefined{mathbb}{%
    \DeclareSymbolFontAlphabet{\mathbb}{AMSb}%
}{}
```

7.1 Setting up the fonts for correct accents in math

There are a few ⟨*math alphabets*⟩ which don't have any or enough accents for math in the corresponding fonts. For example the \mathbb ⟨*math alphabet identifier*⟩ comes from a symbol font with none of the normal accents in the correct places. $\mathcal{AMS}$-TEX has a sophisticated solution for this problem: all math accents are defined in a way that the font where the accents are taken from depends on the current value of a macro called \accentclass@. To support this idea ⟨*math alphabet identfiers*⟩ which come with their own accents should set this macro to the number 7 (variable family in the terminology of the TEXbook); all others should set it to 0 so that the accents are taken from the ⟨*math group*⟩ zero in the current *math version*.

There is a \DeclareFontEncodingDefaults macro which declares such defaults. For the amsfonts package we set this default in the following way. This information will be ignored unless the amsmath package is also loaded.

```
\DeclareFontEncodingDefaults{\relax}{\def\accentclass@{7}}
```

8 Some backward compatibility definitions

The following commands are provided for compatibility with pre-1995 versions of the amsfonts package. If a \documentclass command is used the \@obsolete function will issue a warning saying that the command is obsolete.

Upgraded to robust commands. [bnb, 1996/09/29]

\frak
\Bbb
\bold

```
\DeclareRobustCommand{\frak}[1]{%
  {\@subst@obsolete{amsfonts}\frak\mathfrak{#1}}}
\DeclareRobustCommand{\Bbb}[1]{%
  {\@subst@obsolete{amsfonts}\Bbb\mathbb{#1}}}
\DeclareRobustCommand{\bold}[1]{%
  {\@subst@obsolete{amsfonts}\bold\mathbf{#1}}}
```

The \newsymbol command has different syntax than \DeclareMathSymbol so it cannot use the direct substitution function \@subst@obsolete.

```
\begingroup \catcode`\"=12 \relax
\gdef\newsymbol#1#2#3#4#5{%
  \@obsolete{amsfonts}\newsymbol\DeclareMathSymbol
  \@ifdefinable#1{%
     \edef\next@
       {\ifcase #2 \or
          \hexnumber@\symAMSa\or
          \hexnumber@\symAMSb\fi}%
     \ifx\next@\@empty
       \PackageError{amsfonts}{\Invalid@@\newsymbol}\@ehd%
     \else
      \global\mathchardef#1"#3\next@#4#5
     \fi}}
\endgroup
```

\@obsolete This command gives a warning on the first use that the command given as second argument is deprecated/obsolete, with the third argument recommended as a substitute. In compatibility mode we just continue silently.

```
\long\def\@gobblethree#1#2#3{}
\if@compatibility
  \let\@obsolete\@gobblethree
\else
  \def\@obsolete#1#2#3{\PackageWarning{#1}{%
    Obsolete command \protect#2; \protect#3 should be used instead}}%
\fi
```

\@subst@obsolete For obsolete commands that have a substitute command.

```
\def\@subst@obsolete#1#2#3{\@obsolete{#1}#2#3\gdef#2{#3}#2}

\@ifpackageloaded{stix}{\endinput}{}
```

First we load the symbols under the official AMS names and then define the LaTeX names via \let unless they are already defined (which probably means that the lasy fonts are defined).

```
\begingroup \catcode`\"=12
\DeclareMathSymbol{\square}           {\mathord}{AMSa}{"03}
\DeclareMathSymbol{\lozenge}          {\mathord}{AMSa}{"06}
\ams@DeclareMathSymbol{\vartriangleright} {\mathrel}{AMSa}{"42}
\ams@DeclareMathSymbol{\vartriangleleft}  {\mathrel}{AMSa}{"43}
```

```
\ams@DeclareMathSymbol{\trianglerighteq}  {\mathrel}{AMSa}{"44}
\ams@DeclareMathSymbol{\trianglelefteq}   {\mathrel}{AMSa}{"45}
\ams@DeclareMathSymbol{\rightsquigarrow}  {\mathrel}{AMSa}{"20}
```

Check if the `latexsym` package has already been loaded

```
\@ifpackageloaded{latexsym}{\@tempswafalse}{\@tempswatrue}
\if@tempswa
  \global\let\Box\square
  \global\let\Diamond\lozenge
  \global\let\leadsto\rightsquigarrow
```

`\lhd` and its relatives look like `\vartriangleleft` and its relatives, but the math atom types are different (according to the LaTeX book). So we need to issue different `\DeclareMathSymbol` statements.

```
  \global\let\lhd\@@undefined
  \global\let\unlhd\@@undefined
  \global\let\rhd\@@undefined
  \global\let\unrhd\@@undefined
  \DeclareMathSymbol{\lhd}     {\mathbin}{AMSa}{"43}
  \DeclareMathSymbol{\unlhd}   {\mathbin}{AMSa}{"45}
  \DeclareMathSymbol{\rhd}     {\mathbin}{AMSa}{"42}
  \DeclareMathSymbol{\unrhd}   {\mathbin}{AMSa}{"44}
```

No equivalent of the `\Join` symbol in `lasy` is available in `msam` or `msbm` so we make do with a composite of two characters.

```
\xdef\Join{\mathrel{\mathchar"0\hexnumber@\symAMSb 6F\mkern-13.8mu%
  \mathchar"0\hexnumber@\symAMSb 6E}}
\fi
\endgroup
```

⟨/sty⟩

9 The `.fd` files

9.1 AMS symbol font A

⟨∗msa⟩

```
\DeclareFontFamily{U}{msa}{}
\DeclareFontShape{U}{msa}{m}{n}{%
     <-6>msam5%
    <6-8>msam7%
    <8->msam10%
}{}
```

⟨/msa⟩

9.2 AMS symbol font B

⟨∗msb⟩

```
\DeclareFontFamily{U}{msb}{}
\DeclareFontShape{U}{msb}{m}{n}{%
     <-6>msbm5%
    <6-8>msbm7%
    <8->msbm10%
}{}
```

⟨/msb⟩

9.3 Euler Fraktur

⟨*euf⟩
```
\DeclareFontFamily{U}{euf}{}
\DeclareFontShape{U}{euf}{m}{n}{%
     <-6>eufm5%
    <6-8>eufm7%
    <8->eufm10%
}{}
\DeclareFontShape{U}{euf}{b}{n}{%
     <-6>eufb5%
    <6-8>eufb7%
    <8->eufb10%
}{}
```
⟨/euf⟩

9.4 Euler Script

⟨*eus⟩
```
\DeclareFontFamily{U}{eus}{\skewchar\font'60}
\DeclareFontShape{U}{eus}{m}{n}{%
     <-6>eusm5%
    <6-8>eusm7%
    <8->eusm10%
}{}
\DeclareFontShape{U}{eus}{b}{n}{%
     <-6>eusb5%
    <6-8>eusb7%
    <8->eusb10%
}{}
```
⟨/eus⟩

9.5 Euler math extension

⟨*euex⟩
```
\DeclareFontFamily{U}{euex}{}
\DeclareFontShape{U}{euex}{m}{n}{%
       <-7.5>euex7%
    <7.5-8.5>euex8%
    <8.5-9.5>euex9%
    <9.5->euex10%
}{}
```
⟨/euex⟩

9.6 Euler Math Roman

These fonts are nearly `OML` encoded but some characters are missing so we use `U` encoding again.

⟨*eur⟩
```
\DeclareFontFamily{U}{eur}{\skewchar\font'177}
\DeclareFontShape{U}{eur}{m}{n}{%
     <-6>eurm5%
```

```
      <6-8>eurm7%
      <8->eurm10%
    }{}
  \DeclareFontShape{U}{eur}{b}{n}{%
       <-6>eurb5%
      <6-8>eurb7%
      <8->eurb10%
  }{}
```
⟨/eur⟩

The next line goes into all files and in addition prevents `docstrip` from adding any further code from the main source file (such as a character table).

```
\endinput
```