

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	\bitsetAnd	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	29
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	40
3.2.3	Detection of unwanted space	40
3.2.4	Test macros	40
3.2.5	Test sets	42
4	Installation	57
4.1	Download	57
4.2	Bundle installation	57
4.3	Package installation	57
4.4	Refresh file name databases	58
4.5	Some details for the interested	58
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.

Expandability: 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 `⟨BitSet⟩` 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 `⟨ShiftAmount⟩` is negative, it reverts the meaning of the shift operation. A left shift becomes a right shift. A `⟨ShiftAmount⟩` 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 `⟨BitSet⟩` 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\@undefined
54 \xdef#1{#3}%
55 \fi
56 \ifx#1\relax
57 \xdef#1{#3}%
58 \fi
59 }%
60 \fi
61 \expandafter\x\csname ver@bitset.sty\endcsname
62 \ProvidesPackage{bitset}%
63 [2007/09/28 v1.0 Data type bit set (H0)]

```

2.2 Catcodes

```

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

```

```

83 \catcode#1 \the\catcode#1\relax
84 }%
85 \catcode#1 #2\relax
86 }
87 \TMP@EnsureCode{33}{12}% !
88 \TMP@EnsureCode{39}{12}% '
89 \TMP@EnsureCode{40}{12}% (
90 \TMP@EnsureCode{41}{12}% )
91 \TMP@EnsureCode{42}{12}% *
92 \TMP@EnsureCode{43}{12}% +
93 \TMP@EnsureCode{44}{12}% ,
94 \TMP@EnsureCode{45}{12}% -
95 \TMP@EnsureCode{46}{12}% .
96 \TMP@EnsureCode{47}{12}% /
97 \TMP@EnsureCode{58}{11}% : (letter!)
98 \TMP@EnsureCode{60}{12}% <
99 \TMP@EnsureCode{61}{12}% =
100 \TMP@EnsureCode{62}{12}% >
101 \TMP@EnsureCode{63}{14}% ? (comment!)
102 \TMP@EnsureCode{96}{12}% '
103 \begingroup\expandafter\expandafter\expandafter\endgroup
104 \expandafter\ifx\csname BitSet@TestMode\endcsname\relax
105 \else
106 \catcode63=9 % ? (ignore)
107 \fi
108 ? \let\BitSet@@TestMode\BitSet@TestMode

```

2.3 Package loading

```

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

```

2.4 Help macros

2.4.1 Number constant

```

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

```

2.4.2 General basic macros

```

\BitSet@Empty
120 \def\BitSet@Empty{}

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

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

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

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

```

\BitSet@Space

```
125 \def\BitSet@Space{ }
```

\BitSet@ZapSpace

```
126 \def\BitSet@ZapSpace#1 #2{%
127   #1%
128   \ifx\BitSet@Empty#2%
129   \else
130     \expandafter\BitSet@ZapSpace
131   \fi
132   #2%
133 }
```

2.4.3 Tail recursion

\BitSet@Fi

```
134 \let\BitSet@Fi\fi
```

\BitSet@AfterFi

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

\BitSet@AfterFiFi

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

\BitSet@AfterFiFiFi

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

2.4.4 Check macros

\BitSet@IfUndefined

```
138 \def\BitSet@IfUndefined#1{%
139   \expandafter\ifx\csname BS@#1\endcsname\relax
140   \expandafter\BitSet@FirstOfTwo
141   \else
142   \expandafter\BitSet@SecondOfTwo
143   \fi
144 }
```

\BitSet@CheckIndex #1: continuation code
#2: BitSet
#3: Index

```
145 \def\BitSet@CheckIndex#1#2#3{%
146   \BitSet@IfUndefined{#2}{\bitsetReset{#2}}{%
147     \expandafter\expandafter\expandafter\BitSet@@CheckIndex
148     \intcalcNum{#3}!%
149     {#2}{#1}%
150   }
```

\BitSet@@CheckIndex #1: plain Index
#2: BitSet
#3: continuation code

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

```

160      #3{#2}{#1}%
161    }%
162    \BitSet@Fi
163 }

```

2.5 Miscellaneous

\bitsetReset

```

164 \def\bitsetReset#1{%
165   \expandafter\def\csname BS@#1\endcsname{0}%
166 }

```

\bitsetLet

```

167 \def\bitsetLet#1#2{%
168   \BitSet@IfUndefined{#2}{%
169     \bitsetReset{#1}%
170   }{%
171     \expandafter\let\csname BS@#1\endcsname\expandafter\endcsname
172     \csname BS@#2\endcsname
173   }%
174 }

```

2.6 Import

2.6.1 From binary number

\bitsetSetBin

```

175 \def\bitsetSetBin#1#2{%
176   \edef\BitSet@Temp{#2}%
177   \edef\BitSet@Temp{%
178     \expandafter\expandafter\expandafter\BitSet@ZapSpace
179     \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
180   }%
181   \edef\BitSet@Temp{%
182     \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
183   }%
184   \ifx\BitSet@Temp\BitSet@Empty
185     \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
186   \else
187     \expandafter\edef\csname BS@#1\endcsname{%
188       \expandafter\BitSet@Reverse\BitSet@Temp!%
189     }%
190   \fi
191 }

```

\BitSet@KillZeros

```

192 \def\BitSet@KillZeros#1{%
193   \ifx#10%
194     \expandafter\BitSet@KillZeros
195   \else
196     #1%
197   \fi
198 }

```

\BitSet@Reverse

```

199 \def\BitSet@Reverse#1#2!{%
200   \ifx\#2\%
201     #1%
202   \else
203     \BitSet@AfterFi{%
204       \BitSet@Reverse#2!#1%

```

```

205     }%
206   \BitSet@Fi
207 }

```

2.6.2 From octal/hex number

\bitsetSetOct

```

208 \def\bitsetSetOct{%
209   \BitSet@SetOctHex\BitSet@FromFirstOct
210 }

```

\bitsetSetHex

```

211 \def\bitsetSetHex{%
212   \BitSet@SetOctHex\BitSet@FromFirstHex
213 }

```

\BitSet@SetOctHex

```

214 \def\BitSet@SetOctHex#1#2#3{%
215   \edef\BitSet@Temp{#3}%
216   \edef\BitSet@Temp{%
217     \expandafter\expandafter\expandafter\BitSet@ZapSpace
218     \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
219   }%
220   \edef\BitSet@Temp{%
221     \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
222   }%
223   \ifx\BitSet@Temp\BitSet@Empty
224     \expandafter\let\csname BS@#2\endcsname\BitSet@Zero
225   \else
226     \edef\BitSet@Temp{%
227       \expandafter#1\BitSet@Temp!%
228     }%
229     \ifx\BitSet@Temp\BitSet@Empty
230       \expandafter\let\csname BS@#2\endcsname\BitSet@Zero
231     \else
232       \expandafter\edef\csname BS@#2\endcsname{%
233         \expandafter\BitSet@Reverse\BitSet@Temp!%
234       }%
235     \fi
236   \fi
237 }

```

\BitSet@FromFirstOct

```

238 \def\BitSet@FromFirstOct#1{%
239   \ifx#1!%
240   \else
241     \ifcase#1 \BitSet@AfterFiFi\BitSet@FromFirstOct
242     \or 1%
243     \or 10%
244     \or 11%
245     \or 100%
246     \or 101%
247     \or 110%
248     \or 111%
249     \else \BitSetError:WrongOctalDigit%
250     \fi
251     \expandafter\BitSet@FromOct
252   \BitSet@Fi
253 }

```

\BitSet@FromOct

```

254 \def\BitSet@FromOct#1{%
255   \ifx#1!%
256   \else
257     \ifcase#1 000%
258     \or 001%
259     \or 010%
260     \or 011%
261     \or 100%
262     \or 101%
263     \or 110%
264     \or 111%
265     \else \BitSetError:WrongOctalDigit%
266     \fi
267     \expandafter\BitSet@FromOct
268   \fi
269 }

```

\BitSet@FromFirstHex

```

270 \def\BitSet@FromFirstHex#1{%
271   \ifx#1!%
272   \else
273     \ifx#10%
274       \BitSet@AfterFiFi\BitSet@FromFirstHex
275     \fi
276     \expandafter\ifx\csname BitSet@Hex#1\endcsname\relax
277       \BitSetError:InvalidHexDigit%
278     \else
279       \expandafter\expandafter\expandafter\BitSet@KillZeros
280       \csname BitSet@Hex#1\endcsname
281     \fi
282     \expandafter\BitSet@FromHex
283   \BitSet@Fi
284 }

```

\BitSet@FromHex

```

285 \def\BitSet@FromHex#1{%
286   \ifx#1!%
287   \else
288     \expandafter\ifx\csname BitSet@Hex#1\endcsname\relax
289       \BitSetError:InvalidHexDigit%
290     \else
291       \csname BitSet@Hex#1\endcsname
292     \fi
293     \expandafter\BitSet@FromHex
294   \fi
295 }

```

\BitSet@Hex[0..F]

```

296 \def\BitSet@Temp#1{%
297   \expandafter\def\csname BitSet@Hex#1\endcsname
298 }
299 \BitSet@Temp 0{0000}%
300 \BitSet@Temp 1{0001}%
301 \BitSet@Temp 2{0010}%
302 \BitSet@Temp 3{0011}%
303 \BitSet@Temp 4{0100}%
304 \BitSet@Temp 5{0101}%
305 \BitSet@Temp 6{0110}%
306 \BitSet@Temp 7{0111}%
307 \BitSet@Temp 8{1000}%
308 \BitSet@Temp 9{1001}%
309 \BitSet@Temp A{1010}%

```

```

310 \BitSet@Temp B{1011}%
311 \BitSet@Temp C{1100}%
312 \BitSet@Temp D{1101}%
313 \BitSet@Temp E{1110}%
314 \BitSet@Temp F{1111}%
315 \BitSet@Temp a{1010}%
316 \BitSet@Temp b{1011}%
317 \BitSet@Temp c{1100}%
318 \BitSet@Temp d{1101}%
319 \BitSet@Temp e{1110}%
320 \BitSet@Temp f{1111}%

```

2.6.3 From decimal number

\bitsetSetDec

```

321 \def\bitsetSetDec#1#2{%
322   \edef\BitSet@Temp{#2}%
323   \edef\BitSet@Temp{%
324     \expandafter\expandafter\expandafter\BitSet@ZapSpace
325     \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
326   }%
327   \edef\BitSet@Temp{%
328     \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
329   }%
330   \ifx\BitSet@Temp\BitSet@Empty
331     \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
332   \else
333     \ifcase\bigintcalcSgn{\BitSet@Temp} %
334       \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
335     \or
336       \ifnum\bigintcalcCmp\BitSet@Temp\BitSet@MaxSize>0 %
337         \expandafter\edef\csname BS@#1\endcsname{%
338           \expandafter\BitSet@SetDecBig\BitSet@Temp!%
339         }%
340       \else
341         \expandafter\edef\csname BS@#1\endcsname{%
342           \expandafter\BitSet@SetDec\BitSet@Temp!%
343         }%
344       \fi
345     \else
346       \@PackageError{bitset}{%
347         Bit sets cannot be negative%
348       }\@ehc
349     \fi
350   \fi
351 }

```

\BitSet@SetDecBig

```

352 \def\BitSet@SetDecBig#1#2#3#4#5#6#7#8#9!{%
353   \ifx\#9\%
354     \BitSet@SetDec#1#2#3#4#5#6#7#8!%
355   \else
356     \ifcase\BigIntCalcOdd#1#2#4#5#6#7#8#9! %
357       0%
358     \or
359       1%
360   ? \else\BitSetError:ThisCannotHappen%
361   \fi
362   \BitSet@AfterFi{%
363     \expandafter\expandafter\expandafter\BitSet@SetDecBig
364     \BigIntCalcShr#1#2#3#4#5#6#7#8#9!%
365   }%

```



```

366 \BitSet@Fi
367 }

\BitSet@SetDec

368 \def\BitSet@SetDec#1!{%
369 \ifcase#1 %
370 \or 1%
371 \else
372 \ifodd#1 %
373 1%
374 \else
375 0%
376 \fi
377 \BitSet@AfterFi{%
378 \expandafter\expandafter\expandafter\BitSet@SetDec
379 \IntCalcShr#1!!%
380 }%
381 \BitSet@Fi
382 }

```

2.7 Export

2.7.1 To binary number

```

\bitsetGetBin

383 \def\bitsetGetBin#1#2{%
384 \romannumeral0%
385 \expandafter\expandafter\expandafter\BitSet@@GetBin
386 \intcalcNum{#2}!{#1}%
387 }

\BitSet@@GetBin

388 \def\BitSet@@GetBin#1!#2{%
389 \BitSet@IfUndefined{#2}{%
390 \ifnum#1>1 %
391 \BitSet@AfterFi{%
392 \expandafter\expandafter\expandafter\BitSet@Fill
393 \IntCalcDec#1!!0%
394 }%
395 \else
396 \BitSet@AfterFi{ 0}%
397 \BitSet@Fi
398 }{%
399 \expandafter\expandafter\expandafter\BitSet@NumBinRev
400 \expandafter\expandafter\expandafter1%
401 \expandafter\expandafter\expandafter!%
402 \csname BS@#2\endcsname!!#1!%
403 }%
404 }

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

405 \def\BitSet@Fill#1!{%
406 \ifnum#1>0 %
407 \BitSet@AfterFi{%
408 \expandafter\expandafter\expandafter\BitSet@Fill
409 \IntCalcDec#1!!0%
410 }%
411 \else
412 \BitSet@AfterFi{ }%
413 \BitSet@Fi
414 }

```

```

\BitSet@NumBinRev #1: bit counter (including #2)
#2#3: reverted number
#4: result
#5: min size
415 \def\BitSet@NumBinRev#1!#2#3!{%
416   \ifx\#3\%
417     \BitSet@AfterFi{%
418       \BitSet@NumBinFill#1!#2%
419     }%
420   \else
421     \BitSet@AfterFi{%
422       \expandafter\expandafter\expandafter\BitSet@NumBinRev
423       \IntCalcInc#1!!#3!#2%
424     }%
425   \BitSet@Fi
426 }

```

```

\BitSet@NumBinFill
427 \def\BitSet@NumBinFill#1!#2!#3!{%
428   \ifnum#3>#1 %
429     \BitSet@AfterFi{%
430       \expandafter\expandafter\expandafter\BitSet@Fill
431       \IntCalcSub#3!#1!!#2%
432     }%
433   \else
434     \BitSet@AfterFi{ #2}%
435   \BitSet@Fi
436 }

```

2.7.2 To octal/hexadecimal number

```

\bitsetGetOct
437 \def\bitsetGetOct#1#2{%
438   \romannumeral0%
439   \bitsetIsEmpty{#1}{%
440     \expandafter\expandafter\expandafter\BitSet@@GetOctHex
441     \intcalcNum{#2}!3!230%
442   }{%
443     \expandafter\expandafter\expandafter\BitSet@@GetOct
444     \expandafter\expandafter\expandafter1%
445     \expandafter\expandafter\expandafter!%
446     \expandafter\expandafter\expandafter!%
447     \csname BS@#1\endcsname00%
448     \BitSet@Empty\BitSet@Empty\BitSet@Empty!{#2}%
449   }%
450 }

```

```

\bitsetGetHex
451 \def\bitsetGetHex#1#2{%
452   \romannumeral0%
453   \bitsetIsEmpty{#1}{%
454     \expandafter\expandafter\expandafter\BitSet@@GetOctHex
455     \intcalcNum{#2}!4!340%
456   }{%
457     \expandafter\expandafter\expandafter\BitSet@@GetHex
458     \expandafter\expandafter\expandafter1%
459     \expandafter\expandafter\expandafter!%
460     \expandafter\expandafter\expandafter!%
461     \csname BS@#1\endcsname000%
462     \BitSet@Empty\BitSet@Empty\BitSet@Empty\BitSet@Empty!{#2}%
463   }%
464 }

```

```

\BitSet@@GetOct #1: number of digits
#2: result
#3#4#5: bits
465 \def\BitSet@@GetOct#1!#2!#3#4#5{%
466   \ifx#5\BitSet@Empty
467     \BitSet@AfterFi{%
468       \expandafter\expandafter\expandafter\BitSet@GetOctHex
469       \IntCalcDec#1!!#2!23%
470     }%
471   \else
472     \BitSet@AfterFi{%
473       \expandafter\expandafter\expandafter\BitSet@@GetOct
474       \number\IntCalcInc#1!\expandafter\expandafter\expandafter!%
475       \csname BitSet@Oct#5#4#3\endcsname#2!%
476     }%
477   \BitSet@Fi
478 }

```

\BitSet@Oct[000..111]

```

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

```

```

\BitSet@@GetHex #1: number of digits
#2: result
#3#4#5#6: bits
490 \def\BitSet@@GetHex#1!#2!#3#4#5#6{%
491   \ifx#6\BitSet@Empty
492     \BitSet@AfterFi{%
493       \expandafter\expandafter\expandafter\BitSet@GetOctHex
494       \IntCalcDec#1!!#2!34%
495     }%
496   \else
497     \BitSet@AfterFi{%
498       \expandafter\expandafter\expandafter\BitSet@@GetHex
499       \number\IntCalcInc#1!\expandafter\expandafter\expandafter!%
500       \csname BitSet@Hex#6#5#4#3\endcsname#2!%
501     }%
502   \BitSet@Fi
503 }

```

\BitSet@Hex[0000..1111]

```

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

```

```

515 \BitSet@Temp10008%
516 \BitSet@Temp10019%
517 \BitSet@Temp1010A%
518 \BitSet@Temp1011B%
519 \BitSet@Temp1100C%
520 \BitSet@Temp1101D%
521 \BitSet@Temp1110E%
522 \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

```

523 \def\BitSet@GetOctHex#1!#2!#3#4#5!#6{%
524   \expandafter\BitSet@@GetOctHex
525   \number\intcalNum{#6}\expandafter\expandafter\expandafter!%
526   \IntCalcMul#1!#4!!#3#4#2%
527 }

```

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

```

528 \def\BitSet@@GetOctHex#1!#2!#3#4{%
529   \ifnum#1>#2 %
530     \BitSet@AfterFi{%
531       \expandafter\expandafter\expandafter\expandafter
532       \expandafter\expandafter\expandafter\BitSet@Fill
533       \expandafter\IntCalcDiv\number
534       \expandafter\expandafter\expandafter\IntCalcAdd
535       \IntCalcSub#1!#2!!#3!!#4!!%
536     }%
537   \else
538     \BitSet@AfterFi{ }%
539   \BitSet@Fi
540 }

```

2.7.3 To decimal number

\bitsetGetDec

```

541 \def\bitsetGetDec#1{%
542   \romannumeral0%
543   \BitSet@IfUndefined{#1}{ 0 }{%
544     \expandafter\expandafter\expandafter\BitSet@GetDec
545     \csname BS@#1\endcsname!%
546   }%
547 }

```

\BitSet@GetDec

```

548 \def\BitSet@GetDec#1#2!{%
549   \ifx\#2\%
550     \BitSet@AfterFi{ #1}%
551   \else
552     \BitSet@AfterFi{%
553       \BitSet@@GetDec2!#1!#2!%
554     }%
555   \BitSet@Fi
556 }

```

```

\BitSet@@GetDec #1: power of two
#2: result
#3#4: number
557 \def\BitSet@@GetDec#1!#2!#3#4!{%
558   \ifx\#4\%
559     \ifx#31%
560       \BitSet@AfterFiFi{%
561         \expandafter\expandafter\expandafter\BitSet@Space
562         \IntCalcAdd#1!#2!%
563       }%
564     \else
565       \BitSet@AfterFiFi{ #2}%
566     \fi
567   \else
568     \ifx#31%
569       \BitSet@AfterFiFi{%
570         \csname BitSet@N#1%
571         \expandafter\expandafter\expandafter\endcsname
572         \IntCalcAdd#1!#2!!#4!%
573       }%
574     \else
575       \BitSet@AfterFiFi{%
576         \csname BitSet@N#1\endcsname#2!#4!%
577       }%
578     \fi
579   \BitSet@Fi
580 }

```

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

```

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

```

```

614 \BitSet@Temp{268435456}-{536870912}
615 \BitSet@Temp{536870912}-{1073741824}

\BitSet@N1073741824

616 \expandafter\def\csname BitSet@N1073741824\endcsname{%
617   \BitSet@GetDecBig2147483648!%
618 }%

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

619 \def\BitSet@GetDecBig#1!#2!#3#4!{%
620   \ifx\#4\%
621     \ifx#31%
622       \BitSet@AfterFiFi{%
623         \expandafter\expandafter\expandafter\BitSet@Space
624         \BigIntCalcAdd#1!#2!%
625       }%
626     \else
627       \BitSet@AfterFiFi{ #2}%
628     \fi
629   \else
630     \ifx#31%
631       \BitSet@AfterFiFi{%
632         \expandafter\expandafter\expandafter\BitSet@@GetDecBig
633         \BigIntCalcAdd#1!#2!!#1!#4!%
634       }%
635     \else
636       \BitSet@AfterFiFi{%
637         \expandafter\expandafter\expandafter\BitSet@GetDecBig
638         \BigIntCalcShl#1!!#2!#4!%
639       }%
640     \fi
641   \BitSet@Fi
642 }

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

643 \def\BitSet@@GetDecBig#1!#2!{%
644   \expandafter\expandafter\expandafter\BitSet@GetDecBig
645   \BigIntCalcShl#2!!#1!%
646 }

```

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

```

647 \def\bitsetAnd#1#2{%
648   \bitsetIsEmpty{#1}{%
649     \bitsetReset{#1}%
650   }{%
651     \bitsetIsEmpty{#2}{%
652       \bitsetReset{#1}%
653     }{%

```

```

654 \expandafter\edef\csname BS@#1\endcsname{%
655 \expandafter\expandafter\expandafter\BitSet@And
656 \csname BS@#1\expandafter\expandafter\expandafter\endcsname
657 \expandafter\expandafter\expandafter!%
658 \csname BS@#2\endcsname!%!%
659 }%
660 \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
661 \bitsetReset{#1}%
662 \fi
663 }%
664 }%
665 }

```

\BitSet@And

```

666 \def\BitSet@And#1#2!#3#4!#5!{%
667 \ifx\#2\%
668 \ifnum#1#3=11 #5!\fi
669 \else
670 \ifx\#4\%
671 \ifnum#1#3=11 #5!\fi
672 \else
673 \ifnum#1#3=11 %
674 #5!%
675 \BitSet@AfterFiFiFi{%
676 \BitSet@And#2!#4!%!%
677 }%
678 \else
679 \BitSet@AfterFiFiFi{%
680 \BitSet@And#2!#4!#50!%
681 }%
682 \fi
683 \fi
684 \BitSet@Fi
685 }

```

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

```

686 \def\bitsetAndNot#1#2{%
687 \bitsetIsEmpty{#1}{%
688 \bitsetReset{#1}%
689 }{%
690 \bitsetIsEmpty{#2}{%
691 }{%
692 \expandafter\edef\csname BS@#1\endcsname{%
693 \expandafter\expandafter\expandafter\BitSet@AndNot
694 \csname BS@#1\expandafter\expandafter\expandafter\endcsname
695 \expandafter\expandafter\expandafter!%
696 \csname BS@#2\endcsname!%!%
697 }%
698 \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
699 \bitsetReset{#1}%
700 \fi
701 }%
702 }%
703 }

```

\BitSet@AndNot

```

704 \def\BitSet@AndNot#1#2!#3#4!#5!{%
705   \ifx\#2\%
706     \ifnum#1#3=10 #5!\fi
707   \else
708     \ifx\#4\%
709       #5%
710       \ifnum#1#3=10 1\else 0\fi
711     #2%
712   \else
713     \ifnum#1#3=10 %
714       #51%
715       \BitSet@AfterFiFiFi{%
716         \BitSet@AndNot#2!#4!#5!%
717       }%
718   \else
719     \BitSet@AfterFiFiFi{%
720       \BitSet@AndNot#2!#4!#50!%
721     }%
722   \fi
723 \fi
724 \BitSet@Fi
725 }

```

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

```

726 \def\bitsetOr#1#2{%
727   \bitsetIsEmpty{#2}{%
728     \BitSet@IfUndefined{#1}{\bitsetReset{#1}}{%
729   }{%
730     \bitsetIsEmpty{#1}{%
731       \expandafter\let\csname BS@#1\expandafter\endcsname
732         \csname BS@#2\endcsname
733     }{%
734       \expandafter\edef\csname BS@#1\endcsname{%
735         \expandafter\expandafter\expandafter\BitSet@Or
736         \csname BS@#1\expandafter\expandafter\expandafter\endcsname
737         \expandafter\expandafter\expandafter!%
738         \csname BS@#2\endcsname!%
739       }%
740     }%
741   }%
742 }

```

\BitSet@Or

```

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

```



```

753     }%
754     \fi
755     \BitSet@Fi
756 }

```

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

```

757 \def\bitsetXor#1#2{%
758   \bitsetIsEmpty{#2}{%
759     \BitSet@IfUndefined{#1}{\bitsetReset{#1}}{%
760     }{%
761       \bitsetIsEmpty{#1}{%
762         \expandafter\let\csname BS@#1\expandafter\endcsname
763           \csname BS@#2\endcsname
764       }{%
765         \expandafter\edef\csname BS@#1\endcsname{%
766           \expandafter\expandafter\expandafter\BitSet@Xor
767             \csname BS@#1\expandafter\expandafter\expandafter\endcsname
768             \expandafter\expandafter\expandafter!%
769             \csname BS@#2\endcsname!%!%
770         }%
771         \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
772           \bitsetReset{#1}%
773         \fi
774       }%
775     }%
776 }

```

\BitSet@Xor

```

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

```

```

803 \fi
804 \BitSet@Fi
805 }

```

2.8.5 Shifting

2.8.6 \bitsetShiftLeft

\bitsetShiftLeft

```

806 \def\bitsetShiftLeft#1#2{%
807 \BitSet@IfUndefined{#1}{%
808 \bitsetReset{#1}%
809 }{%
810 \bitsetIsEmpty{#1}{%
811 }{%
812 \expandafter\expandafter\expandafter\BitSet@ShiftLeft
813 \intcalcNum{#2}!{#1}%
814 }%
815 }%
816 }

```

\BitSet@ShiftLeft

```

817 \def\BitSet@ShiftLeft#1!#2{%
818 \ifcase\intcalcSgn{#1} %
819 \or
820 \begingroup
821 \uccode'm='0 %
822 \uppercase\expandafter{\expandafter\endgroup
823 \expandafter\edef\csname BS@#2\expandafter\endcsname
824 \expandafter{%
825 \romannumeral#1000\expandafter\BitSet@Space
826 \csname BS@#2\endcsname
827 }%
828 }%
829 \else
830 \expandafter\BitSet@ShiftRight\BitSet@Gobble#1!{#2}%
831 \fi
832 }

```

2.8.7 \bitsetShiftRight

\bitsetShiftRight

```

833 \def\bitsetShiftRight#1#2{%
834 \BitSet@IfUndefined{#1}{%
835 \bitsetReset{#1}%
836 }{%
837 \bitsetIsEmpty{#1}{%
838 }{%
839 \expandafter\expandafter\expandafter\BitSet@ShiftRight
840 \intcalcNum{#2}!{#1}%
841 }%
842 }%
843 }

```

\BitSet@ShiftRight

```

844 \def\BitSet@ShiftRight#1!#2{%
845 \ifcase\intcalcSgn{#1} %
846 \or
847 \expandafter\edef\csname BS@#2\endcsname{%
848 \expandafter\expandafter\expandafter\BitSet@Kill
849 \csname BS@#2\expandafter\endcsname\expandafter\BitSet@Empty
850 \expandafter=%

```

```

851     \expandafter{\expandafter}\expandafter{\expandafter}%
852     \romannumeral#1000!%
853 }%
854 \else
855     \expandafter\BitSet@ShiftLeft\BitSet@Gobble#1!{#2}%
856 \fi
857 }

```

\BitSet@Kill

```

858 \def\BitSet@Kill#1#2=#3#4#5{%
859     #3#4%
860     \ifx#5!%
861         \ifx#1\BitSet@Empty
862             0%
863         \else
864             #1#2%
865         \fi
866     \else
867         \ifx#1\BitSet@Empty
868             0%
869             \BitSet@AfterFiFi\BitSet@Cleanup
870         \else
871             \BitSet@Kill#2=%
872         \fi
873 \BitSet@Fi
874 }

```

2.9 Bit manipulation

\bitsetClear

```

875 \def\bitsetClear{%
876     \BitSet@CheckIndex\BitSet@Clear
877 }

```

\bitsetSet

```

878 \def\bitsetSet{%
879     \BitSet@CheckIndex\BitSet@Set
880 }

```

\bitsetFlip

```

881 \def\bitsetFlip{%
882     \BitSet@CheckIndex\BitSet@Flip
883 }

```

\bitsetSetValue

```

884 \def\bitsetSetValue#1#2#3{%
885     \expandafter\expandafter\expandafter\BitSet@SetValue
886     \intcalculus{#3}!{#1}{#2}%
887 }

```

\BitSet@SetValue #1: plain value
#2: BitSet
#3: Index

```

888 \def\BitSet@SetValue#1!{%
889     \BitSet@CheckIndex{%
890         \ifcase#1 %
891             \expandafter\BitSet@Clear
892         \or
893             \expandafter\BitSet@Set
894         \else
895             \BitSet@ErrorInvalidBitValue{#1}%

```

```

896     \expandafter\expandafter\expandafter\BitSet@Gobble
897     \expandafter\BitSet@Gobble
898   \fi
899 }%
900 }

```

```

\BitSet@ErrorInvalidBitValue #1: Wrong bit value

901 \def\BitSet@ErrorInvalidBitValue#1{%
902   \@PackageError{bitset}{%
903     Invalid bit value (#1) not in range 0..1%
904   }\@ehc
905 }

```

2.9.1 Clear operation

```

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

906 \def\BitSet@Clear#1#2{%
907   \edef\BitSet@Temp{%
908     \expandafter\expandafter\expandafter\BitSet@@Clear
909     \csname BS@#1\expandafter\endcsname
910     \expandafter\BitSet@Empty\expandafter=\expandafter!%
911     \romannumeral#2000!%
912   }%
913   \expandafter\let\csname BS@#1\expandafter\endcsname
914   \ifx\BitSet@Temp\BitSet@Empty
915     \BitSet@Zero
916   \else
917     \BitSet@Temp
918   \fi
919 }

```

```

\BitSet@@Clear

920 \def\BitSet@@Clear#1#2=#3!#4{%
921   \ifx#4!%
922     \ifx#1\BitSet@Empty
923     \else
924       \ifx\BitSet@Empty#2%
925       \else
926         #30#2%
927       \fi
928     \fi
929   \else
930     \ifx#1\BitSet@Empty
931       \BitSet@AfterFiFi\BitSet@Cleanup
932     \else
933       \ifx#10%
934         \BitSet@AfterFiFiFi{%
935           \BitSet@@Clear#2=#30!%
936         }%
937       \else
938         #31%
939         \BitSet@AfterFiFiFi{%
940           \BitSet@@Clear#2=!%
941         }%
942       \fi
943     \fi
944   \BitSet@Fi
945 }

```

2.9.2 Set operation

```

\BitSet@Set #1: BitSet
#2: plain and checked Index
946 \def\BitSet@Set#1#2{%
947   \expandafter\edef\csname BS@#1\endcsname{%
948     \expandafter\expandafter\expandafter\BitSet@Set
949     \csname BS@#1\expandafter\endcsname
950     \expandafter\BitSet@Empty\expandafter=%
951     \expandafter{\expandafter}\expandafter{\expandafter}%
952     \romannumeral#2000!%
953   }%
954 }

\BitSet@@Set

955 \def\BitSet@@Set#1#2=#3#4#5{%
956   #3#4%
957   \ifx#5!%
958     1#2%
959   \else
960     \ifx#1\BitSet@Empty
961       0%
962       \BitSet@AfterFiFi\BitSet@@@Set
963     \else
964       #1%
965       \BitSet@@Set#2=%
966     \fi
967     \BitSet@Fi
968 }

\BitSet@@@Set

969 \def\BitSet@@@Set#1{%
970   \ifx#1!%
971     1%
972   \else
973     0%
974     \expandafter\BitSet@@@Set
975   \fi
976 }

```

2.9.3 Flip operation

```

\BitSet@Flip #1: BitSet
#2: plain and checked Index
977 \def\BitSet@Flip#1#2{%
978   \edef\BitSet@Temp{%
979     \expandafter\expandafter\expandafter\BitSet@@Flip
980     \csname BS@#1\expandafter\endcsname
981     \expandafter\BitSet@Empty\expandafter=\expandafter!%
982     \romannumeral#2000!%
983   }%
984   \expandafter\let\csname BS@#1\expandafter\endcsname
985   \ifx\BitSet@Temp\BitSet@Empty
986     \BitSet@Zero
987   \else
988     \BitSet@Temp
989   \fi
990 }

\BitSet@@Flip

991 \def\BitSet@@Flip#1#2=#3!#4{%

```

```

992 \ifx#4!%
993 \ifx#11%
994 \ifx\BitSet@Empty#2%
995 \else
996 #30#2%
997 \fi
998 \else
999 #31#2%
1000 \fi
1001 \else
1002 \ifx#1\BitSet@Empty
1003 #30%
1004 \BitSet@AfterFiFi\BitSet@@@Set
1005 \else
1006 \ifx#10%
1007 \BitSet@AfterFiFiFi{%
1008 \BitSet@@Flip#2=#30!%
1009 }%
1010 \else
1011 #31%
1012 \BitSet@AfterFiFiFi{%
1013 \BitSet@@Flip#2=!%
1014 }%
1015 \fi
1016 \fi
1017 \BitSet@Fi
1018 }

```

2.9.4 Range operators

\bitsetClearRange

```

1019 \def\bitsetClearRange{%
1020 \BitSet@Range\BitSet@Clear
1021 }

```

\bitsetSetRange

```

1022 \def\bitsetSetRange{%
1023 \BitSet@Range\BitSet@Set
1024 }

```

\bitsetFlipRange

```

1025 \def\bitsetFlipRange{%
1026 \BitSet@Range\BitSet@Flip
1027 }

```

\bitsetSetValueRange

```

1028 \def\bitsetSetValueRange#1#2#3#4{%
1029 \expandafter\expandafter\expandafter\BitSet@SetValueRange
1030 \intcalcNum{#4}!{#1}{#2}{#3}%
1031 }

```

\BitSet@SetValueRange

```

1032 \def\BitSet@SetValueRange#1!#2#3#4{%
1033 \ifcase#1 %
1034 \BitSet@Range\BitSet@Clear{#2}{#3}{#4}%
1035 \or
1036 \BitSet@Range\BitSet@Set{#2}{#3}{#4}%
1037 \else
1038 \BitSet@ErrorInvalidBitValue{#1}%
1039 \fi
1040 }

```

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

```
1041 \def\BitSet@Range#1#2#3#4{%
1042   \edef\BitSet@Temp{%
1043     \noexpand\BitSet@@Range\noexpand#1{#2}%
1044     \intcalcNum{#3}!\intcalcNum{#4}!%
1045   }%
1046   \BitSet@Temp
1047 }
```

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

```
1048 \def\BitSet@@Range#1#2#3!#4!{%
1049   \ifnum#3<0 %
1050     \BitSet@NegativeIndex#1{#2}#3!#4!0!#4!%
1051   \else
1052     \ifnum#4<0 %
1053       \BitSet@NegativeIndex#1{#2}#3!#4!#3!0!%
1054     \else
1055       \ifcase\intcalcCmp{#3}{#4} %
1056       \or
1057         \@PackageError{bitset}{%
1058           Wrong index numbers in range [#3..#4]\MessageBreak% hash-ok
1059           for clear/set/flip on bit set '#2'.\MessageBreak
1060           The lower index exceeds the upper index.\MessageBreak
1061           Canceling the operation as error recovery%
1062         }\@ehc
1063       \else
1064         \BitSet@@Range#3!#4!#1{#2}%
1065       \fi
1066     \fi
1067   \fi
1068 }
```

\BitSet@NegativeIndex

```
1069 \def\BitSet@NegativeIndex#1#2#3!#4!#5!#6!{%
1070   \@PackageError{bitset}{%
1071     Negative index in range [#3..#4]\MessageBreak % hash-ok
1072     for \string\bitset
1073     \ifx#1\BitSet@Clear
1074       Clear%
1075     \else
1076       \ifx#1\BitSet@Set
1077         Set%
1078       \else
1079         Flip%
1080       \fi
1081     \fi
1082     Range on bit set '#2'.\MessageBreak
1083     Using [#5..#6] as error recovery% hash-ok
1084   }\@ehc
1085   \BitSet@@Range#1{#2}#5!#6!%
1086 }
```

\BitSet@@@Range

```
1087 \def\BitSet@@@Range#1!#2!#3#4{%
1088   \ifnum#1<#2 %
```

```

1089     #3{#4}{#1}%
1090     \BitSet@AfterFi{%
1091         \expandafter\expandafter\expandafter\BitSet@@@Range
1092         \IntCalcInc#1!!#2!#3{#4}%
1093     }%
1094     \BitSet@Fi
1095 }

```

2.10 Bit retrieval

2.10.1 \bitsetGet

\bitsetGet

```

1096 \def\bitsetGet#1#2{%
1097     \number
1098     \expandafter\expandafter\expandafter\BitSet@Get
1099     \intcalcNum{#2}{#1}%
1100 }

```

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

```

1101 \def\BitSet@Get#1!#2{%
1102     \ifnum#1<0 %
1103         \BitSet@AfterFi{%
1104             0 \BitSetError:NegativeIndex%
1105         }%
1106     \else
1107         \BitSet@IfUndefined{#2}{0}{%
1108             \expandafter\expandafter\expandafter\BitSet@@@Get
1109             \csname BS@#2\expandafter\endcsname
1110             \expandafter!\expandafter=%
1111             \expandafter{\expandafter}\expandafter{\expandafter}%
1112             \romannumeral\intcalcNum{#1}000!%
1113         }%
1114         \expandafter\BitSet@Space
1115     \BitSet@Fi
1116 }

```

\BitSet@@@Get

```

1117 \def\BitSet@@@Get#1#2=#3#4#5{%
1118     #3#4%
1119     \ifx#5!%
1120         \ifx#1!%
1121             0%
1122         \else
1123             #1%
1124         \fi
1125     \else
1126         \ifx#1!%
1127             0%
1128             \BitSet@AfterFiFi\BitSet@Cleanup
1129         \else
1130             \BitSet@@@Get#2=%
1131         \fi
1132     \BitSet@Fi
1133 }

```

2.10.2 \bitsetNextClearBit, \bitsetNextSetBit

\bitsetNextClearBit

```

1134 \def\bitsetNextClearBit#1#2{%

```



```

1135 \number
1136 \expandafter\expandafter\expandafter\BitSet@NextClearBit
1137 \intcalcNum{#2}!{#1} %
1138 }

\BitSet@NextClearBit #1: Index
#2: BitSet
1139 \def\BitSet@NextClearBit#1!#2{%
1140 \ifnum#1<0 %
1141 \BitSet@NextClearBit0!{#2}%
1142 \BitSet@AfterFi{%
1143 \expandafter\BitSet@Space
1144 \expandafter\BitSetError:NegativeIndex\romannumeral0%
1145 }%
1146 \else
1147 \bitsetIsEmpty{#2}{#1}{%
1148 \expandafter\BitSet@Skip
1149 \number#1\expandafter\expandafter\expandafter!%
1150 \csname BS@#2\endcsname!!!!!!!!!=%
1151 {\BitSet@@NextClearBit#1!}%
1152 }%
1153 \BitSet@Fi
1154 }

\BitSet@@NextClearBit #1: index for next bit in #2
#2: next bit
1155 \def\BitSet@@NextClearBit#1!#2{%
1156 \ifx#2!%
1157 #1%
1158 \else
1159 \ifx#20%
1160 #1%
1161 \BitSet@AfterFiFi\BitSet@Cleanup
1162 \else
1163 \BitSet@AfterFiFi{%
1164 \expandafter\expandafter\expandafter\BitSet@@NextClearBit
1165 \IntCalcInc#1!!%
1166 }%
1167 \fi
1168 \BitSet@Fi
1169 }

\bitsetNextSetBit
1170 \def\bitsetNextSetBit#1#2{%
1171 \number
1172 \expandafter\expandafter\expandafter\BitSet@NextSetBit
1173 \intcalcNum{#2}!{#1} %
1174 }

\BitSet@NextSetBit #1: Index
#2: BitSet
1175 \def\BitSet@NextSetBit#1!#2{%
1176 \ifnum#1<0 %
1177 \BitSet@NextSetBit0!{#2}%
1178 \BitSet@AfterFi{%
1179 \expandafter\BitSet@Space
1180 \expandafter\BitSetError:NegativeIndex\romannumeral0%
1181 }%
1182 \else
1183 \bitsetIsEmpty{#2}{-1}{%
1184 \expandafter\BitSet@Skip
1185 \number#1\expandafter\expandafter\expandafter!%

```

```

1186     \csname BS@#2\endcsname!!!!!!!!!=%
1187     {\BitSet@@NextSetBit#1!}%
1188 }%
1189 \BitSet@Fi
1190 }

\BitSet@@NextSetBit #1: index for next bit in #2
#2: next bit
1191 \def\BitSet@@NextSetBit#1!#2{%
1192   \ifx#2!%
1193     -1%
1194   \else
1195     \ifx#21%
1196       #1%
1197       \BitSet@AfterFiFi\BitSet@Cleanup
1198     \else
1199       \BitSet@AfterFiFi{%
1200         \expandafter\expandafter\expandafter\BitSet@@NextSetBit
1201         \IntCalcInc#1!!%
1202       }%
1203     \fi
1204   \BitSet@Fi
1205 }

\BitSet@Cleanup
1206 \def\BitSet@Cleanup#1!{%

\BitSet@Skip #1: number of bits to skip
#2: bits
#3: continuation code
1207 \def\BitSet@Skip#1!#2{%
1208   \ifx#2!%
1209     \BitSet@AfterFi{%
1210       \BitSet@SkipContinue%
1211     }%
1212   \else
1213     \ifcase#1 %
1214       \BitSet@AfterFiFi{%
1215         \BitSet@SkipContinue#2%
1216       }%
1217     \or
1218       \BitSet@AfterFiFi\BitSet@SkipContinue
1219     \or
1220       \BitSet@AfterFiFi{%
1221         \expandafter\BitSet@SkipContinue\BitSet@Gobble
1222       }%
1223     \else
1224       \ifnum#1>8 %
1225         \BitSet@AfterFiFiFi{%
1226           \expandafter\BitSet@Skip
1227           \number\IntCalcSub#1!8!\expandafter!%
1228           \BitSet@GobbleSeven
1229         }%
1230       \else
1231         \BitSet@AfterFiFiFi{%
1232           \expandafter\expandafter\expandafter\BitSet@Skip
1233           \IntCalcDec#1!!%
1234         }%
1235       \fi
1236     \fi
1237   \BitSet@Fi
1238 }

```

```

\BitSet@SkipContinue #1: remaining bits
#2: continuation code
1239 \def\BitSet@SkipContinue#1!#2=#3{%
1240   #3#1!%
1241 }

\BitSet@GobbleSeven
1242 \def\BitSet@GobbleSeven#1#2#3#4#5#6#7{}

```

2.10.3 \bitsetGetSetBitList

```

\bitsetGetSetBitList It's just a wrapper for \bitsetNextSetBit.
1243 \def\bitsetGetSetBitList#1{%
1244   \romannumeral0%
1245   \bitsetIsEmpty{#1}{ }{%
1246     \expandafter\BitSet@GetSetBitList
1247     \number\BitSet@NextSetBit0!{#1}!{#1}!{%
1248   }%
1249 }

\BitSet@GetSetBitList #1: found index
#2: BitSet
#3: comma #4: result
1250 \def\BitSet@GetSetBitList#1!#2#3#4!{%
1251   \ifnum#1<0 %
1252     \BitSet@AfterFi{ #4}%
1253   \else
1254     \BitSet@AfterFi{%
1255       \expandafter\BitSet@GetSetBitList\number
1256       \expandafter\expandafter\expandafter\BitSet@NextSetBit
1257       \IntCalcInc#1!!{#2}!{#2},#4#3#1!%
1258     }%
1259   \BitSet@Fi
1260 }

```

2.11 Bit set properties

```

\bitsetSize
1261 \def\bitsetSize#1{%
1262   \number
1263   \BitSet@IfUndefined{#1}{0 }{%
1264     \expandafter\expandafter\expandafter\BitSet@Size
1265     \expandafter\expandafter\expandafter1%
1266     \expandafter\expandafter\expandafter!%
1267     \csname BS@#1\endcsname!0!%
1268   }%
1269 }

\BitSet@Size #1: counter
#2#3: bits
#4: result
1270 \def\BitSet@Size#1!#2#3!#4!{%
1271   \ifx#21%
1272     \ifx\#3\%
1273       \BitSet@AfterFiFi{#1 }%
1274     \else
1275       \BitSet@AfterFiFi{%
1276         \expandafter\expandafter\expandafter\BitSet@Size
1277         \IntCalcInc#1!!#3!#1!%
1278       }%

```

```

1279     \fi
1280 \else
1281     \ifx\\#3\\%
1282         \BitSet@AfterFiFi{#4 }%
1283     \else
1284         \BitSet@AfterFiFi{%
1285             \expandafter\expandafter\expandafter\BitSet@Size
1286             \IntCalcInc#1!!#3!#4!%
1287         }%
1288     \fi
1289 \fi
1290 \BitSet@Fi
1291 }

```

\bitsetCardinality

```

1292 \def\bitsetCardinality#1{%
1293     \number
1294     \BitSet@IfUndefined{#1}{0 }{%
1295         \expandafter\expandafter\expandafter\BitSet@Cardinality
1296         \expandafter\expandafter\expandafter0%
1297         \expandafter\expandafter\expandafter!%
1298         \csname BS@#1\endcsname!%
1299     }%
1300 }

```

\BitSet@Cardinality #1: result
#2#3: bits

```

1301 \def\BitSet@Cardinality#1!#2#3!{%
1302     \ifx#21%
1303         \ifx\\#3\\%
1304             \BitSet@AfterFiFi{\IntCalcInc#1! }%
1305         \else
1306             \BitSet@AfterFiFi{%
1307                 \expandafter\expandafter\expandafter\BitSet@Cardinality
1308                 \IntCalcInc#1!!#3!%
1309             }%
1310         \fi
1311     \else
1312         \ifx\\#3\\%
1313             \BitSet@AfterFiFi{#1 }%
1314         \else
1315             \BitSet@AfterFiFi{%
1316                 \BitSet@Cardinality#1!#3!%
1317             }%
1318         \fi
1319     \fi
1320 \BitSet@Fi
1321 }

```

2.12 Queries

\bitsetIsDefined

```

1322 \def\bitsetIsDefined#1{%
1323     \BitSet@IfUndefined{#1}%
1324     \BitSet@SecondOfTwo
1325     \BitSet@FirstOfTwo
1326 }

```

\bitsetIsEmpty

```

1327 \def\bitsetIsEmpty#1{%
1328     \BitSet@IfUndefined{#1}\BitSet@FirstOfTwo{%

```

```

1329     \expandafter\ifx\csname BS@#1\endcsname\BitSet@Zero
1330     \expandafter\BitSet@FirstOfTwo
1331     \else
1332     \expandafter\BitSet@SecondOfTwo
1333     \fi
1334 }%
1335 }

\BitSet@Zero

1336 \def\BitSet@Zero{0}

\bitsetQuery

1337 \def\bitsetQuery#1#2{%
1338     \ifnum\bitsetGet{#1}{#2}=1 %
1339     \expandafter\BitSet@FirstOfTwo
1340     \else
1341     \expandafter\BitSet@SecondOfTwo
1342     \fi
1343 }

\bitsetEquals

1344 \def\bitsetEquals#1#2{%
1345     \BitSet@IfUndefined{#1}{%
1346     \BitSet@IfUndefined{#2}\BitSet@FirstOfTwo\BitSet@SecondOfTwo
1347     }{%
1348     \BitSet@IfUndefined{#2}\BitSet@SecondOfTwo{%
1349     \expandafter\ifx\csname BS@#1\endcsname
1350     \csname BS@#2\endcsname
1351     \expandafter\BitSet@FirstOfTwo
1352     \else
1353     \expandafter\BitSet@SecondOfTwo
1354     \fi
1355     }%
1356 }%
1357 }

\bitsetIntersects

1358 \def\bitsetIntersects#1#2{%
1359     \bitsetIsEmpty{#1}\BitSet@SecondOfTwo{%
1360     \bitsetIsEmpty{#2}\BitSet@SecondOfTwo{%
1361     \expandafter\expandafter\expandafter\BitSet@Intersects
1362     \csname BS@#1\endcsname\expandafter\expandafter\expandafter\endcsname
1363     \expandafter\expandafter\expandafter!%
1364     \csname BS@#2\endcsname!%
1365     }%
1366 }%
1367 }

\BitSet@Intersects

1368 \def\BitSet@Intersects#1#2!#3#4!{%
1369     \ifnum#1#3=11 %
1370     \BitSet@AfterFi\BitSet@FirstOfTwo
1371     \else
1372     \ifx\#2\%
1373     \BitSet@AfterFiFi\BitSet@SecondOfTwo
1374     \else
1375     \ifx\#4\%
1376     \BitSet@AfterFiFiFi\BitSet@SecondOfTwo
1377     \else
1378     \BitSet@AfterFiFiFiFi%
1379     \BitSet@Intersects#2!#4!%

```

```

1380     }%
1381     \fi
1382     \fi
1383     \BitSet@Fi
1384 }

1385 \BitSet@AtEnd
1386 \endpackage

```

3 Test

3.1 Catcode checks for loading

```

1387 <*test1>

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

```

```

1436 \ifnum\count@<#2\relax
1437 \advance\count@ 1 %
1438 \repeat
1439 }
1440 \expandafter\ifx\csname LoadCommand\endcsname\relax
1441 \def\LoadCommand{\input bitset.sty\relax}%
1442 \fi
1443 \def\Test{%
1444 \RangeCatcodeInvalid{0}{47}%
1445 \RangeCatcodeInvalid{58}{64}%
1446 \RangeCatcodeInvalid{91}{96}%
1447 \RangeCatcodeInvalid{123}{255}%
1448 \catcode'\@=12 %
1449 \catcode'\=0 %
1450 \catcode'\{=1 %
1451 \catcode'\}=2 %
1452 \catcode'\#=6 %
1453 \catcode'\[=12 %
1454 \catcode'\]=12 %
1455 \catcode'\%=14 %
1456 \catcode'\ =10 %
1457 \catcode13=5 %
1458 \LoadCommand
1459 \RestoreCatcodes
1460 }
1461 \Test
1462 \csname @@end\endcsname
1463 \end
1464 </test1>

```

3.2 Macro tests

3.2.1 Preamble

```

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

```

```

1494 \IncludeTests{*}
1495 \LogTests{log}{*}{*}
1496 \makeatletter

```

3.2.2 Time

```

1497 \begingroup\expandafter\expandafter\expandafter\endgroup
1498 \expandafter\ifx\csname pdfresettimer\endcsname\relax
1499 \else
1500 \newcount\SummaryTime
1501 \newcount\TestTime
1502 \SummaryTime=\z@
1503 \newcommand*{\PrintTime}[2]{%
1504 \typeout{%
1505 [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
1506 }%
1507 }%
1508 \newcommand*{\StartTime}[1]{%
1509 \renewcommand*{\TimeDescription}{#1}%
1510 \pdfresettimer
1511 }%
1512 \newcommand*{\TimeDescription}{}%
1513 \newcommand*{\StopTime}{%
1514 \TestTime=\pdfelapseddtime
1515 \global\advance\SummaryTime\TestTime
1516 \PrintTime\TimeDescription\TestTime
1517 }%
1518 \let\saved@qstest\qstest
1519 \let\saved@endqstest\endqstest
1520 \def\qstest#1#2{%
1521 \saved@qstest{#1}{#2}%
1522 \StartTime{#1}%
1523 }%
1524 \def\endqstest{%
1525 \StopTime
1526 \saved@endqstest
1527 }%
1528 \AtEndDocument{%
1529 \PrintTime{summary}\SummaryTime
1530 }%
1531 \fi

```

3.2.3 Detection of unwanted space

```

1532 \let\orig@qstest\qstest
1533 \let\orig@endqstest\endqstest
1534 \def\qstest#1#2{%
1535 \orig@qstest{#1}{#2}%
1536 \setbox0\hbox\bgroup\begingroup\ignorespaces
1537 }
1538 \def\endqstest{%
1539 \endgroup\egroup
1540 \Expect*{\the\wd0}{0.0pt}%
1541 \orig@endqstest
1542 }

```

3.2.4 Test macros

```

1543 \newcounter{Test}
1544
1545 \def\TestError#1#2{%
1546 \begingroup
1547 \setcounter{Test}{0}%
1548 \sbox0{%
1549 \def\@PackageError##1##2##3{%

```



```

1550         \stepcounter{Test}%
1551         \begingroup
1552         \let\MessageBreak\relax
1553 <*noetex>
1554         \ETEXEnable
1555 </noetex>
1556         \Expect{##1}{bitset}%
1557         \Expect*{##2}*{#1}%
1558         \endgroup
1559     }%
1560 <*noetex>
1561         \ETEXDisable
1562 </noetex>
1563     #2%
1564 }%
1565 \Expect*{\theTest}{1}%
1566 \Expect*{\the\wd0}{0.0pt}%
1567 \endgroup
1568 }
1569
1570 \def\TestErrorNegativeIndex#1#2{%
1571 \TestError{Invalid negative index (#1)}{#2}%
1572 }
1573
1574 \def\TestGetterUndefined#1{%
1575 \CheckUndef{dummy}%
1576 \expandafter\expandafter\expandafter\Expect
1577 \expandafter\expandafter\expandafter{#1{dummy}}{0}%
1578 }
1579
1580 \def\ExpectBitSet#1#2{%
1581 \expandafter\expandafter\expandafter\Expect
1582 \expandafter\expandafter\expandafter
1583 {\csname BS@#1\endcsname}*{#2}%
1584 }
1585 \def\Check#1#2{%
1586 \ExpectBitSet{#1}{#2}%
1587 }
1588 \def\CheckUndef#1{%
1589 \begingroup
1590 \Expect*{%
1591 \expandafter
1592 \ifx\csname BS@#1\endcsname\relax true\else false\fi
1593 }{true}%
1594 \endgroup
1595 }
1596 \def\RevCheck#1#2{%
1597 \ExpectBitSet{#1}{\Reverse#2!!}%
1598 }
1599 \def\Set#1#2{%
1600 \expandafter\def\csname BS@#1\endcsname{#2}%
1601 }
1602 \def\RevSet#1#2{%
1603 \expandafter\edef\csname BS@#1\endcsname{%
1604 \Reverse#2!!%
1605 }%
1606 }
1607 \def\Reverse#1#2!#3!{%
1608 \ifx\#2\%
1609 #1#3%
1610 \expandafter\@gobble
1611 \else

```

```

1612     \expandafter\@firstofone
1613     \fi
1614     {\Reverse#2!#1#3!}%
1615 }

```

3.2.5 Test sets

```

1616 \begin{qstest}{Let}{Let}
1617   \CheckUndef{abc}%
1618   \CheckUndef{xyz}%
1619   \bitsetLet{xyz}{abc}%
1620   \CheckUndef{abc}%
1621   \Check{xyz}{0}%
1622   \Set{abc}{1}%
1623   \Check{abc}{1}%
1624   \Check{xyz}{0}%
1625   \bitsetLet{xyz}{abc}%
1626   \Check{abc}{1}%
1627   \Check{xyz}{1}%
1628   \Set{xyz}{11}%
1629   \Check{abc}{1}%
1630   \Check{xyz}{11}%
1631 \end{qstest}
1632
1633 \begin{qstest}{Reset}{Reset}
1634   \bitsetReset{xyz}%
1635   \Check{xyz}{0}%
1636   \bitsetReset{abc}%
1637   \Check{abc}{0}%
1638   \Set{abc}{10101}%
1639   \bitsetReset{abc}%
1640   \Check{abc}{0}%
1641 \end{qstest}
1642
1643 \begin{qstest}{Get/Query}{Get/Query}
1644   \expandafter\expandafter\expandafter\Expect
1645   \expandafter\expandafter\expandafter{%
1646     \bitsetGet{dummy}{0}%
1647   }{0}%
1648   \begingroup
1649     \expandafter\def\csname BitSetError:NegativeIndex\endcsname{}%
1650     \Set{abc}{1}%
1651     \Expect*{\bitsetQuery{abc}{-1}{true}{false}}{false}%
1652   \endgroup
1653   \def\Test#1#2#3{%
1654     \Set{abc}{#1}%
1655     \expandafter\expandafter\expandafter\Expect
1656     \expandafter\expandafter\expandafter{\bitsetGet{abc}{#2}}{#3}%
1657     \Expect*{\bitsetQuery{abc}{#2}{true}{false}}%
1658     *{\ifcase#3 false\or true\else error\fi}%
1659   }%
1660   \Test{1}{100}{0}%
1661   \Test{0}{0}{0}%
1662   \Test{1}{0}{1}%
1663   \Test{11}{1}{1}%
1664   \Test{111}{1}{1}%
1665   \Test{101}{1}{0}%
1666   \Test{101}{2}{1}%
1667   \Test{10100110011}{10}{1}%
1668 \end{qstest}
1669
1670 \begin{qstest}{Size}{Size}
1671   \TestGetterUndefined\bitsetSize
1672   \def\Test#1#2{%

```

```

1673     \Set{abc}{#1}%
1674     \expandafter\expandafter\expandafter\Expect
1675     \expandafter\expandafter\expandafter{\bitsetSize{abc}}{#2}%
1676 }%
1677 \Test{0}{0}%
1678 \Test{1}{1}%
1679 \Test{00}{0}%
1680 \Test{0000000}{0}%
1681 \Test{10}{1}%
1682 \Test{01}{2}%
1683 \Test{11}{2}%
1684 \Test{010}{2}%
1685 \Test{011}{3}%
1686 \Test{100110011}{9}%
1687 \Test{0000011111000001111100000}{20}%
1688 \Test{000000000000000000000000011111111111111111111}{45}%
1689 \end{qstest}
1690
1691 \begin{qstest}{Cardinality}{Cardinality}
1692   \TestGetterUndefined\bitsetCardinality
1693   \def\Test#1#2{%
1694     \Set{abc}{#1}%
1695     \expandafter\expandafter\expandafter\Expect
1696     \expandafter\expandafter\expandafter{%
1697       \bitsetCardinality{abc}%
1698     }{#2}%
1699   }%
1700   \Test{0}{0}%
1701   \Test{1}{1}%
1702   \Test{00}{0}%
1703   \Test{0000000}{0}%
1704   \Test{10}{1}%
1705   \Test{01}{1}%
1706   \Test{11}{2}%
1707   \Test{010}{1}%
1708   \Test{011}{2}%
1709   \Test{100110011}{5}%
1710   \Test{0000011111000001111100000}{10}%
1711   \Test{000000000000000000000000011111111111111111111}{20}%
1712 \end{qstest}
1713
1714 \begin{qstest}{NextClearBit/NextSetBit}{NextClearBit/NextSetBit}
1715   \def\Test#1#2{%
1716     \expandafter\expandafter\expandafter\Expect
1717     \expandafter\expandafter\expandafter{%
1718       \TestOp{abc}{#1}%
1719     }{#2}%
1720   }%
1721   \def\CLEAR{\let\TestOp\bitsetNextClearBit}%
1722   \def\SET{\let\TestOp\bitsetNextSetBit}%
1723   \begingroup
1724     \catcode'\:=11 %
1725     \bitsetSetBin{abc}{1}%
1726     \CLEAR
1727     \Test{-1}{1\BitsetError:NegativeIndex}%
1728     \SET
1729     \Test{-1}{0\BitsetError:NegativeIndex}%
1730   \endgroup
1731   \let\BS@abc\undefined
1732   \CLEAR
1733   \Test{0}{0}%
1734   \Test{1}{1}%

```

```

1735 \Test{2}{2}%
1736 \Test{100}{100}%
1737 \Set
1738 \Test{0}{-1}%
1739 \Test{1}{-1}%
1740 \Test{100}{-1}%
1741 \bitsetReset{abc}%
1742 \Clear
1743 \Test{0}{0}%
1744 \Test{1}{1}%
1745 \Test{2}{2}%
1746 \Test{100}{100}%
1747 \Set
1748 \Test{0}{-1}%
1749 \Test{1}{-1}%
1750 \Test{100}{-1}%
1751 \bitsetSetBin{abc}{1}%
1752 \Clear
1753 \Test{0}{1}%
1754 \Test{1}{1}%
1755 \Test{2}{2}%
1756 \Test{100}{100}%
1757 \Set
1758 \Test{0}{0}%
1759 \Test{1}{-1}%
1760 \Test{100}{-1}%
1761 \bitsetSetBin{abc}{111000111000111000111}%
1762 \Clear
1763 \Test{0}{3}%
1764 \Test{1}{3}%
1765 \Test{2}{3}%
1766 \Test{3}{3}%
1767 \Test{4}{4}%
1768 \Test{5}{5}%
1769 \Test{6}{9}%
1770 \Test{7}{9}%
1771 \Test{8}{9}%
1772 \Test{9}{9}%
1773 \Test{10}{10}%
1774 \Test{11}{11}%
1775 \Test{12}{15}%
1776 \Test{13}{15}%
1777 \Test{14}{15}%
1778 \Test{15}{15}%
1779 \Test{16}{16}%
1780 \Test{17}{17}%
1781 \Test{18}{21}%
1782 \Test{19}{21}%
1783 \Test{20}{21}%
1784 \Test{21}{21}%
1785 \Test{22}{22}%
1786 \Test{100}{100}%
1787 \Set
1788 \Test{0}{0}%
1789 \Test{1}{1}%
1790 \Test{2}{2}%
1791 \Test{3}{6}%
1792 \Test{4}{6}%
1793 \Test{5}{6}%
1794 \Test{6}{6}%
1795 \Test{7}{7}%
1796 \Test{8}{8}%

```

```

1797 \Test{9}{12}%
1798 \Test{10}{12}%
1799 \Test{11}{12}%
1800 \Test{12}{12}%
1801 \Test{13}{13}%
1802 \Test{14}{14}%
1803 \Test{15}{18}%
1804 \Test{16}{18}%
1805 \Test{17}{18}%
1806 \Test{18}{18}%
1807 \Test{19}{19}%
1808 \Test{20}{20}%
1809 \Test{21}{-1}%
1810 \Test{22}{-1}%
1811 \Test{100}{-1}%
1812 \bitsetSetBin{abc}{1111111}%
1813 \Clear
1814 \Test{6}{7}%
1815 \Test{7}{7}%
1816 \Test{8}{8}%
1817 \Test{100}{100}%
1818 \Set
1819 \Test{6}{6}%
1820 \Test{7}{-1}%
1821 \Test{8}{-1}%
1822 \Test{100}{-1}%
1823 \bitsetSetBin{abc}{11111111}%
1824 \Clear
1825 \Test{7}{8}%
1826 \Test{8}{8}%
1827 \Test{9}{9}%
1828 \Test{100}{100}%
1829 \Set
1830 \Test{7}{7}%
1831 \Test{8}{-1}%
1832 \Test{9}{-1}%
1833 \Test{100}{-1}%
1834 \bitsetSetBin{abc}{111111111}%
1835 \Clear
1836 \Test{8}{9}%
1837 \Test{9}{9}%
1838 \Test{10}{10}%
1839 \Test{100}{100}%
1840 \Set
1841 \Test{8}{8}%
1842 \Test{9}{-1}%
1843 \Test{10}{-1}%
1844 \Test{100}{-1}%
1845 \bitsetSetBin{abc}{1111111111}%
1846 \Clear
1847 \Test{9}{10}%
1848 \Test{10}{10}%
1849 \Test{11}{11}%
1850 \Test{100}{100}%
1851 \Set
1852 \Test{9}{9}%
1853 \Test{10}{-1}%
1854 \Test{11}{-1}%
1855 \Test{100}{-1}%
1856 \end{qstest}
1857
1858 \begin{qstest}{GetSetBitList}{GetSetBitList}

```

```

1859 \let\BS@abc\@undefined
1860 \expandafter\expandafter\expandafter\Expect
1861 \expandafter\expandafter\expandafter{%
1862   \bitsetGetSetBitList{abc}%
1863 }{}%
1864 \def\Test#1#2{%
1865   \bitsetSetBin{abc}{#1}%
1866   \expandafter\expandafter\expandafter\Expect
1867   \expandafter\expandafter\expandafter{%
1868     \bitsetGetSetBitList{abc}%
1869   }{#2}%
1870 }%
1871 \Test{0}{}%
1872 \Test{1}{0}%
1873 \Test{10}{1}%
1874 \Test{11}{0,1}%
1875 \Test{10110100}{2,4,5,7}%
1876 \Test{101101001010011}{0,1,4,6,9,11,12,14}%
1877 \end{qstest}
1878
1879 \begin{qstest}{GetDec}{GetDec}
1880 \TestGetterUndefined\bitsetGetDec
1881 \def\Test#1#2{%
1882   \RevSet{abc}{#1}%
1883 }*noetex
1884 \begin{group}\expandafter\expandafter\expandafter\endgroup
1885 

```

```

1921 \bitsetClear{abc}{2}%
1922 \RevCheck{abc}{0}%
1923 \TestErrorNegativeIndex{-1}{\bitsetClear{abc}{-1}}%
1924 \RevCheck{abc}{0}%
1925 \Test{0}{0}{0}%
1926 \Test{1}{0}{0}%
1927 \Test{111}{1}{101}%
1928 \Test{111}{30}{111}%
1929 \Test{0000111}{5}{0000111}% 111 would also be ok
1930 \Test{10000111}{5}{10000111}%
1931 \Test{1001001}{3}{1000001}%
1932 \end{qstest}
1933
1934 \begin{qstest}{Set}{Set}
1935 \def\Test#1#2#3{%
1936   \RevSet{abc}{#1}%
1937   \bitsetSet{abc}{#2}%
1938   \Expect*{\BS@abc}*{\Reverse#3!!}%
1939 }%
1940 \bitsetSet{abc}{2}%
1941 \RevCheck{abc}{100}%
1942 \TestErrorNegativeIndex{-1}{\bitsetSet{abc}{-1}}%
1943 \RevCheck{abc}{100}%
1944 \Test{0}{0}{1}%
1945 \Test{1}{0}{1}%
1946 \Test{100}{1}{110}%
1947 \Test{111}{1}{111}%
1948 \Test{11}{1}{11}%
1949 \Test{11}{2}{111}%
1950 \Test{11}{3}{1011}%
1951 \Test{111}{10}{1000000111}%
1952 \Test{0000111}{5}{0100111}% 100111 would also be ok
1953 \Test{10000111}{5}{10100111}%
1954 \Test{1000001}{3}{1001001}%
1955 \Test{1001001}{3}{1001001}%
1956 \end{qstest}
1957
1958 \begin{qstest}{Flip}{Flip}
1959 \def\Test#1#2#3{%
1960   \RevSet{abc}{#1}%
1961   \bitsetFlip{abc}{#2}%
1962   \Expect*{\BS@abc}*{\Reverse#3!!}%
1963 }%
1964 \bitsetFlip{abc}{2}%
1965 \RevCheck{abc}{100}%
1966 \TestErrorNegativeIndex{-1}{\bitsetFlip{abc}{-1}}%
1967 \RevCheck{abc}{100}%
1968 \Test{0}{0}{1}%
1969 \Test{1}{0}{0}%
1970 \Test{0}{2}{100}%
1971 \Test{100}{1}{110}%
1972 \Test{111}{1}{101}%
1973 \Test{11}{1}{1}%
1974 \Test{11}{2}{111}%
1975 \Test{11}{3}{1011}%
1976 \Test{111}{10}{1000000111}%
1977 \Test{0000111}{5}{0100111}% 100111 would also be ok
1978 \Test{10000111}{5}{10100111}%
1979 \Test{1000001}{3}{1001001}%
1980 \Test{1001001}{3}{1000001}%
1981 \Test{11111}{2}{11011}%
1982 \end{qstest}

```

```

1983
1984 \begin{qstest}{SetValue}{SetValue}
1985   \def\Test#1#2{%
1986     \TestError{Invalid bit value (#2) not in range 0..1}{%
1987       \bitsetSetValue{abc}{#1}{#2}%
1988     }%
1989   }%
1990   \Test{0}{-1}%
1991   \Test{0}{2}%
1992   \Test{0}{10}%
1993   \def\Test#1#2#3{%
1994     \let\BS@abc\@undefined
1995     \bitsetSetValue{abc}{#1}{#2}%
1996     \bitsetSetBin{result}{#3}%
1997     \Expect*{\BS@abc}*{\BS@result}%
1998   }%
1999   \Test{0}{0}{0}%
2000   \Test{0}{1}{1}%
2001   \Test{1}{0}{0}%
2002   \Test{1}{1}{10}%
2003   \def\Test#1#2#3#4{%
2004     \bitsetSetBin{abc}{#1}%
2005     \bitsetSetBin{result}{#4}%
2006     \bitsetSetValue{abc}{#2}{#3}%
2007     \Expect*{\BS@abc}*{\BS@result}%
2008   }%
2009   \Test{0}{0}{0}{0}%
2010   \Test{0}{0}{0}{0}%
2011   \Test{0}{0}{1}{1}%
2012   \Test{0}{1}{0}{0}%
2013   \Test{0}{1}{1}{10}%
2014   \Test{1010}{2}{1}{1110}%
2015   \Test{1010}{4}{1}{11010}%
2016   \Test{1010}{6}{1}{1001010}%
2017   \Test{1010}{1}{0}{1000}%
2018   \Test{1010}{2}{0}{1010}%
2019   \Test{1010}{3}{0}{10}%
2020   \Test{1010}{4}{0}{1010}%
2021   \Test{1010}{6}{0}{1010}%
2022   \Test{1010}{2}{\csname iffalse\endcsname 0\else 1\fi}{1110}%
2023   \Test{1010}{1}{\csname iffalse\endcsname 1\else 0\fi}{1000}%
2024 \end{qstest}
2025
2026 \begin{qstest}{IsDefined}{IsDefined}
2027   \let\BS@abc\@undefined
2028   \Expect*{\bitsetIsDefined{abc}{true}{false}}{false}%
2029   \bitsetReset{abc}%
2030   \Expect*{\bitsetIsDefined{abc}{true}{false}}{true}%
2031 \end{qstest}
2032
2033 \begin{qstest}{IsEmpty}{IsEmpty}
2034   \let\BS@abc\@undefined
2035   \Expect*{\bitsetIsEmpty{abc}{true}{false}}{true}%
2036   \bitsetReset{abc}%
2037   \Expect*{\bitsetIsEmpty{abc}{true}{false}}{true}%
2038   \bitsetSet{abc}{1}%
2039   \Expect*{\bitsetIsEmpty{abc}{true}{false}}{false}%
2040 \end{qstest}
2041
2042 \begin{qstest}{Equals}{Equals}
2043   \def\Test#1#2#3{%
2044     \Expect*{\bitsetEquals{#1}{#2}{true}{false}}{#3}%

```



```

2045 }%
2046 \let\BS@abc\@undefined
2047 \Test{abc}{abc}{true}%
2048 \Test{abc}{foo}{true}%
2049 \Test{foo}{abc}{true}%
2050 \bitsetReset{abc}%
2051 \Test{abc}{abc}{true}%
2052 \Test{abc}{foo}{false}%
2053 \Test{foo}{abc}{false}%
2054 \bitsetReset{foo}%
2055 \Test{abc}{foo}{true}%
2056 \Test{foo}{abc}{true}%
2057 \bitsetSet{abc}{4}%
2058 \Test{abc}{foo}{false}%
2059 \Test{foo}{abc}{false}%
2060 \bitsetFlip{foo}{4}%
2061 \Test{abc}{foo}{true}%
2062 \Test{foo}{abc}{true}%
2063 \end{qstest}
2064
2065 \begin{qstest}{Intersects}{Intersects}
2066   \def\Test#1{%
2067     \Expect*{\bitsetIntersects{abc}{foo}{true}{false}}{#1}%
2068   }%
2069   \let\BS@abc\@undefined
2070   \let\BS@foo\@undefined
2071   \Test{false}%
2072   \Set{abc}{0}%
2073   \Test{false}%
2074   \Set{foo}{0}%
2075   \Test{false}%
2076   \let\BS@abc\@undefined
2077   \Test{false}%
2078   \Set{foo}{1}%
2079   \Test{false}%
2080   \Set{abc}{0}%
2081   \Test{false}%
2082   \Set{abc}{1}%
2083   \Test{true}%
2084   \let\BS@foo\@undefined
2085   \Test{false}%
2086   \Set{foo}{0}%
2087   \Test{false}%
2088   \def\Test#1#2#3{%
2089     \bitsetSetBin{abc}{#1}%
2090     \bitsetSetBin{foo}{#2}%
2091     \Expect*{\bitsetIntersects{abc}{foo}{true}{false}}{#3}%
2092   }%
2093   \Test{1010}{0101}{false}%
2094   \Test{0}{10}{false}%
2095   \Test{1}{11}{true}%
2096   \Test{11}{1}{true}%
2097   \Test{10}{1}{false}%
2098 \end{qstest}
2099
2100 \begin{qstest}{And/AndNot/Or/Xor}{And/AndNot/Or/Xor}
2101   \def\@Test#1#2#3#4#5{%
2102     \begingroup
2103       #5%
2104     \begingroup
2105       \let\BS@foo\@undefined
2106       \csname bitset#1\endcsname{abc}{foo}%

```

```

2107         \CheckUndef{foo}%
2108         \Check{abc}{#2}%
2109     \endgroup
2110     \begingroup
2111         \bitsetReset{foo}%
2112         \csname bitset#1\endcsname{abc}{foo}%
2113         \Check{foo}{0}%
2114         \Check{abc}{#3}%
2115     \endgroup
2116     \begingroup
2117         \def\BS@foo{0101}%
2118         \csname bitset#1\endcsname{abc}{foo}%
2119         \Check{foo}{0101}%
2120         \Check{abc}{#4}%
2121     \endgroup
2122 \endgroup
2123 }%
2124 \def\Test#1{%
2125     \def\Op{#1}%
2126     \Test@
2127 }%
2128 \def\Test@#1#2#3#4#5#6#7#8#9{%
2129     \@Test\Op{#1}{#2}{#3}{%
2130         \let\BS@abc\undefined
2131     }%
2132     \@Test\Op{#4}{#5}{#6}{%
2133         \bitsetReset{abc}%
2134     }%
2135     \@Test\Op{#7}{#8}{#9}{%
2136         \def\BS@abc{1001}%
2137     }%
2138 }%
2139 \Test{And}%
2140     {0}{0}{0}%
2141     {0}{0}{0}%
2142     {0}{0}{0001}%
2143 \Test{AndNot}%
2144     {0}{0}{0}%
2145     {0}{0}{0}%
2146     {1001}{1001}{1}%
2147 \Test{Or}%
2148     {0}{0}{0101}%
2149     {0}{0}{0101}%
2150     {1001}{1001}{1101}%
2151 \Test{Xor}%
2152     {0}{0}{0101}%
2153     {0}{0}{0101}%
2154     {1001}{1001}{11}%
2155 \def\Test#1#2#3{%
2156     \bitsetSetBin{abc}{#1}%
2157     \bitsetSetBin{foo}{#2}%
2158     \csname bitset\Op\endcsname{abc}{foo}%
2159     \RevCheck{foo}{#2}%
2160     \RevCheck{abc}{#3}%
2161 }%
2162 \def\Op{And}%
2163 \Test{1}{111}{1}%
2164 \Test{111}{1}{1}%
2165 \Test{10}{111}{10}%
2166 \Test{111}{10}{10}%
2167 \Test{111}{1000}{0}%
2168 \Test{1000}{111}{0}%

```

```

2169 \def\Op{AndNot}%
2170 \Test{1010}{11}{1000}%
2171 \Test{100}{100}{0}%
2172 \Test{111}{1111}{0}%
2173 \Test{100}{111}{0}%
2174 \def\Op{Or}%
2175 \Test{0}{0}{0}%
2176 \Test{1}{0}{1}%
2177 \Test{0}{1}{1}%
2178 \Test{1}{1}{1}%
2179 \Test{1000}{10}{1010}%
2180 \Test{10}{1000}{1010}%
2181 \def\Op{Xor}%
2182 \Test{0}{0}{0}%
2183 \Test{1}{0}{1}%
2184 \Test{0}{1}{1}%
2185 \Test{1}{1}{0}%
2186 \Test{1000}{10}{1010}%
2187 \Test{10}{1000}{1010}%
2188 \Test {110011001100}%
2189 {111000111000111}%
2190 {111110100001011}%
2191 \Test{111000111000111}%
2192 {110011001100}%
2193 {111110100001011}%
2194 \end{qstest}
2195
2196 \begin{qstest}{GetUndef}{GetUndef, GetBin, GetOct, GetHex}
2197 \def\TestUndef#1#2{%
2198 \let\BS@abc\@undefined
2199 \expandafter\expandafter\expandafter\Expect
2200 \expandafter\expandafter\expandafter{%
2201 \x{abc}{#1}%
2202 }{#2}%
2203 }%
2204 \let\x\bitsetGetBin
2205 \TestUndef{-1}{0}%
2206 \TestUndef{0}{0}%
2207 \TestUndef{1}{0}%
2208 \TestUndef{2}{00}%
2209 \TestUndef{8}{00000000}%
2210 \let\x\bitsetGetOct
2211 \TestUndef{-1}{0}%
2212 \TestUndef{0}{0}%
2213 \TestUndef{1}{0}%
2214 \TestUndef{2}{0}%
2215 \TestUndef{3}{0}%
2216 \TestUndef{4}{00}%
2217 \TestUndef{5}{00}%
2218 \TestUndef{6}{00}%
2219 \TestUndef{7}{000}%
2220 \TestUndef{8}{000}%
2221 \TestUndef{9}{000}%
2222 \TestUndef{10}{0000}%
2223 \let\x\bitsetGetHex
2224 \TestUndef{-1}{0}%
2225 \TestUndef{0}{0}%
2226 \TestUndef{1}{0}%
2227 \TestUndef{2}{0}%
2228 \TestUndef{3}{0}%
2229 \TestUndef{4}{0}%
2230 \TestUndef{5}{00}%

```

```

2231 \TestUndef{6}{00}%
2232 \TestUndef{7}{00}%
2233 \TestUndef{8}{00}%
2234 \TestUndef{9}{000}%
2235 \TestUndef{10}{000}%
2236 \TestUndef{12}{000}%
2237 \TestUndef{13}{0000}%
2238 \TestUndef{16}{0000}%
2239 \TestUndef{17}{00000}%
2240 \end{qstest}
2241
2242 \begin{qstest}{SetBin}{SetBin}
2243 \def\Test#1#2{%
2244 \let\BS@abc\@undefined
2245 \bitsetSetBin{abc}{#1}%
2246 \expandafter\Expect\expandafter{\BS@abc}{#2}%
2247 }%
2248 \Test{}{0}%
2249 \Test{0}{0}%
2250 \Test{1}{1}%
2251 \Test{10}{01}%
2252 \Test{11}{11}%
2253 \Test{010}{01}%
2254 \Test{011}{11}%
2255 \Test{0010}{01}%
2256 \Test{1010}{0101}%
2257 \end{qstest}
2258
2259 \begin{qstest}{SetOct}{SetOct}
2260 \def\Test#1#2{%
2261 \bitsetSetOct{abc}{#1}%
2262 \expandafter\Expect\expandafter{\BS@abc}{#2}%
2263 }%
2264 \Test{}{0}%
2265 \Test{0}{0}%
2266 \Test{000}{0}%
2267 \Test{1}{1}%
2268 \Test{001}{1}%
2269 \Test{010}{0001}%
2270 \Test{020}{00001}%
2271 \Test{42}{010001}%
2272 \Test{377}{11111111}%
2273 \Test{0377}{11111111}%
2274 \Test{76543210}{000100010110001101011111}%
2275 \Test{ 0 7 0 7 1 }{100111000111}%
2276 \end{qstest}
2277
2278 \begin{qstest}{SetHex}{SetHex}
2279 \def\Test#1#2{%
2280 \bitsetSetHex{abc}{#1}%
2281 \expandafter\Expect\expandafter{\BS@abc}{#2}%
2282 }%
2283 \Test{}{0}%
2284 \Test{0}{0}%
2285 \Test{000}{0}%
2286 \Test{1}{1}%
2287 \Test{001}{1}%
2288 \Test{010}{00001}%
2289 \Test{020}{000001}%
2290 \Test{42}{0100001}%
2291 \Test{3F}{111111}%
2292 \Test{03F}{111111}%

```

```

2293 \Test{43210}{0000100001001100001}%
2294 \Test{98765}{10100110111000011001}%
2295 \Test{FEDCBA}{010111010011101101111111}%
2296 \Test{ 0 F 0 F 1 }{1000111100001111}%
2297 \end{qstest}
2298
2299 \begin{qstest}{SetDec}{SetDec}
2300 \def\Test#1#2{%
2301     \bitsetSetDec{abc}{#1}%
2302     \expandafter\Expect\expandafter{\BS@abc}{#2}%
2303 }%
2304 \Test{}{0}%
2305 \Test{0}{0}%
2306 \Test{000}{0}%
2307 \Test{1}{1}%
2308 \Test{7}{111}%
2309 \Test{8}{0001}%
2310 \Test{001}{1}%
2311 \Test{010}{0101}%
2312 \Test{020}{00101}%
2313 \Test{53}{101011}%
2314 \Test{255}{11111111}%
2315 \Test{256}{000000001}%
2316 \Test{999999999}{11111111001001101011001110111}%
2317 \Test{1000000000}{000000000101001101011001110111}%
2318 \Test{4210987654}{01100001010010010111111101011111}%
2319 \Test{2147483647}{11111111111111111111111111111111}%
2320 \Test{2147483648}{00000000000000000000000000000001}%
2321 \end{qstest}
2322
2323 \begin{qstest}{GetBin}{GetBin}
2324 \def\TestUndef#1#2{%
2325     \let\BS@abc\@undefined
2326     \expandafter\expandafter\expandafter\Expect
2327     \expandafter\expandafter\expandafter{%
2328         \bitsetGetBin{abc}{#1}%
2329     }{#2}%
2330 }%
2331 \TestUndef{-1}{0}%
2332 \TestUndef{0}{0}%
2333 \TestUndef{1}{0}%
2334 \TestUndef{2}{00}%
2335 \TestUndef{8}{00000000}%
2336 \def\Test#1#2{%
2337     \bitsetSetBin{abc}{#2}%
2338     \expandafter\expandafter\expandafter\Expect
2339     \expandafter\expandafter\expandafter{%
2340         \bitsetGetBin{abc}{#1}%
2341     }{#2}%
2342 }%
2343 \Test{-1}{0}%
2344 \Test{0}{0}%
2345 \Test{1}{0}%
2346 \Test{1}{1}%
2347 \Test{2}{01}%
2348 \Test{2}{10}%
2349 \Test{3}{010}%
2350 \Test{2}{00}%
2351 \Test{2}{01}%
2352 \Test{8}{00101100}%
2353 \Test{2}{10101}%
2354 \Test{-100}{11011}%

```

```

2355 \end{qstest}
2356
2357 \begin{qstest}{GetOct}{GetOct}
2358   \def\Test#1#2#3{%
2359     \edef\x{\zap@space#1 \@empty}%
2360     \edef\x{\noexpand\bitsetSetBin{abc}{\x}}%
2361     \x
2362     \expandafter\expandafter\expandafter\Expect
2363     \expandafter\expandafter\expandafter{%
2364       \bitsetGetOct{abc}{#2}%
2365     }{#3}%
2366   }%
2367   \Test{111 110 101 100 011 010 001 000}{0}{76543210}%
2368   \Test{000 111}{0}{7}%
2369   \Test{101 000}{-1}{50}%
2370   \Test{111}{-1}{7}%
2371   \Test{111}{0}{7}%
2372   \Test{111}{1}{7}%
2373   \Test{111}{3}{7}%
2374   \Test{111}{4}{07}%
2375   \Test{111}{6}{07}%
2376   \Test{111}{7}{007}%
2377   \Test{111 010}{6}{72}%
2378   \Test{111 010}{7}{072}%
2379   \Test{011 111}{0}{37}%
2380   \Test{011 111}{6}{37}%
2381   \Test{011 111}{7}{037}%
2382   \Test{001 111}{0}{17}%
2383   \Test{001 111}{6}{17}%
2384   \Test{001 111}{7}{017}%
2385 \end{qstest}
2386
2387 \begin{qstest}{GetHex}{GetHex}
2388   \def\Test#1#2#3{%
2389     \bitsetSetBin{abc}{#1}%
2390     \expandafter\expandafter\expandafter\Expect
2391     \expandafter\expandafter\expandafter{%
2392       \bitsetGetHex{abc}{#2}%
2393     }{#3}%
2394   }%
2395   \Test{1111 1110 1101 1100 1011 1010 1001 1000}{0}{FEDCBA98}%
2396   \Test{0111 0110 0101 0100 0011 0010 0001 0000}{0}{76543210}%
2397   \Test{0000 1111}{0}{F}%
2398   \Test{0101 0000}{-1}{50}%
2399   \Test{1111}{-1}{F}%
2400   \Test{1111}{0}{F}%
2401   \Test{1111}{1}{F}%
2402   \Test{1111}{4}{F}%
2403   \Test{1111}{5}{0F}%
2404   \Test{1111}{8}{0F}%
2405   \Test{1111}{9}{00F}%
2406   \Test{1111 0010}{8}{F2}%
2407   \Test{1111 0010}{9}{0F2}%
2408   \Test{0111 1111}{0}{7F}%
2409   \Test{0111 1111}{8}{7F}%
2410   \Test{0111 1111}{9}{07F}%
2411   \Test{0011 1111}{0}{3F}%
2412   \Test{0011 1111}{8}{3F}%
2413   \Test{0011 1111}{9}{03F}%
2414   \Test{0001 1111}{0}{1F}%
2415   \Test{0001 1111}{8}{1F}%
2416   \Test{0001 1111}{9}{01F}%

```

```

2417 \end{qstest}
2418
2419 \begin{qstest}{Range}{Range}
2420 \TestError{%
2421     Wrong index numbers in range [9..8]\MessageBreak% hash-ok
2422     for clear/set/flip on bit set 'abc'.\MessageBreak
2423     The lower index exceeds the upper index.\MessageBreak
2424     Canceling the operation as error recovery%
2425 }{%
2426     \bitsetSetRange{abc}{9}{8}%
2427 }%
2428 \def\TestErrorNegInd#1#2#3#4#5#6{%
2429     \TestError{%
2430         Negative index in range [#2..#3]\MessageBreak % hash-ok
2431         for \string\bitset #1Range on bit set 'abc'.\MessageBreak
2432         Using [#4..#5] as error recovery% hash-ok
2433     }{%
2434         \csname bitset#1Range\endcsname{abc}{#2}{#3}%
2435         \global\let\BS@global\BS@abc
2436     }%
2437     \Check{global}{#6}%
2438 }%
2439 \Set{abc}{111}%
2440 \TestErrorNegInd{Clear}{-1}{0}{0}{0}{111}%
2441 \TestErrorNegInd{Clear}{0}{-1}{0}{0}{111}%
2442 \TestErrorNegInd{Clear}{-2}{2}{0}{2}{001}%
2443 \bitsetReset{abc}%
2444 \TestErrorNegInd{Set}{-1}{0}{0}{0}{0}%
2445 \TestErrorNegInd{Set}{0}{-1}{0}{0}{0}%
2446 \TestErrorNegInd{Set}{-2}{2}{0}{2}{11}%
2447 \Set{abc}{101}%
2448 \TestErrorNegInd{Flip}{-1}{0}{0}{0}{101}%
2449 \TestErrorNegInd{Flip}{0}{-1}{0}{0}{101}%
2450 \TestErrorNegInd{Flip}{-2}{2}{0}{2}{011}%
2451 \def\Test#1#2#3#4{%
2452     \bitsetSetBin{abc}{#1}%
2453     \csname bitset\TestOp Range\endcsname{abc}{#2}{#3}%
2454     \Expect*{\bitsetGetBin{abc}{0}}{#4}%
2455 }%
2456 \def\TestOp{Clear}%
2457 \Test{0}{0}{1}{0}%
2458 \Test{1111}{1}{2}{1101}%
2459 \Test{1111}{1}{3}{1001}%
2460 \Test{1111111000000000}{12}{14}{1100111000000000}%
2461 \def\TestOp{Set}%
2462 \Test{0}{0}{1}{1}%
2463 \Test{1000}{1}{2}{1010}%
2464 \Test{0}{1}{2}{10}%
2465 \Test{1}{12}{15}{111000000000001}%
2466 \Test{1111}{1}{3}{1111}%
2467 \Test{1000000000000000}{12}{14}{1011000000000000}%
2468 \def\TestOp{Flip}%
2469 \Test{0}{0}{1}{1}%
2470 \Test{1}{0}{1}{0}%
2471 \Test{10101010}{1}{5}{10110100}%
2472 \def\Test#1#2#3#4#5{%
2473     \bitsetSetBin{abc}{#1}%
2474     \bitsetSetValueRange{abc}{#2}{#3}{#4}%
2475     \Expect*{\bitsetGetBin{abc}{0}}{#5}%
2476 }%
2477 \Test{0}{0}{1}{0}{0}%
2478 \Test{0}{0}{1}{1}{1}%

```

[illegible]


```
2541 \end{document}
2542 </test2>
```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/bitset.dtx](#) The source file.

[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`.

¹<http://ftp.ctan.org/tex-archive/>

4.4 Refresh file name databases

If your \TeX distribution ($\text{te}\text{\TeX}$, $\text{mik}\text{\TeX}$, ...) relies on file name databases, you must refresh these. For example, $\text{te}\text{\TeX}$ 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 \LaTeX . The `.dtx` chooses its action depending on the format:

plain- \TeX : Run `docstrip` and extract the files.

\LaTeX : Generate the documentation.

If you insist on using \LaTeX for `docstrip` (really, `docstrip` does not need \LaTeX), 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 $\text{pdf}\text{\LaTeX}$:

```
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		2129, 2132, 2135, 2484, 2490, 2491
<code>\#</code> 1390, 1452	<code>\@ehc</code> 156, 348, 904, 1062, 1084
<code>\%</code> 1455	<code>\@empty</code> 2359
<code>\:</code> 1724	<code>\@firstofone</code> 1399, 1402, 1612
<code>\@</code> 1391, 1448	<code>\@gobble</code> 1396, 1404, 1610
<code>\@PackageError</code>	<code>\@undefined</code> .. 53, 1475, 1476, 1477,
	.. 154, 346, 902, 1057, 1070, 1549	1731, 1859, 1994, 2027, 2034,
<code>\@Test</code> 2101,	2046, 2069, 2070, 2076, 2084,

2105, 2130, 2198, 2244, 2325, 2485
 \[..... 1453
 \\ 200, 353, 416, 549, 558,
 620, 667, 670, 705, 708, 745,
 748, 778, 780, 788, 1272, 1281,
 1303, 1312, 1372, 1375, 1449, 1608
 \{ 1388, 1450
 \} 1389, 1451
 \] 1454

 _ 1456

A
 \advance 1429, 1437, 1515
 \AtEndDocument 1528

B
 \begin 1616, 1633, 1643, 1670,
 1691, 1714, 1858, 1879, 1915,
 1934, 1958, 1984, 2026, 2033,
 2042, 2065, 2100, 2196, 2242,
 2259, 2278, 2299, 2323, 2357,
 2387, 2419, 2483, 2531, 2536, 2540
 \BigIntCalcAdd 624, 633
 \bigintcalcCmp 336
 \BigIntCalcOdd 356
 \bigintcalcSgn 333
 \BigIntCalcShl 638, 645
 \BigIntCalcShr 364
 \bitset 1072, 2431
 \BitSet@@Range 1064, 1087, 1091
 \BitSet@@@Set 962, 969, 1004
 \BitSet@@CheckIndex 147, 151
 \BitSet@@Clear 908, 920
 \BitSet@@Flip 979, 991
 \BitSet@@Get 1108, 1117
 \BitSet@@GetBin 385, 388
 \BitSet@@GetDec 553, 557, 583
 \BitSet@@GetDecBig 632, 643
 \BitSet@@GetHex 457, 490
 \BitSet@@GetOct 443, 465
 \BitSet@@GetOctHex . 440, 454, 524, 528
 \BitSet@@NextClearBit ... 1151, 1155
 \BitSet@@NextSetBit 1187, 1191
 \BitSet@@Range . 1043, 1048, 1085, 1087
 \BitSet@@Set 948, 955
 \BitSet@@TestMode 108
 \BitSet@AfterFi
 . 135, 153, 159, 203, 362, 377,
 391, 396, 407, 412, 417, 421,
 429, 434, 467, 472, 492, 497,
 530, 538, 550, 552, 1090, 1103,
 1142, 1178, 1209, 1252, 1254, 1370
 \BitSet@AfterFiFi
 136, 241, 274, 560, 565,
 569, 575, 622, 627, 631, 636,
 751, 869, 931, 962, 1004, 1128,
 1161, 1163, 1197, 1199, 1214,
 1218, 1220, 1273, 1275, 1282,
 1284, 1304, 1306, 1313, 1315, 1373
 \BitSet@AfterFiFiFi 137, 675, 679,
 715, 719, 794, 799, 934, 939,
 1007, 1012, 1225, 1231, 1376, 1378
 \BitSet@And 655, 666
 \BitSet@AndNot 693, 704
 \BitSet@AtEnd 81, 82, 1385
 \BitSet@Cardinality 1295, 1301
 \BitSet@CheckIndex
 145, 876, 879, 882, 889
 \BitSet@Cleanup
 . 869, 931, 1128, 1161, 1197, 1206
 \BitSet@Clear
 .. 876, 891, 906, 1020, 1034, 1073
 \BitSet@Empty 120, 128,
 179, 182, 184, 218, 221, 223,
 229, 325, 328, 330, 448, 462,
 466, 491, 660, 698, 771, 849,
 861, 867, 910, 914, 922, 924,
 930, 950, 960, 981, 985, 994, 1002
 \BitSet@ErrorInvalidBitValue ...
 895, 901, 1038
 \BitSet@Fi 134, 135, 136, 137, 162,
 206, 252, 283, 366, 381, 397,
 413, 425, 435, 477, 502, 539,
 555, 579, 641, 684, 724, 755,
 804, 873, 944, 967, 1017, 1094,
 1115, 1132, 1153, 1168, 1189,
 1204, 1237, 1259, 1290, 1320, 1383
 \BitSet@Fill 392, 405, 430, 532
 \BitSet@FirstOfOne 121
 \BitSet@FirstOfTwo 123, 140, 1325,
 1328, 1330, 1339, 1346, 1351, 1370
 \BitSet@Flip 882, 977, 1026
 \BitSet@FromFirstHex 212, 270
 \BitSet@FromFirstOct 209, 238
 \BitSet@FromHex 282, 285
 \BitSet@FromOct 251, 254
 \BitSet@Get 1098, 1101
 \BitSet@GetDec 544, 548
 \BitSet@GetDecBig 617, 619, 644
 \BitSet@GetOctHex 468, 493, 523
 \BitSet@GetSetBitList ... 1246, 1250
 \BitSet@Gobble
 122, 830, 855, 896, 897, 1221
 \BitSet@GobbleSeven 1228, 1242
 \BitSet@Hex[0..F] 296
 \BitSet@Hex[0000..1111] 504
 \BitSet@IfUndefined
 138, 146, 168, 389, 543,
 728, 759, 807, 834, 1107, 1263,
 1294, 1323, 1328, 1345, 1346, 1348
 \BitSet@Intersects 1361, 1368
 \BitSet@Kill 848, 858
 \BitSet@KillZeros
 182, 192, 221, 279, 328
 \BitSet@MaxSize 119, 336
 \BitSet@N1073741824 616
 \BitSet@N[1,2,4,...] 581
 \BitSet@NegativeIndex 1050, 1053, 1069
 \BitSet@NextClearBit 1136, 1139
 \BitSet@NextSetBit
 1172, 1175, 1247, 1256

\BitSet@NumBinFill	418, <u>427</u>	\bitsetGetBin	6, <u>383</u> , 2204, 2328, 2340, 2454, 2475, 2502
\BitSet@NumBinRev	399, <u>415</u>	\bitsetGetDec	7, <u>541</u> , 1880, 1888
\BitSet@Oct[000..111]	<u>479</u>	\bitsetGetHex	<u>451</u> , 2223, 2392
\BitSet@Or	<u>735</u> , <u>743</u>	\bitsetGetOct	<u>437</u> , 2210, 2364
\BitSet@Range	<u>1020</u> , <u>1023</u> , <u>1026</u> , <u>1034</u> , <u>1036</u> , <u>1041</u>	\bitsetGetSetBitList	8, <u>1243</u> , 1862, 1868
\BitSet@Reverse	188, <u>199</u> , 233	\bitsetIntersects	9, <u>1358</u> , 2067, 2091
\BitSet@SecondOfTwo	<u>124</u> , 142, 1324, 1332, 1341, 1346, 1348, 1353, 1359, 1360, 1373, 1376	\bitsetIsDefined	8, <u>1322</u> , 2028, 2030
\BitSet@Set	879, 893, <u>946</u> , 1023, 1036, 1076	\bitsetIsEmpty	9, 439, 453, 648, 651, 687, 690, 727, 730, 758, 761, 810, 837, 1147, 1183, 1245, <u>1327</u> , 1359, 1360, 2035, 2037, 2039
\BitSet@SetDec	342, 354, <u>368</u>	\bitsetLet	6, <u>167</u> , 1619, 1625
\BitSet@SetDecBig	338, <u>352</u>	\bitsetNextClearBit	8, <u>1134</u> , 1721
\BitSet@SetOctHex	209, 212, <u>214</u>	\bitsetNextSetBit	8, <u>1170</u> , 1722
\BitSet@SetValue	885, <u>888</u>	\bitsetOr	7, <u>726</u>
\BitSet@SetValueRange	1029, <u>1032</u>	\bitsetQuery	9, <u>1337</u> , 1651, 1657
\BitSet@ShiftLeft	812, <u>817</u> , 855	\bitsetReset	6, 146, <u>164</u> , 169, 649, 652, 661, 688, 699, 728, 759, 772, 808, 835, 1634, 1636, 1639, 1741, 2029, 2036, 2050, 2054, 2111, 2133, 2443
\BitSet@ShiftRight	830, 839, <u>844</u>	\bitsetSet	878, 1937, 1940, 1942, 2038, 2057, 2532
\BitSet@Size	1264, <u>1270</u>	\bitsetSetBin	6, <u>175</u> , 1725, 1751, 1761, 1812, 1823, 1834, 1845, 1865, 1996, 2004, 2005, 2089, 2090, 2156, 2157, 2245, 2337, 2360, 2389, 2452, 2473, 2499, 2500
\BitSet@Skip	1148, 1184, <u>1207</u>	\bitsetSetDec	6, <u>321</u> , 2301
\BitSet@SkipContinue	1210, 1215, 1218, 1221, <u>1239</u>	\bitsetSetHex	<u>211</u> , 2280
\BitSet@Space	<u>125</u> , 179, 218, 325, 561, 623, 825, 1114, 1143, 1179	\bitsetSetOct	<u>208</u> , 2261
\BitSet@Temp	176, 177, 179, 181, 182, 184, 188, 215, 216, 218, 220, 221, 223, 226, 227, 229, 233, 296, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 322, 323, 325, 327, 328, 330, 333, 336, 338, 342, 479, 482, 483, 484, 485, 486, 487, 488, 489, 504, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 581, 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, 613, 614, 615, 907, 914, 917, 978, 985, 988, 1042, 1046	\bitsetSetRange	<u>1022</u> , 2426
\BitSet@TestMode	108, 1482	\bitsetSetValue	8, <u>884</u> , 1987, 1995, 2006
\BitSet@Xor	766, <u>777</u>	\bitsetSetValueRange	<u>1028</u> , 2474
\BitSet@ZapSpace	<u>126</u> , 178, 217, 324	\bitsetShiftLeft	7, <u>806</u>
\BitSet@Zero	185, 224, 230, 331, 334, 915, 986, 1329, <u>1336</u>	\bitsetShiftRight	<u>833</u>
\bitsetAnd	7, <u>647</u>	\bitsetSize	8, <u>1261</u> , 1671, 1675
\bitsetAndNot	7, <u>686</u>	\bitsetXor	7, <u>757</u>
\bitsetCardinality	8, <u>1292</u> , 1692, 1697	\body	1408, 1412
\bitsetClear	7, <u>875</u> , 1918, 1921, 1923	\BS@abc	1731, 1859, 1919, 1938, 1962, 1994, 1997, 2007, 2027, 2034, 2046, 2069, 2076, 2130, 2136, 2198, 2244, 2246, 2262, 2281, 2302, 2325, 2435, 2485, 2487, 2533
\bitsetClearRange	<u>1019</u>	\BS@foo	2070, 2084, 2105, 2117
\bitsetEquals	9, <u>1344</u> , 2044	\BS@global	2435, 2533
\BitSetError	249, 265, 277, 289, 360, 1104, 1144, 1180, 1727, 1729	\BS@result	1997, 2007
\bitsetFlip	<u>881</u> , 1961, 1964, 1966, 2060		
\bitsetFlipRange	<u>1025</u>		
\bitsetGet	8, <u>1096</u> , 1338, 1646, 1656, 2537		

C

\catcode	3, 4, 5, 6, 7, 18, 19, 20, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 65, 66, 69, 70, 71, 72, 76, 77, 78, 79, 83, 85, 106, 1388, 1389, 1390, 1391, 1426, 1435, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1724
\chardef	1482
\Check	1585, 1621, 1623, 1624, 1626, 1627, 1629, 1630, 1635, 1637, 1640, 2108, 2113, 2114, 2119, 2120, 2437

<code>\CheckUndef</code>	1575, 1588, 1617, 1618, 1620, 2107	1919, 1938, 1962, 1997, 2007, 2028, 2030, 2035, 2037, 2039, 2044, 2067, 2091, 2199, 2246, 2262, 2281, 2302, 2326, 2338, 2362, 2390, 2454, 2475, 2487, 2502
<code>\Clear</code>	1721, 1726, 1732, 1742, 1752, 1762, 1813, 1824, 1835, 1846	<code>\ExpectBitSet</code> 1580, 1586, 1597
<code>\count@</code>	1393, 1422, 1426, 1428, 1429, 1433, 1435, 1436, 1437	
<code>\countdef</code>	1393	
<code>\csname</code>	8, 21, 45, 61, 68, 104, 110, 139, 165, 171, 172, 185, 187, 224, 230, 232, 276, 280, 288, 291, 297, 331, 334, 337, 341, 402, 447, 461, 475, 480, 500, 505, 545, 570, 576, 582, 616, 654, 656, 658, 660, 692, 694, 696, 698, 731, 732, 734, 736, 738, 762, 763, 765, 767, 769, 771, 823, 826, 847, 849, 909, 913, 947, 949, 980, 984, 1109, 1150, 1186, 1267, 1298, 1329, 1349, 1350, 1362, 1364, 1392, 1395, 1398, 1401, 1440, 1462, 1498, 1583, 1592, 1600, 1603, 1649, 2022, 2023, 2106, 2112, 2118, 2158, 2434, 2453, 2486, 2501	
<code>\currentgrouplevel</code> . .	1473, 1477, 1489	
D		
<code>\dimexpr</code>	1505	
<code>\documentclass</code>	1468	
E		
<code>\empty</code>	12	
<code>\end</code>	1463, 1631, 1641, 1668, 1689, 1712, 1856, 1877, 1913, 1932, 1956, 1982, 2024, 2031, 2040, 2063, 2098, 2194, 2240, 2257, 2276, 2297, 2321, 2355, 2385, 2417, 2481, 2529, 2534, 2538, 2541	
<code>\endcsname</code>	8, 21, 45, 61, 68, 104, 110, 139, 165, 171, 172, 185, 187, 224, 230, 232, 276, 280, 288, 291, 297, 331, 334, 337, 341, 402, 447, 461, 475, 480, 500, 505, 545, 571, 576, 582, 616, 654, 656, 658, 660, 692, 694, 696, 698, 731, 732, 734, 736, 738, 762, 763, 765, 767, 769, 771, 823, 826, 847, 849, 909, 913, 947, 949, 980, 984, 1109, 1150, 1186, 1267, 1298, 1329, 1349, 1350, 1362, 1364, 1392, 1395, 1398, 1401, 1440, 1462, 1498, 1583, 1592, 1600, 1603, 1649, 2022, 2023, 2106, 2112, 2118, 2158, 2434, 2453, 2486, 2501	
<code>\endinput</code>	30	
<code>\endqstest</code>	1519, 1524, 1533, 1538	
<code>\ETeXDisable</code>	1474, 1479, 1561	
<code>\ETeXEnable</code>	1486, 1491, 1554	
<code>\Expect</code>	1540, 1556, 1557, 1565, 1566, 1576, 1581, 1590, 1644, 1651, 1655, 1657, 1674, 1695, 1716, 1860, 1866, 1886,	
H		
<code>\hbox</code>	1536	
I		
<code>\ifcase</code>	9, 241, 257, 333, 356, 369, 818, 845, 890, 1033, 1055, 1213, 1658	
<code>\ifcsname</code>	1472, 1475, 1488	
<code>\ifnum</code>	152, 336, 390, 406, 428, 529, 668, 671, 673, 706, 710, 713, 744, 1049, 1052, 1088, 1102, 1140, 1176, 1224, 1251, 1338, 1369, 1428, 1436	
<code>\ifodd</code>	372	
<code>\ifx</code>	10, 12, 21, 45, 53, 56, 104, 110, 128, 139, 184, 193, 200, 223, 229, 239, 255, 271, 273, 276, 286, 288, 330, 353, 416, 466, 491, 549, 558, 559, 568, 620, 621, 630, 660, 667, 670, 698, 705, 708, 745, 748, 771, 778, 779, 780, 788, 790, 793, 860, 861, 867, 914, 921, 922, 924, 930, 933, 957, 960, 970, 985, 992, 993, 994, 1002, 1006, 1073, 1076, 1119, 1120, 1126, 1156, 1159, 1192, 1195, 1208, 1271, 1272, 1281, 1302, 1303, 1312, 1329, 1349, 1372, 1375, 1392, 1395, 1398, 1401, 1440, 1498, 1592, 1608	
<code>\ignorespaces</code>	1536	
<code>\immediate</code>	23, 47	
<code>\IncludeTests</code>	1494	
<code>\input</code>	111, 112, 113, 1441	
<code>\IntCalcAdd</code>	534, 562, 572	
<code>\intcalcCmp</code>	1055	
<code>\IntCalcDec</code>	393, 409, 469, 494, 1233	
<code>\IntCalcDiv</code>	533	
<code>\IntCalcInc</code>	423, 474, 499, 1092, 1165, 1201, 1257, 1277, 1286, 1304, 1308	
<code>\IntCalcMul</code>	526	
<code>\intcalcNum</code>	148, 386, 441, 455, 525, 813, 840, 886, 1030, 1044, 1099, 1112, 1137, 1173	
<code>\intcalcSgn</code>	818, 845	
<code>\IntCalcShr</code>	379	
<code>\IntCalcSub</code>	431, 535, 1227	
<code>\iterate</code>	1409, 1411, 1413	
L		
<code>\LoadCommand</code>	1441, 1458	
<code>\LogTests</code>	1495	
<code>\loop</code>	1407, 1423, 1434	
M		
<code>\makeatletter</code>	1469, 1481, 1496	
<code>\makeatother</code>	1483	

\MessageBreak	\StartTime 1508, 1522
. 1058, 1059, 1060, 1071, 1082,	\stepcounter 1550
1552, 2421, 2422, 2423, 2430, 2431	\StopTime 1513, 1525
	\strip@pt 1505
	\SummaryTime . . . 1500, 1502, 1515, 1529
N	
\NeedsTeXFormat 1466	
\newcommand 1503, 1508, 1512, 1513	
\newcount 1500, 1501	
\newcounter 1543	
\next 1413, 1415, 1417	
\nofiles 1467	
\number 474, 499, 525, 533,	
1097, 1135, 1149, 1171, 1185,	
1227, 1247, 1255, 1262, 1293, 1505	
\numexpr 1471, 1476, 1487	
O	
\Op 2125, 2129, 2132, 2135, 2158, 2162,	
2169, 2174, 2181, 2501, 2504, 2515	
\orig@endqstest 1533, 1541	
\orig@qstest 1532, 1535	
P	
\PackageInfo 26	
\pdfelapsedtime 1514	
\pdfresettimer 1510	
\PrintTime 1503, 1516, 1529	
\ProvidesPackage 62	
Q	
\qstest 1518, 1520, 1532, 1534	
R	
\RangeCatcodeInvalid	
. . . . 1432, 1444, 1445, 1446, 1447	
\renewcommand 1509	
\repeat 1407, 1419, 1430, 1438	
\RequirePackage 115, 116, 117	
\RestoreCatcodes 1421, 1424, 1425, 1459	
\RevCheck 1596, 1922, 1924,	
1941, 1943, 1965, 1967, 2159, 2160	
\Reverse 1597,	
1604, 1607, 1614, 1919, 1938, 1962	
\RevSet . . 1602, 1882, 1917, 1936, 1960	
\romannumeral 384,	
438, 452, 542, 825, 852, 911,	
952, 982, 1112, 1144, 1180, 1244	
S	
\saved@endqstest 1519, 1526	
\saved@qstest 1518, 1521	
\SavedCurrentgrouplevel . 1473, 1489	
\SavedIfcsname 1472, 1488	
\SavedNumexpr 1471, 1487	
\sbox 1548	
\Set . . 1599, 1622, 1628, 1638, 1650,	
1654, 1673, 1694, 1722, 1728,	
1737, 1747, 1757, 1787, 1818,	
1829, 1840, 1851, 2072, 2074,	
2078, 2080, 2082, 2086, 2439, 2447	
\setbox 1536	
\setcounter 1547	
\space 1505	
	T
	\Test 1443, 1461, 1653, 1660,
	1661, 1662, 1663, 1664, 1665,
	1666, 1667, 1672, 1677, 1678,
	1679, 1680, 1681, 1682, 1683,
	1684, 1685, 1686, 1687, 1688,
	1693, 1700, 1701, 1702, 1703,
	1704, 1705, 1706, 1707, 1708,
	1709, 1710, 1711, 1715, 1727,
	1729, 1733, 1734, 1735, 1736,
	1738, 1739, 1740, 1743, 1744,
	1745, 1746, 1748, 1749, 1750,
	1753, 1754, 1755, 1756, 1758,
	1759, 1760, 1763, 1764, 1765,
	1766, 1767, 1768, 1769, 1770,
	1771, 1772, 1773, 1774, 1775,
	1776, 1777, 1778, 1779, 1780,
	1781, 1782, 1783, 1784, 1785,
	1786, 1788, 1789, 1790, 1791,
	1792, 1793, 1794, 1795, 1796,
	1797, 1798, 1799, 1800, 1801,
	1802, 1803, 1804, 1805, 1806,
	1807, 1808, 1809, 1810, 1811,
	1814, 1815, 1816, 1817, 1819,
	1820, 1821, 1822, 1825, 1826,
	1827, 1828, 1830, 1831, 1832,
	1833, 1836, 1837, 1838, 1839,
	1841, 1842, 1843, 1844, 1847,
	1848, 1849, 1850, 1852, 1853,
	1854, 1855, 1864, 1871, 1872,
	1873, 1874, 1875, 1876, 1881,
	1891, 1892, 1893, 1894, 1895,
	1896, 1897, 1898, 1899, 1900,
	1901, 1903, 1905, 1907, 1909,
	1911, 1916, 1925, 1926, 1927,
	1928, 1929, 1930, 1931, 1935,
	1944, 1945, 1946, 1947, 1948,
	1949, 1950, 1951, 1952, 1953,
	1954, 1955, 1959, 1968, 1969,
	1970, 1971, 1972, 1973, 1974,
	1975, 1976, 1977, 1978, 1979,
	1980, 1981, 1985, 1990, 1991,
	1992, 1993, 1999, 2000, 2001,
	2002, 2003, 2009, 2010, 2011,
	2012, 2013, 2014, 2015, 2016,
	2017, 2018, 2019, 2020, 2021,
	2022, 2023, 2043, 2047, 2048,
	2049, 2051, 2052, 2053, 2055,
	2056, 2058, 2059, 2061, 2062,
	2066, 2071, 2073, 2075, 2077,
	2079, 2081, 2083, 2085, 2087,
	2088, 2093, 2094, 2095, 2096,
	2097, 2124, 2139, 2143, 2147,
	2151, 2155, 2163, 2164, 2165,
	2166, 2167, 2168, 2170, 2171,
	2172, 2173, 2175, 2176, 2177,
	2178, 2179, 2180, 2182, 2183,

2184, 2185, 2186, 2187, 2188,	2444, 2445, 2446, 2448, 2449, 2450
2191, 2243, 2248, 2249, 2250,	\TestGetterUndefined
2251, 2252, 2253, 2254, 2255,	1574, 1671, 1692, 1880
2256, 2260, 2264, 2265, 2266,	\TestOp 1718,
2267, 2268, 2269, 2270, 2271,	1721, 1722, 2453, 2456, 2461, 2468
2272, 2273, 2274, 2275, 2279,	\TestTime 1501, 1514, 1515, 1516
2283, 2284, 2285, 2286, 2287,	\TestUndef 2197, 2205, 2206, 2207,
2288, 2289, 2290, 2291, 2292,	2208, 2209, 2211, 2212, 2213,
2293, 2294, 2295, 2296, 2300,	2214, 2215, 2216, 2217, 2218,
2304, 2305, 2306, 2307, 2308,	2219, 2220, 2221, 2222, 2224,
2309, 2310, 2311, 2312, 2313,	2225, 2226, 2227, 2228, 2229,
2314, 2315, 2316, 2317, 2318,	2230, 2231, 2232, 2233, 2234,
2319, 2320, 2336, 2343, 2344,	2235, 2236, 2237, 2238, 2239,
2345, 2346, 2347, 2348, 2349,	2324, 2331, 2332, 2333, 2334, 2335
2350, 2351, 2352, 2353, 2354,	\the . 69, 70, 71, 72, 83, 1426, 1540, 1566
2358, 2367, 2368, 2369, 2370,	\theTest 1565
2371, 2372, 2373, 2374, 2375,	\TimeDescription . . . 1509, 1512, 1516
2376, 2377, 2378, 2379, 2380,	\TMP@EnsureCode 80,
2381, 2382, 2383, 2384, 2388,	87, 88, 89, 90, 91, 92, 93, 94,
2395, 2396, 2397, 2398, 2399,	95, 96, 97, 98, 99, 100, 101, 102
2400, 2401, 2402, 2403, 2404,	\typeout 1504
2405, 2406, 2407, 2408, 2409,	
2410, 2411, 2412, 2413, 2414,	U
2415, 2416, 2451, 2457, 2458,	\uccode 821
2459, 2460, 2462, 2463, 2464,	\uppercase 822
2465, 2466, 2467, 2469, 2470,	\usepackage 1484, 1493
2471, 2472, 2477, 2478, 2479,	
2480, 2489, 2493, 2494, 2495,	W
2496, 2497, 2498, 2505, 2506,	\wd 1540, 1566
2507, 2508, 2509, 2510, 2511,	\write 23, 47
2512, 2513, 2514, 2516, 2517,	
2518, 2519, 2520, 2521, 2522,	X
2523, 2524, 2525, 2526, 2527, 2528	\x 8, 10, 12, 22, 26, 28,
\Test@ 2126, 2128	46, 51, 61, 67, 75, 2201, 2204,
\TestError 1545, 1571, 1986, 2420, 2429	2210, 2223, 2359, 2360, 2361, 2537
\TestErrorNegativeIndex	
1570, 1923, 1942, 1966	Z
\TestErrorNegInd	\z@ 1502
2428, 2440, 2441, 2442,	\zap@space 2359