

The `bitset` package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/09/28 v1.0

Abstract

This package defines and implements the data type bit set, a vector of bits. The size of the vector may grow dynamically. Individual bits can be manipulated.

Contents

1	Documentation	3
1.1	Introduction	3
1.2	Glossary	3
1.3	Design principles	4
1.4	Operator overview	5
1.5	Package loading	5
1.6	Operators	5
1.6.1	Miscellaneous	6
1.6.2	Import	6
1.6.3	Export	6
1.6.4	Logical operators	7
1.6.5	Shifting	7
1.6.6	Bit manipulation	7
1.6.7	Bit retrieval	8
1.6.8	Bit set properties	8
1.6.9	Queries	8
2	Implementation	9
2.1	Reload check and package identification	9
2.2	Catcodes	10
2.3	Package loading	11
2.4	Help macros	11
2.4.1	Number constant	11
2.4.2	General basic macros	11
2.4.3	Tail recursion	12
2.4.4	Check macros	12
2.5	Miscellaneous	13
2.6	Import	13
2.6.1	From binary number	13
2.6.2	From octal/hex number	14
2.6.3	From decimal number	16
2.7	Export	17
2.7.1	To binary number	17
2.7.2	To octal/hexadecimal number	18
2.7.3	To decimal number	20
2.8	Logical operators	22
2.8.1	<code>\bitsetAnd</code>	22

2.8.2	<code>\bitsetAndNot</code>	23
2.8.3	<code>\bitsetOr</code>	24
2.8.4	<code>\bitsetXor</code>	25
2.8.5	Shifting	26
2.8.6	<code>\bitsetShiftLeft</code>	26
2.8.7	<code>\bitsetShiftRight</code>	26
2.9	Bit manipulation	27
2.9.1	Clear operation	28
2.9.2	Set operation	28
2.9.3	Flip operation	29
2.9.4	Range operators	30
2.10	Bit retrieval	32
2.10.1	<code>\bitsetGet</code>	32
2.10.2	<code>\bitsetNextClearBit</code> , <code>\bitsetNextSetBit</code>	32
2.10.3	<code>\bitsetGetSetBitList</code>	35
2.11	Bit set properties	35
2.12	Queries	36
3	Test	38
3.1	Catcode checks for loading	38
3.2	Macro tests	39
3.2.1	Preamble	39
3.2.2	Time	39
3.2.3	Detection of unwanted space	40
3.2.4	Test macros	40
3.2.5	Test sets	41
4	Installation	56
4.1	Download	56
4.2	Bundle installation	57
4.3	Package installation	57
4.4	Refresh file name databases	57
4.5	Some details for the interested	57
5	History	58
	[2007/09/28 v1.0]	58
6	Index	58

1 Documentation

1.1 Introduction

Annotations in the PDF format know entries whose values are integers. These numbers are interpreted as set of flags specifying properties. For example, annotation dictionaries can have a key `/F`. The bits of its integer value are interpreted the following way:

Bit position	Property name
1	Invisible
2	Hidden
3	Print
4	NoZoom
5	NoRotate
6	NoView
7	ReadOnly
...	...

Now, let's see how these values are set in package `hyperref` before it uses this package (before v6.77a):

```
\ifFld@hidden /F 6\else /F 4\fi
```

Where are the other flags? The following example for key `/Ff` in a widget annotation supports at least three properties:

```
\ifFld@multiline
  \ifFld@readonly /Ff 4097\else /Ff 4096\fi
\else
  \ifFld@password
    \ifFld@readonly /Ff 8193\else /Ff 8192\fi
  \else
    \ifFld@readonly /Ff 1\fi
\fi
\fi
```

But you see the point. It would be a nightmare to continue this way in supporting the missing flag settings. This kind of integers may have up to 32 bits.

Therefore I wanted a data structure for setting and clearing individual bits. Also it should provide an export as decimal number. The snippets above are executed in expansion contexts without \TeX 's stomach commands. It would be convenient to have an expandable conversion from the data structure to the integer that gets written to the PDF file.

This package `bitset` implements such a data structure. The interface is quite close to Java's class `BitSet` in order not to learn too many interfaces for the same kind of data structure.

1.2 Glossary

Bit set: A bit set is a vector of bits or flags. The vector size is unlimited and grows dynamically. An undefined bit set is treated as bit set where all bits are cleared.

Bit sets are addressed by name. A name should consist of letters or digits. Technically it must survive `\csname`, see \LaTeX 's environment names for other names with such a constraint. Package `babel`'s shorthands are not supported due to technical reasons. Shorthand support would break expandable operations.

Size: A size of a bit set is the number of bits in use. It's the number of the highest index, incremented by one. Sizes are in the range 0 up to 2147483647, the highest number supported by \TeX .

Index: Bit positions in a bit set are addressed by an index number. The bit vector is zero based. The first and least significant bit is addressed by index 0 and the highest possible bit by 2147483646.

Bit: A bit is encoded as 0 for cleared/disabled or 1 for set/enabled.

1.3 Design principles

Name conventions: To avoid conflicts with existing macro names, the operations are prefixed by the package name.

Zero based indexes: The first bit is addressed by zero. (Convention of array indexing in C, Java, ...)

Unlimited size: There is no restriction on the size of a bit set other than usual memory limitations. `\bitsetSetDec` and `\bitsetGetDec` transparently switch to package `bigintcalc` if the numbers get too large for `TEX`'s number limit.

Expandibility: Any operation that does not change the bit set is expandable. And all operations that extract or calculate some result do this in exact two expansion steps. For example, a macro `\Macro` wants a bit set as decimal number. But the argument must be a plain number without macros. Thus you could prefix `\bitsetGetDec` with `\number`. However this won't work for bit sets with 31 or more bits because of `TEX`'s number limit of $2^{31} - 1$. then just hit the operator with two `\expandafter`:

```
\expandafter\expandafter\expandafter
\Macro\bitsetGetDec{foo}
```

`\bitsetGetDec` is hit first by the third `\expandafter` and then by the second one.

Format independence: This package is written as `LATEX` package, but it does not depend on `LATEX`. It will also work for other formats such as plain-`TEX`.

Independence from `TEX` engines: Vanilla `TEX` is all you need. Calculations are delegated to packages `intcalc` and `bigintcalc`. They don't need any special features, but they will switch to a little more efficient implementation if features such as `\numexpr` are available.

Numeric arguments: Anything that is accepted by `\number`. If ε -`TEX` is detected, also expressions for `\numexpr` are supported. The only exception so far is the number for `\bitsetSetDec`. The number might be too large for `\number` or `\numexpr`.

Error messages: In expandable contexts, only a limited set of `TEX` primitive commands work as expected. So called stomach commands behave like `\relax` and don't get expanded or executed. Unhappily also the error commands belong to this category. The expandable operations will throw an unknown control sequence instead to get `TEX`'s and user's attention. The name of these control sequences starts with `\BitSetError:` with the type of error after the colon.

1.4 Operator overview

Miscellaneous (section 1.6.1)

<code>\bitsetReset</code>	$\langle BitSet \rangle$
<code>\bitsetLet</code>	$\langle BitSet A \rangle \langle BitSet B \rangle$

Import (section 1.6.2)

<code>\bitsetSetBin, \bitsetSetOct, \bitsetSetHex</code>	$\langle BitSet \rangle \langle Value \rangle$
<code>\bitsetSetDec</code>	$\langle BitSet \rangle \langle Value \rangle$

Export^a (section 1.6.3)

<code>\bitsetGetBin, \bitsetGetOct, \bitsetGetHex</code>	$\langle BitSet \rangle \langle MinSize \rangle$
<code>\bitsetGetDec</code>	$\langle BitSet \rangle$

Logical operators (section 1.6.4)

<code>\bitsetAnd, \bitsetAndNot</code>	$\langle BitSet A \rangle \langle BitSet B \rangle$
<code>\bitsetOr, \bitsetXor</code>	$\langle BitSet A \rangle \langle BitSet B \rangle$

Shifting (section 1.6.5)

<code>\bitsetShiftLeft, \bitsetShiftRight</code>	$\langle BitSet \rangle \langle ShiftAmount \rangle$
--	--

Bit manipulation (section 1.6.6)

<code>\bitsetClear, \bitsetSet, \bitsetFlip</code>	$\langle BitSet \rangle \langle Index \rangle$
<code>\bitsetSetValue</code>	$\langle BitSet \rangle \langle Index \rangle \langle Value \rangle$
<code>\bitsetClearRange, \bitsetSetRange, \bitsetFlipRange</code>	$\langle BitSet \rangle \langle IndexFrom \rangle \langle IndexTo \rangle$
<code>\bitsetSetValueRange</code>	$\langle BitSet \rangle \langle IndexFrom \rangle \langle IndexTo \rangle$

Bit retrieval^a (section 1.6.7)

<code>\bitsetGet</code>	$\langle BitSet \rangle \langle Index \rangle$
<code>\bitsetNextClearBit, \bitsetNextSetBit</code>	$\langle BitSet \rangle \langle Index \rangle$
<code>\bitsetGetSetBitList</code>	$\langle BitSet \rangle$

Bit set properties (section 1.6.8)

<code>\bitsetSize, \bitsetCardinality</code>	$\langle BitSet \rangle$
--	--------------------------

Queries^b (section 1.6.9)

<code>\bitsetIsDefined, \bitsetIsEmpty</code>	$\langle BitSet \rangle \langle Then \rangle \langle Else \rangle$
<code>\bitsetEquals, \bitsetIntersects</code>	$\langle BitSet A \rangle \langle BitSet B \rangle \langle Then \rangle \langle Else \rangle$
<code>\bitsetQuery</code>	$\langle BitSet \rangle \langle Index \rangle \langle Then \rangle \langle Else \rangle$

^aMacros are expandable, full expansion by two steps.

^bMacros are expandable.

1.5 Package loading

The package can be used as normal L^AT_EX package:

```
\usepackage{bitset}
```

Also plain-T_EX is supported:

```
\input bitset.sty\relax
```

1.6 Operators

The following macros work on and with bit sets. A bit set $\langle BitSet \rangle$ is represented by a name. The should consist of letters and digits. Technically it must survive `\csname`. It is the same constraint that must be satisfied by label or environment names in L^AT_EX.

However active characters that are shorthands of package `babel` are not supported. Support for shorthands works by an assignment. But many operators such

as `\bitsetGetDec` must be usable in expandable contexts. There assignments will not be executed in the best case or they will cause errors.

The bits in a bit set are addressed by non-negative integers starting from zero. Thus negative index numbers cause an error message. Because index numbers are \TeX numbers. The largest index is 2147483647. But in practice memory limits and patience limits will be very likely reached much before.

1.6.1 Miscellaneous

There isn't a separate operation for bit set creation. For simplicity an undefined bit set is treated as bit set with all bits cleared.

`\bitsetReset {\langle BitSet \rangle}`

Macro `\bitsetReset` clears all bits. The result is an empty bit set. It may also be used as replacement for an operation “new”, because an undefined bit set is defined afterwards.

`\bitsetLet {\langle BitSet A \rangle} {\langle BitSet B \rangle}`

Macro `\bitsetLet` performs a simple assignment similar to \TeX 's `\let`. After the operation `\langle BitSet A \rangle` has the same value as `\langle BitSet B \rangle`. If `\langle BitSet B \rangle` is undefined, then `\langle BitSet A \rangle` will be the empty bit set.

Note: If `\langle BitSet A \rangle` exists, it will be overwritten.

1.6.2 Import

`\bitsetSetBin {\langle BitSet \rangle} {\langle BinaryNumber \rangle}`
`\bitsetSetOct {\langle BitSet \rangle} {\langle OctalNumber \rangle}`
`\bitsetSetHex {\langle BitSet \rangle} {\langle HexadecimalNumber \rangle}`

The numbers are interpreted as bit vectors and the flags in the bit `\langle BitSet \rangle` set are set accordingly. These numeric arguments are the only arguments where spaces are allowed. Then the numbers are easier to read.

`\bitsetSetDec {\langle BitSet \rangle} {\langle DecimalNumber \rangle}`

Macro `\bitsetSetDec` uses `\langle DecimalNumber \rangle` to set the bit set `\langle BitSet \rangle`. The numeric argument must expand to a plain number consisting of decimal digits without command tokens or spaces. Internally this argument is expanded only. It cannot be passed to `\number` or `\numexpr`, because the number may be too large for them. However `\number` or `\the\numexpr` may be used explicitly. This also helps for unexpandable number command tokens or registers (`\z@`, `\@ne`, `\count@`, ...). Also \LaTeX ' `\value` needs prefixing:

`\bitsetSetDec{foo}{\number\value{bar}}`

1.6.3 Export

`\bitsetGetBin {\langle BitSet \rangle} {\langle MinSize \rangle}`
`\bitsetGetOct {\langle BitSet \rangle} {\langle MinSize \rangle}`
`\bitsetGetHex {\langle BitSet \rangle} {\langle MinSize \rangle}`

These macros returns the bit set as binary, octal or hexadecimal number. If the bit size is smaller than `\langle MinSize \rangle` the gap is filled with leading zeros. Example:

```

\bitsetReset{abc}
\bitsetSet{abc}{2}
\bitsetGetBin{abc}{8} → 00000100
\bitsetSet{abc}{5}\bitsetSet{abc}{7}
\bitsetGetHex{abc}{16} → 00A2

```

Macro `\bitsetGetHex` uses the uppercase letters A to F. The catcode of the letters is one of 11 (letter) or 12 (other).

`\bitsetGetDec {⟨BitSet⟩}`

Macro `\bitsetGetDec` returns the bit set $\langle BitSet \rangle$ as decimal number. The returned number can be larger than T_EX's number limit of $2^{31} - 1$.

1.6.4 Logical operators

`\bitsetAnd {⟨BitSet A⟩} {⟨BitSet B⟩}`

$$A_{\text{new}} := A_{\text{old}} \text{ and } B \quad (\forall \text{ bits})$$

`\bitsetAndNot {⟨BitSet A⟩} {⟨BitSet B⟩}`

$$A_{\text{new}} := A_{\text{old}} \text{ and (not } B) \quad (\forall \text{ bits})$$

`\bitsetOr {⟨BitSet A⟩} {⟨BitSet B⟩}`

$$A_{\text{new}} := A_{\text{old}} \text{ or } B \quad (\forall \text{ bits})$$

`\bitsetXor {⟨BitSet A⟩} {⟨BitSet B⟩}`

$$A_{\text{new}} := A_{\text{old}} \text{ xor } B \quad (\forall \text{ bits})$$

1.6.5 Shifting

`\bitsetShiftLeft {⟨BitSet⟩} {⟨ShiftAmount⟩}`
`\bitsetShiftRight {⟨BitSet⟩} {⟨ShiftAmount⟩}`

A left shift by one is a multiplication by two, thus left shifting moves the flags to higher positions. The new created low positions are filled by zeros.

A right shift is the opposite, dividing by two, moving the bits to lower positions. The number will become smaller, the lowest bits are lost.

If the $\langle ShiftAmount \rangle$ is negative, it reverts the meaning of the shift operation. A left shift becomes a right shift. A $\langle ShiftAmount \rangle$ of zero is ignored.

1.6.6 Bit manipulation

`\bitsetClear {⟨BitSet⟩} {⟨Index⟩}`
`\bitsetSet {⟨BitSet⟩} {⟨Index⟩}`
`\bitsetFlip {⟨BitSet⟩} {⟨Index⟩}`

This macros manipulate a single bit in $\langle BitSet \rangle$ addressed by `\Index`. Macro `\bitsetClear` disables the bit, `\bitsetSet` enables it and `\bitsetFlip` reverts the current setting of the bit.

`\bitsetSetValue {⟨BitSet⟩} {⟨Index⟩} {⟨Bit⟩}`

Macro `\bitsetSetValue` puts bit `⟨Bit⟩` at position `⟨Index⟩` in bit set `⟨BitSet⟩`. `⟨Bit⟩` must be a valid T_EX number equals to zero (disabled/cleared) or one (enabled/set).

1.6.7 Bit retrieval

`\bitsetGet {⟨BitSet⟩} {⟨Index⟩}`

Macro `\bitsetGet` extracts the status of the bit at position `⟨Index⟩` in bit set `⟨BitSet⟩`. Digit 1 is returned if the bit is set/enabled. If the bit is cleared/disabled and in cases of an undefined bitset or an index number out of range the return value is 0.

`\bitsetNextClearBit {⟨BitSet⟩} {⟨Index⟩}`

Starting at position `⟨Index⟩` (inclusive) the bits are inspected. The first position without a set bit is returned. Possible results are decimal numbers: `⟨Index⟩`, `⟨Index⟩ + 1`, ..., (∞)

`\bitsetNextSetBit {⟨BitSet⟩} {⟨Index⟩}`

Starting at position `⟨Index⟩` (inclusive) the bits are inspected and the index position of the first found set bit is returned. If there isn't such a bit, then the result is -1. In summary possible results are decimal numbers: -1, `⟨Index⟩`, `⟨Index⟩ + 1`, ..., (∞)

`\bitsetGetSetBitList {⟨BitSet⟩}`

Macro `\bitsetGetSetBitList` is an application for `\bitsetNextSetBit`. The set bits are iterated and returned as comma separated list of index positions in increasing order. The list is empty in case of an empty bit set.

1.6.8 Bit set properties

`\bitsetSize {⟨BitSet⟩}`

Macro `\bitsetSize` returns number of bits in use. It is the same as the index number of the highest set/enabled bit incremented by one.

`\bitsetCardinality {⟨BitSet⟩}`

Macro `\bitsetCardinality` counts the number of set/enabled bits.

1.6.9 Queries

Also the query procedures are expandable. They ask for a piece of information about a bit set and execute code depending on the answer.

`\bitsetIsDefined {⟨BitSet⟩} {⟨Then⟩} {⟨Else⟩}`

If the bit set with the name `⟨BitSet⟩` exists the code given in `⟨Then⟩` is executed, otherwise `⟨Else⟩` is used.

`\bitsetIsEmpty {⟨BitSet⟩} {⟨Then⟩} {⟨Else⟩}`

If the bit set $\langle BitSet \rangle$ exists and at least one bit is set/enabled, the code in $\langle Then \rangle$ is executed, $\langle Else \rangle$ otherwise.

`\bitsetEquals {⟨BitSet A⟩} {⟨BitSet B⟩} {⟨Then⟩} {⟨Else⟩}`

Both bit sets are equal if and only if either both are undefined or both are defined and represents the same bit values at the same positions. Thus this definition is reflexive, symmetric, and transitive, enough for an equivalent relation.

`\bitsetIntersects {⟨BitSet A⟩} {⟨BitSet B⟩} {⟨Then⟩} {⟨Else⟩}`

If and only if $\langle BitSet A \rangle$ and $\langle BitSet B \rangle$ have at least one bit at the same position that is set, then code part $\langle Then \rangle$ is executed.

`\bitsetQuery {⟨BitSet⟩} {⟨Index⟩} {⟨Then⟩} {⟨Else⟩}`

It's just a wrapper for `\bitsetGet`. If the bit at position $\langle Index \rangle$ is enabled, code $\langle Then \rangle$ is called.

2 Implementation

The internal format of a bit set is quite simple, a sequence of digits 0 and 1. The least significant bit is left. A bit set without any flag set is encoded by 0. Also undefined bit sets are treated that way. After the highest bit that is set there are no further zeroes. A regular expression of valid bit sets values:

```
0|[01]*1
1 ⟨*package⟩
```

2.1 Reload check and package identification

Reload check, especially if the package is not used with L^AT_EX.

```

2 \begingroup
3 \catcode44 12 % ,
4 \catcode45 12 % -
5 \catcode46 12 % .
6 \catcode58 12 % :
7 \catcode64 11 % @
8 \expandafter\let\expandafter\x\csname ver@bitset.sty\endcsname
9 \ifcase 0%
10 \ifx\x\relax % plain
11 \else
12 \ifx\x\empty % LaTeX
13 \else
14 1%
15 \fi
16 \fi
17 \else
18 \catcode35 6 % #
19 \catcode123 1 % {
20 \catcode125 2 % }
21 \expandafter\ifx\csname PackageInfo\endcsname\relax
22 \def\x#1#2{%
23 \immediate\write-1{Package #1 Info: #2.}%
24 }%
```

```

25 \else
26 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27 \fi
28 \x{bitset}{The package is already loaded}%
29 \endgroup
30 \expandafter\endinput
31 \fi
32 \endgroup

```

Package identification:

```

33 \begingroup
34 \catcode35 6 % #
35 \catcode40 12 % (
36 \catcode41 12 % )
37 \catcode44 12 % ,
38 \catcode45 12 % -
39 \catcode46 12 % .
40 \catcode47 12 % /
41 \catcode58 12 % :
42 \catcode64 11 % @
43 \catcode123 1 % {
44 \catcode125 2 % }
45 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
46 \def\x#1#2#3[#4]{\endgroup
47 \immediate\write-1{Package: #3 #4}%
48 \xdef#1{#4}%
49 }%
50 \else
51 \def\x#1#2[#3]{\endgroup
52 #2[{#3}]%
53 \ifx#1\relax
54 \xdef#1{#3}%
55 \fi
56 }%
57 \fi
58 \expandafter\x\csname ver@bitset.sty\endcsname
59 \ProvidesPackage{bitset}%
60 [2007/09/28 v1.0 Data type bit set (H0)]

```

2.2 Catcodes

```

61 \begingroup
62 \catcode123 1 % {
63 \catcode125 2 % }
64 \def\x{\endgroup
65 \expandafter\edef\csname BitSet@AtEnd\endcsname{%
66 \catcode35 \the\catcode35\relax
67 \catcode64 \the\catcode64\relax
68 \catcode123 \the\catcode123\relax
69 \catcode125 \the\catcode125\relax
70 }%
71 }%
72 \x
73 \catcode35 6 % #
74 \catcode64 11 % @
75 \catcode123 1 % {
76 \catcode125 2 % }
77 \def\TMP@EnsureCode#1#2{%
78 \edef\BitSet@AtEnd{%
79 \BitSet@AtEnd
80 \catcode#1 \the\catcode#1\relax
81 }%
82 \catcode#1 #2\relax

```

```

83 }
84 \TMP@EnsureCode{33}{12}% !
85 \TMP@EnsureCode{39}{12}% '
86 \TMP@EnsureCode{40}{12}% (
87 \TMP@EnsureCode{41}{12}% )
88 \TMP@EnsureCode{42}{12}% *
89 \TMP@EnsureCode{43}{12}% +
90 \TMP@EnsureCode{44}{12}% ,
91 \TMP@EnsureCode{45}{12}% -
92 \TMP@EnsureCode{46}{12}% .
93 \TMP@EnsureCode{47}{12}% /
94 \TMP@EnsureCode{58}{11}% : (letter!)
95 \TMP@EnsureCode{60}{12}% <
96 \TMP@EnsureCode{61}{12}% =
97 \TMP@EnsureCode{62}{12}% >
98 \TMP@EnsureCode{63}{14}% ? (comment!)
99 \TMP@EnsureCode{96}{12}% '
100 \begingroup\expandafter\expandafter\expandafter\endgroup
101 \expandafter\ifx\csname BitSet@TestMode\endcsname\relax
102 \else
103   \catcode63=9 % ? (ignore)
104 \fi
105 ? \let\BitSet@@TestMode\BitSet@TestMode

```

2.3 Package loading

```

106 \begingroup\expandafter\expandafter\expandafter\endgroup
107 \expandafter\ifx\csname RequirePackage\endcsname\relax
108   \input infwarerr.sty\relax
109   \input intcalc.sty\relax
110   \input bigintcalc.sty\relax
111 \else
112   \RequirePackage{infwarerr}[2007/09/09]%
113   \RequirePackage{intcalc}[2007/09/27]%
114   \RequirePackage{bigintcalc}[2007/09/27]%
115 \fi

```

2.4 Help macros

2.4.1 Number constant

```

\BitSet@MaxSize
116 \def\BitSet@MaxSize{2147483647}%

```

2.4.2 General basic macros

```

\BitSet@Empty
117 \def\BitSet@Empty{}

\BitSet@FirstOfOne
118 \def\BitSet@FirstOfOne#1{#1}

\BitSet@Gobble
119 \def\BitSet@Gobble#1{}

\BitSet@FirstOfTwo
120 \def\BitSet@FirstOfTwo#1#2{#1}

\BitSet@SecondOfTwo
121 \def\BitSet@SecondOfTwo#1#2{#2}

\BitSet@Space
122 \def\BitSet@Space{ }

```

\BitSet@ZapSpace

```
123 \def\BitSet@ZapSpace#1 #2{%
124   #1%
125   \ifx\BitSet@Empty#2%
126   \else
127     \expandafter\BitSet@ZapSpace
128   \fi
129   #2%
130 }
```

2.4.3 Tail recursion

\BitSet@Fi

```
131 \let\BitSet@Fi\fi
```

\BitSet@AfterFi

```
132 \def\BitSet@AfterFi#1#2\BitSet@Fi{\fi#1}
```

\BitSet@AfterFiFi

```
133 \def\BitSet@AfterFiFi#1#2\BitSet@Fi{\fi\fi#1}%
```

\BitSet@AfterFiFiFi

```
134 \def\BitSet@AfterFiFiFi#1#2\BitSet@Fi{\fi\fi\fi#1}%
```

2.4.4 Check macros

\BitSet@IfUndefined

```
135 \def\BitSet@IfUndefined#1{%
136   \expandafter\ifx\csname BS@#1\endcsname\relax
137     \expandafter\BitSet@FirstOfTwo
138   \else
139     \expandafter\BitSet@SecondOfTwo
140   \fi
141 }
```

\BitSet@CheckIndex

#1: continuation code
#2: BitSet
#3: Index

```
142 \def\BitSet@CheckIndex#1#2#3{%
143   \BitSet@IfUndefined{#2}{\bitsetReset{#2}}{}%
144   \expandafter\expandafter\expandafter\BitSet@@CheckIndex
145   \intcalcNum{#3}!%
146   {#2}{#1}%
147 }
```

\BitSet@@CheckIndex

#1: plain Index
#2: BitSet
#3: continuation code

```
148 \def\BitSet@@CheckIndex#1!#2#3{%
149   \ifnum#1<0 %
150     \BitSet@AfterFi{%
151       \@PackageError{bitset}{%
152         Invalid negative index (#1)%
153       }\@ehc
154     }%
155   \else
156     \BitSet@AfterFi{%
157       #3{#2}{#1}%
158     }%
159   \BitSet@Fi
160 }
```

2.5 Miscellaneous

`\bitsetReset`

```
161 \def\bitsetReset#1{%
162   \expandafter\def\csname BS@#1\endcsname{0}%
163 }
```

`\bitsetLet`

```
164 \def\bitsetLet#1#2{%
165   \BitSet@IfUndefined{#2}{%
166     \bitsetReset{#1}%
167   }{%
168     \expandafter\let\csname BS@#1\expandafter\endcsname
169                       \csname BS@#2\endcsname
170   }%
171 }
```

2.6 Import

2.6.1 From binary number

`\bitsetSetBin`

```
172 \def\bitsetSetBin#1#2{%
173   \edef\BitSet@Temp{#2}%
174   \edef\BitSet@Temp{%
175     \expandafter\expandafter\expandafter\BitSet@ZapSpace
176     \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
177   }%
178   \edef\BitSet@Temp{%
179     \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
180   }%
181   \ifx\BitSet@Temp\BitSet@Empty
182     \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
183   \else
184     \expandafter\edef\csname BS@#1\endcsname{%
185       \expandafter\BitSet@Reverse\BitSet@Temp!%
186     }%
187   \fi
188 }
```

`\BitSet@KillZeros`

```
189 \def\BitSet@KillZeros#1{%
190   \ifx#10%
191     \expandafter\BitSet@KillZeros
192   \else
193     #1%
194   \fi
195 }
```

`\BitSet@Reverse`

```
196 \def\BitSet@Reverse#1#2!{%
197   \ifx\#2\%
198     #1%
199   \else
200     \BitSet@AfterFi{%
201       \BitSet@Reverse#2!#1%
202     }%
203   \BitSet@Fi
204 }
```

2.6.2 From octal/hex number

\bitsetSetOct

```
205 \def\bitsetSetOct{%
206   \BitSet@SetOctHex\BitSet@FromFirstOct
207 }
```

\bitsetSetHex

```
208 \def\bitsetSetHex{%
209   \BitSet@SetOctHex\BitSet@FromFirstHex
210 }
```

\BitSet@SetOctHex

```
211 \def\BitSet@SetOctHex#1#2#3{%
212   \edef\BitSet@Temp{#3}%
213   \edef\BitSet@Temp{%
214     \expandafter\expandafter\expandafter\BitSet@ZapSpace
215     \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
216   }%
217   \edef\BitSet@Temp{%
218     \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
219   }%
220   \ifx\BitSet@Temp\BitSet@Empty
221     \expandafter\let\csname BS@#2\endcsname\BitSet@Zero
222   \else
223     \edef\BitSet@Temp{%
224       \expandafter#1\BitSet@Temp!%
225     }%
226     \ifx\BitSet@Temp\BitSet@Empty
227       \expandafter\let\csname BS@#2\endcsname\BitSet@Zero
228     \else
229       \expandafter\edef\csname BS@#2\endcsname{%
230         \expandafter\BitSet@Reverse\BitSet@Temp!%
231       }%
232     \fi
233   \fi
234 }
```

\BitSet@FromFirstOct

```
235 \def\BitSet@FromFirstOct#1{%
236   \ifx#1!%
237   \else
238     \ifcase#1 \BitSet@AfterFiFi\BitSet@FromFirstOct
239     \or 1%
240     \or 10%
241     \or 11%
242     \or 100%
243     \or 101%
244     \or 110%
245     \or 111%
246     \else \BitSetError:WrongOctalDigit%
247     \fi
248     \expandafter\BitSet@FromOct
249   \BitSet@Fi
250 }
```

\BitSet@FromOct

```
251 \def\BitSet@FromOct#1{%
252   \ifx#1!%
253   \else
254     \ifcase#1 000%
255     \or 001%
```

```

256     \or 010%
257     \or 011%
258     \or 100%
259     \or 101%
260     \or 110%
261     \or 111%
262     \else \BitSetError:WrongOctalDigit%
263     \fi
264     \expandafter\BitSet@FromOct
265 \fi
266 }

```

\BitSet@FromFirstHex

```

267 \def\BitSet@FromFirstHex#1{%
268   \ifx#1!%
269   \else
270     \ifx#10%
271       \BitSet@AfterFiFi\BitSet@FromFirstHex
272     \fi
273     \expandafter\ifx\csname BitSet@Hex#1\endcsname\relax
274     \BitSetError:InvalidHexDigit%
275   \else
276     \expandafter\expandafter\expandafter\BitSet@KillZeros
277     \csname BitSet@Hex#1\endcsname
278   \fi
279   \expandafter\BitSet@FromHex
280 \BitSet@Fi
281 }

```

\BitSet@FromHex

```

282 \def\BitSet@FromHex#1{%
283   \ifx#1!%
284   \else
285     \expandafter\ifx\csname BitSet@Hex#1\endcsname\relax
286     \BitSetError:InvalidHexDigit%
287   \else
288     \csname BitSet@Hex#1\endcsname
289   \fi
290   \expandafter\BitSet@FromHex
291 \fi
292 }

```

\BitSet@Hex[0..F]

```

293 \def\BitSet@Temp#1{%
294   \expandafter\def\csname BitSet@Hex#1\endcsname
295 }
296 \BitSet@Temp 0{0000}%
297 \BitSet@Temp 1{0001}%
298 \BitSet@Temp 2{0010}%
299 \BitSet@Temp 3{0011}%
300 \BitSet@Temp 4{0100}%
301 \BitSet@Temp 5{0101}%
302 \BitSet@Temp 6{0110}%
303 \BitSet@Temp 7{0111}%
304 \BitSet@Temp 8{1000}%
305 \BitSet@Temp 9{1001}%
306 \BitSet@Temp A{1010}%
307 \BitSet@Temp B{1011}%
308 \BitSet@Temp C{1100}%
309 \BitSet@Temp D{1101}%
310 \BitSet@Temp E{1110}%
311 \BitSet@Temp F{1111}%

```

```

312 \BitSet@Temp a{1010}%
313 \BitSet@Temp b{1011}%
314 \BitSet@Temp c{1100}%
315 \BitSet@Temp d{1101}%
316 \BitSet@Temp e{1110}%
317 \BitSet@Temp f{1111}%

```

2.6.3 From decimal number

\bitsetSetDec

```

318 \def\bitsetSetDec#1#2{%
319   \edef\BitSet@Temp{#2}%
320   \edef\BitSet@Temp{%
321     \expandafter\expandafter\expandafter\BitSet@ZapSpace
322     \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
323   }%
324   \edef\BitSet@Temp{%
325     \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
326   }%
327   \ifx\BitSet@Temp\BitSet@Empty
328     \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
329   \else
330     \ifcase\bigintcalcSgn{\BitSet@Temp} %
331       \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
332     \or
333       \ifnum\bigintcalcCmp\BitSet@Temp\BitSet@MaxSize>0 %
334         \expandafter\edef\csname BS@#1\endcsname{%
335           \expandafter\BitSet@SetDecBig\BitSet@Temp!%
336         }%
337       \else
338         \expandafter\edef\csname BS@#1\endcsname{%
339           \expandafter\BitSet@SetDec\BitSet@Temp!%
340         }%
341       \fi
342     \else
343       \@PackageError{bitset}{%
344         Bit sets cannot be negative%
345       }\@ehc
346     \fi
347   \fi
348 }

```

\BitSet@SetDecBig

```

349 \def\BitSet@SetDecBig#1#2#3#4#5#6#7#8#9!{%
350   \ifx\#9\%
351     \BitSet@SetDec#1#2#3#4#5#6#7#8!%
352   \else
353     \ifcase\BigIntCalcOdd#1#2#4#5#6#7#8#9! %
354       0%
355     \or
356       1%
357   ? \else\BitSetError:ThisCannotHappen%
358     \fi
359     \BitSet@AfterFi{%
360       \expandafter\expandafter\expandafter\BitSet@SetDecBig
361       \BigIntCalcShr#1#2#3#4#5#6#7#8#9!!%
362     }%
363   \BitSet@Fi
364 }

```

\BitSet@SetDec

```

365 \def\BitSet@SetDec#1!{%

```



```

366 \ifcase#1 %
367 \or 1%
368 \else
369 \ifodd#1 %
370 1%
371 \else
372 0%
373 \fi
374 \BitSet@AfterFi{%
375 \expandafter\expandafter\expandafter\BitSet@SetDec
376 \IntCalcShr#1!!%
377 }%
378 \BitSet@Fi
379 }

```

2.7 Export

2.7.1 To binary number

\bitsetGetBin

```

380 \def\bitsetGetBin#1#2{%
381 \romannumeral0%
382 \expandafter\expandafter\expandafter\BitSet@@GetBin
383 \intcalcNum{#2}!{#1}%
384 }

```

\BitSet@@GetBin

```

385 \def\BitSet@@GetBin#1!#2{%
386 \BitSet@IfUndefined{#2}{%
387 \ifnum#1>1 %
388 \BitSet@AfterFi{%
389 \expandafter\expandafter\expandafter\BitSet@Fill
390 \IntCalcDec#1!!0%
391 }%
392 \else
393 \BitSet@AfterFi{ 0}%
394 \BitSet@Fi
395 }{%
396 \expandafter\expandafter\expandafter\BitSet@NumBinRev
397 \expandafter\expandafter\expandafter1%
398 \expandafter\expandafter\expandafter!%
399 \csname BS@#2\endcsname!!#1!%
400 }%
401 }

```

\BitSet@Fill #1: number of leading digits 0

#2: result

```

402 \def\BitSet@Fill#1!{%
403 \ifnum#1>0 %
404 \BitSet@AfterFi{%
405 \expandafter\expandafter\expandafter\BitSet@Fill
406 \IntCalcDec#1!!0%
407 }%
408 \else
409 \BitSet@AfterFi{ }%
410 \BitSet@Fi
411 }

```

\BitSet@NumBinRev #1: bit counter (including #2)

#2#3: reverted number

#4: result

#5: min size

```

412 \def\BitSet@NumBinRev#1!#2#3!{%
413   \ifx\#3\%
414     \BitSet@AfterFi{%
415       \BitSet@NumBinFill#1!#2%
416     }%
417   \else
418     \BitSet@AfterFi{%
419       \expandafter\expandafter\expandafter\BitSet@NumBinRev
420       \IntCalcInc#1!!#3!#2%
421     }%
422   \BitSet@Fi
423 }

```

\BitSet@NumBinFill

```

424 \def\BitSet@NumBinFill#1!#2!#3!{%
425   \ifnum#3>#1 %
426     \BitSet@AfterFi{%
427       \expandafter\expandafter\expandafter\BitSet@Fill
428       \IntCalcSub#3!#1!!#2%
429     }%
430   \else
431     \BitSet@AfterFi{ #2}%
432   \BitSet@Fi
433 }

```

2.7.2 To octal/hexadecimal number

\bitsetGetOct

```

434 \def\bitsetGetOct#1#2{%
435   \romannumeral0%
436   \bitsetIsEmpty{#1}{%
437     \expandafter\expandafter\expandafter\BitSet@@GetOctHex
438     \intcalcNum{#2}!3!230%
439   }{%
440     \expandafter\expandafter\expandafter\BitSet@@GetOct
441     \expandafter\expandafter\expandafter1%
442     \expandafter\expandafter\expandafter!%
443     \expandafter\expandafter\expandafter!%
444     \csname BS@#1\endcsname00%
445     \BitSet@Empty\BitSet@Empty\BitSet@Empty!{#2}%
446   }%
447 }

```

\bitsetGetHex

```

448 \def\bitsetGetHex#1#2{%
449   \romannumeral0%
450   \bitsetIsEmpty{#1}{%
451     \expandafter\expandafter\expandafter\BitSet@@GetOctHex
452     \intcalcNum{#2}!4!340%
453   }{%
454     \expandafter\expandafter\expandafter\BitSet@@GetHex
455     \expandafter\expandafter\expandafter1%
456     \expandafter\expandafter\expandafter!%
457     \expandafter\expandafter\expandafter!%
458     \csname BS@#1\endcsname000%
459     \BitSet@Empty\BitSet@Empty\BitSet@Empty\BitSet@Empty!{#2}%
460   }%
461 }

```

\BitSet@@GetOct #1: number of digits
#2: result
#3#4#5: bits

```

462 \def\BitSet@@GetOct#1!#2!#3#4#5{%
463   \ifx#5\BitSet@Empty
464     \BitSet@AfterFi{%
465       \expandafter\expandafter\expandafter\BitSet@GetOctHex
466       \IntCalcDec#1!#2!23%
467     }%
468   \else
469     \BitSet@AfterFi{%
470       \expandafter\expandafter\expandafter\BitSet@@GetOct
471       \number\IntCalcInc#1!\expandafter\expandafter\expandafter!%
472       \csname BitSet@Oct#5#4#3\endcsname#2!%
473     }%
474   \BitSet@Fi
475 }

```

\BitSet@Oct[000..111]

```

476 \def\BitSet@Temp#1#2#3#4{%
477   \expandafter\def\csname BitSet@Oct#1#2#3\endcsname{#4}%
478 }
479 \BitSet@Temp0000%
480 \BitSet@Temp0011%
481 \BitSet@Temp0102%
482 \BitSet@Temp0113%
483 \BitSet@Temp1004%
484 \BitSet@Temp1015%
485 \BitSet@Temp1106%
486 \BitSet@Temp1117%

```

\BitSet@@GetHex #1: number of digits
#2: result
#3#4#5#6: bits

```

487 \def\BitSet@@GetHex#1!#2!#3#4#5#6{%
488   \ifx#6\BitSet@Empty
489     \BitSet@AfterFi{%
490       \expandafter\expandafter\expandafter\BitSet@GetOctHex
491       \IntCalcDec#1!#2!34%
492     }%
493   \else
494     \BitSet@AfterFi{%
495       \expandafter\expandafter\expandafter\BitSet@@GetHex
496       \number\IntCalcInc#1!\expandafter\expandafter\expandafter!%
497       \csname BitSet@Hex#6#5#4#3\endcsname#2!%
498     }%
499   \BitSet@Fi
500 }

```

\BitSet@Hex[0000..1111]

```

501 \def\BitSet@Temp#1#2#3#4#5{%
502   \expandafter\def\csname BitSet@Hex#1#2#3#4\endcsname{#5}%
503 }
504 \BitSet@Temp00000%
505 \BitSet@Temp00011%
506 \BitSet@Temp00102%
507 \BitSet@Temp00113%
508 \BitSet@Temp01004%
509 \BitSet@Temp01015%
510 \BitSet@Temp01106%
511 \BitSet@Temp01117%
512 \BitSet@Temp10008%
513 \BitSet@Temp10019%
514 \BitSet@Temp1010A%
515 \BitSet@Temp1011B%

```

```

516 \BitSet@Temp1100C%
517 \BitSet@Temp1101D%
518 \BitSet@Temp1110E%
519 \BitSet@Temp1111F%

\BitSet@GetOctHex Leading zeros  $(\#4 - \#1 * 3 + 2)/3$  if  $\#4 > \#1 * 3$ 
#1: digit size
#2: result
#3: bits per digit - 1
#4: bits per digit #5: garbage
#6: min size

520 \def\BitSet@GetOctHex#1!#2!#3#4#5!#6{%
521   \expandafter\BitSet@@GetOctHex
522   \number\intcalcNum{#6}\expandafter\expandafter\expandafter!%
523   \IntCalcMul#1!#4!!#3#4#2%
524 }

\BitSet@@GetOctHex #1: plain min size
#2: digits * (bits per digit)
#3: bits per digit - 1
#4: bits per digit

525 \def\BitSet@@GetOctHex#1!#2!#3#4{%
526   \ifnum#1>#2 %
527     \BitSet@AfterFi{%
528       \expandafter\expandafter\expandafter\expandafter
529       \expandafter\expandafter\expandafter\BitSet@Fill
530       \expandafter\IntCalcDiv\number
531       \expandafter\expandafter\expandafter\IntCalcAdd
532       \IntCalcSub#1!#2!!#3!!#4!!%
533     }%
534   \else
535     \BitSet@AfterFi{ }%
536   \BitSet@Fi
537 }

```

2.7.3 To decimal number

```

\bitsetGetDec

538 \def\bitsetGetDec#1{%
539   \romannumeral0%
540   \BitSet@IfUndefined{#1}{ 0}{%
541     \expandafter\expandafter\expandafter\BitSet@GetDec
542     \csname BS@#1\endcsname!%
543   }%
544 }

\BitSet@GetDec

545 \def\BitSet@GetDec#1#2!{%
546   \ifx\#2\%
547     \BitSet@AfterFi{ #1}%
548   \else
549     \BitSet@AfterFi{%
550       \BitSet@@GetDec2!#1!#2!%
551     }%
552   \BitSet@Fi
553 }

\BitSet@@GetDec #1: power of two
#2: result
#3#4: number

554 \def\BitSet@@GetDec#1!#2!#3#4!{%

```

```

555 \ifx\#4\%
556 \ifx#31%
557 \BitSet@AfterFiFi{%
558 \expandafter\expandafter\expandafter\BitSet@Space
559 \IntCalcAdd#1!#2!%
560 }%
561 \else
562 \BitSet@AfterFiFi{ #2}%
563 \fi
564 \else
565 \ifx#31%
566 \BitSet@AfterFiFi{%
567 \csname BitSet@N#1%
568 \expandafter\expandafter\expandafter\endcsname
569 \IntCalcAdd#1!#2!!#4!%
570 }%
571 \else
572 \BitSet@AfterFiFi{%
573 \csname BitSet@N#1\endcsname#2!#4!%
574 }%
575 \fi
576 \BitSet@Fi
577 }

```

\BitSet@N[1,2,4,...]

```

578 \def\BitSet@Temp#1#2{%
579 \expandafter\def\csname BitSet@N#1\endcsname{%
580 \BitSet@@GetDec#2!%
581 }%
582 }
583 \BitSet@Temp{1}{2}
584 \BitSet@Temp{2}{4}
585 \BitSet@Temp{4}{8}
586 \BitSet@Temp{8}{16}
587 \BitSet@Temp{16}{32}
588 \BitSet@Temp{32}{64}
589 \BitSet@Temp{64}{128}
590 \BitSet@Temp{128}{256}
591 \BitSet@Temp{256}{512}
592 \BitSet@Temp{512}{1024}
593 \BitSet@Temp{1024}{2048}
594 \BitSet@Temp{2048}{4096}
595 \BitSet@Temp{4096}{8192}
596 \BitSet@Temp{8192}{16384}
597 \BitSet@Temp{16384}{32768}
598 \BitSet@Temp{32768}{65536}
599 \BitSet@Temp{65536}{131072}
600 \BitSet@Temp{131072}{262144}
601 \BitSet@Temp{262144}{524288}
602 \BitSet@Temp{524288}{1048576}
603 \BitSet@Temp{1048576}{2097152}
604 \BitSet@Temp{2097152}{4194304}
605 \BitSet@Temp{4194304}{8388608}
606 \BitSet@Temp{8388608}{16777216}
607 \BitSet@Temp{16777216}{33554432}
608 \BitSet@Temp{33554432}{67108864}
609 \BitSet@Temp{67108864}{134217728}
610 \BitSet@Temp{134217728}{268435456}
611 \BitSet@Temp{268435456}{536870912}
612 \BitSet@Temp{536870912}{1073741824}

```

\BitSet@N1073741824

```

613 \expandafter\def\csname BitSet@N1073741824\endcsname{%
614   \BitSet@GetDecBig2147483648!%
615 }%

\BitSet@GetDecBig #1: current power of two
#2: result
#3#4: number

616 \def\BitSet@GetDecBig#1!#2!#3#4!{%
617   \ifx\#4\%
618     \ifx#31%
619       \BitSet@AfterFiFi{%
620         \expandafter\expandafter\expandafter\BitSet@Space
621         \BigIntCalcAdd#1!#2!%
622       }%
623     \else
624       \BitSet@AfterFiFi{ #2}%
625     \fi
626   \else
627     \ifx#31%
628       \BitSet@AfterFiFi{%
629         \expandafter\expandafter\expandafter\BitSet@GetDecBig
630         \BigIntCalcAdd#1!#2!!#1!#4!%
631       }%
632     \else
633       \BitSet@AfterFiFi{%
634         \expandafter\expandafter\expandafter\BitSet@GetDecBig
635         \BigIntCalcShl#1!!#2!#4!%
636       }%
637     \fi
638   \BitSet@Fi
639 }

\BitSet@@GetDecBig #1: result
#2: power of two
#3#4: number

640 \def\BitSet@@GetDecBig#1!#2!{%
641   \expandafter\expandafter\expandafter\BitSet@GetDecBig
642   \BigIntCalcShl#2!!#1!%
643 }

```

2.8 Logical operators

2.8.1 \bitsetAnd

\bitsetAnd Decision table for \bitsetAnd:

	undef(B)	empty(B)	cardinality(B)>0
undef(A)	A := empty	A := empty	A := empty
empty(A)			
cardinality(A)>0	A := empty	A := empty	A &= B

```

644 \def\bitsetAnd#1#2{%
645   \bitsetIsEmpty{#1}{%
646     \bitsetReset{#1}%
647   }{%
648     \bitsetIsEmpty{#2}{%
649       \bitsetReset{#1}%
650     }{%
651       \expandafter\edef\csname BS@#1\endcsname{%
652         \expandafter\expandafter\expandafter\BitSet@And
653         \csname BS@#1\expandafter\expandafter\expandafter\endcsname
654         \expandafter\expandafter\expandafter!%

```

```

655         \csname BS@#2\endcsname!!%
656     }%
657     \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
658         \bitsetReset{#1}%
659     \fi
660 }%
661 }%
662 }

```

\BitSet@And

```

663 \def\BitSet@And#1#2!#3#4!#5!{%
664     \ifx\#2\%
665         \ifnum#1#3=11 #51\fi
666     \else
667         \ifx\#4\%
668             \ifnum#1#3=11 #51\fi
669         \else
670             \ifnum#1#3=11 %
671                 #51%
672                 \BitSet@AfterFiFiFi{%
673                     \BitSet@And#2!#4!%!%
674                 }%
675             \else
676                 \BitSet@AfterFiFiFi{%
677                     \BitSet@And#2!#4!#50!%
678                 }%
679             \fi
680         \fi
681     \BitSet@Fi
682 }

```

2.8.2 \bitsetAndNot

\bitsetAndNot Decision table for \bitsetAndNot:

	undef(B)	empty(B)	cardinality(B)>0
undef(A)	A := empty	A := empty	A := empty
empty(A)			
cardinality(A)>0			A &= !B

```

683 \def\bitsetAndNot#1#2{%
684     \bitsetIsEmpty{#1}{%
685         \bitsetReset{#1}%
686     }{%
687         \bitsetIsEmpty{#2}{%
688             }{%
689                 \expandafter\edef\csname BS@#1\endcsname{%
690                     \expandafter\expandafter\expandafter\BitSet@AndNot
691                     \csname BS@#1\expandafter\expandafter\expandafter\endcsname
692                     \expandafter\expandafter\expandafter!%
693                     \csname BS@#2\endcsname!!%
694                 }%
695                 \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
696                     \bitsetReset{#1}%
697                 \fi
698             }%
699         }%
700     }

```

\BitSet@AndNot

```

701 \def\BitSet@AndNot#1#2!#3#4!#5!{%
702     \ifx\#2\%

```

```

703   \ifnum#1#3=10 #51\fi
704 \else
705   \ifx\\#4\\%
706     #5%
707     \ifnum#1#3=10 1\else 0\fi
708     #2%
709   \else
710     \ifnum#1#3=10 %
711       #51%
712       \BitSet@AfterFiFiFi{%
713         \BitSet@AndNot#2!#4!%!%
714       }%
715   \else
716     \BitSet@AfterFiFiFi{%
717       \BitSet@AndNot#2!#4!#50!%!%
718     }%
719   \fi
720 \fi
721 \BitSet@Fi
722 }

```

2.8.3 \bitsetOr

\bitsetOr Decision table for \bitsetOr:

	undef(B)	empty(B)	cardinality(B)>0
undef(A)	A := empty	A := empty	A := B
empty(A)			A := B
cardinality(A)>0			A = B

```

723 \def\bitsetOr#1#2{%
724   \bitsetIsEmpty{#2}{%
725     \BitSet@IfUndefined{#1}{\bitsetReset{#1}}{}}%
726 }{%
727   \bitsetIsEmpty{#1}{%
728     \expandafter\let\csname BS@#1\expandafter\endcsname
729       \csname BS@#2\endcsname
730   }{%
731     \expandafter\edef\csname BS@#1\endcsname{%
732       \expandafter\expandafter\expandafter\BitSet@Or
733       \csname BS@#1\expandafter\expandafter\expandafter\endcsname
734       \expandafter\expandafter\expandafter!%
735       \csname BS@#2\endcsname!%
736     }%
737   }%
738 }%
739 }

```

\BitSet@Or

```

740 \def\BitSet@Or#1#2!#3#4!{%
741   \ifnum#1#3>0 1\else 0\fi
742   \ifx\\#2\\%
743     #4%
744   \else
745     \ifx\\#4\\%
746       #2%
747     \else
748       \BitSet@AfterFiFiFi{%
749         \BitSet@Or#2!#4!%!%
750       }%
751     \fi
752   \BitSet@Fi
753 }

```


2.8.4 \bitsetXor

\bitsetXor Decision table for \bitsetXor:

	undef(B)	empty(B)	cardinality(B)>0
undef(A)	A := empty	A := empty	A := B
empty(A)			A := B
cardinality(A)>0			A $\hat{=}$ B

```

754 \def\bitsetXor#1#2{%
755   \bitsetIsEmpty{#2}{%
756     \BitSet@IfUndefined{#1}{\bitsetReset{#1}}{}%
757   }{%
758     \bitsetIsEmpty{#1}{%
759       \expandafter\let\csname BS@#1\expandafter\endcsname
760         \csname BS@#2\endcsname
761     }{%
762       \expandafter\edef\csname BS@#1\endcsname{%
763         \expandafter\expandafter\expandafter\BitSet@Xor
764         \csname BS@#1\expandafter\expandafter\expandafter\endcsname
765         \expandafter\expandafter\expandafter!%
766         \csname BS@#2\endcsname!%!%
767       }%
768       \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
769         \bitsetReset{#1}%
770       \fi
771     }%
772   }%
773 }

```

\BitSet@Xor

```

774 \def\BitSet@Xor#1#2!#3#4!#5!{%
775   \ifx\#2\%
776     \ifx#1#3%
777       \ifx\#4\%
778         \else
779           #50#4%
780         \fi
781       \else
782         #51#4%
783       \fi
784     \else
785       \ifx\#4\%
786         #5%
787         \ifx#1#30\else 1\fi
788         #2%
789       \else
790         \ifx#1#3%
791           \BitSet@AfterFiFiFi{%
792             \BitSet@Xor#2!#4!#50!%
793           }%
794         \else
795           #51%
796           \BitSet@AfterFiFiFi{%
797             \BitSet@Xor#2!#4!%!%
798           }%
799         \fi
800       \fi
801     \BitSet@Fi
802 }

```

2.8.5 Shifting

2.8.6 \bitsetShiftLeft

\bitsetShiftLeft

```
803 \def\bitsetShiftLeft#1#2{%
804   \BitSet@IfUndefined{#1}{%
805     \bitsetReset{#1}%
806   }{%
807     \bitsetIsEmpty{#1}{%
808       {%
809         \expandafter\expandafter\expandafter\BitSet@ShiftLeft
810         \intcalcNum{#2}!{#1}%
811       }%
812     }%
813 }
```

\BitSet@ShiftLeft

```
814 \def\BitSet@ShiftLeft#1!#2{%
815   \ifcase\intcalcSgn{#1} %
816   \or
817     \begingroup
818       \uccode'm='0 %
819       \uppercase\expandafter{\expandafter\endgroup
820         \expandafter\edef\csname BS@#2\expandafter\endcsname
821         \expandafter{%
822           \romannumeral#1000\expandafter\BitSet@Space
823           \csname BS@#2\endcsname
824         }%
825       }%
826   \else
827     \expandafter\BitSet@ShiftRight\BitSet@Gobble#1!{#2}%
828   \fi
829 }
```

2.8.7 \bitsetShiftRight

\bitsetShiftRight

```
830 \def\bitsetShiftRight#1#2{%
831   \BitSet@IfUndefined{#1}{%
832     \bitsetReset{#1}%
833   }{%
834     \bitsetIsEmpty{#1}{%
835       {%
836         \expandafter\expandafter\expandafter\BitSet@ShiftRight
837         \intcalcNum{#2}!{#1}%
838       }%
839     }%
840 }
```

\BitSet@ShiftRight

```
841 \def\BitSet@ShiftRight#1!#2{%
842   \ifcase\intcalcSgn{#1} %
843   \or
844     \expandafter\edef\csname BS@#2\endcsname{%
845       \expandafter\expandafter\expandafter\BitSet@Kill
846       \csname BS@#2\expandafter\endcsname\expandafter\BitSet@Empty
847       \expandafter=%
848       \expandafter{\expandafter}\expandafter{\expandafter}%
849       \romannumeral#1000!%
850     }%
851   \else
```

```

852 \expandafter\BitSet@ShiftLeft\BitSet@Gobble#1!{#2}%
853 \fi
854 }

```

\BitSet@Kill

```

855 \def\BitSet@Kill#1#2=#3#4#5{%
856   #3#4%
857   \ifx#5!%
858     \ifx#1\BitSet@Empty
859       0%
860     \else
861       #1#2%
862     \fi
863   \else
864     \ifx#1\BitSet@Empty
865       0%
866     \BitSet@AfterFiFi\BitSet@Cleanup
867   \else
868     \BitSet@Kill#2=%
869   \fi
870 \BitSet@Fi
871 }

```

2.9 Bit manipulation

\bitsetClear

```

872 \def\bitsetClear{%
873   \BitSet@CheckIndex\BitSet@Clear
874 }

```

\bitsetSet

```

875 \def\bitsetSet{%
876   \BitSet@CheckIndex\BitSet@Set
877 }

```

\bitsetFlip

```

878 \def\bitsetFlip{%
879   \BitSet@CheckIndex\BitSet@Flip
880 }

```

\bitsetSetValue

```

881 \def\bitsetSetValue#1#2#3{%
882   \expandafter\expandafter\expandafter\BitSet@SetValue
883   \intcalcNum{#3}!{#1}{#2}%
884 }

```

\BitSet@SetValue #1: plain value

#2: BitSet

#3: Index

```

885 \def\BitSet@SetValue#1!{%
886   \BitSet@CheckIndex{%
887     \ifcase#1 %
888       \expandafter\BitSet@Clear
889     \or
890       \expandafter\BitSet@Set
891     \else
892       \BitSet@ErrorInvalidBitValue{#1}%
893     \expandafter\expandafter\expandafter\BitSet@Gobble
894     \expandafter\BitSet@Gobble
895   \fi
896   }%
897 }

```

```

\BitSet@ErrorInvalidBitValue #1: Wrong bit value

898 \def\BitSet@ErrorInvalidBitValue#1{%
899   \@PackageError{bitset}{%
900     Invalid bit value (#1) not in range 0..1%
901   }\@ehc
902 }

```

2.9.1 Clear operation

```

\BitSet@Clear #1: BitSet
#2: plain and checked index

903 \def\BitSet@Clear#1#2{%
904   \edef\BitSet@Temp{%
905     \expandafter\expandafter\expandafter\BitSet@@Clear
906     \csname BS@#1\expandafter\endcsname
907     \expandafter\BitSet@Empty\expandafter=\expandafter!%
908     \romannumeral#2000!%
909   }%
910   \expandafter\let\csname BS@#1\expandafter\endcsname
911   \ifx\BitSet@Temp\BitSet@Empty
912     \BitSet@Zero
913   \else
914     \BitSet@Temp
915   \fi
916 }

```

```

\BitSet@@Clear

917 \def\BitSet@@Clear#1#2=#3!#4{%
918   \ifx#4!%
919     \ifx#1\BitSet@Empty
920       \else
921         \ifx\BitSet@Empty#2%
922           \else
923             #30#2%
924           \fi
925         \fi
926       \else
927         \ifx#1\BitSet@Empty
928           \BitSet@AfterFiFi\BitSet@Cleanup
929         \else
930           \ifx#10%
931             \BitSet@AfterFiFiFi{%
932               \BitSet@@Clear#2=#30!%
933             }%
934           \else
935             #31%
936             \BitSet@AfterFiFiFi{%
937               \BitSet@@Clear#2=!%
938             }%
939           \fi
940         \fi
941       \BitSet@Fi
942 }

```

2.9.2 Set operation

```

\BitSet@Set #1: BitSet
#2: plain and checked Index

943 \def\BitSet@Set#1#2{%
944   \expandafter\edef\csname BS@#1\endcsname{%
945     \expandafter\expandafter\expandafter\BitSet@@Set

```

```

946 \csname BS@#1\expandafter\endcsname
947 \expandafter\BitSet@Empty\expandafter=%
948 \expandafter{\expandafter}\expandafter{\expandafter}%
949 \romannumeral#2000!%
950 }%
951 }

```

\BitSet@@Set

```

952 \def\BitSet@@Set#1#2=#3#4#5{%
953   #3#4%
954   \ifx#5!%
955     1#2%
956   \else
957     \ifx#1\BitSet@Empty
958       0%
959       \BitSet@AfterFiFi\BitSet@@@Set
960     \else
961       #1%
962       \BitSet@@Set#2=%
963     \fi
964   \BitSet@Fi
965 }

```

\BitSet@@@Set

```

966 \def\BitSet@@@Set#1{%
967   \ifx#1!%
968     1%
969   \else
970     0%
971     \expandafter\BitSet@@@Set
972   \fi
973 }

```

2.9.3 Flip operation

\BitSet@Flip #1: BitSet

#2: plain and checked Index

```

974 \def\BitSet@Flip#1#2{%
975   \edef\BitSet@Temp{%
976     \expandafter\expandafter\expandafter\BitSet@@Flip
977     \csname BS@#1\expandafter\endcsname
978     \expandafter\BitSet@Empty\expandafter=\expandafter!%
979     \romannumeral#2000!%
980   }%
981   \expandafter\let\csname BS@#1\expandafter\endcsname
982   \ifx\BitSet@Temp\BitSet@Empty
983     \BitSet@Zero
984   \else
985     \BitSet@Temp
986   \fi
987 }

```

\BitSet@@@Flip

```

988 \def\BitSet@@@Flip#1#2=#3!#4{%
989   \ifx#4!%
990     \ifx#11%
991       \ifx\BitSet@Empty#2%
992         \else
993           #30#2%
994         \fi
995       \else
996         #31#2%

```

```

997     \fi
998   \else
999     \ifx#1\BitSet@Empty
1000       #30%
1001       \BitSet@AfterFiFi\BitSet@@@Set
1002     \else
1003       \ifx#10%
1004         \BitSet@AfterFiFiFi{%
1005           \BitSet@@Flip#2=#30!%
1006         }%
1007       \else
1008         #31%
1009         \BitSet@AfterFiFiFi{%
1010           \BitSet@@Flip#2=!%
1011         }%
1012       \fi
1013     \fi
1014   \BitSet@Fi
1015 }

```

2.9.4 Range operators

\bitsetClearRange

```

1016 \def\bitsetClearRange{%
1017   \BitSet@Range\BitSet@Clear
1018 }

```

\bitsetSetRange

```

1019 \def\bitsetSetRange{%
1020   \BitSet@Range\BitSet@Set
1021 }

```

\bitsetFlipRange

```

1022 \def\bitsetFlipRange{%
1023   \BitSet@Range\BitSet@Flip
1024 }

```

\bitsetSetValueRange

```

1025 \def\bitsetSetValueRange#1#2#3#4{%
1026   \expandafter\expandafter\expandafter\BitSet@SetValueRange
1027   \intcalNum{#4}!{#1}{#2}{#3}%
1028 }

```

\BitSet@SetValueRange

```

1029 \def\BitSet@SetValueRange#1!#2#3#4{%
1030   \ifcase#1 %
1031     \BitSet@Range\BitSet@Clear{#2}{#3}{#4}%
1032   \or
1033     \BitSet@Range\BitSet@Set{#2}{#3}{#4}%
1034   \else
1035     \BitSet@ErrorInvalidBitValue{#1}%
1036   \fi
1037 }

```

\BitSet@Range #1: clear/set/flip macro

#2: BitSet

#3: Index from

#4: Index to

```

1038 \def\BitSet@Range#1#2#3#4{%
1039   \edef\BitSet@Temp{%
1040     \noexpand\BitSet@@Range\noexpand#1{#2}%

```

```

1041     \intcalcNum{#3}!\intcalcNum{#4}!%
1042 }%
1043 \BitSet@Temp
1044 }

\BitSet@@Range #1: clear/set/flip macro
#2: BitSet
#3: Index from
#4: Index to

1045 \def\BitSet@@Range#1#2#3!#4!{%
1046   \ifnum#3<0 %
1047     \BitSet@NegativeIndex#1{#2}#3!#4!0!#4!%
1048   \else
1049     \ifnum#4<0 %
1050       \BitSet@NegativeIndex#1{#2}#3!#4!#3!0!%
1051     \else
1052       \ifcase\intcalcCmp{#3}{#4} %
1053       \or
1054         \@PackageError{bitset}{%
1055           Wrong index numbers in range [#3..#4]\MessageBreak% hash-ok
1056           for clear/set/flip on bit set '#2'.\MessageBreak
1057           The lower index exceeds the upper index.\MessageBreak
1058           Canceling the operation as error recovery%
1059         }\@ehc
1060       \else
1061         \BitSet@@Range#3!#4!#1{#2}%
1062       \fi
1063     \fi
1064   \fi
1065 }

\BitSet@NegativeIndex

1066 \def\BitSet@NegativeIndex#1#2#3!#4!#5!#6!{%
1067   \@PackageError{bitset}{%
1068     Negative index in range [#3..#4]\MessageBreak % hash-ok
1069     for \string\bitset
1070     \ifx#1\BitSet@Clear
1071       Clear%
1072     \else
1073       \ifx#1\BitSet@Set
1074         Set%
1075       \else
1076         Flip%
1077       \fi
1078     \fi
1079     Range on bit set '#2'.\MessageBreak
1080     Using [#5..#6] as error recovery% hash-ok
1081   }\@ehc
1082   \BitSet@@Range#1{#2}#5!#6!%
1083 }

\BitSet@@Range

1084 \def\BitSet@@Range#1!#2!#3#4{%
1085   \ifnum#1<#2 %
1086     #3{#4}{#1}%
1087   \BitSet@AfterFi{%
1088     \expandafter\expandafter\expandafter\BitSet@@Range
1089     \IntCalcInc#1!#2!#3{#4}%
1090   }%
1091   \BitSet@Fi
1092 }

```

2.10 Bit retrieval

2.10.1 \bitsetGet

\bitsetGet

```
1093 \def\bitsetGet#1#2{%
1094   \number
1095   \expandafter\expandafter\expandafter\BitSet@Get
1096   \intcalcNum{#2}!\{#1}%
1097 }
```

\BitSet@Get #1: plain index
#2: BitSet

```
1098 \def\BitSet@Get#1!#2{%
1099   \ifnum#1<0 %
1100     \BitSet@AfterFi{%
1101       0 \BitSetError:NegativeIndex%
1102     }%
1103   \else
1104     \BitSet@IfUndefined{#2}{0}{%
1105       \expandafter\expandafter\expandafter\BitSet@@Get
1106       \csname BS@#2\expandafter\endcsname
1107       \expandafter!\expandafter=%
1108       \expandafter{\expandafter}\expandafter{\expandafter}%
1109       \romannumeral\intcalcNum{#1}000!%
1110     }%
1111     \expandafter\BitSet@Space
1112     \BitSet@Fi
1113 }
```

\BitSet@@Get

```
1114 \def\BitSet@@Get#1#2=#3#4#5{%
1115   #3#4%
1116   \ifx#5!%
1117     \ifx#1!%
1118       0%
1119     \else
1120       #1%
1121     \fi
1122   \else
1123     \ifx#1!%
1124       0%
1125       \BitSet@AfterFiFi\BitSet@Cleanup
1126     \else
1127       \BitSet@@Get#2=%
1128     \fi
1129   \BitSet@Fi
1130 }
```

2.10.2 \bitsetNextClearBit, \bitsetNextSetBit

\bitsetNextClearBit

```
1131 \def\bitsetNextClearBit#1#2{%
1132   \number
1133   \expandafter\expandafter\expandafter\BitSet@NextClearBit
1134   \intcalcNum{#2}!\{#1} %
1135 }
```

\BitSet@NextClearBit #1: Index
#2: BitSet

```
1136 \def\BitSet@NextClearBit#1!#2{%
1137   \ifnum#1<0 %
```



```

1138 \BitSet@NextClearBit0!{#2}%
1139 \BitSet@AfterFi{%
1140 \expandafter\BitSet@Space
1141 \expandafter\BitSetError:NegativeIndex\romannumeral0%
1142 }%
1143 \else
1144 \bitsetIsEmpty{#2}{#1}{%
1145 \expandafter\BitSet@Skip
1146 \number#1\expandafter\expandafter\expandafter!%
1147 \csname BS@#2\endcsname!!!!!!!!!=%
1148 {\BitSet@@NextClearBit#1!}%
1149 }%
1150 \BitSet@Fi
1151 }

```

\BitSet@@NextClearBit #1: index for next bit in #2
#2: next bit

```

1152 \def\BitSet@@NextClearBit#1!#2{%
1153 \ifx#2!%
1154 #1%
1155 \else
1156 \ifx#20%
1157 #1%
1158 \BitSet@AfterFiFi\BitSet@Cleanup
1159 \else
1160 \BitSet@AfterFiFi{%
1161 \expandafter\expandafter\expandafter\BitSet@@NextClearBit
1162 \IntCalcInc#1!!%
1163 }%
1164 \fi
1165 \BitSet@Fi
1166 }

```

\bitsetNextSetBit

```

1167 \def\bitsetNextSetBit#1#2{%
1168 \number
1169 \expandafter\expandafter\expandafter\BitSet@NextSetBit
1170 \intcalcNum{#2}{#1} %
1171 }

```

\BitSet@NextSetBit #1: Index
#2: BitSet

```

1172 \def\BitSet@NextSetBit#1!#2{%
1173 \ifnum#1<0 %
1174 \BitSet@NextSetBit0!{#2}%
1175 \BitSet@AfterFi{%
1176 \expandafter\BitSet@Space
1177 \expandafter\BitSetError:NegativeIndex\romannumeral0%
1178 }%
1179 \else
1180 \bitsetIsEmpty{#2}{-1}{%
1181 \expandafter\BitSet@Skip
1182 \number#1\expandafter\expandafter\expandafter!%
1183 \csname BS@#2\endcsname!!!!!!!!!=%
1184 {\BitSet@@NextSetBit#1!}%
1185 }%
1186 \BitSet@Fi
1187 }

```

\BitSet@@NextSetBit #1: index for next bit in #2
#2: next bit

```

1188 \def\BitSet@@NextSetBit#1!#2{%

```

```

1189 \ifx#2!%
1190 -1%
1191 \else
1192 \ifx#21%
1193 #1%
1194 \BitSet@AfterFiFi\BitSet@Cleanup
1195 \else
1196 \BitSet@AfterFiFi{%
1197 \expandafter\expandafter\expandafter\BitSet@@NextSetBit
1198 \IntCalcInc#1!!%
1199 }%
1200 \fi
1201 \BitSet@Fi
1202 }

\BitSet@Cleanup
1203 \def\BitSet@Cleanup#1!{%

\BitSet@Skip #1: number of bits to skip
#2: bits
#3: continuation code
1204 \def\BitSet@Skip#1!#2{%
1205 \ifx#2!%
1206 \BitSet@AfterFiFi{%
1207 \BitSet@SkipContinue%
1208 }%
1209 \else
1210 \ifcase#1 %
1211 \BitSet@AfterFiFi{%
1212 \BitSet@SkipContinue#2%
1213 }%
1214 \or
1215 \BitSet@AfterFiFi\BitSet@SkipContinue
1216 \or
1217 \BitSet@AfterFiFi{%
1218 \expandafter\BitSet@SkipContinue\BitSet@Gobble
1219 }%
1220 \else
1221 \ifnum#1>8 %
1222 \BitSet@AfterFiFiFi{%
1223 \expandafter\BitSet@Skip
1224 \number\IntCalcSub#1!8!\expandafter!%
1225 \BitSet@GobbleSeven
1226 }%
1227 \else
1228 \BitSet@AfterFiFiFi{%
1229 \expandafter\expandafter\expandafter\BitSet@Skip
1230 \IntCalcDec#1!!%
1231 }%
1232 \fi
1233 \fi
1234 \BitSet@Fi
1235 }

\BitSet@SkipContinue #1: remaining bits
#2: continuation code
1236 \def\BitSet@SkipContinue#1!#2=#3{%
1237 #3#1!%
1238 }

\BitSet@GobbleSeven
1239 \def\BitSet@GobbleSeven#1#2#3#4#5#6#7{%

```

2.10.3 \bitsetGetSetBitList

\bitsetGetSetBitList It's just a wrapper for \bitsetNextSetBit.

```

1240 \def\bitsetGetSetBitList#1{%
1241   \romannumeral0%
1242   \bitsetIsEmpty{#1}{ }{%
1243     \expandafter\BitSet@GetSetBitList
1244     \number\BitSet@NextSetBit0!{#1}!{#1}{ }!%
1245   }%
1246 }
```

```

\BitSet@GetSetBitList #1: found index
#2: BitSet
#3: comma #4: result

1247 \def\BitSet@GetSetBitList#1!#2#3#4!{%
1248   \ifnum#1<0 %
1249     \BitSet@AfterFi{ #4}%
1250   \else
1251     \BitSet@AfterFi{%
1252       \expandafter\BitSet@GetSetBitList\number
1253       \expandafter\expandafter\expandafter\BitSet@NextSetBit
1254       \IntCalcInc#1!!{#2}!{#2},#4#3#1!%
1255     }%
1256   \BitSet@Fi
1257 }
```

2.11 Bit set properties

\bitsetSize

```

1258 \def\bitsetSize#1{%
1259   \number
1260   \BitSet@IfUndefined{#1}{0 }{%
1261     \expandafter\expandafter\expandafter\BitSet@Size
1262     \expandafter\expandafter\expandafter1%
1263     \expandafter\expandafter\expandafter!%
1264     \csname BS@#1\endcsname!0!%
1265   }%
1266 }
```

```

\BitSet@Size #1: counter
#2#3: bits
#4: result

1267 \def\BitSet@Size#1!#2#3!#4!{%
1268   \ifx\#21%
1269     \ifx\#3\%
1270       \BitSet@AfterFiFi{#1 }%
1271     \else
1272       \BitSet@AfterFiFi{%
1273         \expandafter\expandafter\expandafter\BitSet@Size
1274         \IntCalcInc#1!!#3!#1!%
1275       }%
1276     \fi
1277   \else
1278     \ifx\#3\%
1279       \BitSet@AfterFiFi{#4 }%
1280     \else
1281       \BitSet@AfterFiFi{%
1282         \expandafter\expandafter\expandafter\BitSet@Size
1283         \IntCalcInc#1!!#3!#4!%
1284       }%
1285     \fi
1286 }
```

```

1286 \fi
1287 \BitSet@Fi
1288 }

\bitsetCardinality
1289 \def\bitsetCardinality#1{%
1290 \number
1291 \BitSet@IfUndefined{#1}{0 }{%
1292 \expandafter\expandafter\expandafter\BitSet@Cardinality
1293 \expandafter\expandafter\expandafter0%
1294 \expandafter\expandafter\expandafter!%
1295 \csname BS@#1\endcsname!%
1296 }%
1297 }

\BitSet@Cardinality #1: result
#2#3: bits
1298 \def\BitSet@Cardinality#1!#2#3!{%
1299 \ifx#21%
1300 \ifx\#3\%
1301 \BitSet@AfterFiFi{\IntCalcInc#1! }%
1302 \else
1303 \BitSet@AfterFiFi{%
1304 \expandafter\expandafter\expandafter\BitSet@Cardinality
1305 \IntCalcInc#1!!#3!%
1306 }%
1307 \fi
1308 \else
1309 \ifx\#3\%
1310 \BitSet@AfterFiFi{#1 }%
1311 \else
1312 \BitSet@AfterFiFi{%
1313 \BitSet@Cardinality#1!#3!%
1314 }%
1315 \fi
1316 \fi
1317 \BitSet@Fi
1318 }

```

2.12 Queries

```

\bitsetIsDefined
1319 \def\bitsetIsDefined#1{%
1320 \BitSet@IfUndefined{#1}%
1321 \BitSet@SecondOfTwo
1322 \BitSet@FirstOfTwo
1323 }

\bitsetIsEmpty
1324 \def\bitsetIsEmpty#1{%
1325 \BitSet@IfUndefined{#1}\BitSet@FirstOfTwo{%
1326 \expandafter\ifx\csname BS@#1\endcsname\BitSet@Zero
1327 \expandafter\BitSet@FirstOfTwo
1328 \else
1329 \expandafter\BitSet@SecondOfTwo
1330 \fi
1331 }%
1332 }

\BitSet@Zero
1333 \def\BitSet@Zero{0}

```

\bitsetQuery

```

1334 \def\bitsetQuery#1#2{%
1335   \ifnum\bitsetGet{#1}-{#2}=1 %
1336     \expandafter\BitSet@FirstOfTwo
1337   \else
1338     \expandafter\BitSet@SecondOfTwo
1339   \fi
1340 }

```

\bitsetEquals

```

1341 \def\bitsetEquals#1#2{%
1342   \BitSet@IfUndefined{#1}{%
1343     \BitSet@IfUndefined{#2}\BitSet@FirstOfTwo\BitSet@SecondOfTwo
1344   }{%
1345     \BitSet@IfUndefined{#2}\BitSet@SecondOfTwo{%
1346       \expandafter\ifx\csname BS@#1\expandafter\endcsname
1347         \csname BS@#2\endcsname
1348       \expandafter\BitSet@FirstOfTwo
1349     \else
1350       \expandafter\BitSet@SecondOfTwo
1351     \fi
1352   }%
1353 }%
1354 }

```

\bitsetIntersects

```

1355 \def\bitsetIntersects#1#2{%
1356   \bitsetIsEmpty{#1}\BitSet@SecondOfTwo{%
1357     \bitsetIsEmpty{#2}\BitSet@SecondOfTwo{%
1358       \expandafter\expandafter\expandafter\BitSet@Intersects
1359       \csname BS@#1\expandafter\expandafter\expandafter\endcsname
1360       \expandafter\expandafter\expandafter!%
1361       \csname BS@#2\endcsname!%
1362     }%
1363   }%
1364 }

```

\BitSet@Intersects

```

1365 \def\BitSet@Intersects#1#2!#3#4!{%
1366   \ifnum#1#3=11 %
1367     \BitSet@AfterFi\BitSet@FirstOfTwo
1368   \else
1369     \ifx\#2\%
1370       \BitSet@AfterFiFi\BitSet@SecondOfTwo
1371     \else
1372       \ifx\#4\%
1373         \BitSet@AfterFiFiFi\BitSet@SecondOfTwo
1374       \else
1375         \BitSet@AfterFiFiFi{%
1376           \BitSet@Intersects#2!#4!%
1377         }%
1378       \fi
1379     \fi
1380   \BitSet@Fi
1381 }

1382 \BitSet@AtEnd
1383 </package>

```

3 Test

3.1 Catcode checks for loading

```
1384 <*test1>
1385 \catcode'\{=1 %
1386 \catcode'\}=2 %
1387 \catcode'\#=6 %
1388 \catcode'\@=11 %
1389 \expandafter\ifx\csname count@\endcsname\relax
1390 \countdef\count@=255 %
1391 \fi
1392 \expandafter\ifx\csname @gobble\endcsname\relax
1393 \long\def\@gobble#1{}%
1394 \fi
1395 \expandafter\ifx\csname @firstofone\endcsname\relax
1396 \long\def\@firstofone#1{#1}%
1397 \fi
1398 \expandafter\ifx\csname loop\endcsname\relax
1399 \expandafter\@firstofone
1400 \else
1401 \expandafter\@gobble
1402 \fi
1403 {%
1404 \def\loop#1\repeat{%
1405 \def\body{#1}%
1406 \iterate
1407 }%
1408 \def\iterate{%
1409 \body
1410 \let\next\iterate
1411 \else
1412 \let\next\relax
1413 \fi
1414 \next
1415 }%
1416 \let\repeat=\fi
1417 }%
1418 \def\RestoreCatcodes{}
1419 \count@=0 %
1420 \loop
1421 \edef\RestoreCatcodes{%
1422 \RestoreCatcodes
1423 \catcode\the\count@=\the\catcode\count@\relax
1424 }%
1425 \ifnum\count@<255 %
1426 \advance\count@ 1 %
1427 \repeat
1428
1429 \def\RangeCatcodeInvalid#1#2{%
1430 \count@=#1\relax
1431 \loop
1432 \catcode\count@=15 %
1433 \ifnum\count@<#2\relax
1434 \advance\count@ 1 %
1435 \repeat
1436 }
1437 \expandafter\ifx\csname LoadCommand\endcsname\relax
1438 \def\LoadCommand{\input bitset.sty\relax}%
1439 \fi
1440 \def\Test{%
1441 \RangeCatcodeInvalid{0}{47}%
```

```

1442 \RangeCatcodeInvalid{58}{64}%
1443 \RangeCatcodeInvalid{91}{96}%
1444 \RangeCatcodeInvalid{123}{255}%
1445 \catcode'\@=12 %
1446 \catcode'\=0 %
1447 \catcode'\{=1 %
1448 \catcode'\}=2 %
1449 \catcode'\#=6 %
1450 \catcode'\[=12 %
1451 \catcode'\]=12 %
1452 \catcode'\%=14 %
1453 \catcode'\ =10 %
1454 \catcode13=5 %
1455 \LoadCommand
1456 \RestoreCatcodes
1457 }
1458 \Test
1459 \csname @@end\endcsname
1460 \end
1461 </test1>

```

3.2 Macro tests

3.2.1 Preamble

```

1462 <*test2>
1463 \NeedsTeXFormat{LaTeX2e}
1464 \nofiles
1465 \documentclass{article}
1466 \makeatletter
1467 <*noetex>
1468 \let\SavedNumexpr\numexpr
1469 \let\SavedIfcsname\ifcsname
1470 \let\SavedCurrentgrouplevel\currentgrouplevel
1471 \def\ETeXDisable{%
1472   \let\ifcsname\@undefined
1473   \let\numexpr\@undefined
1474   \let\currentgrouplevel\@undefined
1475 }
1476 \ETeXDisable
1477 </noetex>
1478 \makeatletter
1479 \chardef\BitSet@TestMode=1 %
1480 \makeatother
1481 \usepackage{bitset}[2007/09/28]
1482 <*noetex>
1483 \def\ETeXEnable{%
1484   \let\numexpr\SavedNumexpr
1485   \let\ifcsname\SavedIfcsname
1486   \let\currentgrouplevel\SavedCurrentgrouplevel
1487 }
1488 \ETeXEnable
1489 </noetex>
1490 \usepackage{qstest}
1491 \IncludeTests{*}
1492 \LogTests{log}{*}{*}
1493 \makeatletter

```

3.2.2 Time

```

1494 \begingroup\expandafter\expandafter\expandafter\endgroup
1495 \expandafter\ifx\csname pdfresettimer\endcsname\relax
1496 \else
1497   \newcount\SummaryTime

```

```

1498 \newcount\TestTime
1499 \SummaryTime=\z@
1500 \newcommand*\PrintTime}[2]{%
1501   \typeout{%
1502     [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
1503   }%
1504 }%
1505 \newcommand*\StartTime#[1]{%
1506   \renewcommand*\TimeDescription}{#1}%
1507   \pdfresettimer
1508 }%
1509 \newcommand*\TimeDescription{}%
1510 \newcommand*\StopTime{}%
1511   \TestTime=\pdfelapsedtime
1512   \global\advance\SummaryTime\TestTime
1513   \PrintTime\TimeDescription\TestTime
1514 }%
1515 \let\saved@qstest\qstest
1516 \let\saved@endqstest\endqstest
1517 \def\qstest#1#2{%
1518   \saved@qstest{#1}{#2}%
1519   \StartTime{#1}%
1520 }%
1521 \def\endqstest{%
1522   \StopTime
1523   \saved@endqstest
1524 }%
1525 \AtEndDocument{%
1526   \PrintTime{summary}\SummaryTime
1527 }%
1528 \fi

```

3.2.3 Detection of unwanted space

```

1529 \let\orig@qstest\qstest
1530 \let\orig@endqstest\endqstest
1531 \def\qstest#1#2{%
1532   \orig@qstest{#1}{#2}%
1533   \setbox0\hbox\bgroup\beginingroup\ignorespaces
1534 }
1535 \def\endqstest{%
1536   \endgroup\egroup
1537   \Expect*\the\wd0}{0.0pt}%
1538   \orig@endqstest
1539 }

```

3.2.4 Test macros

```

1540 \newcounter{Test}
1541
1542 \def\TestError#1#2{%
1543   \beginingroup
1544     \setcounter{Test}{0}%
1545     \sbox0{%
1546       \def\@PackageError##1##2##3{%
1547         \stepcounter{Test}%
1548         \beginingroup
1549           \let\MessageBreak\relax
1550 \noetex}
1551       \ETEXEnable
1552 \noetex}
1553       \Expect{##1}{bitset}%
1554       \Expect*{##2}*{#1}%
1555     \endgroup
1556   }%

```



```

1557 <*noetex>
1558     \ETeXDisable
1559 </noetex>
1560     #2%
1561 }%
1562 \Expect*{\theTest}{1}%
1563 \Expect*{\the\wd0}{0.0pt}%
1564 \endgroup
1565 }
1566
1567 \def\TestErrorNegativeIndex#1#2{%
1568   \TestError{Invalid negative index (#1)}{#2}%
1569 }
1570
1571 \def\TestGetterUndefined#1{%
1572   \CheckUndef{dummy}%
1573   \expandafter\expandafter\expandafter\Expect
1574   \expandafter\expandafter\expandafter{#1{dummy}}{0}%
1575 }
1576
1577 \def\ExpectBitSet#1#2{%
1578   \expandafter\expandafter\expandafter\Expect
1579   \expandafter\expandafter\expandafter
1580   {\csname BS@#1\endcsname}*{#2}%
1581 }
1582 \def\Check#1#2{%
1583   \ExpectBitSet{#1}{#2}%
1584 }
1585 \def\CheckUndef#1{%
1586   \begingroup
1587     \Expect*{%
1588       \expandafter
1589       \ifx\csname BS@#1\endcsname\relax true\else false\fi
1590     }{true}%
1591   \endgroup
1592 }
1593 \def\RevCheck#1#2{%
1594   \ExpectBitSet{#1}{\Reverse#2!!}%
1595 }
1596 \def\Set#1#2{%
1597   \expandafter\def\csname BS@#1\endcsname{#2}%
1598 }
1599 \def\RevSet#1#2{%
1600   \expandafter\edef\csname BS@#1\endcsname{%
1601     \Reverse#2!!%
1602   }%
1603 }
1604 \def\Reverse#1#2!#3!{%
1605   \ifx\#2\%
1606     #1#3%
1607     \expandafter\@gobble
1608   \else
1609     \expandafter\@firstofone
1610   \fi
1611   {\Reverse#2!#1#3!}%
1612 }

```

3.2.5 Test sets

```

1613 \begin{qstest}{Let}{Let}
1614   \CheckUndef{abc}%
1615   \CheckUndef{xyz}%
1616   \bitsetLet{xyz}{abc}%
1617   \CheckUndef{abc}%

```

```

1618 \Check{xyz}{0}%
1619 \Set{abc}{1}%
1620 \Check{abc}{1}%
1621 \Check{xyz}{0}%
1622 \bitsetLet{xyz}{abc}%
1623 \Check{abc}{1}%
1624 \Check{xyz}{1}%
1625 \Set{xyz}{11}%
1626 \Check{abc}{1}%
1627 \Check{xyz}{11}%
1628 \end{qstest}
1629
1630 \begin{qstest}{Reset}{Reset}
1631 \bitsetReset{xyz}%
1632 \Check{xyz}{0}%
1633 \bitsetReset{abc}%
1634 \Check{abc}{0}%
1635 \Set{abc}{10101}%
1636 \bitsetReset{abc}%
1637 \Check{abc}{0}%
1638 \end{qstest}
1639
1640 \begin{qstest}{Get/Query}{Get/Query}
1641 \expandafter\expandafter\expandafter\Expect
1642 \expandafter\expandafter\expandafter{%
1643 \bitsetGet{dummy}{0}%
1644 }{0}%
1645 \begingroup
1646 \expandafter\def\csname BitSetError:NegativeIndex\endcsname{}%
1647 \Set{abc}{1}%
1648 \Expect*{\bitsetQuery{abc}{-1}{true}{false}}{false}%
1649 \endgroup
1650 \def\Test#1#2#3{%
1651 \Set{abc}{#1}%
1652 \expandafter\expandafter\expandafter\Expect
1653 \expandafter\expandafter\expandafter{\bitsetGet{abc}{#2}}{#3}%
1654 \Expect*{\bitsetQuery{abc}{#2}{true}{false}}%
1655 *{\ifcase#3 false\or true\else error\fi}%
1656 }%
1657 \Test{1}{100}{0}%
1658 \Test{0}{0}{0}%
1659 \Test{1}{0}{1}%
1660 \Test{11}{1}{1}%
1661 \Test{111}{1}{1}%
1662 \Test{101}{1}{0}%
1663 \Test{101}{2}{1}%
1664 \Test{10100110011}{10}{1}%
1665 \end{qstest}
1666
1667 \begin{qstest}{Size}{Size}
1668 \TestGetterUndefined\bitsetSize
1669 \def\Test#1#2{%
1670 \Set{abc}{#1}%
1671 \expandafter\expandafter\expandafter\Expect
1672 \expandafter\expandafter\expandafter{\bitsetSize{abc}}{#2}%
1673 }%
1674 \Test{0}{0}%
1675 \Test{1}{1}%
1676 \Test{00}{0}%
1677 \Test{0000000}{0}%
1678 \Test{10}{1}%
1679 \Test{01}{2}%

```

```

1680 \Test{11}{2}%
1681 \Test{010}{2}%
1682 \Test{011}{3}%
1683 \Test{100110011}{9}%
1684 \Test{0000011111000001111100000}{20}%
1685 \Test{00000000000000000000000001111111111111111111}{45}%
1686 \end{qstest}
1687
1688 \begin{qstest}{Cardinality}{Cardinality}
1689 \TestGetterUndefined\bitsetCardinality
1690 \def\Test#1#2{%
1691   \Set{abc}{#1}%
1692   \expandafter\expandafter\expandafter\Expect
1693   \expandafter\expandafter\expandafter{%
1694     \bitsetCardinality{abc}%
1695   }{#2}%
1696 }%
1697 \Test{0}{0}%
1698 \Test{1}{1}%
1699 \Test{00}{0}%
1700 \Test{0000000}{0}%
1701 \Test{10}{1}%
1702 \Test{01}{1}%
1703 \Test{11}{2}%
1704 \Test{010}{1}%
1705 \Test{011}{2}%
1706 \Test{100110011}{5}%
1707 \Test{0000011111000001111100000}{10}%
1708 \Test{00000000000000000000000000011111111111111111}{20}%
1709 \end{qstest}
1710
1711 \begin{qstest}{NextClearBit/NextSetBit}{NextClearBit/NextSetBit}
1712 \def\Test#1#2{%
1713   \expandafter\expandafter\expandafter\Expect
1714   \expandafter\expandafter\expandafter{%
1715     \TestOp{abc}{#1}%
1716   }{#2}%
1717 }%
1718 \def\CLEAR{\let\TestOp\bitsetNextClearBit}%
1719 \def\SET{\let\TestOp\bitsetNextSetBit}%
1720 \begingroup
1721   \catcode'\:=11 %
1722   \bitsetSetBin{abc}{1}%
1723   \CLEAR
1724   \Test{-1}{1\BitSetError:NegativeIndex}%
1725   \SET
1726   \Test{-1}{0\BitSetError:NegativeIndex}%
1727 \endgroup
1728 \let\BS@abc\undefined
1729 \CLEAR
1730 \Test{0}{0}%
1731 \Test{1}{1}%
1732 \Test{2}{2}%
1733 \Test{100}{100}%
1734 \SET
1735 \Test{0}{-1}%
1736 \Test{1}{-1}%
1737 \Test{100}{-1}%
1738 \bitsetReset{abc}%
1739 \CLEAR
1740 \Test{0}{0}%
1741 \Test{1}{1}%

```

```

1742 \Test{2}{2}%
1743 \Test{100}{100}%
1744 \Set
1745 \Test{0}{-1}%
1746 \Test{1}{-1}%
1747 \Test{100}{-1}%
1748 \bitsetSetBin{abc}{1}%
1749 \Clear
1750 \Test{0}{1}%
1751 \Test{1}{1}%
1752 \Test{2}{2}%
1753 \Test{100}{100}%
1754 \Set
1755 \Test{0}{0}%
1756 \Test{1}{-1}%
1757 \Test{100}{-1}%
1758 \bitsetSetBin{abc}{111000111000111000111}%
1759 \Clear
1760 \Test{0}{3}%
1761 \Test{1}{3}%
1762 \Test{2}{3}%
1763 \Test{3}{3}%
1764 \Test{4}{4}%
1765 \Test{5}{5}%
1766 \Test{6}{9}%
1767 \Test{7}{9}%
1768 \Test{8}{9}%
1769 \Test{9}{9}%
1770 \Test{10}{10}%
1771 \Test{11}{11}%
1772 \Test{12}{15}%
1773 \Test{13}{15}%
1774 \Test{14}{15}%
1775 \Test{15}{15}%
1776 \Test{16}{16}%
1777 \Test{17}{17}%
1778 \Test{18}{21}%
1779 \Test{19}{21}%
1780 \Test{20}{21}%
1781 \Test{21}{21}%
1782 \Test{22}{22}%
1783 \Test{100}{100}%
1784 \Set
1785 \Test{0}{0}%
1786 \Test{1}{1}%
1787 \Test{2}{2}%
1788 \Test{3}{6}%
1789 \Test{4}{6}%
1790 \Test{5}{6}%
1791 \Test{6}{6}%
1792 \Test{7}{7}%
1793 \Test{8}{8}%
1794 \Test{9}{12}%
1795 \Test{10}{12}%
1796 \Test{11}{12}%
1797 \Test{12}{12}%
1798 \Test{13}{13}%
1799 \Test{14}{14}%
1800 \Test{15}{18}%
1801 \Test{16}{18}%
1802 \Test{17}{18}%
1803 \Test{18}{18}%

```

```

1804 \Test{19}{19}%
1805 \Test{20}{20}%
1806 \Test{21}{-1}%
1807 \Test{22}{-1}%
1808 \Test{100}{-1}%
1809 \bitsetSetBin{abc}{1111111}%
1810 \Clear
1811 \Test{6}{7}%
1812 \Test{7}{7}%
1813 \Test{8}{8}%
1814 \Test{100}{100}%
1815 \Set
1816 \Test{6}{6}%
1817 \Test{7}{-1}%
1818 \Test{8}{-1}%
1819 \Test{100}{-1}%
1820 \bitsetSetBin{abc}{11111111}%
1821 \Clear
1822 \Test{7}{8}%
1823 \Test{8}{8}%
1824 \Test{9}{9}%
1825 \Test{100}{100}%
1826 \Set
1827 \Test{7}{7}%
1828 \Test{8}{-1}%
1829 \Test{9}{-1}%
1830 \Test{100}{-1}%
1831 \bitsetSetBin{abc}{111111111}%
1832 \Clear
1833 \Test{8}{9}%
1834 \Test{9}{9}%
1835 \Test{10}{10}%
1836 \Test{100}{100}%
1837 \Set
1838 \Test{8}{8}%
1839 \Test{9}{-1}%
1840 \Test{10}{-1}%
1841 \Test{100}{-1}%
1842 \bitsetSetBin{abc}{1111111111}%
1843 \Clear
1844 \Test{9}{10}%
1845 \Test{10}{10}%
1846 \Test{11}{11}%
1847 \Test{100}{100}%
1848 \Set
1849 \Test{9}{9}%
1850 \Test{10}{-1}%
1851 \Test{11}{-1}%
1852 \Test{100}{-1}%
1853 \end{qstest}
1854
1855 \begin{qstest}{GetSetBitList}{GetSetBitList}
1856 \let\BS@abc\@undefined
1857 \expandafter\expandafter\expandafter\Expect
1858 \expandafter\expandafter\expandafter{%
1859 \bitsetGetSetBitList{abc}%
1860 }{}%
1861 \def\Test#1#2{%
1862 \bitsetSetBin{abc}{#1}%
1863 \expandafter\expandafter\expandafter\Expect
1864 \expandafter\expandafter\expandafter{%
1865 \bitsetGetSetBitList{abc}%

```

```

1866     }{#2}%
1867 }%
1868 \Test{0}{}%
1869 \Test{1}{0}%
1870 \Test{10}{1}%
1871 \Test{11}{0,1}%
1872 \Test{10110100}{2,4,5,7}%
1873 \Test{101101001010011}{0,1,4,6,9,11,12,14}%
1874 \end{qstest}
1875
1876 \begin{qstest}{GetDec}{GetDec}
1877 \TestGetterUndefined\bitsetGetDec
1878 \def\Test#1#2{%
1879     \RevSet{abc}{#1}%
1880 }*noetex
1881 \begin{group}\expandafter\expandafter\expandafter\endgroup
1882 

```

```

1928 \Test{1001001}{3}{1000001}%
1929 \end{qstest}
1930
1931 \begin{qstest}{Set}{Set}
1932 \def\Test#1#2#3{%
1933 \RevSet{abc}{#1}%
1934 \bitsetSet{abc}{#2}%
1935 \Expect*{\BS@abc}*{\Reverse#3!!}%
1936 }%
1937 \bitsetSet{abc}{2}%
1938 \RevCheck{abc}{100}%
1939 \TestErrorNegativeIndex{-1}{\bitsetSet{abc}{-1}}%
1940 \RevCheck{abc}{100}%
1941 \Test{0}{0}{1}%
1942 \Test{1}{0}{1}%
1943 \Test{100}{1}{110}%
1944 \Test{111}{1}{111}%
1945 \Test{11}{1}{11}%
1946 \Test{11}{2}{111}%
1947 \Test{11}{3}{1011}%
1948 \Test{111}{10}{10000000111}%
1949 \Test{0000111}{5}{0100111}% 100111 would also be ok
1950 \Test{10000111}{5}{10100111}%
1951 \Test{1000001}{3}{1001001}%
1952 \Test{1001001}{3}{1001001}%
1953 \end{qstest}
1954
1955 \begin{qstest}{Flip}{Flip}
1956 \def\Test#1#2#3{%
1957 \RevSet{abc}{#1}%
1958 \bitsetFlip{abc}{#2}%
1959 \Expect*{\BS@abc}*{\Reverse#3!!}%
1960 }%
1961 \bitsetFlip{abc}{2}%
1962 \RevCheck{abc}{100}%
1963 \TestErrorNegativeIndex{-1}{\bitsetFlip{abc}{-1}}%
1964 \RevCheck{abc}{100}%
1965 \Test{0}{0}{1}%
1966 \Test{1}{0}{0}%
1967 \Test{0}{2}{100}%
1968 \Test{100}{1}{110}%
1969 \Test{111}{1}{101}%
1970 \Test{11}{1}{1}%
1971 \Test{11}{2}{111}%
1972 \Test{11}{3}{1011}%
1973 \Test{111}{10}{10000000111}%
1974 \Test{0000111}{5}{0100111}% 100111 would also be ok
1975 \Test{10000111}{5}{10100111}%
1976 \Test{1000001}{3}{1001001}%
1977 \Test{1001001}{3}{1000001}%
1978 \Test{11111}{2}{11011}%
1979 \end{qstest}
1980
1981 \begin{qstest}{SetValue}{SetValue}
1982 \def\Test#1#2{%
1983 \TestError{Invalid bit value (#2) not in range 0..1}{%
1984 \bitsetSetValue{abc}{#1}{#2}%
1985 }%
1986 }%
1987 \Test{0}{-1}%
1988 \Test{0}{2}%
1989 \Test{0}{10}%

```

```

1990 \def\Test#1#2#3{%
1991   \let\BS@abc\@undefined
1992   \bitsetSetValue{abc}{#1}{#2}%
1993   \bitsetSetBin{result}{#3}%
1994   \Expect*{\BS@abc}*{\BS@result}%
1995 }%
1996 \Test{0}{0}{0}%
1997 \Test{0}{1}{1}%
1998 \Test{1}{0}{0}%
1999 \Test{1}{1}{10}%
2000 \def\Test#1#2#3#4{%
2001   \bitsetSetBin{abc}{#1}%
2002   \bitsetSetBin{result}{#4}%
2003   \bitsetSetValue{abc}{#2}{#3}%
2004   \Expect*{\BS@abc}*{\BS@result}%
2005 }%
2006 \Test{0}{0}{0}{0}%
2007 \Test{0}{0}{0}{0}%
2008 \Test{0}{0}{1}{1}%
2009 \Test{0}{1}{0}{0}%
2010 \Test{0}{1}{1}{10}%
2011 \Test{1010}{2}{1}{1110}%
2012 \Test{1010}{4}{1}{11010}%
2013 \Test{1010}{6}{1}{1001010}%
2014 \Test{1010}{1}{0}{1000}%
2015 \Test{1010}{2}{0}{1010}%
2016 \Test{1010}{3}{0}{10}%
2017 \Test{1010}{4}{0}{1010}%
2018 \Test{1010}{6}{0}{1010}%
2019 \Test{1010}{2}{\csname iffalse\endcsname 0\else 1\fi}{1110}%
2020 \Test{1010}{1}{\csname iffalse\endcsname 1\else 0\fi}{1000}%
2021 \end{qstest}
2022
2023 \begin{qstest}{IsDefined}{IsDefined}
2024   \let\BS@abc\@undefined
2025   \Expect*{\bitsetIsDefined{abc}{true}{false}}{false}%
2026   \bitsetReset{abc}%
2027   \Expect*{\bitsetIsDefined{abc}{true}{false}}{true}%
2028 \end{qstest}
2029
2030 \begin{qstest}{IsEmpty}{IsEmpty}
2031   \let\BS@abc\@undefined
2032   \Expect*{\bitsetIsEmpty{abc}{true}{false}}{true}%
2033   \bitsetReset{abc}%
2034   \Expect*{\bitsetIsEmpty{abc}{true}{false}}{true}%
2035   \bitsetSet{abc}{1}%
2036   \Expect*{\bitsetIsEmpty{abc}{true}{false}}{false}%
2037 \end{qstest}
2038
2039 \begin{qstest}{Equals}{Equals}
2040   \def\Test#1#2#3{%
2041     \Expect*{\bitsetEquals{#1}{#2}{true}{false}}{#3}%
2042   }%
2043   \let\BS@abc\@undefined
2044   \Test{abc}{abc}{true}%
2045   \Test{abc}{foo}{true}%
2046   \Test{foo}{abc}{true}%
2047   \bitsetReset{abc}%
2048   \Test{abc}{abc}{true}%
2049   \Test{abc}{foo}{false}%
2050   \Test{foo}{abc}{false}%
2051   \bitsetReset{foo}%

```



```

2052 \Test{abc}{foo}{true}%
2053 \Test{foo}{abc}{true}%
2054 \bitsetSet{abc}{4}%
2055 \Test{abc}{foo}{false}%
2056 \Test{foo}{abc}{false}%
2057 \bitsetFlip{foo}{4}%
2058 \Test{abc}{foo}{true}%
2059 \Test{foo}{abc}{true}%
2060 \end{qstest}
2061
2062 \begin{qstest}{Intersects}{Intersects}
2063 \def\Test#1{%
2064 \Expect*{\bitsetIntersects{abc}{foo}{true}{false}}{#1}%
2065 }%
2066 \let\BS@abc\@undefined
2067 \let\BS@foo\@undefined
2068 \Test{false}%
2069 \Set{abc}{0}%
2070 \Test{false}%
2071 \Set{foo}{0}%
2072 \Test{false}%
2073 \let\BS@abc\@undefined
2074 \Test{false}%
2075 \Set{foo}{1}%
2076 \Test{false}%
2077 \Set{abc}{0}%
2078 \Test{false}%
2079 \Set{abc}{1}%
2080 \Test{true}%
2081 \let\BS@foo\@undefined
2082 \Test{false}%
2083 \Set{foo}{0}%
2084 \Test{false}%
2085 \def\Test#1#2#3{%
2086 \bitsetSetBin{abc}{#1}%
2087 \bitsetSetBin{foo}{#2}%
2088 \Expect*{\bitsetIntersects{abc}{foo}{true}{false}}{#3}%
2089 }%
2090 \Test{1010}{0101}{false}%
2091 \Test{0}{10}{false}%
2092 \Test{1}{11}{true}%
2093 \Test{11}{1}{true}%
2094 \Test{10}{1}{false}%
2095 \end{qstest}
2096
2097 \begin{qstest}{And/AndNot/Or/Xor}{And/AndNot/Or/Xor}
2098 \def\@Test#1#2#3#4#5{%
2099 \begingroup
2100 #5%
2101 \begingroup
2102 \let\BS@foo\@undefined
2103 \csname bitset#1\endcsname{abc}{foo}%
2104 \CheckUndef{foo}%
2105 \Check{abc}{#2}%
2106 \endgroup
2107 \begingroup
2108 \bitsetReset{foo}%
2109 \csname bitset#1\endcsname{abc}{foo}%
2110 \Check{foo}{0}%
2111 \Check{abc}{#3}%
2112 \endgroup
2113 \begingroup

```

```

2114         \def\BS@foo{0101}%
2115         \csname bitset#1\endcsname{abc}{foo}%
2116         \Check{foo}{0101}%
2117         \Check{abc}{#4}%
2118     \endgroup
2119 \endgroup
2120 }%
2121 \def\Test#1{%
2122     \def\Op{#1}%
2123     \Test@
2124 }%
2125 \def\Test@#1#2#3#4#5#6#7#8#9{%
2126     \@Test\Op{#1}{#2}{#3}{%
2127         \let\BS@abc\undefined
2128     }%
2129     \@Test\Op{#4}{#5}{#6}{%
2130         \bitsetReset{abc}%
2131     }%
2132     \@Test\Op{#7}{#8}{#9}{%
2133         \def\BS@abc{1001}%
2134     }%
2135 }%
2136 \Test{And}%
2137     {0}{0}{0}%
2138     {0}{0}{0}%
2139     {0}{0}{0001}%
2140 \Test{AndNot}%
2141     {0}{0}{0}%
2142     {0}{0}{0}%
2143     {1001}{1001}{1}%
2144 \Test{Or}%
2145     {0}{0}{0101}%
2146     {0}{0}{0101}%
2147     {1001}{1001}{1101}%
2148 \Test{Xor}%
2149     {0}{0}{0101}%
2150     {0}{0}{0101}%
2151     {1001}{1001}{11}%
2152 \def\Test#1#2#3{%
2153     \bitsetSetBin{abc}{#1}%
2154     \bitsetSetBin{foo}{#2}%
2155     \csname bitset\Op\endcsname{abc}{foo}%
2156     \RevCheck{foo}{#2}%
2157     \RevCheck{abc}{#3}%
2158 }%
2159 \def\Op{And}%
2160 \Test{1}{111}{1}%
2161 \Test{111}{1}{1}%
2162 \Test{10}{111}{10}%
2163 \Test{111}{10}{10}%
2164 \Test{111}{1000}{0}%
2165 \Test{1000}{111}{0}%
2166 \def\Op{AndNot}%
2167 \Test{1010}{11}{1000}%
2168 \Test{100}{100}{0}%
2169 \Test{111}{1111}{0}%
2170 \Test{100}{111}{0}%
2171 \def\Op{Or}%
2172 \Test{0}{0}{0}%
2173 \Test{1}{0}{1}%
2174 \Test{0}{1}{1}%
2175 \Test{1}{1}{1}%

```

```

2176 \Test{1000}{10}{1010}%
2177 \Test{10}{1000}{1010}%
2178 \def\Op{Xor}%
2179 \Test{0}{0}{0}%
2180 \Test{1}{0}{1}%
2181 \Test{0}{1}{1}%
2182 \Test{1}{1}{0}%
2183 \Test{1000}{10}{1010}%
2184 \Test{10}{1000}{1010}%
2185 \Test {110011001100}%
2186 {111000111000111}%
2187 {111110100001011}%
2188 \Test{111000111000111}%
2189 {110011001100}%
2190 {111110100001011}%
2191 \end{qstest}
2192
2193 \begin{qstest}{\GetUndef}{\GetUndef, GetBin, GetOct, GetHex}
2194 \def\TestUndef#1#2{%
2195 \let\BS@abc\@undefined
2196 \expandafter\expandafter\expandafter\Expect
2197 \expandafter\expandafter\expandafter{%
2198 \x{abc}{#1}%
2199 }{#2}%
2200 }%
2201 \let\x\bitsetGetBin
2202 \TestUndef{-1}{0}%
2203 \TestUndef{0}{0}%
2204 \TestUndef{1}{0}%
2205 \TestUndef{2}{00}%
2206 \TestUndef{8}{00000000}%
2207 \let\x\bitsetGetOct
2208 \TestUndef{-1}{0}%
2209 \TestUndef{0}{0}%
2210 \TestUndef{1}{0}%
2211 \TestUndef{2}{0}%
2212 \TestUndef{3}{0}%
2213 \TestUndef{4}{00}%
2214 \TestUndef{5}{00}%
2215 \TestUndef{6}{00}%
2216 \TestUndef{7}{000}%
2217 \TestUndef{8}{000}%
2218 \TestUndef{9}{000}%
2219 \TestUndef{10}{0000}%
2220 \let\x\bitsetGetHex
2221 \TestUndef{-1}{0}%
2222 \TestUndef{0}{0}%
2223 \TestUndef{1}{0}%
2224 \TestUndef{2}{0}%
2225 \TestUndef{3}{0}%
2226 \TestUndef{4}{0}%
2227 \TestUndef{5}{00}%
2228 \TestUndef{6}{00}%
2229 \TestUndef{7}{00}%
2230 \TestUndef{8}{00}%
2231 \TestUndef{9}{000}%
2232 \TestUndef{10}{000}%
2233 \TestUndef{12}{000}%
2234 \TestUndef{13}{0000}%
2235 \TestUndef{16}{0000}%
2236 \TestUndef{17}{00000}%
2237 \end{qstest}

```

```

2238
2239 \begin{qstest}{SetBin}{SetBin}
2240   \def\Test#1#2{%
2241     \let\BS@abc\@undefined
2242     \bitsetSetBin{abc}{#1}%
2243     \expandafter\Expect\expandafter{\BS@abc}{#2}%
2244   }%
2245   \Test{}{0}%
2246   \Test{0}{0}%
2247   \Test{1}{1}%
2248   \Test{10}{01}%
2249   \Test{11}{11}%
2250   \Test{010}{01}%
2251   \Test{011}{11}%
2252   \Test{0010}{01}%
2253   \Test{1010}{0101}%
2254 \end{qstest}
2255
2256 \begin{qstest}{SetOct}{SetOct}
2257   \def\Test#1#2{%
2258     \bitsetSetOct{abc}{#1}%
2259     \expandafter\Expect\expandafter{\BS@abc}{#2}%
2260   }%
2261   \Test{}{0}%
2262   \Test{0}{0}%
2263   \Test{000}{0}%
2264   \Test{1}{1}%
2265   \Test{001}{1}%
2266   \Test{010}{0001}%
2267   \Test{020}{00001}%
2268   \Test{42}{010001}%
2269   \Test{377}{11111111}%
2270   \Test{0377}{11111111}%
2271   \Test{76543210}{000100010110001101011111}%
2272   \Test{ 0 7 0 7 1 }{100111000111}%
2273 \end{qstest}
2274
2275 \begin{qstest}{SetHex}{SetHex}
2276   \def\Test#1#2{%
2277     \bitsetSetHex{abc}{#1}%
2278     \expandafter\Expect\expandafter{\BS@abc}{#2}%
2279   }%
2280   \Test{}{0}%
2281   \Test{0}{0}%
2282   \Test{000}{0}%
2283   \Test{1}{1}%
2284   \Test{001}{1}%
2285   \Test{010}{00001}%
2286   \Test{020}{000001}%
2287   \Test{42}{0100001}%
2288   \Test{3F}{111111}%
2289   \Test{03F}{111111}%
2290   \Test{43210}{0000100001001100001}%
2291   \Test{98765}{10100110111000011001}%
2292   \Test{FEDCBA}{010111010011101101111111}%
2293   \Test{ 0 F 0 F 1 }{1000111100001111}%
2294 \end{qstest}
2295
2296 \begin{qstest}{SetDec}{SetDec}
2297   \def\Test#1#2{%
2298     \bitsetSetDec{abc}{#1}%
2299     \expandafter\Expect\expandafter{\BS@abc}{#2}%

```

```

2300 }%
2301 \Test{}{0}%
2302 \Test{0}{0}%
2303 \Test{000}{0}%
2304 \Test{1}{1}%
2305 \Test{7}{111}%
2306 \Test{8}{0001}%
2307 \Test{001}{1}%
2308 \Test{010}{0101}%
2309 \Test{020}{00101}%
2310 \Test{53}{101011}%
2311 \Test{255}{11111111}%
2312 \Test{256}{000000001}%
2313 \Test{999999999}{11111111001001101011001110111}%
2314 \Test{1000000000}{000000000101001101011001110111}%
2315 \Test{4210987654}{011000010100100101111110101111}%
2316 \Test{2147483647}{11111111111111111111111111111111}%
2317 \Test{2147483648}{00000000000000000000000000000001}%
2318 \end{qstest}
2319
2320 \begin{qstest}{GetBin}{GetBin}
2321 \def\TestUndef#1#2{%
2322 \let\BS@abc\@undefined
2323 \expandafter\expandafter\expandafter\Expect
2324 \expandafter\expandafter\expandafter{%
2325 \bitsetGetBin{abc}{#1}%
2326 }{#2}%
2327 }%
2328 \TestUndef{-1}{0}%
2329 \TestUndef{0}{0}%
2330 \TestUndef{1}{0}%
2331 \TestUndef{2}{00}%
2332 \TestUndef{8}{00000000}%
2333 \def\Test#1#2{%
2334 \bitsetSetBin{abc}{#2}%
2335 \expandafter\expandafter\expandafter\Expect
2336 \expandafter\expandafter\expandafter{%
2337 \bitsetGetBin{abc}{#1}%
2338 }{#2}%
2339 }%
2340 \Test{-1}{0}%
2341 \Test{0}{0}%
2342 \Test{1}{0}%
2343 \Test{1}{1}%
2344 \Test{2}{01}%
2345 \Test{2}{10}%
2346 \Test{3}{010}%
2347 \Test{2}{00}%
2348 \Test{2}{01}%
2349 \Test{8}{00101100}%
2350 \Test{2}{10101}%
2351 \Test{-100}{11011}%
2352 \end{qstest}
2353
2354 \begin{qstest}{GetOct}{GetOct}
2355 \def\Test#1#2#3{%
2356 \edef\x{\zap@space#1 \@empty}%
2357 \edef\x{\noexpand\bitsetSetBin{abc}{\x}}%
2358 \x
2359 \expandafter\expandafter\expandafter\Expect
2360 \expandafter\expandafter\expandafter{%
2361 \bitsetGetOct{abc}{#2}%

```

```

2362     }{#3}%
2363 }%
2364 \Test{111 110 101 100 011 010 001 000}{0}{76543210}%
2365 \Test{000 111}{0}{7}%
2366 \Test{101 000}{-1}{50}%
2367 \Test{111}{-1}{7}%
2368 \Test{111}{0}{7}%
2369 \Test{111}{1}{7}%
2370 \Test{111}{3}{7}%
2371 \Test{111}{4}{07}%
2372 \Test{111}{6}{07}%
2373 \Test{111}{7}{007}%
2374 \Test{111 010}{6}{72}%
2375 \Test{111 010}{7}{072}%
2376 \Test{011 111}{0}{37}%
2377 \Test{011 111}{6}{37}%
2378 \Test{011 111}{7}{037}%
2379 \Test{001 111}{0}{17}%
2380 \Test{001 111}{6}{17}%
2381 \Test{001 111}{7}{017}%
2382 \end{qstest}
2383
2384 \begin{qstest}{GetHex}{GetHex}
2385   \def\Test#1#2#3{%
2386     \bitsetSetBin{abc}{#1}%
2387     \expandafter\expandafter\expandafter\Expect
2388     \expandafter\expandafter\expandafter{%
2389       \bitsetGetHex{abc}{#2}%
2390     }{#3}%
2391   }%
2392   \Test{1111 1110 1101 1100 1011 1010 1001 1000}{0}{FEDCBA98}%
2393   \Test{0111 0110 0101 0100 0011 0010 0001 0000}{0}{76543210}%
2394   \Test{0000 1111}{0}{F}%
2395   \Test{0101 0000}{-1}{50}%
2396   \Test{1111}{-1}{F}%
2397   \Test{1111}{0}{F}%
2398   \Test{1111}{1}{F}%
2399   \Test{1111}{4}{F}%
2400   \Test{1111}{5}{0F}%
2401   \Test{1111}{8}{0F}%
2402   \Test{1111}{9}{00F}%
2403   \Test{1111 0010}{8}{F2}%
2404   \Test{1111 0010}{9}{0F2}%
2405   \Test{0111 1111}{0}{7F}%
2406   \Test{0111 1111}{8}{7F}%
2407   \Test{0111 1111}{9}{07F}%
2408   \Test{0011 1111}{0}{3F}%
2409   \Test{0011 1111}{8}{3F}%
2410   \Test{0011 1111}{9}{03F}%
2411   \Test{0001 1111}{0}{1F}%
2412   \Test{0001 1111}{8}{1F}%
2413   \Test{0001 1111}{9}{01F}%
2414 \end{qstest}
2415
2416 \begin{qstest}{Range}{Range}
2417   \TestError{%
2418     Wrong index numbers in range [9..8]\MessageBreak% hash-ok
2419     for clear/set/flip on bit set 'abc'.\MessageBreak
2420     The lower index exceeds the upper index.\MessageBreak
2421     Canceling the operation as error recovery%
2422   }{%
2423     \bitsetSetRange{abc}{9}{8}%

```

```

2424 }%
2425 \def\TestErrorNegInd#1#2#3#4#5#6{%
2426   \TestError{%
2427     Negative index in range [#2..#3]\MessageBreak % hash-ok
2428     for \string\bitset #1Range on bit set 'abc'.\MessageBreak
2429     Using [#4..#5] as error recovery% hash-ok
2430   }{%
2431     \csname bitset#1Range\endcsname{abc}{#2}{#3}%
2432     \global\let\BS@global\BS@abc
2433   }%
2434   \Check{global}{#6}%
2435 }%
2436 \Set{abc}{111}%
2437 \TestErrorNegInd{Clear}{-1}{0}{0}{0}{111}%
2438 \TestErrorNegInd{Clear}{0}{-1}{0}{0}{111}%
2439 \TestErrorNegInd{Clear}{-2}{2}{0}{2}{001}%
2440 \bitsetReset{abc}%
2441 \TestErrorNegInd{Set}{-1}{0}{0}{0}{0}%
2442 \TestErrorNegInd{Set}{0}{-1}{0}{0}{0}%
2443 \TestErrorNegInd{Set}{-2}{2}{0}{2}{11}%
2444 \Set{abc}{101}%
2445 \TestErrorNegInd{Flip}{-1}{0}{0}{0}{101}%
2446 \TestErrorNegInd{Flip}{0}{-1}{0}{0}{101}%
2447 \TestErrorNegInd{Flip}{-2}{2}{0}{2}{011}%
2448 \def\Test#1#2#3#4{%
2449   \bitsetSetBin{abc}{#1}%
2450   \csname bitset\TestOp Range\endcsname{abc}{#2}{#3}%
2451   \Expect*{\bitsetGetBin{abc}{0}}{#4}%
2452 }%
2453 \def\TestOp{Clear}%
2454 \Test{0}{0}{1}{0}%
2455 \Test{1111}{1}{2}{1101}%
2456 \Test{1111}{1}{3}{1001}%
2457 \Test{1111111100000000}{12}{14}{1100111100000000}%
2458 \def\TestOp{Set}%
2459 \Test{0}{0}{1}{1}%
2460 \Test{1000}{1}{2}{1010}%
2461 \Test{0}{1}{2}{10}%
2462 \Test{1}{12}{15}{111000000000001}%
2463 \Test{1111}{1}{3}{1111}%
2464 \Test{1000000000000000}{12}{14}{1011000000000000}%
2465 \def\TestOp{Flip}%
2466 \Test{0}{0}{1}{1}%
2467 \Test{1}{0}{1}{0}%
2468 \Test{10101010}{1}{5}{10110100}%
2469 \def\Test#1#2#3#4#5{%
2470   \bitsetSetBin{abc}{#1}%
2471   \bitsetSetValueRange{abc}{#2}{#3}{#4}%
2472   \Expect*{\bitsetGetBin{abc}{0}}{#5}%
2473 }%
2474 \Test{0}{0}{1}{0}{0}%
2475 \Test{0}{0}{1}{1}{1}%
2476 \Test{1010}{1}{3}{0}{1000}%
2477 \Test{1010}{1}{3}{1}{1110}%
2478 \end{qstest}
2479
2480 \begin{qstest}{ShiftLeft/ShiftRight}{ShiftLeft/ShiftRight}
2481   \def\@Test#1#2{%
2482     \let\BS@abc\@undefined
2483     \csname bitsetShift#1\endcsname{abc}{#2}%
2484     \Expect*{\BS@abc}{0}%
2485   }%

```


[CTAN:macros/latex/contrib/oberdiek/bitset.pdf](#) Documentation.

Bundle. All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

[CTAN:install/macros/latex/contrib/oberdiek.tds.zip](#)

TDS refers to the standard “A Directory Structure for T_EX Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

4.2 Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

4.3 Package installation

Unpacking. The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain-T_EX:

```
tex bitset.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

<code>bitset.sty</code>	→ <code>tex/generic/oberdiek/bitset.sty</code>
<code>bitset.pdf</code>	→ <code>doc/latex/oberdiek/bitset.pdf</code>
<code>test/bitset-test1.tex</code>	→ <code>doc/latex/oberdiek/test/bitset-test1.tex</code>
<code>test/bitset-test2.tex</code>	→ <code>doc/latex/oberdiek/test/bitset-test2.tex</code>
<code>test/bitset-test3.tex</code>	→ <code>doc/latex/oberdiek/test/bitset-test3.tex</code>
<code>bitset.dtx</code>	→ <code>source/latex/oberdiek/bitset.dtx</code>

If you have a `docstrip.cfg` that configures and enables `docstrip`’s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

4.4 Refresh file name databases

If your T_EX distribution (teT_EX, miK_TE_X, ...) relies on file name databases, you must refresh these. For example, teT_EX users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk bitset.pdf unpack_files output .
```

Unpacking with L^AT_EX. The .dtx chooses its action depending on the format:

plain-T_EX: Run docstrip and extract the files.

L^AT_EX: Generate the documentation.

If you insist on using L^AT_EX for docstrip (really, docstrip does not need L^AT_EX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{bitset.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the .dtx or the .drv to generate the documentation. The process can be configured by the configuration file ltxdoc.cfg. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfL^AT_EX:

```
pdflatex bitset.dtx
makeindex -s gind.ist bitset.idx
pdflatex bitset.dtx
makeindex -s gind.ist bitset.idx
pdflatex bitset.dtx
```

5 History

[2007/09/28 v1.0]

- First version.

6 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.

Symbols		1300, 1309, 1369, 1372, 1446, 1605
\#	1387, 1449	\{ 1385, 1447
\%	1452	\} 1386, 1448
\:	1721	\] 1451
\@	1388, 1445	
\@PackageError	151, 343, 899, 1054, 1067, 1546	_ 1453
\@Test	2098, 2126, 2129, 2132, 2481, 2487, 2488	
\@ehc	153, 345, 901, 1059, 1081	A
\@empty	2356	\advance 1426, 1434, 1512
\@firstofone	1396, 1399, 1609	\AtEndDocument 1525
\@gobble	1393, 1401, 1607	B
\@undefined	1472, 1473, 1474, 1728, 1856, 1991, 2024, 2031, 2043, 2066, 2067, 2073, 2081, 2102, 2127, 2195, 2241, 2322, 2482	\begin 1613, 1630, 1640, 1667, 1688, 1711, 1855, 1876, 1912, 1931, 1955, 1981, 2023, 2030, 2039, 2062, 2097, 2193, 2239, 2256, 2275, 2296, 2320, 2354, 2384, 2416, 2480, 2528, 2533, 2537
\[1450	\BigIntCalcAdd 621, 630
\%	197, 350, 413, 546, 555, 617, 664, 667, 702, 705, 742, 745, 775, 777, 785, 1269, 1278,	\bigintcalcCmp 333
		\BigIntCalcOdd 353

\bigintcalcSgn	330	\BitSet@Fill	389, <u>402</u> , 427, 529
\BigIntCalcShl	635, 642	\BitSet@FirstOfOne	<u>118</u>
\BigIntCalcShr	361	\BitSet@FirstOfTwo	<u>120</u> , 137, 1322,
\bitset	1069, 2428		1325, 1327, 1336, 1343, 1348, 1367
\BitSet@@@Range	1061, 1084, 1088	\BitSet@Flip	879, <u>974</u> , 1023
\BitSet@@@Set	959, <u>966</u> , 1001	\BitSet@FromFirstHex	209, <u>267</u>
\BitSet@@@CheckIndex	144, <u>148</u>	\BitSet@FromFirstOct	206, <u>235</u>
\BitSet@@@Clear	905, <u>917</u>	\BitSet@FromHex	279, <u>282</u>
\BitSet@@@Flip	976, <u>988</u>	\BitSet@FromOct	248, <u>251</u>
\BitSet@@@Get	1105, <u>1114</u>	\BitSet@Get	1095, <u>1098</u>
\BitSet@@@GetBin	382, <u>385</u>	\BitSet@GetDec	541, <u>545</u>
\BitSet@@@GetDec	550, <u>554</u> , 580	\BitSet@GetDecBig	614, <u>616</u> , 641
\BitSet@@@GetDecBig	629, <u>640</u>	\BitSet@GetOctHex	465, 490, <u>520</u>
\BitSet@@@GetHex	454, <u>487</u>	\BitSet@GetSetBitList ...	<u>1243</u> , <u>1247</u>
\BitSet@@@GetOct	440, <u>462</u>	\BitSet@Gobble	
\BitSet@@@GetOctHex ..	437, 451, 521, <u>525</u>		119, 827, 852, 893, 894, 1218
\BitSet@@@NextClearBit ...	1148, <u>1152</u>	\BitSet@GobbleSeven	1225, <u>1239</u>
\BitSet@@@NextSetBit	1184, <u>1188</u>	\BitSet@Hex[0..F]	<u>293</u>
\BitSet@@@Range ..	1040, <u>1045</u> , 1082, <u>1084</u>	\BitSet@Hex[0000..1111]	<u>501</u>
\BitSet@@@Set	945, <u>952</u>	\BitSet@IfUndefined	
\BitSet@@@TestMode	105		135, 143, 165, 386, 540,
\BitSet@AfterFi			725, 756, 804, 831, 1104, 1260,
	132, 150, 156, 200, 359, 374,		1291, 1320, 1325, 1342, 1343, 1345
	388, 393, 404, 409, 414, 418,	\BitSet@Intersects	1358, <u>1365</u>
	426, 431, 464, 469, 489, 494,	\BitSet@Kill	845, <u>855</u>
	527, 535, 547, 549, 1087, 1100,	\BitSet@KillZeros	
	1139, 1175, 1206, 1249, 1251, 1367		179, <u>189</u> , 218, 276, 325
\BitSet@AfterFiFi		\BitSet@MaxSize	<u>116</u> , 333
	133, 238, 271, 557, 562,	\BitSet@N1073741824	<u>613</u>
	566, 572, 619, 624, 628, 633,	\BitSet@N[1,2,4,...]	<u>578</u>
	748, 866, 928, 959, 1001, 1125,	\BitSet@NegativeIndex	1047, 1050, <u>1066</u>
	1158, 1160, 1194, 1196, 1211,	\BitSet@NextClearBit	1133, <u>1136</u>
	1215, 1217, 1270, 1272, 1279,	\BitSet@NextSetBit	
	1281, 1301, 1303, 1310, 1312, 1370		1169, <u>1172</u> , 1244, 1253
\BitSet@AfterFiFiFi	134, 672, 676,	\BitSet@NumBinFill	415, <u>424</u>
	712, 716, 791, 796, 931, 936,	\BitSet@NumBinRev	396, <u>412</u>
	1004, 1009, 1222, 1228, 1373, 1375	\BitSet@Oct[000..111]	<u>476</u>
\BitSet@And	652, <u>663</u>	\BitSet@Or	732, <u>740</u>
\BitSet@AndNot	690, <u>701</u>	\BitSet@Range	
\BitSet@AtEnd	78, 79, 1382		1017, 1020, 1023, 1031, 1033, <u>1038</u>
\BitSet@Cardinality	1292, <u>1298</u>	\BitSet@Reverse	185, <u>196</u> , 230
\BitSet@CheckIndex		\BitSet@SecondOfTwo	<u>121</u> ,
	142, 873, 876, 879, 886		139, 1321, 1329, 1338, 1343,
\BitSet@Cleanup			1345, 1350, 1356, 1357, 1370, 1373
	866, 928, 1125, 1158, 1194, <u>1203</u>	\BitSet@Set	
\BitSet@Clear			876, 890, <u>943</u> , 1020, 1033, 1073
	873, 888, <u>903</u> , 1017, 1031, 1070	\BitSet@SetDec	339, 351, <u>365</u>
\BitSet@Empty	<u>117</u> , 125,	\BitSet@SetDecBig	335, <u>349</u>
	176, 179, 181, 215, 218, 220,	\BitSet@SetOctHex	206, 209, <u>211</u>
	226, 322, 325, 327, 445, 459,	\BitSet@SetValue	882, <u>885</u>
	463, 488, 657, 695, 768, 846,	\BitSet@SetValueRange ...	1026, <u>1029</u>
	858, 864, 907, 911, 919, 921,	\BitSet@ShiftLeft	809, <u>814</u> , 852
	927, 947, 957, 978, 982, 991, 999	\BitSet@ShiftRight	827, 836, <u>841</u>
\BitSet@ErrorInvalidBitValue ...		\BitSet@Size	1261, <u>1267</u>
	892, <u>898</u> , 1035	\BitSet@Skip	1145, 1181, <u>1204</u>
\BitSet@Fi	<u>131</u> , 132, 133, 134, 159,	\BitSet@SkipContinue	
	203, 249, 280, 363, 378, 394,		1207, 1212, 1215, 1218, <u>1236</u>
	410, 422, 432, 474, 499, 536,	\BitSet@Space	<u>122</u> , 176, 215,
	552, 576, 638, 681, 721, 752,		322, 558, 620, 822, 1111, 1140, 1176
	801, 870, 941, 964, 1014, 1091,	\BitSet@Temp	
	1112, 1129, 1150, 1165, 1186,		173, 174, 176, 178, 179, 181,
	1201, 1234, 1256, 1287, 1317, 1380		185, 212, 213, 215, 217, 218,

220, 223, 224, 226, 230, 293,
 296, 297, 298, 299, 300, 301,
 302, 303, 304, 305, 306, 307,
 308, 309, 310, 311, 312, 313,
 314, 315, 316, 317, 319, 320,
 322, 324, 325, 327, 330, 333,
 335, 339, 476, 479, 480, 481,
 482, 483, 484, 485, 486, 501,
 504, 505, 506, 507, 508, 509,
 510, 511, 512, 513, 514, 515,
 516, 517, 518, 519, 578, 583,
 584, 585, 586, 587, 588, 589,
 590, 591, 592, 593, 594, 595,
 596, 597, 598, 599, 600, 601,
 602, 603, 604, 605, 606, 607,
 608, 609, 610, 611, 612, 904,
 911, 914, 975, 982, 985, 1039, 1043
 \BitSet@TestMode 105, 1479
 \BitSet@Xor 763, 774
 \BitSet@ZapSpace .. 123, 175, 214, 321
 \BitSet@Zero 182, 221,
 227, 328, 331, 912, 983, 1326, 1333
 \bitsetAnd 7, 644
 \bitsetAndNot 7, 683
 \bitsetCardinality 8, 1289, 1689, 1694
 \bitsetClear .. 7, 872, 1915, 1918, 1920
 \bitsetClearRange 1016
 \bitsetEquals 9, 1341, 2041
 \BitSetError ... 246, 262, 274, 286,
 357, 1101, 1141, 1177, 1724, 1726
 \bitsetFlip 878, 1958, 1961, 1963, 2057
 \bitsetFlipRange 1022
 \bitsetGet
 .. 8, 1093, 1335, 1643, 1653, 2534
 \bitsetGetBin 6, 380,
 2201, 2325, 2337, 2451, 2472, 2499
 \bitsetGetDec 7, 538, 1877, 1885
 \bitsetGetHex 448, 2220, 2389
 \bitsetGetOct 434, 2207, 2361
 \bitsetGetSetBitList
 8, 1240, 1859, 1865
 \bitsetIntersects 9, 1355, 2064, 2088
 \bitsetIsDefined . 8, 1319, 2025, 2027
 \bitsetIsEmpty .. 9, 436, 450, 645,
 648, 684, 687, 724, 727, 755,
 758, 807, 834, 1144, 1180, 1242,
 1324, 1356, 1357, 2032, 2034, 2036
 \bitsetLet 6, 164, 1616, 1622
 \bitsetNextClearBit ... 8, 1131, 1718
 \bitsetNextSetBit 8, 1167, 1719
 \bitsetOr 7, 723
 \bitsetQuery 9, 1334, 1648, 1654
 \bitsetReset 6, 143,
 161, 166, 646, 649, 658, 685,
 696, 725, 756, 769, 805, 832,
 1631, 1633, 1636, 1738, 2026,
 2033, 2047, 2051, 2108, 2130, 2440
 \bitsetSet 875,
 1934, 1937, 1939, 2035, 2054, 2529
 \bitsetSetBin .. 6, 172, 1722, 1748,
 1758, 1809, 1820, 1831, 1842,
 1862, 1993, 2001, 2002, 2086,
 2087, 2153, 2154, 2242, 2334,
 2357, 2386, 2449, 2470, 2496, 2497
 \bitsetSetDec 6, 318, 2298
 \bitsetSetHex 208, 2277
 \bitsetSetOct 205, 2258
 \bitsetSetRange 1019, 2423
 \bitsetSetValue 8, 881, 1984, 1992, 2003
 \bitsetSetValueRange 1025, 2471
 \bitsetShiftLeft 7, 803
 \bitsetShiftRight 830
 \bitsetSize 8, 1258, 1668, 1672
 \bitsetXor 7, 754
 \body 1405, 1409
 \BS@abc 1728, 1856, 1916, 1935, 1959,
 1991, 1994, 2004, 2024, 2031,
 2043, 2066, 2073, 2127, 2133,
 2195, 2241, 2243, 2259, 2278,
 2299, 2322, 2432, 2482, 2484, 2530
 \BS@foo 2067, 2081, 2102, 2114
 \BS@global 2432, 2530
 \BS@result 1994, 2004

C

 \catcode 3, 4, 5, 6, 7, 18, 19,
 20, 34, 35, 36, 37, 38, 39, 40, 41,
 42, 43, 44, 62, 63, 66, 67, 68, 69,
 73, 74, 75, 76, 80, 82, 103, 1385,
 1386, 1387, 1388, 1423, 1432,
 1445, 1446, 1447, 1448, 1449,
 1450, 1451, 1452, 1453, 1454, 1721
 \chardef 1479
 \Check 1582,
 1618, 1620, 1621, 1623, 1624,
 1626, 1627, 1632, 1634, 1637,
 2105, 2110, 2111, 2116, 2117, 2434
 \CheckUndef
 1572, 1585, 1614, 1615, 1617, 2104
 \Clear 1718, 1723, 1729, 1739,
 1749, 1759, 1810, 1821, 1832, 1843
 \count@ 1390, 1419, 1423,
 1425, 1426, 1430, 1432, 1433, 1434
 \countdef 1390
 \csname . 8, 21, 45, 58, 65, 101, 107,
 136, 162, 168, 169, 182, 184,
 221, 227, 229, 273, 277, 285,
 288, 294, 328, 331, 334, 338,
 399, 444, 458, 472, 477, 497,
 502, 542, 567, 573, 579, 613,
 651, 653, 655, 657, 689, 691,
 693, 695, 728, 729, 731, 733,
 735, 759, 760, 762, 764, 766,
 768, 820, 823, 844, 846, 906,
 910, 944, 946, 977, 981, 1106,
 1147, 1183, 1264, 1295, 1326,
 1346, 1347, 1359, 1361, 1389,
 1392, 1395, 1398, 1437, 1459,
 1495, 1580, 1589, 1597, 1600,
 1646, 2019, 2020, 2103, 2109,
 2115, 2155, 2431, 2450, 2483, 2498
 \currentgrouplevel .. 1470, 1474, 1486

D

 \dimexpr 1502

<code>\documentclass</code>	1465	864, 911, 918, 919, 921, 927, 930, 954, 957, 967, 982, 989, 990, 991, 999, 1003, 1070, 1073, 1116, 1117, 1123, 1153, 1156, 1189, 1192, 1205, 1268, 1269, 1278, 1299, 1300, 1309, 1326, 1346, 1369, 1372, 1389, 1392, 1395, 1398, 1437, 1495, 1589, 1605
E		
<code>\empty</code>	12	
<code>\end</code> ..	1460, 1628, 1638, 1665, 1686, 1709, 1853, 1874, 1910, 1929, 1953, 1979, 2021, 2028, 2037, 2060, 2095, 2191, 2237, 2254, 2273, 2294, 2318, 2352, 2382, 2414, 2478, 2526, 2531, 2535, 2538	
<code>\endcsname</code>	8, 21, 45, 58, 65, 101, 107, 136, 162, 168, 169, 182, 184, 221, 227, 229, 273, 277, 285, 288, 294, 328, 331, 334, 338, 399, 444, 458, 472, 477, 497, 502, 542, 568, 573, 579, 613, 651, 653, 655, 657, 689, 691, 693, 695, 728, 729, 731, 733, 735, 759, 760, 762, 764, 766, 768, 820, 823, 844, 846, 906, 910, 944, 946, 977, 981, 1106, 1147, 1183, 1264, 1295, 1326, 1346, 1347, 1359, 1361, 1389, 1392, 1395, 1398, 1437, 1459, 1495, 1580, 1589, 1597, 1600, 1646, 2019, 2020, 2103, 2109, 2115, 2155, 2431, 2450, 2483, 2498	
<code>\endinput</code>	30	
<code>\endqstest</code>	1516, 1521, 1530, 1535	
<code>\ETeXDisable</code>	1471, 1476, 1558	
<code>\ETeXEnable</code>	1483, 1488, 1551	
<code>\Expect</code>	1537, 1553, 1554, 1562, 1563, 1573, 1578, 1587, 1641, 1648, 1652, 1654, 1671, 1692, 1713, 1857, 1863, 1883, 1916, 1935, 1959, 1994, 2004, 2025, 2027, 2032, 2034, 2036, 2041, 2064, 2088, 2196, 2243, 2259, 2278, 2299, 2323, 2335, 2359, 2387, 2451, 2472, 2484, 2499	
<code>\ExpectBitSet</code>	1577, 1583, 1594	
H		
<code>\hbox</code>	1533	
I		
<code>\ifcase</code>	9, 238, 254, 330, 353, 366, 815, 842, 887, 1030, 1052, 1210, 1655	
<code>\ifcsname</code>	1469, 1472, 1485	
<code>\ifnum</code>	149, 333, 387, 403, 425, 526, 665, 668, 670, 703, 707, 710, 741, 1046, 1049, 1085, 1099, 1137, 1173, 1221, 1248, 1335, 1366, 1425, 1433	
<code>\ifodd</code>	369	
<code>\ifx</code>	10, 12, 21, 45, 53, 101, 107, 125, 136, 181, 190, 197, 220, 226, 236, 252, 268, 270, 273, 283, 285, 327, 350, 413, 463, 488, 546, 555, 556, 565, 617, 618, 627, 657, 664, 667, 695, 702, 705, 742, 745, 768, 775, 776, 777, 785, 787, 790, 857, 858,	
<code>\ignorespaces</code>	1533	
<code>\immediate</code>	23, 47	
<code>\IncludeTests</code>	1491	
<code>\input</code>	108, 109, 110, 1438	
<code>\IntCalcAdd</code>	531, 559, 569	
<code>\intcalcCmp</code>	1052	
<code>\IntCalcDec</code> ...	390, 406, 466, 491, 1230	
<code>\IntCalcDiv</code>	530	
<code>\IntCalcInc</code>	420, 471, 496, 1089, 1162, 1198, 1254, 1274, 1283, 1301, 1305	
<code>\IntCalcMul</code>	523	
<code>\intcalcNum</code>	145, 383, 438, 452, 522, 810, 837, 883, 1027, 1041, 1096, 1109, 1134, 1170	
<code>\intcalcSgn</code>	815, 842	
<code>\IntCalcShr</code>	376	
<code>\IntCalcSub</code>	428, 532, 1224	
<code>\iterate</code>	1406, 1408, 1410	
L		
<code>\LoadCommand</code>	1438, 1455	
<code>\LogTests</code>	1492	
<code>\loop</code>	1404, 1420, 1431	
M		
<code>\makeatletter</code>	1466, 1478, 1493	
<code>\makeatother</code>	1480	
<code>\MessageBreak</code> 1055, 1056, 1057, 1068, 1079, 1549, 2418, 2419, 2420, 2427, 2428	
N		
<code>\NeedsTeXFormat</code>	1463	
<code>\newcommand</code>	1500, 1505, 1509, 1510	
<code>\newcount</code>	1497, 1498	
<code>\newcounter</code>	1540	
<code>\next</code>	1410, 1412, 1414	
<code>\nofiles</code>	1464	
<code>\number</code>	471, 496, 522, 530, 1094, 1132, 1146, 1168, 1182, 1224, 1244, 1252, 1259, 1290, 1502	
<code>\numexpr</code>	1468, 1473, 1484	
O		
<code>\Op</code>	2122, 2126, 2129, 2132, 2155, 2159, 2166, 2171, 2178, 2498, 2501, 2512	
<code>\orig@endqstest</code>	1530, 1538	
<code>\orig@qstest</code>	1529, 1532	
P		
<code>\PackageInfo</code>	26	
<code>\pdfelapsedtime</code>	1511	
<code>\pdfresettimer</code>	1507	
<code>\PrintTime</code>	1500, 1513, 1526	
<code>\ProvidesPackage</code>	59	

Q		1811, 1812, 1813, 1814, 1816,
\qstest	1515, 1517, 1529, 1531	1817, 1818, 1819, 1822, 1823,
R		1824, 1825, 1827, 1828, 1829,
\RangeCatcodeInvalid		1830, 1833, 1834, 1835, 1836,
.	1429, 1441, 1442, 1443, 1444	1838, 1839, 1840, 1841, 1844,
\renewcommand	1506	1845, 1846, 1847, 1849, 1850,
\repeat	1404, 1416, 1427, 1435	1851, 1852, 1861, 1868, 1869,
\RequirePackage	112, 113, 114	1870, 1871, 1872, 1873, 1878,
\RestoreCatcodes	1418, 1421, 1422, 1456	1888, 1889, 1890, 1891, 1892,
\RevCheck	1593, 1919, 1921,	1893, 1894, 1895, 1896, 1897,
.	1938, 1940, 1962, 1964, 2156, 2157	1898, 1900, 1902, 1904, 1906,
\Reverse	1594,	1908, 1913, 1922, 1923, 1924,
.	1601, 1604, 1611, 1916, 1935, 1959	1925, 1926, 1927, 1928, 1932,
\RevSet	1599, 1879, 1914, 1933, 1957	1941, 1942, 1943, 1944, 1945,
\romannumeral	381,	1946, 1947, 1948, 1949, 1950,
.	435, 449, 539, 822, 849, 908,	1951, 1952, 1956, 1965, 1966,
.	949, 979, 1109, 1141, 1177, 1241	1967, 1968, 1969, 1970, 1971,
S		1972, 1973, 1974, 1975, 1976,
\saved@endqstest	1516, 1523	1977, 1978, 1982, 1987, 1988,
\saved@qstest	1515, 1518	1989, 1990, 1996, 1997, 1998,
\SavedCurrentgrouplevel	1470, 1486	1999, 2000, 2006, 2007, 2008,
\SavedIfcsname	1469, 1485	2009, 2010, 2011, 2012, 2013,
\SavedNumexpr	1468, 1484	2014, 2015, 2016, 2017, 2018,
\sbox	1545	2019, 2020, 2040, 2044, 2045,
\Set	1596, 1619, 1625, 1635, 1647,	2046, 2048, 2049, 2050, 2052,
.	1651, 1670, 1691, 1719, 1725,	2053, 2055, 2056, 2058, 2059,
.	1734, 1744, 1754, 1784, 1815,	2063, 2068, 2070, 2072, 2074,
.	1826, 1837, 1848, 2069, 2071,	2076, 2078, 2080, 2082, 2084,
.	2075, 2077, 2079, 2083, 2436, 2444	2085, 2090, 2091, 2092, 2093,
\setbox	1533	2094, 2121, 2136, 2140, 2144,
\setcounter	1544	2148, 2152, 2160, 2161, 2162,
\space	1502	2163, 2164, 2165, 2167, 2168,
\StartTime	1505, 1519	2169, 2170, 2172, 2173, 2174,
\stepcounter	1547	2175, 2176, 2177, 2179, 2180,
\StopTime	1510, 1522	2181, 2182, 2183, 2184, 2185,
\strip@pt	1502	2188, 2240, 2245, 2246, 2247,
\SummaryTime	1497, 1499, 1512, 1526	2248, 2249, 2250, 2251, 2252,
T		2253, 2257, 2261, 2262, 2263,
\Test	1440, 1458, 1650, 1657,	2264, 2265, 2266, 2267, 2268,
.	1658, 1659, 1660, 1661, 1662,	2269, 2270, 2271, 2272, 2276,
.	1663, 1664, 1669, 1674, 1675,	2280, 2281, 2282, 2283, 2284,
.	1676, 1677, 1678, 1679, 1680,	2285, 2286, 2287, 2288, 2289,
.	1681, 1682, 1683, 1684, 1685,	2290, 2291, 2292, 2293, 2297,
.	1690, 1697, 1698, 1699, 1700,	2301, 2302, 2303, 2304, 2305,
.	1701, 1702, 1703, 1704, 1705,	2306, 2307, 2308, 2309, 2310,
.	1706, 1707, 1708, 1712, 1724,	2311, 2312, 2313, 2314, 2315,
.	1726, 1730, 1731, 1732, 1733,	2316, 2317, 2333, 2340, 2341,
.	1735, 1736, 1737, 1740, 1741,	2342, 2343, 2344, 2345, 2346,
.	1742, 1743, 1745, 1746, 1747,	2347, 2348, 2349, 2350, 2351,
.	1750, 1751, 1752, 1753, 1755,	2355, 2364, 2365, 2366, 2367,
.	1756, 1757, 1760, 1761, 1762,	2368, 2369, 2370, 2371, 2372,
.	1763, 1764, 1765, 1766, 1767,	2373, 2374, 2375, 2376, 2377,
.	1768, 1769, 1770, 1771, 1772,	2378, 2379, 2380, 2381, 2385,
.	1773, 1774, 1775, 1776, 1777,	2392, 2393, 2394, 2395, 2396,
.	1778, 1779, 1780, 1781, 1782,	2397, 2398, 2399, 2400, 2401,
.	1783, 1785, 1786, 1787, 1788,	2402, 2403, 2404, 2405, 2406,
.	1789, 1790, 1791, 1792, 1793,	2407, 2408, 2409, 2410, 2411,
.	1794, 1795, 1796, 1797, 1798,	2412, 2413, 2448, 2454, 2455,
.	1799, 1800, 1801, 1802, 1803,	2456, 2457, 2459, 2460, 2461,
.	1804, 1805, 1806, 1807, 1808,	2462, 2463, 2464, 2466, 2467,
		2468, 2469, 2474, 2475, 2476,
		2477, 2486, 2490, 2491, 2492,

2493, 2494, 2495, 2502, 2503,	\the . 66, 67, 68, 69, 80, 1423, 1537, 1563
2504, 2505, 2506, 2507, 2508,	\theTest 1562
2509, 2510, 2511, 2513, 2514,	\TimeDescription ... 1506, 1509, 1513
2515, 2516, 2517, 2518, 2519,	\TMP@EnsureCode
2520, 2521, 2522, 2523, 2524, 2525 77, 84, 85, 86, 87, 88, 89,
\Test@ 2123, 2125	90, 91, 92, 93, 94, 95, 96, 97, 98, 99
\TestError 1542, 1568, 1983, 2417, 2426	\typeout 1501
\TestErrorNegativeIndex	
..... 1567, 1920, 1939, 1963	U
\TestErrorNegInd	\uccode 818
..... 2425, 2437, 2438, 2439,	\uppercase 819
2441, 2442, 2443, 2445, 2446, 2447	\usepackage 1481, 1490
\TestGetterUndefined	
..... 1571, 1668, 1689, 1877	W
\TestOp 1715,	\wd 1537, 1563
1718, 1719, 2450, 2453, 2458, 2465	\write 23, 47
\TestTime 1498, 1511, 1512, 1513	
\TestUndef 2194, 2202, 2203, 2204,	X
2205, 2206, 2208, 2209, 2210,	\x 8, 10, 12, 22, 26, 28,
2211, 2212, 2213, 2214, 2215,	46, 51, 58, 64, 72, 2198, 2201,
2216, 2217, 2218, 2219, 2221,	2207, 2220, 2356, 2357, 2358, 2534
2222, 2223, 2224, 2225, 2226,	
2227, 2228, 2229, 2230, 2231,	Z
2232, 2233, 2234, 2235, 2236,	\z@ 1499
2321, 2328, 2329, 2330, 2331, 2332	\zap@space 2356