

The AMS Export Bib_TE_X style [`amsxport`]

American Mathematical Society
Michael Downes
David M. Jones

Version 2.01, 2007/10/01

Contents

1	Introduction	1
2	Implementation	2
3	Macros for month names	3
4	Formatting author names and editor names	9
5	Handling year, month, date	10
6	Journal abbreviations	11
7	Type info	12
8	Miscellaneous tail-end information	12
9	Wrapper for each entry	12
10	Entry types	13
11	Sorting the entries	16
12	Final pass for output	21
13	Journal abbreviations	22
14	To Do	25

1 Introduction

`amsxport` is the collective name for a collection of Bib_TE_X styles that form a bridge between Bib_TE_X and `amsrefs`. It produces output in which the structural information of the original Bib_TE_X database file is retained. No ad hoc visual formatting is introduced that would hamper the application of alternative design specifications when rendering the information on paper or screen. This makes it possible to have Bib_TE_X handle only extraction and sorting and leave the visual appearance to be controlled entirely by L^AT_EX. The process of applying typical bibliography specs to exported data is illustrated in the `amsrefs` package. The authoritative description of the export format is in the `amsrefs` documentation, inasmuch as interpreting the format is the chief task of the `amsrefs` package.

Each of the five `.bst` files generated from this file corresponds to a combination of `amsrefs` package options, as follows:

amsrn: This is the default style, used when none of the following is appropriate.

amsru: This is used when the `citation-order` option is specified. It is identical to `amsrn` except that the bibliography items are not sorted.

amsra: This is when with the `alphabetic` option is specified, byt the `y2k` option is *not* also specified.

amsry: This is used when both the `alphabetic` and `y2k` options are specified.

amsrs: This is used when the `shortalphabetic` option is specified.

Readers may wish to refer also to the standard Bib_TE_X documentation (`btxdoc.tex`, `btxhak.tex`, `btxbst.doc`) for background information.

2 Implementation

The list below gives the fields that are expected to occur. (Other fields will be discarded without comment.) suggested fields.

```

1 <*bst>
2 ENTRY
3 {
4   address      institution  number      status
5   archive      isbn          organization subtitle
6   author       issn          pages       title
7   booktitle    journal      part        translator
8   chapter      key          pii         type
9   date         language    preprint    url
10  edition      meeting     publisher   volume
11  editor       month       review      xid
12  eprint       mrnumber    school      year
13  howpublished note        series
14 }
15 { }
16 { label }
```

Some comments about specific fields:

archive: The archive that holds the eprint listed in the `eprint` field.

author: Bib_T_EX doesn't allow this field (or any field) to be used more than once, but in the output its contents will be split into a separate key-value pair for each name. Likewise for `editor` and `translator`.

crossref: This is a built-in field type, so it doesn't occur in the argument of ENTRY above.

date: This is a generalization of the `year` and `month` fields. Its value should be written in ISO 8601 format, e.g., 1987-06-05; but the day and month are omissible, so this can be easily be used instead of the `year` field.

edition: For books. The Bib_T_EX documentation suggests that the value of this field should be an ordinal word such as "Second". This may be because the Bib_T_EX language provides only the weakest sort of support for an *is-numeric* test.

eprint: Electronic preprint information such as for www.arXiv.org.

institution: What's the difference between an `institution` and an `organization`? The Bib_T_EX documentation says that `institution` should be used for technical reports and `organization` for other entry types.

isbn: International Standard Book Number.

issn: International Standard Serial Number. See the discussion of journal abbreviations.

language: Language of the work. This is especially desirable when the value of the title field is a translation of the true title (most often when the original language is one like Arabic or Chinese that poses rendering problems for non-native software).

meeting: Since Bib_T_EX doesn't allow a field name to be the same as a function name, we have to use `meeting` instead of `conference` if we want to give the title of a conference in a dedicated field.

mrnumber: An alias for `review` which we provide because MathSciNet has been using it in the Bib_T_EX output that it provides for a couple of years already.

organization: See the comment for `institution`.

pii: Publisher Item Identifier

part: This is for a long journal article that is published in separate issues of the same journal. There should be a separate BibTeX entry for each part (though all the ones after part 1 could crossref the first one).

preprint: It is not possible quite yet to assume that all preprints should be classified as eprints. If you have a preprint that is not an eprint, use this field to give the “address” where the preprint may be found (institution, preprint number, etc.).

review: Review numbers or similar pointers, e.g., for Mathematical Reviews or Zentralblatt.

school: The analog of “organization”, for a thesis.

status: Typically used for notes such as “to appear” or “in preparation” with journal articles.

subtitle: Typically used with a multipart journal article to give a subtitle for each part.

translator: This needs no explanation, surely? Except perhaps to note that the standard BibTeX styles don’t provide it.

url: Universal Resource Locator.

xid: This is used by a cross-referenced item to pass its identity to child entries that refer to it. It would not be necessary if BibTeX left the `crossref` field value accessible, but for some reason that I have not found yet (skimming in the source code) it is cleared internally before there is any opportunity to save it.

Practically speaking: For every `crossref` in your `.bib` files, the target should contain a matching `xid` field.

More on extra fields. In the Shelah bibliography, the `author` field is given without any accents in the author names, and the normal TeX version of the author names is given in a separate field called `trueauthor`.

Also, for MR reviews:

```
review = {MR 48:3735},
```

And for each author a country is indicated with the `fromwhere` field, e.g.,

```
fromwhere = {UK, IL},
```

This means the first author is from the United Kingdom and the second one from Israel.

3 Macros for month names

```
17 MACRO {jan} {"-01"}
18 MACRO {feb} {"-02"}
19 MACRO {mar} {"-03"}
20 MACRO {apr} {"-04"}
21 MACRO {may} {"-05"}
22 MACRO {jun} {"-06"}
23 MACRO {jul} {"-07"}
24 MACRO {aug} {"-08"}
25 MACRO {sep} {"-09"}
26 MACRO {oct} {"-10"}
27 MACRO {nov} {"-11"}
28 MACRO {dec} {"-12"}
```

Some scratch variables and global variables.

```
29 STRINGS { p s t l }
```

```
30 INTEGERS { len ptr }
31 <bst>
```

Note: BibTeX 0.99 requires a blank line in certain contexts, such as after the READ command, and will choke if it is missing. When it is necessary to add a blank line at the boundary of a macro code section we use a <bst> marker.

Some utility functions.

```
32 FUNCTION {NOT} { { #0 } { #1 } if$ }
33
34 FUNCTION {AND} { 'skip$ { pop$ #0 } if$ }
35
36 FUNCTION {OR} { { pop$ #1 } 'skip$ if$ }
37
38 FUNCTION {TRUE} { #1 }
39
40 FUNCTION {FALSE} { #0 }
41 <bst>
```

incr.ptr

```
42 FUNCTION {incr.ptr} { ptr #1 + 'ptr := }
43 <bst>
```

current.char Returns: first character of STRING

```
44 FUNCTION {current.char} { ptr #1 substring$ }
```

copy.two.chars The function copy.two.chars appends the next two characters from s to STRING.

Arg: STRING (partial copy of pages string)

Returns: Modified version of STRING

Side effects: advances ptr

```
45 FUNCTION {copy.two.chars} { t * incr.ptr s current.char * incr.ptr }
46 <bst>
```

is.digit

```
47 FUNCTION {is.digit} {
48   chr.to.int$
49   duplicate$
50   "0" chr.to.int$ #1 - >
51   swap$
52   "9" chr.to.int$ #1 + <
53   AND
54 }
55 <bst>
```

is.lowercase.letter In a previous implementation the is.letter function first lowercased the character using change.case\$, then tested it against the range a–z. But change.case\$ issues a warning if it is applied to a lone brace character.

```
56 FUNCTION {is.lowercase.letter} {
57   chr.to.int$ duplicate$
58   "a" chr.to.int$ #1 - >
59   swap$
60   "z" chr.to.int$ #1 + <
61   AND
62 }
```

`is.uppercase.letter`

```
63 FUNCTION {is.uppercase.letter} {
64   chr.to.int$ duplicate$
65   "A" chr.to.int$ #1 - >
66   swap$
67   "Z" chr.to.int$ #1 + <
68   AND
69 }
```

`is.letter`

```
70 FUNCTION {is.letter} {
71   duplicate$
72   is.lowercase.letter
73   { pop$ TRUE }
74   { is.uppercase.letter }
75   if$
76 }
```

`skip.accent` This implementation of `skip.accent` simply skips ahead one character whenever a backslash is found. This handles things like `\v`, `\u`, `\k`, `\r`. Although this is overly simplistic it is still an improvement over the previous method which was to not even make the attempt. [mjd,2001-10-27]

```
77 FUNCTION {skip.accent} { pop$ incr.ptr }
```

`strip.label`

```
78 FUNCTION {strip.label} {
79   's :=
80   #1 'ptr :=
81   ""
82   { s ptr #1 substring$ duplicate$ empty$ NOT }
83   { duplicate$ is.letter
84     { * }
85     { duplicate$ "\" =
86       { skip.accent }
87       'pop$
88       if$
89     }
90     if$
91     incr.ptr
92   }
93   while$
```

When the loop ends we have an extra empty-string on top of the stack to get rid of.

```
94   pop$
95 }
```

`sortify`

```
96 FUNCTION {sortify} {
97   purify$
98   "l" change.case$
99 }
```

`lpad` Warning: The use of `text.length$` here restricts `lpad`'s use to strings that we know do not contain any special characters. Since we only use it for field names, this is ok.

```

    <string> <int> lpad
100 FUNCTION {lpad} {
101     swap$
102     'l :=
103     l text.length$ - 'len :=
104     { len #0 > }
105     {
106         " " l * 'l :=
107         len #1 - 'len :=
108     }
109     while$
110     l
111 }

```

field.or.null

```

112 FUNCTION {field.or.null} {
113     duplicate$ empty$
114     { pop$ "" }
115     'skip$
116     if$
117 }

```

missing.or.empty

```

118 FUNCTION {missing.or.empty} { duplicate$ missing$ swap$ empty$ OR }
119 <bst>

```

`ndash.and.skip.hyphens` The function `append.ndash.and.skip.hyphens` adds `\ndash` to `STRING` when a hyphen is found and advances `ptr` to the next non-hyphen character in `s`.

Arg: `STRING` (partial copy of pages string)

Returns: modified version of `STRING`

Side effects: advances `ptr`

```

120 FUNCTION {append.ndash.and.skip.hyphens} {
121     "\ndash " *
122     incr.ptr
123     { s current.char "-" = }
124     { incr.ptr }
125     while$
126 }

```

`n.dashify` In the `n.dashify` function we store the given string in `s`, push an empty string on the stack, and start examining the characters of `s`. For a non-hyphen character, we just append a copy of it to the top string. For a hyphen, we append `\ndash` to the top string and advance the pointer until we reach a non-hyphen character. And there is one more exceptional case: for a backslash, we copy two characters instead of one. This keeps us from erroneously translating `\-` to `\ndash`. One would scarcely expect to see an instance of `\-` in a page number field but let's face it, in actual use nearly anything can and does happen.

Arg: `STRING` (value of `pages` field)

Returns: `STRING` (dashified version of input string)

```

127 FUNCTION {n.dashify} {
128     's :=
129     #1 'ptr :=
130     ""
131     { s current.char 't := t "" = NOT }

```

```

132     { t "\" =
133         { copy.two.chars }
134     { t "-" =
135         { append.ndash.and.skip.hyphens }
136         { t * incr.ptr }
137     if$
138     }
139     if$
140     }
141     while$
142 }

```

missing.warning Standard warning message for a missing or empty field. For the user we call any such field ‘missing’ without respect to the distinction made by BibTeX between missing and empty.

```

143 FUNCTION {missing.warning} {
144     "missing " swap$ * " in " * cite$ * warning$
145 }
146 <bst>

```

string.length Because BibTeX doesn’t provide a straightforward `string.length` function (`text.length$` counts “special characters” as a single character), it appears necessary to implement one the hard way.

```

147 INTEGERS { string.ptr }
148
149 FUNCTION {string.length} {
150     #1 'string.ptr :=
151     { duplicate$ string.ptr #1 substring$ "" = 'FALSE 'TRUE if$ }
152     { string.ptr #1 + 'string.ptr := }
153     while$
154     pop$
155     string.ptr #1 -
156 }

```

format.title

```

157 FUNCTION {format.title}{
158     duplicate$
159     missing.or.empty
160     { pop$ "" }
161     { "t" change.case$ }
162     if$
163 }

```

start.field

```

164 FUNCTION {start.field} {
165     #12 lpad "=" * write$
166 }

```

fin.field

```

167 FUNCTION {fin.field} { "}," write$ newline$ }
168 <bst>

```

write.field

```

169 FUNCTION {write.field} {

```

```

170     duplicate$
171     missing$
172         { pop$ missing.warning }
173         { duplicate$ empty$
174             { pop$ missing.warning }
175             { swap$ start.field write$ fin.field }
176         if$
177     }
178     if$
179 }

```

optional.field

```

180 FUNCTION {optional.field} {
181     duplicate$
182     missing.or.empty
183     { pop$ pop$ }
184     { swap$ start.field write$ fin.field }
185     if$
186 }

```

optional.title.field

```

187 FUNCTION {optional.title.field} {
188     duplicate$
189     missing.or.empty
190     { pop$ pop$ }
191     { format.title swap$ start.field write$ fin.field }
192     if$
193 }

```

optional.pages.field

```

194 FUNCTION {optional.pages.field} {
195     duplicate$
196     missing.or.empty
197     { pop$ pop$ }
198     { swap$ start.field
199         n.dashify
200         write$ fin.field
201     }
202     if$
203 }

```

optional.mr.field

```

204 FUNCTION {optional.mr.field} {
205     duplicate$
206     missing.or.empty
207     { pop$ pop$ }
208     {
209         swap$ start.field
210         "\MR{" swap$ * "}" * write$ fin.field
211     }
212     if$
213 }

```

remove.ordinal.suffix

```

214 FUNCTION {remove.ordinal.suffix} {
215   's :=
216   s string.length 'len :=
217   #1 'ptr :=
218   { ptr len < s ptr #1 substring$ is.digit AND }
219   'incr.ptr
220   while$
221   s ptr global.max$ substring$ 'p :=
222   p "st" =
223     { TRUE }
224     { p "nd" =
225       { TRUE }
226       { p "rd" =
227         { TRUE }
228         { p "th" =
229           { TRUE }
230           { FALSE }
231           if$
232         }
233         if$
234       }
235       if$
236     }
237   if$
238   { s #1 ptr #1 - substring$ }
239   { s }
240   if$
241 }

```

optional.edition.field

```

242 FUNCTION {optional.edition.field} {
243   duplicate$ missing.or.empty
244   { pop$ pop$ }
245   { swap$ start.field
246     remove.ordinal.suffix
247     write$ fin.field
248   }
249   if$
250 }

```

4 Formatting author names and editor names

Take a name list in Bib_TE_X form (names separated by the word “and”) and output the desired form of each name.

```

251 INTEGERS { nameptr numnames namesleft }
252 STRINGS { namelist fieldname }

```

`format.name` The `format.name` function operates on a single name, producing a string of the form

von Last, First, Jr.

Args: *namelist*, *index* (top; integer)

Returns: formatted version of the *n*th name in *namelist* where integer *index* specifies *n*.

```

253 FUNCTION {format.name} { "{vv~}{ll}{, ff}{, jj}" format.name$ }
254 <bst>

```

`optional.name.field` The function `optional.name.field` handles the task of splitting a multiple-name field value into multiple fields with single values.

Args: *fieldname*, *namelist* (top)

Side effects: writes a key/value pair to the output file for each name in *namelist*.

```

255 FUNCTION {optional.name.field}{
256     duplicate$ missing.or.empty
257     { pop$ pop$ }
258     { swap$ 'fieldname :=
259       duplicate$ num.names$
260       'namesleft :=
261       #1 'ptr :=
262       { namesleft #0 > }
263       { fieldname start.field

```

Copy the name string before running `format.name` on it.

```

264         duplicate$ ptr format.name write$
265         fin.field
266         ptr #1 + 'ptr :=
267         namesleft #1 - 'namesleft :=
268     }
269     while$

```

At this point we have an extra copy of the name string on the stack.

```

270     pop$
271     }
272     if$
273 }

```

5 Handling year, month, date

`assemble.date.field` In the output we produce a `date` field instead of the year and month fields normally used in Bib_TE_X files. In the date field we use ISO date notation (e.g., 1987-06-05) to facilitate switching between full and abbreviated month names. The month and day parts are frequently absent, making the date field equivalent to a year field.

```

274 FUNCTION {assemble.date.field}
275 { date missing$
276   { year missing.or.empty
277     { status missing.or.empty
278       { "No year or other date information for "
279         cite$ * warning$ }
280       'skip$
281       if$
282       "status"
283     }
284     { year month missing$ { "" } { month } if$ * }
285     if$
286   }
287   { date }
288   if$

```

If the date string is now equal to “status”, it means that we have a status field or we already gave a warning about lack of date info; in either case omit the date field.

```

289 duplicate$ "status" =
290   { pop$ pop$ }
291   { swap$ start.field write$ fin.field }
292 if$
293 }

```

6 Journal abbreviations

Use of journal abbreviations is recommended to make supplying ISSN numbers easier. To use this feature, you need a STRING definition of the form

```

@STRING{cpam="cpam/0010-3640/"
        #"Communications in Pure and Applied Mathematics"}

```

Then when you write

```
journal=cpam,
```

exporting with `amsxport` will produce

```
journal={cpam},
ISSN={0010-3640},
```

The text before the ISSN number in the STRING definition is preferably a repetition of the abbreviation name (leaving expansion to be done on the \LaTeX side); or it could be the full journal name.

Warning: If you include braces and write `journal={cpam}`, \BibTeX will not expand the abbreviation but leave it as the field value; and then you will not get automatic lookup of the ISSN number.

`optional.journal.field`

```

294 FUNCTION {optional.journal.field} {
295   duplicate$ missing.or.empty
296   { pop$ pop$ }
297   {
298     's :=
299     start.field
300     #1 'ptr :=
301     s current.char is.lowercase.letter
302     {
303       s "/" * 's :=
304       { s ptr #1 substring$ "/" = NOT }
305       { incr.ptr }
306       while$

```

Journal name started with a lowercase letter, but we couldn't find no slash? All right, just let it go through as it is.

```

307     s ptr #1 + #1 substring$ empty$
308     { s #1 ptr #1 - substring$ }

```

But if we did find a slash, then we probably have found us an ISSN number that we can write out.

```

309     {
310       s #1 ptr #1 - substring$
311       write$
312       fin.field

```

ISSNs are always nine characters in length (four digits, hyphen, four digits).

```

313             "ISSN" start.field
314             s ptr #1 + #9 substring$
315             }
316             if$
317             }
318             { s }
319             if$
320             write$
321             fin.field
322         }
323     if$
324 }
```

7 Type info

`optional.type.field` Putz around with the type info a little, for theses mainly. Two types here: the type field, and the entry type.

```

325 FUNCTION {optional.type.field}
326 { duplicate$ missing.or.empty
327   { type$ "mastersthesis" = { "Master's Thesis" }
328     { type$ "phdthesis" = { "Ph.D. Thesis" }
329       { "" } if$ } if$
330     swap$ pop$
331   }
332   'skip$
333   if$
334   duplicate$ empty$
335   { pop$ pop$ }
336   { swap$ start.field write$ fin.field }
337   if$
338 }
```

8 Miscellaneous tail-end information

`url.note.status.review` The following items are common to all entry types are output at the tail end of the entry.

```

339 FUNCTION {url.note.status.review} {
340   "url" url optional.field
341   "note" note optional.field
342   "status" status optional.field
```

The `review` field should perhaps be translated into multiple fields like author names, if more than one review is given.

```

343   "review" review optional.field
```

Data from MathSciNet will have the Math Reviews number in an `mrnumber` field. We assume that a given entry will have a `review` field or an `mrnumber` field but not both.

```

344   "review" mrnumber optional.mr.field
345 }
```

9 Wrapper for each entry

`start.entry`

```

346 FUNCTION {start.entry} {
347     newline$
348     "\bib{" cite$ * "}{ " *
Merge Master's thesis and Ph.D. thesis into a single type.
349     type$ "mastersthesis" = type$ "phdthesis" = OR
350     { "thesis" }
351     { type$ }
352     if$
353     * "}{ " * write$
354     newline$
355 <*debug>
356     "sort.label" label sortify optional.field
357     "sort.key" sort.key$ optional.field
358 </debug>
359 }

```

`fin.entry`

```

360 FUNCTION {fin.entry} {
361     "}" write$
362     newline$
363 }

```

10 Entry types

Here are the types of entries that are normally allowed in a BibTeX file:

book: A book with an explicit publisher.

booklet: A work that is printed and bound, but without a named publisher or sponsoring institution.

inbook: A part of a book, which may be a chapter (or section or whatever) and/or a range of pages.

incollection: A part of a book having its own title.

manual: Technical documentation.

mastersthesis: A Master's thesis.

phdthesis: A PhD thesis.

proceedings: The proceedings of a conference.

techreport: A report published by a school or other institution, usually numbered within a series.

article: An article from a journal or magazine.

inproceedings: An article in a conference proceedings.

conference: An alias for `inproceedings`.

unpublished: A document having an author and title, but not formally published.

misc: Use this type when nothing else fits.

article A journal article differs from an `inproceedings` article by not having `booktitle`, `publisher`, `editor`, and other such info.

```

364 FUNCTION {article} {
365     start.entry
366     "author" author optional.name.field
367     "translator" translator optional.name.field
368     "title" title format.title write.field
369     "subtitle" subtitle optional.title.field

```

```

370     "language" language optional.field
371     "organization" organization optional.field
372     "how" howpublished optional.field

```

Construct a date value from date, year, month fields

```

373     "date" assemble.date.field
374     "ISSN" issn optional.field
375     "journal" journal optional.journal.field
376     "volume" volume optional.field
377     "number" number optional.field
378     "pages" pages optional.pages.field
379     "PII" pii optional.field
380     "archive" archive optional.field
381     "eprint" eprint optional.field
382     "preprint" preprint optional.field
383     url.note.status.review
384     fin.entry
385 }

```

inproceedings An inproceedings entry may have xid, booktitle, meeting, publisher info; it is not expected to have journal, eprint, or preprint info.

```

386 FUNCTION {inproceedings} {
387     start.entry
388     "author" author optional.name.field
389     "translator" translator optional.name.field
390     "title" title format.title write.field
391     "subtitle" subtitle optional.title.field
392     "language" language optional.field
393     "organization" organization optional.field
394     "how" howpublished optional.field

```

Construct a date value from date, year, month fields

```

395     "date" assemble.date.field
396     "xid" xid optional.field

```

The following fields might be inherited from the parent of an article in a proceedings volume or collection.

```

397     "conference" meeting optional.field
398     "booktitle" booktitle optional.title.field

```

Include the edition in case this is an inbook entry.

```

399     "edition" edition optional.edition.field
400     "editor" editor optional.name.field
401     "series" series optional.field
402     "volume" volume optional.field
403     "publisher" publisher optional.field
404     "address" address optional.field
405     "pages" pages optional.pages.field
406     url.note.status.review
407     fin.entry
408 }

```

inbook

```

409 FUNCTION {inbook} { inproceedings }

```

incollection

```

410 FUNCTION {incollection} { inproceedings }

```

inproceedings

```
411 FUNCTION {conference} { inproceedings }
```

book The book type includes a meeting field because otherwise we'd have to repeat the entire definition for the proceedings type with that as the sole difference.

```
412 FUNCTION {book} {
413     start.entry
414     "author" author optional.name.field
415     "editor" editor optional.name.field
416     "translator" translator optional.name.field
417     "title" title missing$ { booktitle } { title } if$
418     format.title write.field
419     "subtitle" subtitle optional.title.field
420     "type" type optional.type.field
421     "language" language optional.field
422     "conference" meeting optional.field
423     "edition" edition optional.edition.field
424     "series" series optional.field
425     "publisher" publisher optional.field
426     organization missing.or.empty
427     { "institution" institution optional.field }
428     { "organization" organization optional.field }
429     if$
430     "address" address optional.field
431     "how" howpublished optional.field
432     "date" year write.field
433     "volume" volume optional.field
```

The number here is intended for a tech report.

```
434     "number" number optional.field
435     "ISBN" isbn optional.field
436     url.note.status.review
437     fin.entry
438 }
```

booklet

```
439 {booklet}
440 FUNCTION {booklet} { book }
```

manual

```
441 FUNCTION {manual} { book }
```

mastersthesis

```
442 FUNCTION {mastersthesis} { book }
443
444 FUNCTION {phdthesis} { book }
445
446 FUNCTION {proceedings} { book }
447
448 FUNCTION {collection} { book }
449
450 FUNCTION {techreport} { book }
451
452 FUNCTION {unpublished} { book }
```

```

453
454 FUNCTION {misc} { book }
455
456 FUNCTION {default.type} { misc }
457 <bst>

```

Bib_TEX 0.99 keels over if READ does not have a blank line after it.

```

458 <bst>
459 READ
460 <bst>

```

11 Sorting the entries

Since the final labels will be produced by the `amsrefs` package, we don't need to worry about generating suffixes to disambiguate between equal stems. All we have to do is strive to generate stems that are identical to the ones that `amsrefs` will generate, so that `amsxport` will sort its output in a fashion that will be consistent with the labels.

Overall, this simplifies our job, since we don't have to keep track of duplicate labels, so we can process each entry in isolation. However, we do have to be more careful about the handling of text accents and text symbols in order to maintain compatibility with `amsrefs`.

See (*inter alia*) the sections "Lexical structure of names" and "Generating alphabetic labels" in the `amsrefs` implementation documentation for more information.

```

461 <*sort>

```

`chop.word`

```

462 FUNCTION {chop.word} {
463   's :=
464   'len :=
465   s #1 len substring$ =
466   { s len #1 + global.max$ substring$ }
467   's
468   if$
469 }

```

`sort.format.title`

```

470 FUNCTION {sort.format.title} {
471   't :=
472   "A " #2
473   "An " #3
474   "The " #4 t chop.word
475   chop.word
476   chop.word
477   sortify
478   #1 global.max$ substring$
479 }

```

`format.lab.names`

```

480 <*alpha | short>
481 FUNCTION {format.lab.names} {
482   's :=
483   s num.names$ 'numnames :=

```

```

484     numnames #1 >
485     { numnames #4 >
486         { #3 'namesleft := }
487         { numnames 'namesleft := }
488         if$
489         #1 'nameptr :=
490         ""
491         { namesleft #0 > }
492         { nameptr numnames =
493             { s nameptr "{ff }{vv }{ll}{ jj}" format.name$ "others" =
494                 { "{\etalchar{+}}" * }
495                 { s nameptr "{v{}}{l{}}" format.name$ * }
496                 if$
497                 }
498                 { s nameptr "{v{}}{l{}}" format.name$ * }
499                 if$
500                 nameptr #1 + 'nameptr :=
501                 namesleft #1 - 'namesleft :=
502                 }
503             while$
504             numnames #4 >
505                 { "{\etalchar{+}}" * }
506                 'skip$
507             if$
508             }
509             { s #1 "{v{}}{l{}}" format.name$
510             duplicate$ text.length$ #2 <
511 <alpha>             { pop$ s #1 "{ll}" format.name$ #3 text.prefix$ }
512 <short>            { pop$ s #1 "{ll}" format.name$ #1 text.prefix$ }
513                 'skip$
514             if$
515             }
516         if$
517     }
518
519 FUNCTION {author.key.label}
520 { author empty$
521     { key empty$
522     { cite$ #1 #3 substring$ }
523     { key #3 text.prefix$ }
524     if$
525     }
526     { author format.lab.names }
527     if$
528 }
529
530 FUNCTION {author.editor.key.label}
531 { author empty$
532     { editor empty$
533     { key empty$
534     { cite$ #1 #3 substring$ }
535     { key #3 text.prefix$ }
536     if$
537 }

```

```

538 { editor format.lab.names }
539     if$
540     }
541     { author format.lab.names }
542     if$
543 }
544
545 FUNCTION {author.key.organization.label}
546 { author empty$
547     { key empty$
548     { organization empty$
549         { cite$ #1 #3 substring$ }
550         { "The " #4 organization chop.word #3 text.prefix$ }
551         if$
552     }
553 { key #3 text.prefix$ }
554     if$
555     }
556     { author format.lab.names }
557     if$
558 }
559
560 FUNCTION {editor.key.organization.label}
561 { editor empty$
562     { key empty$
563     { organization empty$
564         { cite$ #1 #3 substring$ }
565         { "The " #4 organization chop.word #3 text.prefix$ }
566         if$
567     }
568 { key #3 text.prefix$ }
569     if$
570     }
571     { editor format.lab.names }
572     if$
573 }
574
575 FUNCTION {calc.label}{
576     type$ "book" = type$ "inbook" = OR
577     'author.editor.key.label
578     { type$ "proceedings" =
579     'editor.key.organization.label
580 { type$ "manual" =
581     'author.key.organization.label
582     'author.key.label
583     if$
584 }
585     if$
586     }
587     if$
588     strip.label
589     sortify
590 <*alpha>
591     year field.or.null

```

```

592 </ly2k>    #3 #8 substring$
593    purify$ *
594 </alpha>
595    'label :=
596 }
597 </alpha | short>

```

sort.format.names

```

598 FUNCTION {sort.format.names} {
599     's :=
600     #1 'ptr :=
601     ""
602     s num.names$ 'numnames :=
603     numnames 'namesleft :=
604     { namesleft #0 > }
605     { ptr #1 >
606       { " " * }
607       'skip$
608       if$
609       s ptr "{vv{ } }{ll{ }}{ ff{ }}{ jj{ }}" format.name$ 't :=
610       ptr numnames = t "others" = AND
611       { "et al" }
612       { t sortify }
613       if$
614       *
615       ptr #1 + 'ptr :=
616       namesleft #1 - 'namesleft :=
617     }
618     while$
619 }

```

sort.warning

```

620 FUNCTION {sort.warning} {
621     "to sort, need " swap$ * " or key in " * cite$ * warning$
622 }

```

author.sort

```

623 FUNCTION {author.sort}
624 { author empty$
625   { key empty$
626     { "author" sort.warning
627       ""
628     }
629     { key sortify }
630   if$
631 }
632 { author sort.format.names }
633 if$
634 }

```

author.editor.sort

```

635 FUNCTION {author.editor.sort}
636 { author empty$
637   { editor empty$

```

```

638     { key empty$
639         { "author, editor," sort.warning
640             ""
641         }
642         { key sortify }
643         if$
644     }
645     { editor sort.format.names }
646     if$
647 }
648 { author sort.format.names }
649 if$
650 }

```

author.organization.sort

```

651 FUNCTION {author.organization.sort}
652 { author empty$
653     { organization empty$
654         { key empty$
655             { "author, organization," sort.warning
656                 ""
657             }
658             { key sortify }
659             if$
660         }
661         { "The " #4 organization chop.word sortify }
662         if$
663     }
664     { author sort.format.names }
665     if$
666 }

```

editor.organization.sort

```

667 FUNCTION {editor.organization.sort}
668 { editor empty$
669     { organization empty$
670         { key empty$
671             { "editor, organization," sort.warning
672                 ""
673             }
674             { key sortify }
675             if$
676         }
677         { "The " #4 organization chop.word sortify }
678         if$
679     }
680     { editor sort.format.names }
681     if$
682 }

```

presort

```

683 FUNCTION {presort} {
684 <*alpha | short>
685     calc.label

```

```

686     label
687 </alpha | short>
688     type$ "book" = type$ "inbook" = OR type$ "collection" = OR
689         'author.editor.sort
690         { type$ "proceedings" =
691             'editor.organization.sort
692             { type$ "manual" =
693                 'author.organization.sort
694                 'author.sort
695             }
696         }
697     if$
698 }
699 if$
700 <alpha | short> *
701 " " *
702 <!alpha | short> year field.or.null sortify * " " *
703 title field.or.null sort.format.title *
704 #1 entry.max$ substring$
705 'sort.key$ :=
706 }

707 ITERATE {presort}
708
709 SORT
710 </sort>

```

12 Final pass for output

`write.with.newlines` The `write.with.newlines` function stores a string in `s` and runs through it looking for occurrences of `^M`; when one is found, the substring before it is written out with a newline, the remainder is left in `s`, and the cycle begins again.

```

711 FUNCTION {write.with.newlines}
712 { 's := #1 'ptr :=
713   { s ptr #2 + #1 substring$ "" = NOT }
714   { s ptr #3 substring$ "^M" =
715     { s #1 ptr #1 - substring$ write$ newline$
716       ptr #3 + 'ptr :=
717       s ptr global.max$ substring$ 's :=
718       #1 'ptr :=
719     }
720     { incr.ptr }
721     if$
722   }
723   while$
724   s write$ newline$
725 }

```

`begin.bib`

```

726 FUNCTION {begin.bib} {
727   preamble$ empty$
728   'skip$
729   { preamble$ write.with.newlines }

```

```

730     if$
731     "% \bib, bibdiv, biblist are defined by the amsrefs package."
732         write$ newline$
733     "\begin{bibdiv}" write$ newline$
734     "\begin{biblist}" write$ newline$
735 }

```

end.bin

```

736 FUNCTION {end.bib} {
737     newline$
738     "\end{biblist}" write$ newline$
739     "\end{bibdiv}" write$ newline$
740 }

741 EXECUTE {begin.bib}
742
743 ITERATE {call.type$}
744
745 EXECUTE {end.bib}
746 </bst>

```

13 Journal abbreviations

By putting the ISSN into the L^AT_EX document, we make it easier to construct www queries from the bibliography data. These string definitions tie together journal abbreviations, ISSN numbers, and journal names.

These are the journals that are cited most frequently by American Mathematical Society authors at the present time (2000 CE).

The format used here is designed to make it easy for BibTeX to retrieve an ISSN number when given a journal abbreviation.

```

747 <*abbrevs>
748 @string{aa="aa/0065-1036/"
749     #"Acta Arithmetica"}
750 @string{actamath="actamath/0001-5962/"
751     #"Acta Mathematica"}
752 @string{asms="asms/0001-6969/"
753     #"Acta Scientiarum Mathematicarum"}
754 @string{advmath="advmath/0001-8708/"
755     #"Advances in Mathematics"}
756 @string{ajm="ajm/0002-9327/"
757     #"American Journal of Mathematics"}
758 @string{amm="amm/0002-9890/"
759     #"American Mathematical Monthly"}
760 @string{aifg="aifg/0373-0956/"
761     #"Annales de l'Institut Fourier"}
762 @string{am2="am2/0003-486X/Annals of Mathematics"}
763 @string{ap="ap/0091-1798/"
764     #"The Annals of Probability"}
765 @string{asens4="asens4/0012-9593/"
766     #"Annales Scientifiques de l'École Normale Supérieure"}
767 @string{amb="amb/0003-889X/Archiv der Mathematik"}
768 @string{arma="arma/0003-9527/"
769     #"Archive for Rational Mechanics and Analysis"}
770 @string{bamsn="bamsn/0273-0979/"
771     #"Bulletin (New Series) of the American Mathematical Society"}

```

772 @string{bams="bams/0004-9727/"
 773 # "Bulletin of the Australian Mathematical Society"}
 774 @string{blms="blms/0024-6093/"
 775 # "Bulletin of the London Mathematical Society"}
 776 @string{bsmf="bsmf/0037-9484/"
 777 # "Bulletin de la Soci\`{e}t\`{e} Math\`{e}matique de France"}
 778 @string{craspi="craspi/0764-4442/"
 779 # "Comptes Rendus de l'Acad\`{e}mie des Sciences (Paris)"
 780 # "S\`{e}rie I Math\`{e}matique"}
 781 @string{cjm="cjm/0008-414X/Canadian Journal of Mathematics"}
 782 @string{cmb="cmb/0008-4395/"
 783 # "Canadian Mathematical Bulletin"}
 784 @string{colloqmath="colloqmath/0010-1354/"
 785 # "Colloquium Mathematicum"}
 786 @string{commalgebra="commalgebra/0092-7872/"
 787 # "Communications in Algebra"}
 788 @string{cmp="cmp/0010-3616/"
 789 # "Communications in Mathematical Physics"}
 790 @string{cpam="cpam/0010-3640/"
 791 # "Communications on Pure and Applied Mathematics"}
 792 @string{cmh="cmh/0010-2571/"
 793 # "Commentarii Mathematici Helvetici"}
 794 @string{cvta="cvta/0278-1077/"
 795 # "Complex Variables"}
 796 @string{compositiomath="compositiomath/0010-437X/"
 797 # "Compositio Mathematica"}
 798 @string{c="c/0010-485X/Computing"}
 799 @string{constrapprox="constrapprox/0176-4276/"
 800 # "Constructive Approximation"}
 801 @string{dmj="dmj/0012-7094/"
 802 # "Duke Mathematical Journal"}
 803 @string{eraams="eraams/1079-6762/"
 804 # "Electronic Research Announcements"
 805 # " of the American Mathematical Society"}
 806 @string{etds="etds/0143-3857/"
 807 # "Ergodic Theory and Dynamical Systems"}
 808 @string{fm="fm/0016-2736/"
 809 # "Fundamenta Mathematicae"}
 810 @string{gd="gd/0046-5755/"
 811 # "Geometriae Dedicata"}
 812 @string{illinoisjmath="illinoisjmath/0019-2082/"
 813 # "Illinois Journal of Mathematics"}
 814 @string{iumj="iumj/0022-2518/"
 815 # "Indiana University Mathematics"}
 816 @string{im="im/0020-9910/"
 817 # "Inventiones Mathematicae"}
 818 @string{israeljmath="israeljmath/0021-2172/"
 819 # "Israel Journal of Mathematics"}
 820 @string{iansm="iansm/??/"
 821 # "Izvestiya Akademii Nauk SSSR. Seriya Matematicheskaya"}
 822 @string{ja="ja/0021-8693/"
 823 # "Journal of Algebra"}
 824 @string{jams="jams/0894-0347/"
 825 # "Journal of the American Mathematical Society"}

```

826 @string{jam="jam/0021-7670/"
827   #"Journal d'Analyse Math\`{e}matique"}
828 @string{jamm="jamm/0021-8928/"
829   #"Journal of Applied Mathematics and Mechanics"}
830 @string{jat="jat/0021-9045/"
831   #"Journal of Approximation Theory"}
832 @string{jamsa="jamsa/0263-6115/"
833   #"Journal of the Australian Mathematical Society"}
834 @string{jcta="jcta/0097-3165/"
835   #"Journal of Combinatorial Theory"}
836 @string{jde="jde/0022-0396/"
837   #"Journal of Differential Equations"}
838 @string{jdg="jdg/0022-040X/Journal of Differential Geometry"}
839 @string{jfa="jfa/0022-1236/"
840   #"Journal of Functional Analysis"}
841 @string{jllms2="jllms2/0024-6107/"
842   #"Journal of the London Mathematical Society"}
843 @string{jmaa="jmaa/0022-247X/"
844   #"Journal of Mathematical Analysis and Applications"}
845 @string{jmp="jmp/0022-2488/"
846   #"Journal of Mathematical Physics"}
847 @string{jmsj="jmsj/0025-5645/"
848   #"Journal of the Mathematical Society of Japan"}
849 @string{jnt="jnt/0022-314X/Journal of Number Theory"}
850 @string{jot="jot/0379-4024/"
851   #"Journal of Operator Theory"}
852 @string{jpa="jpa/0022-4049/"
853   #"Journal of Pure and Applied Algebra"}
854 % Braces instead of quotes, to hide the embedded \" control sequence
855 @string{jram={jram/0075-4102/}
856   #{Journal f\"{u}r die Reine und Angewandte Mathematik}}
857 @string{jsl="jsl/0022-4812/"
858   #"The Journal of Symbolic Logic"}
859 @string{laa="laa/0024-3795/"
860   #"Linear Algebra and its Applications"}
861 @string{mm="mm/0025-2611/"
862   #"Manuscripta Mathematica"}
863 @string{matzametki="matzametki/0025-567X/Matematicheskie Zametki"}
864 @string{ma="ma/0025-5831/"
865   #"Mathematische Annalen"}
866 @string{mc="mc/0025-5718/"
867   #"Mathematics of Computation"}
868 @string{mj="mj/0025-5513/"
869   #"Mathematica Japonica"}
870 @string{mn="mn/0025-584X/Mathematische Nachrichten"}
871 @string{mpcps="mpcps/0305-0041/"
872   #"Mathematical Proceedings of the"}
873   #" Cambridge Philosophical Society"}
874 @string{mrl="mrl/1073-2780/"
875   #"Mathematical Research Letters"}
876 @string{ms="ms/0025-5521/"
877   #"Mathematica Scandinavica"}
878 @string{mathz="mathz/0025-5874/"
879   #"Mathematische Zeitschrift"}

```

```

880 @string{m="m/0025-5793/"
881   #"Mathematika"}
882 @string{mmj="mmj/0026-2285/"
883   #"The Michigan Mathematical Journal"}
884 @string{nmj="nmj/0027-7630/"
885   #"Nagoya Mathematical Journal"}
886 @string{na="na/0362-546X/Nonlinear Analysis"}
887 @string{nm="nm/0029-599X/Numerische Mathematik"}
888 @string{ojm="ojm/0030-6126/"
889   #"Osaka Journal of Mathematics"}
890 @string{pjm="pjm/0030-8730/"
891   #"Pacific Journal of Mathematics"}
892 @string{pams="pams/0002-9939/"
893   #"Proceedings of the American Mathematical Society"}
894 @string{pems2="pems2/0013-0915/"
895   #"Proceedings of the Edinburgh Mathematical Society"}
896 @string{pjaa="pjaa/0386-2194/"
897   #"Proceedings of the Japan Academy"}
898 @string{plms3="plms3/0024-6115/"
899   #"Proceedings of the London Mathematical Society"}
900 @string{qjmo2="qjmo2/0033-5606/"
901   #"The Quarterly Journal of Mathematics"}
902 @string{rmi="rmi/0213-2230/"
903   #"Revista Matem\`{a}tica Iberoamericana"}
904 @string{rmjm="rmjm/0035-7596/"
905   #"Rocky Mountain Journal of Mathematics"}
906 @string{sjam="sjam/0036-1399/"
907   #"SIAM Journal on Applied Mathematics"}
908 @string{sjc="sjc/0097-5397/"
909   #"SIAM Journal on Computing"}
910 @string{sjma="sjma/0036-1410/"
911   #"SIAM Journal on Mathematical Analysis"}
912 @string{sjna="sjna/0036-1429/"
913   #"SIAM Journal on Numerical Analysis"}
914 @string{sr="sr/0036-1445/"
915   #"SIAM Review"}
916 @string{sm="sm/0039-3223/"
917   #"Studia Mathematica"}
918 @string{tjm="tjm/0387-3870/"
919   #"Tokyo Journal of Mathematics"}
920 @string{ta="ta/0166-8641/"
921   #"Topology and its Applications"}
922 @string{t="t/0040-9383/"
923   #"Topology"}
924 @string{tams="tams/0002-9947/"
925   #"Transactions of the American Mathematical Society"}
926 </abbrevs>

```

14 To Do

—When there is no author or editor info to create a label from: Use the initial letters of all the words in the title. Plus the year. Maybe year first.

- Add `linelength` and `indent` options to control the output?
- Discuss `apacite` "originalfoo" fields etc.
- Discuss Beebe's work.

- Add bibliography.
 - Add AUDIO, VIDEO entry types.
 - Add a field for MR subject classification numbers and the like?
 - Write a converter to convert from typical .bib data to preferred form.
- Have to go via perl or something to capture STRING defs and PREAMBLE and interspersed comments? Or emacs lisp? Provide .bst version as fallback.
- Alpha labels: add numbers after sorting? subscript option? L3Y2a option.
 - Test multi-author articles with ten-plus authors. What happens with the alpha labels then?
 - For url field especially, check length and prebreak by hand instead of relying on BibT_EX’s slapdash stick-in-a-percent-at-column-72 regardless of where it happens to fall.
 - How about a general ‘contributor’ field.
 - Test with a wimpy older version of BibT_EX.

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

- | | | |
|---|--|---|
| <p>Symbols</p> <p>\ " . . . 85, 132, 854, 856</p> <p>\ ' 766, 777,
779, 780, 827, 903</p> <p>.bib 3</p> <p>.bst 1</p> <p>A</p> <p>alphabetic option . 1, 1</p> <p>amsrefs package . . .
. 1, 1, 1,
1, 1, 16, 16, 16, 16</p> <p>amsrn BIBT_EX style . . . 1</p> <p>amsxport 1, 11</p> <p>amsxport BIBT_EX style
. 1, 16</p> <p>append.ndash.and.skip.hyphens
end.bin 6</p> <p>append.ndash.and.skip.hyphens
eprint
\etalchar 120</p> <p>article 364</p> <p>assemble.date.field 274</p> <p>author 3</p> <p>author.editor.sort 635</p> <p>author.organization.sort
. 651</p> <p>author.sort 623</p> <p>B</p> <p>begin.bin 726</p> <p>\bib 348, 731</p> <p>book 412</p> <p>booklet 439</p> <p>btxbst.doc 1</p> <p>btxdoc.tex 1</p> <p>btxhak.tex 1</p> | <p>C</p> <p>chop.word 462</p> <p>citation-order op-
tion 1</p> <p>conference 2</p> <p>copy.two.chars 4</p> <p>copy.two.chars 45</p> <p>crossref 3, 3</p> <p>current.char 44</p> <p>D</p> <p>date 10</p> <p>E</p> <p>editor 2</p> <p>editor.organization.sort
. 667</p> <p>end.bin 736</p> <p>eprint
\etalchar 494, 505</p> <p>F</p> <p>field.or.null 112</p> <p>fin.entry 360</p> <p>fin.field 167</p> <p>format.lab.names 480</p> <p>format.name 9, 10</p> <p>format.name 253</p> <p>format.title 157</p> <p>fromwhere 3</p> <p>I</p> <p>inbook 409</p> <p>incollection 410</p> <p>incr.ptr 42</p> <p>inproceedings 13</p> <p>inproceedings 386, 411</p> | <p>institution 2, 2, 2</p> <p>is.digit 47</p> <p>is.letter 4</p> <p>is.letter 70</p> <p>is.lowercase.letter 56</p> <p>is.uppercase.letter 63</p> <p>L</p> <p>lpad 5, 6</p> <p>lpad 100</p> <p>M</p> <p>manual 441</p> <p>mastersthesis 442</p> <p>meeting 2</p> <p>missing.or.empty 118</p> <p>missing.warning 143</p> <p>month 2</p> <p>\MR 210</p> <p>N</p> <p>n.dashify 6</p> <p>n.dashify 127</p> <p>\ndash 6, 121</p> <p>O</p> <p>optional.edition.field
. 242</p> <p>optional.field 180</p> <p>optional.journal.field
. 294</p> <p>optional.mr.field 204</p> <p>optional.name.field 10</p> <p>optional.name.field 255</p> <p>optional.pages.field
. 194</p> <p>optional.title.field
. 187</p> |
|---|--|---|

- optional.type.field [325](#)
 organization [2, 2](#)
- P**
- pages [6](#)
 presort [683](#)
- R**
- READ [4](#)
 remove.ordinal.suffix
 [214](#)
 review [2, 12](#)
- S**
- s [6, 6, 6, 21, 21](#)
 shortalphabetic op-
 tion [1](#)
- skip.accent [5](#)
 skip.accent [77](#)
 sort.format.names [598](#)
 sort.format.title [470](#)
 sort.warning [620](#)
 sortify [96](#)
 start.entry [346](#)
 start.field [164](#)
 string.length [7](#)
 string.length [147](#)
 strip.label [78](#)
- T**
- translator [2](#)
 trueauthor [3](#)
- U**
- url.note.status.review
 [339](#)
- W**
- write.field [169](#)
 write.with.newlines [21](#)
 write.with.newlines [711](#)
- X**
- xid [3](#)
- Y**
- y2k option [1, 1](#)
 year [2, 2](#)