

# 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

<b>1</b>	<b>Documentation</b>	<b>3</b>
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
<b>2</b>	<b>Implementation</b>	<b>9</b>
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 . . . . .	12
2.6	Import . . . . .	13
2.6.1	From binary number . . . . .	13
2.6.2	From octal/hex number . . . . .	13
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>	24
2.8.5	Shifting	25
2.8.6	<code>\bitsetShiftLeft</code>	25
2.8.7	<code>\bitsetShiftRight</code>	26
2.9	Bit manipulation	27
2.9.1	Clear operation	28
2.9.2	Set operation	28
2.9.3	Flip operation	29
2.9.4	Range operators	30
2.10	Bit retrieval	31
2.10.1	<code>\bitsetGet</code>	31
2.10.2	<code>\bitsetNextClearBit</code> , <code>\bitsetNextSetBit</code>	32
2.10.3	<code>\bitsetGetSetBitList</code>	34
2.11	Bit set properties	35
2.12	Queries	36
<b>3</b>	<b>Test</b>	<b>37</b>
3.1	Catcode checks for loading	37
3.2	Macro tests	38
3.2.1	Preamble	38
3.2.2	Time	39
3.2.3	Detection of unwanted space	39
3.2.4	Test macros	40
3.2.5	Test sets	41
<b>4</b>	<b>Installation</b>	<b>56</b>
4.1	Download	56
4.2	Bundle installation	56
4.3	Package installation	56
4.4	Refresh file name databases	57
4.5	Some details for the interested	57
<b>5</b>	<b>History</b>	<b>57</b>
	[2007/09/28 v1.0]	57
<b>6</b>	<b>Index</b>	<b>58</b>

# 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  $\varepsilon$ -`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<sup>a</sup> (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<sup>a</sup> (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<sup>b</sup> (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>	$\langle BitSet \rangle \langle Index \rangle \langle Then \rangle \langle Else \rangle$

---

<sup>a</sup>Macros are expandable, full expansion by two steps.

<sup>b</sup>Macros are expandable.

## 1.5 Package loading

The package can be used as normal L<sup>A</sup>T<sub>E</sub>X package:

```
\usepackage{bitset}
```

Also plain-T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X.

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

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

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

### 1.6.1 Miscellaneous

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

`\bitsetReset {\langle BitSet \rangle}`

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

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

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

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

### 1.6.2 Import

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

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

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

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

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

### 1.6.3 Export

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

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

```

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

```

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

`\bitsetGetDec {⟨BitSet⟩}`

Macro `\bitsetGetDec` returns the bit set  $\langle BitSet \rangle$  as decimal number. The returned number can be larger than T<sub>E</sub>X's number limit of  $2^{31} - 1$ .

#### 1.6.4 Logical operators

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

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

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

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

`\bitsetOr {⟨BitSet A⟩} {⟨BitSet B⟩}`

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

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

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

#### 1.6.5 Shifting

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

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

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

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

#### 1.6.6 Bit manipulation

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

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

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

Macro `\bitsetSetValue` puts bit `⟨Bit⟩` at position `⟨Index⟩` in bit set `⟨BitSet⟩`. `⟨Bit⟩` must be a valid T<sub>E</sub>X 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`, ...,  $(\infty)$

`\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`, ...,  $(\infty)$

`\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<sup>A</sup>T<sub>E</sub>X.

```
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 \expandafter\ifx\csname PackageInfo\endcsname\relax
19 \def\x#1#2{%
20 \immediate\write-1{Package #1 Info: #2.}%
21 }%
22 \else
23 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
24 \fi
```

```

25 \x{bitset}{The package is already loaded}%
26 \endgroup
27 \expandafter\endinput
28 \fi
29 \endgroup

```

Package identification:

```

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

```

## 2.2 Catcodes

```

55 \expandafter\edef\csname BitSet@AtEnd\endcsname{%
56 \catcode64 \the\catcode64\relax
57 }
58 \catcode64 11 % @
59 \def\TMP@EnsureCode#1#2{%
60 \edef\BitSet@AtEnd{%
61 \BitSet@AtEnd
62 \catcode#1 \the\catcode#1\relax
63 }%
64 \catcode#1 #2\relax
65 }
66 \TMP@EnsureCode{33}{12}% !
67 \TMP@EnsureCode{39}{12}% '
68 \TMP@EnsureCode{40}{12}% (
69 \TMP@EnsureCode{41}{12}% )
70 \TMP@EnsureCode{42}{12}% *
71 \TMP@EnsureCode{43}{12}% +
72 \TMP@EnsureCode{44}{12}% ,
73 \TMP@EnsureCode{45}{12}% -
74 \TMP@EnsureCode{46}{12}% .
75 \TMP@EnsureCode{47}{12}% /
76 \TMP@EnsureCode{58}{11}% : (letter!)
77 \TMP@EnsureCode{60}{12}% <
78 \TMP@EnsureCode{61}{12}% =
79 \TMP@EnsureCode{62}{12}% >
80 \TMP@EnsureCode{63}{14}% ? (comment!)
81 \TMP@EnsureCode{96}{12}% `
82 \begingroup\expandafter\expandafter\expandafter\endgroup

```

```

83 \expandafter\ifx\csname BitSet@TestMode\endcsname\relax
84 \else
85   \catcode63=9 % ? (ignore)
86 \fi
87 ? \let\BitSet@@TestMode\BitSet@TestMode

```

## 2.3 Package loading

```

88 \begingroup\expandafter\expandafter\expandafter\endgroup
89 \expandafter\ifx\csname RequirePackage\endcsname\relax
90   \input infwarerr.sty\relax
91   \input intcalc.sty\relax
92   \input bigintcalc.sty\relax
93 \else
94   \RequirePackage{infwarerr}[2007/09/09]%
95   \RequirePackage{intcalc}[2007/09/27]%
96   \RequirePackage{bigintcalc}[2007/09/27]%
97 \fi

```

## 2.4 Help macros

### 2.4.1 Number constant

```

\BitSet@MaxSize

98 \def\BitSet@MaxSize{2147483647}%

```

### 2.4.2 General basic macros

```

\BitSet@Empty

99 \def\BitSet@Empty{}

\BitSet@FirstOfOne

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

\BitSet@Gobble

101 \def\BitSet@Gobble#1{}

\BitSet@FirstOfTwo

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

\BitSet@SecondOfTwo

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

\BitSet@Space

104 \def\BitSet@Space{ }

\BitSet@ZapSpace

105 \def\BitSet@ZapSpace#1 #2{%
106   #1%
107   \ifx\BitSet@Empty#2%
108   \else
109     \expandafter\BitSet@ZapSpace
110   \fi
111   #2%
112 }

```

### 2.4.3 Tail recursion

```

\BitSet@Fi
113 \let\BitSet@Fi\fi

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

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

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

```

### 2.4.4 Check macros

```

\BitSet@IfUndefined
117 \def\BitSet@IfUndefined#1{%
118   \expandafter\ifx\csname BS@#1\endcsname\relax
119     \expandafter\BitSet@FirstOfTwo
120   \else
121     \expandafter\BitSet@SecondOfTwo
122   \fi
123 }

\BitSet@CheckIndex #1: continuation code
#2: BitSet
#3: Index
124 \def\BitSet@CheckIndex#1#2#3{%
125   \BitSet@IfUndefined{#2}{\bitsetReset{#2}}{}%
126   \expandafter\expandafter\expandafter\BitSet@@CheckIndex
127   \intcalcNum{#3}!%
128   {#2}{#1}%
129 }

\BitSet@@CheckIndex #1: plain Index
#2: BitSet
#3: continuation code
130 \def\BitSet@@CheckIndex#1!#2#3{%
131   \ifnum#1<0 %
132     \BitSet@AfterFi{%
133       \@PackageError{bitset}{%
134         Invalid negative index (#1)%
135       }\@ehc
136     }%
137   \else
138     \BitSet@AfterFi{%
139       #3{#2}{#1}%
140     }%
141     \BitSet@Fi
142   }

```

## 2.5 Miscellaneous

```

\bitsetReset
143 \def\bitsetReset#1{%
144   \expandafter\def\csname BS@#1\endcsname{0}%
145 }

```

\bitsetLet

```
146 \def\bitsetLet#1#2{%
147   \BitSet@IfUndefined{#2}{%
148     \bitsetReset{#1}%
149   }{%
150     \expandafter\let\csname BS@#1\expandafter\endcsname
151       \csname BS@#2\endcsname
152   }%
153 }
```

## 2.6 Import

### 2.6.1 From binary number

\bitsetSetBin

```
154 \def\bitsetSetBin#1#2{%
155   \edef\BitSet@Temp{#2}%
156   \edef\BitSet@Temp{%
157     \expandafter\expandafter\expandafter\BitSet@ZapSpace
158     \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
159   }%
160   \edef\BitSet@Temp{%
161     \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
162   }%
163   \ifx\BitSet@Temp\BitSet@Empty
164     \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
165   \else
166     \expandafter\edef\csname BS@#1\endcsname{%
167       \expandafter\BitSet@Reverse\BitSet@Temp!%
168     }%
169   \fi
170 }
```

\BitSet@KillZeros

```
171 \def\BitSet@KillZeros#1{%
172   \ifx#10%
173     \expandafter\BitSet@KillZeros
174   \else
175     #1%
176   \fi
177 }
```

\BitSet@Reverse

```
178 \def\BitSet@Reverse#1#2!{%
179   \ifx\#2\%
180     #1%
181   \else
182     \BitSet@AfterFi{%
183       \BitSet@Reverse#2!#1%
184     }%
185   \BitSet@Fi
186 }
```

### 2.6.2 From octal/hex number

\bitsetSetOct

```
187 \def\bitsetSetOct{%
188   \BitSet@SetOctHex\BitSet@FromFirstOct
189 }
```

\bitsetSetHex

```
190 \def\bitsetSetHex{%
191   \BitSet@SetOctHex\BitSet@FromFirstHex
192 }
```

\BitSet@SetOctHex

```
193 \def\BitSet@SetOctHex#1#2#3{%
194   \edef\BitSet@Temp{#3}%
195   \edef\BitSet@Temp{%
196     \expandafter\expandafter\expandafter\BitSet@ZapSpace
197     \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
198   }%
199   \edef\BitSet@Temp{%
200     \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
201   }%
202   \ifx\BitSet@Temp\BitSet@Empty
203     \expandafter\let\csname BS@#2\endcsname\BitSet@Zero
204   \else
205     \edef\BitSet@Temp{%
206       \expandafter#1\BitSet@Temp!%
207     }%
208     \ifx\BitSet@Temp\BitSet@Empty
209       \expandafter\let\csname BS@#2\endcsname\BitSet@Zero
210     \else
211       \expandafter\edef\csname BS@#2\endcsname{%
212         \expandafter\BitSet@Reverse\BitSet@Temp!%
213       }%
214     \fi
215   \fi
216 }
```

\BitSet@FromFirstOct

```
217 \def\BitSet@FromFirstOct#1{%
218   \ifx#1!%
219   \else
220     \ifcase#1 \BitSet@AfterFiFi\BitSet@FromFirstOct
221     \or 1%
222     \or 10%
223     \or 11%
224     \or 100%
225     \or 101%
226     \or 110%
227     \or 111%
228     \else \BitSetError:WrongOctalDigit%
229     \fi
230     \expandafter\BitSet@FromOct
231   \BitSet@Fi
232 }
```

\BitSet@FromOct

```
233 \def\BitSet@FromOct#1{%
234   \ifx#1!%
235   \else
236     \ifcase#1 000%
237     \or 001%
238     \or 010%
239     \or 011%
240     \or 100%
241     \or 101%
242     \or 110%
243     \or 111%
244     \else \BitSetError:WrongOctalDigit%
```

```

245     \fi
246     \expandafter\BitSet@FromOct
247 \fi
248 }

\BitSet@FromFirstHex

249 \def\BitSet@FromFirstHex#1{%
250   \ifx#1!%
251   \else
252     \ifx#10%
253       \BitSet@AfterFiFi\BitSet@FromFirstHex
254     \fi
255     \expandafter\ifx\csname BitSet@Hex#1\endcsname\relax
256       \BitSetError:InvalidHexDigit%
257     \else
258       \expandafter\expandafter\expandafter\BitSet@KillZeros
259       \csname BitSet@Hex#1\endcsname
260     \fi
261     \expandafter\BitSet@FromHex
262   \BitSet@Fi
263 }

\BitSet@FromHex

264 \def\BitSet@FromHex#1{%
265   \ifx#1!%
266   \else
267     \expandafter\ifx\csname BitSet@Hex#1\endcsname\relax
268       \BitSetError:InvalidHexDigit%
269     \else
270       \csname BitSet@Hex#1\endcsname
271     \fi
272     \expandafter\BitSet@FromHex
273   \fi
274 }

\BitSet@Hex[0..F]

275 \def\BitSet@Temp#1{%
276   \expandafter\def\csname BitSet@Hex#1\endcsname
277 }
278 \BitSet@Temp 0{0000}%
279 \BitSet@Temp 1{0001}%
280 \BitSet@Temp 2{0010}%
281 \BitSet@Temp 3{0011}%
282 \BitSet@Temp 4{0100}%
283 \BitSet@Temp 5{0101}%
284 \BitSet@Temp 6{0110}%
285 \BitSet@Temp 7{0111}%
286 \BitSet@Temp 8{1000}%
287 \BitSet@Temp 9{1001}%
288 \BitSet@Temp A{1010}%
289 \BitSet@Temp B{1011}%
290 \BitSet@Temp C{1100}%
291 \BitSet@Temp D{1101}%
292 \BitSet@Temp E{1110}%
293 \BitSet@Temp F{1111}%
294 \BitSet@Temp a{1010}%
295 \BitSet@Temp b{1011}%
296 \BitSet@Temp c{1100}%
297 \BitSet@Temp d{1101}%
298 \BitSet@Temp e{1110}%
299 \BitSet@Temp f{1111}%

```

### 2.6.3 From decimal number

\bitsetSetDec

```

300 \def\bitsetSetDec#1#2{%
301   \edef\BitSet@Temp{#2}%
302   \edef\BitSet@Temp{%
303     \expandafter\expandafter\expandafter\BitSet@ZapSpace
304     \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
305   }%
306   \edef\BitSet@Temp{%
307     \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
308   }%
309   \ifx\BitSet@Temp\BitSet@Empty
310     \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
311   \else
312     \ifcase\bigintcalcSgn{\BitSet@Temp} %
313       \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
314     \or
315       \ifnum\bigintcalcCmp\BitSet@Temp\BitSet@MaxSize>0 %
316         \expandafter\edef\csname BS@#1\endcsname{%
317           \expandafter\BitSet@SetDecBig\BitSet@Temp!%
318         }%
319       \else
320         \expandafter\edef\csname BS@#1\endcsname{%
321           \expandafter\BitSet@SetDec\BitSet@Temp!%
322         }%
323     \fi
324   \else
325     \@PackageError{bitset}{%
326       Bit sets cannot be negative%
327     }\@ehc
328   \fi
329 \fi
330 }
```

\BitSet@SetDecBig

```

331 \def\BitSet@SetDecBig#1#2#3#4#5#6#7#8#9!{%
332   \ifx\#9\%
333     \BitSet@SetDec#1#2#3#4#5#6#7#8!%
334   \else
335     \ifcase\BigIntCalcOdd#1#2#4#5#6#7#8#9! %
336       0%
337     \or
338       1%
339   ? \else\BitSetError:ThisCannotHappen%
340   \fi
341   \BitSet@AfterFi{%
342     \expandafter\expandafter\expandafter\BitSet@SetDecBig
343     \BigIntCalcShr#1#2#3#4#5#6#7#8#9!!%
344   }%
345   \BitSet@Fi
346 }
```

\BitSet@SetDec

```

347 \def\BitSet@SetDec#1!{%
348   \ifcase#1 %
349     \or 1%
350   \else
351     \ifodd#1 %
352       1%
353     \else
354       0%
```



```

355     \fi
356     \BitSet@AfterFi{%
357         \expandafter\expandafter\expandafter\BitSet@SetDec
358         \IntCalcShr#1!!%
359     }%
360     \BitSet@Fi
361 }

```

## 2.7 Export

### 2.7.1 To binary number

\bitsetGetBin

```

362 \def\bitsetGetBin#1#2{%
363     \romannumeral0%
364     \expandafter\expandafter\expandafter\BitSet@@GetBin
365     \intcalcNum{#2}\{#1}%
366 }

```

\BitSet@@GetBin

```

367 \def\BitSet@@GetBin#1!#2{%
368     \BitSet@IfUndefined{#2}{%
369         \ifnum#1>1 %
370             \BitSet@AfterFi{%
371                 \expandafter\expandafter\expandafter\BitSet@Fill
372                 \IntCalcDec#1!!0%
373             }%
374         \else
375             \BitSet@AfterFi{ 0}%
376             \BitSet@Fi
377         }{%
378             \expandafter\expandafter\expandafter\BitSet@NumBinRev
379             \expandafter\expandafter\expandafter1%
380             \expandafter\expandafter\expandafter!%
381             \csname BS@#2\endcsname!!#1!%
382         }%
383 }

```

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

#2: result

```

384 \def\BitSet@Fill#1!{%
385     \ifnum#1>0 %
386         \BitSet@AfterFi{%
387             \expandafter\expandafter\expandafter\BitSet@Fill
388             \IntCalcDec#1!!0%
389         }%
390     \else
391         \BitSet@AfterFi{ }%
392     \BitSet@Fi
393 }

```

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

#2#3: reverted number

#4: result

#5: min size

```

394 \def\BitSet@NumBinRev#1!#2#3!{%
395     \ifx\#3\%
396         \BitSet@AfterFi{%
397             \BitSet@NumBinFill#1!#2%
398         }%
399     \else
400         \BitSet@AfterFi{%

```

```

401     \expandafter\expandafter\expandafter\BitSet@NumBinRev
402     \IntCalcInc#1!!#3!#2%
403 }%
404 \BitSet@Fi
405 }

```

\BitSet@NumBinFill

```

406 \def\BitSet@NumBinFill#1!#2!#3!{%
407   \ifnum#3>#1 %
408     \BitSet@AfterFi{%
409       \expandafter\expandafter\expandafter\BitSet@Fill
410       \IntCalcSub#3!#1!!#2%
411     }%
412   \else
413     \BitSet@AfterFi{ #2}%
414   \BitSet@Fi
415 }

```

## 2.7.2 To octal/hexadecimal number

\bitsetGetOct

```

416 \def\bitsetGetOct#1#2{%
417   \romannumeral0%
418   \bitsetIsEmpty{#1}{%
419     \expandafter\expandafter\expandafter\BitSet@@GetOctHex
420     \intcalcNum{#2}!3!230%
421   }{%
422     \expandafter\expandafter\expandafter\BitSet@@GetOct
423     \expandafter\expandafter\expandafter1%
424     \expandafter\expandafter\expandafter!%
425     \expandafter\expandafter\expandafter!%
426     \csname BS@#1\endcsname00%
427     \BitSet@Empty\BitSet@Empty\BitSet@Empty!{#2}%
428   }%
429 }

```

\bitsetGetHex

```

430 \def\bitsetGetHex#1#2{%
431   \romannumeral0%
432   \bitsetIsEmpty{#1}{%
433     \expandafter\expandafter\expandafter\BitSet@@GetOctHex
434     \intcalcNum{#2}!4!340%
435   }{%
436     \expandafter\expandafter\expandafter\BitSet@@GetHex
437     \expandafter\expandafter\expandafter1%
438     \expandafter\expandafter\expandafter!%
439     \expandafter\expandafter\expandafter!%
440     \csname BS@#1\endcsname000%
441     \BitSet@Empty\BitSet@Empty\BitSet@Empty\BitSet@Empty!{#2}%
442   }%
443 }

```

\BitSet@@GetOct #1: number of digits

#2: result

#3#4#5: bits

```

444 \def\BitSet@@GetOct#1!#2!#3#4#5{%
445   \ifx#5\BitSet@Empty
446     \BitSet@AfterFi{%
447       \expandafter\expandafter\expandafter\BitSet@GetOctHex
448       \IntCalcDec#1!!#2!23%
449     }%

```

```

450 \else
451 \BitSet@AfterFi{%
452 \expandafter\expandafter\expandafter\BitSet@@GetOct
453 \number\IntCalcInc#1!\expandafter\expandafter\expandafter!%
454 \csname BitSet@Oct#5#4#3\endcsname#2!%
455 }%
456 \BitSet@Fi
457 }

```

\BitSet@Oct[000..111]

```

458 \def\BitSet@Temp#1#2#3#4{%
459 \expandafter\def\csname BitSet@Oct#1#2#3\endcsname{#4}%
460 }
461 \BitSet@Temp0000%
462 \BitSet@Temp0011%
463 \BitSet@Temp0102%
464 \BitSet@Temp0113%
465 \BitSet@Temp1004%
466 \BitSet@Temp1015%
467 \BitSet@Temp1106%
468 \BitSet@Temp1117%

```

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

```

469 \def\BitSet@@GetHex#1!#2!#3#4#5#6{%
470 \ifx#6\BitSet@Empty
471 \BitSet@AfterFi{%
472 \expandafter\expandafter\expandafter\BitSet@@GetOctHex
473 \IntCalcDec#1!#2!34%
474 }%
475 \else
476 \BitSet@AfterFi{%
477 \expandafter\expandafter\expandafter\BitSet@@GetHex
478 \number\IntCalcInc#1!\expandafter\expandafter\expandafter!%
479 \csname BitSet@Hex#6#5#4#3\endcsname#2!%
480 }%
481 \BitSet@Fi
482 }

```

\BitSet@Hex[0000..1111]

```

483 \def\BitSet@Temp#1#2#3#4#5{%
484 \expandafter\def\csname BitSet@Hex#1#2#3#4\endcsname{#5}%
485 }
486 \BitSet@Temp00000%
487 \BitSet@Temp00011%
488 \BitSet@Temp00102%
489 \BitSet@Temp00113%
490 \BitSet@Temp01004%
491 \BitSet@Temp01015%
492 \BitSet@Temp01106%
493 \BitSet@Temp01117%
494 \BitSet@Temp10008%
495 \BitSet@Temp10019%
496 \BitSet@Temp1010A%
497 \BitSet@Temp1011B%
498 \BitSet@Temp1100C%
499 \BitSet@Temp1101D%
500 \BitSet@Temp1110E%
501 \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
502 \def\BitSet@GetOctHex#1!#2!#3#4#5!#6{%
503   \expandafter\BitSet@@GetOctHex
504   \number\intcalcNum{#6}\expandafter\expandafter\expandafter!%
505   \IntCalcMul#1!#4!!#3#4#2%
506 }

```

```

\BitSet@@GetOctHex #1: plain min size
#2: digits * (bits per digit)
#3: bits per digit - 1
#4: bits per digit
507 \def\BitSet@@GetOctHex#1!#2!#3#4{%
508   \ifnum#1>#2 %
509     \BitSet@AfterFi{%
510       \expandafter\expandafter\expandafter\expandafter
511       \expandafter\expandafter\expandafter\BitSet@Fill
512       \expandafter\IntCalcDiv\number
513       \expandafter\expandafter\expandafter\IntCalcAdd
514       \IntCalcSub#1!#2!!#3!!#4!!%
515     }%
516   \else
517     \BitSet@AfterFi{ }%
518   \BitSet@Fi
519 }

```

### 2.7.3 To decimal number

\bitsetGetDec

```

520 \def\bitsetGetDec#1{%
521   \romannumeral0%
522   \BitSet@IfUndefined{#1}{ 0}{%
523     \expandafter\expandafter\expandafter\BitSet@GetDec
524     \csname BS@#1\endcsname!%
525   }%
526 }

```

\BitSet@GetDec

```

527 \def\BitSet@GetDec#1#2!{%
528   \ifx\#2\%
529     \BitSet@AfterFi{ #1}%
530   \else
531     \BitSet@AfterFi{%
532       \BitSet@@GetDec2!#1!#2!%
533     }%
534   \BitSet@Fi
535 }

```

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

```

536 \def\BitSet@@GetDec#1!#2!#3#4!{%
537   \ifx\#4\%
538     \ifx#31%
539       \BitSet@AfterFiFi{%
540         \expandafter\expandafter\expandafter\BitSet@Space
541         \IntCalcAdd#1!#2!%
542       }%
543     \else

```

```

544     \BitSet@AfterFiFi{ #2}%
545     \fi
546   \else
547     \ifx#31%
548       \BitSet@AfterFiFi{%
549         \csname BitSet@N#1%
550           \expandafter\expandafter\expandafter\endcsname
551           \IntCalcAdd#1!#2!!#4!%
552       }%
553     \else
554       \BitSet@AfterFiFi{%
555         \csname BitSet@N#1\endcsname#2!#4!%
556       }%
557     \fi
558   \BitSet@Fi
559 }

```

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

```

560 \def\BitSet@Temp#1#2{%
561   \expandafter\def\csname BitSet@N#1\endcsname{%
562     \BitSet@@GetDec#2!%
563   }%
564 }
565 \BitSet@Temp{1}{2}
566 \BitSet@Temp{2}{4}
567 \BitSet@Temp{4}{8}
568 \BitSet@Temp{8}{16}
569 \BitSet@Temp{16}{32}
570 \BitSet@Temp{32}{64}
571 \BitSet@Temp{64}{128}
572 \BitSet@Temp{128}{256}
573 \BitSet@Temp{256}{512}
574 \BitSet@Temp{512}{1024}
575 \BitSet@Temp{1024}{2048}
576 \BitSet@Temp{2048}{4096}
577 \BitSet@Temp{4096}{8192}
578 \BitSet@Temp{8192}{16384}
579 \BitSet@Temp{16384}{32768}
580 \BitSet@Temp{32768}{65536}
581 \BitSet@Temp{65536}{131072}
582 \BitSet@Temp{131072}{262144}
583 \BitSet@Temp{262144}{524288}
584 \BitSet@Temp{524288}{1048576}
585 \BitSet@Temp{1048576}{2097152}
586 \BitSet@Temp{2097152}{4194304}
587 \BitSet@Temp{4194304}{8388608}
588 \BitSet@Temp{8388608}{16777216}
589 \BitSet@Temp{16777216}{33554432}
590 \BitSet@Temp{33554432}{67108864}
591 \BitSet@Temp{67108864}{134217728}
592 \BitSet@Temp{134217728}{268435456}
593 \BitSet@Temp{268435456}{536870912}
594 \BitSet@Temp{536870912}{1073741824}

```

\BitSet@N1073741824

```

595 \expandafter\def\csname BitSet@N1073741824\endcsname{%
596   \BitSet@GetDecBig2147483648!%
597 }%

```

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

```

598 \def\BitSet@GetDecBig#1!#2!#3#4!{%
599   \ifx\#4\%
600     \ifx#31%
601       \BitSet@AfterFiFi{%
602         \expandafter\expandafter\expandafter\BitSet@Space
603         \BigIntCalcAdd#1!#2!%
604       }%
605     \else
606       \BitSet@AfterFiFi{ #2}%
607     \fi
608   \else
609     \ifx#31%
610       \BitSet@AfterFiFi{%
611         \expandafter\expandafter\expandafter\BitSet@@GetDecBig
612         \BigIntCalcAdd#1!#2!!#1!#4!%
613       }%
614     \else
615       \BitSet@AfterFiFi{%
616         \expandafter\expandafter\expandafter\BitSet@GetDecBig
617         \BigIntCalcShl#1!!#2!#4!%
618       }%
619     \fi
620   \BitSet@Fi
621 }

```

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

```

622 \def\BitSet@@GetDecBig#1!#2!{%
623   \expandafter\expandafter\expandafter\BitSet@GetDecBig
624   \BigIntCalcShl#2!!#1!%
625 }

```

## 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

```

626 \def\bitsetAnd#1#2{%
627   \bitsetIsEmpty{#1}{%
628     \bitsetReset{#1}%
629   }{%
630     \bitsetIsEmpty{#2}{%
631       \bitsetReset{#1}%
632     }{%
633       \expandafter\edef\csname BS@#1\endcsname{%
634         \expandafter\expandafter\expandafter\BitSet@And
635         \csname BS@#1\expandafter\expandafter\expandafter\endcsname
636         \expandafter\expandafter\expandafter!%
637         \csname BS@#2\endcsname!%!%
638       }%
639       \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
640         \bitsetReset{#1}%
641       \fi
642     }%
643   }%
644 }

```

\BitSet@And

```

645 \def\BitSet@And#1#2!#3#4!#5!{%
646   \ifx\#2\%
647     \ifnum#1#3=11 #51\fi
648   \else
649     \ifx\#4\%
650       \ifnum#1#3=11 #51\fi
651     \else
652       \ifnum#1#3=11 %
653         #51%
654         \BitSet@AfterFiFiFi{%
655           \BitSet@And#2!#4!%!%
656         }%
657     \else
658       \BitSet@AfterFiFiFi{%
659         \BitSet@And#2!#4!#50!%
660       }%
661     \fi
662   \fi
663   \BitSet@Fi
664 }

```

## 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

```

665 \def\bitsetAndNot#1#2{%
666   \bitsetIsEmpty{#1}{%
667     \bitsetReset{#1}%
668   }{%
669     \bitsetIsEmpty{#2}{%
670     }{%
671       \expandafter\edef\csname BS@#1\endcsname{%
672         \expandafter\expandafter\expandafter\BitSet@AndNot
673         \csname BS@#1\expandafter\expandafter\expandafter\endcsname
674         \expandafter\expandafter\expandafter!%
675         \csname BS@#2\endcsname!%!%
676       }%
677       \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
678         \bitsetReset{#1}%
679       \fi
680     }%
681   }%
682 }

```

\BitSet@AndNot

```

683 \def\BitSet@AndNot#1#2!#3#4!#5!{%
684   \ifx\#2\%
685     \ifnum#1#3=10 #51\fi
686   \else
687     \ifx\#4\%
688       #5%
689       \ifnum#1#3=10 1\else 0\fi
690     #2%
691   \else
692     \ifnum#1#3=10 %
693     #51%

```

```

694      \BitSet@AfterFiFiFi{%
695      \BitSet@AndNot#2!#4!%!%
696    }%
697    \else
698      \BitSet@AfterFiFiFi{%
699      \BitSet@AndNot#2!#4!#50!%
700    }%
701    \fi
702  \fi
703  \BitSet@Fi
704 }

```

### 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

```

705 \def\bitsetOr#1#2{%
706   \bitsetIsEmpty{#2}{%
707     \BitSet@IfUndefined{#1}{\bitsetReset{#1}}{%
708   }{%
709     \bitsetIsEmpty{#1}{%
710       \expandafter\let\csname BS@#1\expandafter\endcsname
711       \csname BS@#2\endcsname
712     }{%
713       \expandafter\edef\csname BS@#1\endcsname{%
714         \expandafter\expandafter\expandafter\BitSet@Or
715         \csname BS@#1\expandafter\expandafter\expandafter\endcsname
716         \expandafter\expandafter\expandafter!%
717         \csname BS@#2\endcsname!%
718       }%
719     }%
720   }%
721 }

```

\BitSet@Or

```

722 \def\BitSet@Or#1#2!#3#4!{%
723   \ifnum#1#3>0 1\else 0\fi
724   \ifx\#2\%
725     #4%
726   \else
727     \ifx\#4\%
728       #2%
729     \else
730       \BitSet@AfterFiFi{%
731       \BitSet@Or#2!#4!%
732     }%
733     \fi
734   \BitSet@Fi
735 }

```

### 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 ^= B



```

736 \def\bitsetXor#1#2{%
737   \bitsetIsEmpty{#2}{%
738     \BitSet@IfUndefined{#1}{\bitsetReset{#1}}{}%
739   }{%
740     \bitsetIsEmpty{#1}{%
741       \expandafter\let\csname BS@#1\expandafter\endcsname
742         \csname BS@#2\endcsname
743     }{%
744       \expandafter\edef\csname BS@#1\endcsname{%
745         \expandafter\expandafter\expandafter\BitSet@Xor
746         \csname BS@#1\expandafter\expandafter\expandafter\endcsname
747         \expandafter\expandafter\expandafter!%
748         \csname BS@#2\endcsname!!%
749       }%
750       \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
751         \bitsetReset{#1}%
752       \fi
753     }%
754   }%
755 }

```

\BitSet@Xor

```

756 \def\BitSet@Xor#1#2!#3#4!#5!{%
757   \ifx\#2\%
758     \ifx#1#3%
759       \ifx\#4\%
760         \else
761           #50#4%
762         \fi
763       \else
764         #51#4%
765       \fi
766     \else
767       \ifx\#4\%
768         #5%
769         \ifx#1#30\else 1\fi
770         #2%
771       \else
772         \ifx#1#3%
773           \BitSet@AfterFiFiFi{%
774             \BitSet@Xor#2!#4!#50!%
775           }%
776         \else
777           #51%
778           \BitSet@AfterFiFiFi{%
779             \BitSet@Xor#2!#4!%
780           }%
781         \fi
782       \fi
783     \BitSet@Fi
784 }

```

## 2.8.5 Shifting

### 2.8.6 \bitsetShiftLeft

\bitsetShiftLeft

```

785 \def\bitsetShiftLeft#1#2{%
786   \BitSet@IfUndefined{#1}{%
787     \bitsetReset{#1}%
788   }{%
789     \bitsetIsEmpty{#1}{%

```

```

790   }{%
791     \expandafter\expandafter\expandafter\BitSet@ShiftLeft
792     \intcalcsNum{#2}!{#1}%
793   }%
794 }%
795 }

```

\BitSet@ShiftLeft

```

796 \def\BitSet@ShiftLeft#1!#2{%
797   \ifcase\intcalcsGn{#1} %
798   \or
799     \begingroup
800     \uccode'm='0 %
801     \uppercase\expandafter{\expandafter\endgroup
802     \expandafter\edef\csname BS@#2\expandafter\endcsname
803     \expandafter{%
804       \romannumeral#1000\expandafter\BitSet@Space
805       \csname BS@#2\endcsname
806     }%
807   }%
808   \else
809     \expandafter\BitSet@ShiftRight\BitSet@Gobble#1!{#2}%
810   \fi
811 }

```

### 2.8.7 \bitsetShiftRight

\bitsetShiftRight

```

812 \def\bitsetShiftRight#1#2{%
813   \BitSet@IfUndefined{#1}{%
814     \bitsetReset{#1}%
815   }{%
816     \bitsetIsEmpty{#1}{%
817     }{%
818       \expandafter\expandafter\expandafter\BitSet@ShiftRight
819       \intcalcsNum{#2}!{#1}%
820     }%
821   }%
822 }

```

\BitSet@ShiftRight

```

823 \def\BitSet@ShiftRight#1!#2{%
824   \ifcase\intcalcsGn{#1} %
825   \or
826     \expandafter\edef\csname BS@#2\endcsname{%
827       \expandafter\expandafter\expandafter\BitSet@Kill
828       \csname BS@#2\expandafter\endcsname\expandafter\BitSet@Empty
829       \expandafter=%
830       \expandafter{\expandafter}\expandafter{\expandafter}%
831       \romannumeral#1000!%
832     }%
833   \else
834     \expandafter\BitSet@ShiftLeft\BitSet@Gobble#1!{#2}%
835   \fi
836 }

```

\BitSet@Kill

```

837 \def\BitSet@Kill#1#2=#3#4#5{%
838   #3#4%
839   \ifx#5!%
840     \ifx#1\BitSet@Empty
841     0%

```

```

842     \else
843         #1#2%
844     \fi
845 \else
846     \ifx#1\BitSet@Empty
847         0%
848         \BitSet@AfterFiFi\BitSet@Cleanup
849     \else
850         \BitSet@Kill#2=%
851     \fi
852 \BitSet@Fi
853 }

```

## 2.9 Bit manipulation

\bitsetClear

```

854 \def\bitsetClear{%
855     \BitSet@CheckIndex\BitSet@Clear
856 }

```

\bitsetSet

```

857 \def\bitsetSet{%
858     \BitSet@CheckIndex\BitSet@Set
859 }

```

\bitsetFlip

```

860 \def\bitsetFlip{%
861     \BitSet@CheckIndex\BitSet@Flip
862 }

```

\bitsetSetValue

```

863 \def\bitsetSetValue#1#2#3{%
864     \expandafter\expandafter\expandafter\BitSet@SetValue
865     \intcalcNum{#3}!{#1}{#2}%
866 }

```

\BitSet@SetValue #1: plain value

#2: BitSet

#3: Index

```

867 \def\BitSet@SetValue#1!{%
868     \BitSet@CheckIndex{%
869         \ifcase#1 %
870             \expandafter\BitSet@Clear
871         \or
872             \expandafter\BitSet@Set
873         \else
874             \BitSet@ErrorInvalidBitValue{#1}%
875             \expandafter\expandafter\expandafter\BitSet@Gobble
876             \expandafter\BitSet@Gobble
877         \fi
878     }%
879 }

```

\BitSet@ErrorInvalidBitValue #1: Wrong bit value

```

880 \def\BitSet@ErrorInvalidBitValue#1{%
881     \@PackageError{bitset}{%
882         Invalid bit value (#1) not in range 0..1%
883     }\@ehc
884 }

```

### 2.9.1 Clear operation

```
\BitSet@Clear #1: BitSet
#2: plain and checked index
885 \def\BitSet@Clear#1#2{%
886   \edef\BitSet@Temp{%
887     \expandafter\expandafter\expandafter\BitSet@@Clear
888     \csname BS@#1\expandafter\endcsname
889     \expandafter\BitSet@Empty\expandafter=\expandafter!%
890     \romannumeral#2000!%
891   }%
892   \expandafter\let\csname BS@#1\expandafter\endcsname
893   \ifx\BitSet@Temp\BitSet@Empty
894     \BitSet@Zero
895   \else
896     \BitSet@Temp
897   \fi
898 }
```

```
\BitSet@@Clear
899 \def\BitSet@@Clear#1#2=#3!#4{%
900   \ifx#4!%
901     \ifx#1\BitSet@Empty
902     \else
903       \ifx\BitSet@Empty#2%
904       \else
905         #30#2%
906       \fi
907     \fi
908   \else
909     \ifx#1\BitSet@Empty
910       \BitSet@AfterFiFi\BitSet@Cleanup
911     \else
912       \ifx#10%
913         \BitSet@AfterFiFiFi{%
914           \BitSet@@Clear#2=#30!%
915         }%
916       \else
917         #31%
918         \BitSet@AfterFiFiFi{%
919           \BitSet@@Clear#2=!%
920         }%
921       \fi
922     \fi
923   \BitSet@Fi
924 }
```

### 2.9.2 Set operation

```
\BitSet@Set #1: BitSet
#2: plain and checked Index
925 \def\BitSet@Set#1#2{%
926   \expandafter\edef\csname BS@#1\endcsname{%
927     \expandafter\expandafter\expandafter\BitSet@@Set
928     \csname BS@#1\expandafter\endcsname
929     \expandafter\BitSet@Empty\expandafter=%
930     \expandafter{\expandafter}\expandafter{\expandafter}%
931     \romannumeral#2000!%
932   }%
933 }
```

```
\BitSet@@Set
```

```

934 \def\BitSet@@Set#1#2=#3#4#5{%
935   #3#4%
936   \ifx#5!%
937     1#2%
938   \else
939     \ifx#1\BitSet@Empty
940       0%
941     \BitSet@AfterFiFi\BitSet@@@Set
942   \else
943     #1%
944     \BitSet@@Set#2=%
945   \fi
946 \BitSet@Fi
947 }

```

\BitSet@@@Set

```

948 \def\BitSet@@@Set#1{%
949   \ifx#1!%
950     1%
951   \else
952     0%
953   \expandafter\BitSet@@@Set
954 \fi
955 }

```

### 2.9.3 Flip operation

\BitSet@Flip #1: BitSet

#2: plain and checked Index

```

956 \def\BitSet@Flip#1#2{%
957   \edef\BitSet@Temp{%
958     \expandafter\expandafter\expandafter\BitSet@@Flip
959     \csname BS@#1\expandafter\endcsname
960     \expandafter\BitSet@Empty\expandafter=\expandafter!%
961     \romannumeral#2000!%
962   }%
963   \expandafter\let\csname BS@#1\expandafter\endcsname
964   \ifx\BitSet@Temp\BitSet@Empty
965     \BitSet@Zero
966   \else
967     \BitSet@Temp
968   \fi
969 }

```

\BitSet@@Flip

```

970 \def\BitSet@@Flip#1#2=#3!#4{%
971   \ifx#4!%
972     \ifx#11%
973       \ifx\BitSet@Empty#2%
974       \else
975         #30#2%
976       \fi
977     \else
978       #31#2%
979     \fi
980   \else
981     \ifx#1\BitSet@Empty
982       #30%
983     \BitSet@AfterFiFi\BitSet@@@Set
984   \else
985     \ifx#10%
986       \BitSet@AfterFiFiFi{%

```

```

987         \BitSet@@Flip#2=#30!%
988     }%
989     \else
990         #31%
991         \BitSet@AfterFiFiFi{%
992         \BitSet@@Flip#2=!%
993     }%
994     \fi
995     \fi
996     \BitSet@Fi
997 }

```

## 2.9.4 Range operators

\bitsetClearRange

```

998 \def\bitsetClearRange{%
999     \BitSet@Range\BitSet@Clear
1000 }

```

\bitsetSetRange

```

1001 \def\bitsetSetRange{%
1002     \BitSet@Range\BitSet@Set
1003 }

```

\bitsetFlipRange

```

1004 \def\bitsetFlipRange{%
1005     \BitSet@Range\BitSet@Flip
1006 }

```

\bitsetSetValueRange

```

1007 \def\bitsetSetValueRange#1#2#3#4{%
1008     \expandafter\expandafter\expandafter\BitSet@SetValueRange
1009     \intcalcNum{#4}!\{#1\}{#2}\{#3}%
1010 }

```

\BitSet@SetValueRange

```

1011 \def\BitSet@SetValueRange#1!#2#3#4{%
1012     \ifcase#1 %
1013         \BitSet@Range\BitSet@Clear{#2}\{#3}\{#4}%
1014     \or
1015         \BitSet@Range\BitSet@Set{#2}\{#3}\{#4}%
1016     \else
1017         \BitSet@ErrorInvalidBitValue{#1}%
1018     \fi
1019 }

```

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

#2: BitSet

#3: Index from

#4: Index to

```

1020 \def\BitSet@Range#1#2#3#4{%
1021     \edef\BitSet@Temp{%
1022         \noexpand\BitSet@@Range\noexpand#1{#2}%
1023         \intcalcNum{#3}!\intcalcNum{#4}!%
1024     }%
1025     \BitSet@Temp
1026 }

```

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

#2: BitSet

```

#3: Index from
#4: Index to
1027 \def\BitSet@@Range#1#2#3!#4!{%
1028   \ifnum#3<0 %
1029     \BitSet@NegativeIndex#1{#2}#3!#4!0!#4!%
1030   \else
1031     \ifnum#4<0 %
1032       \BitSet@NegativeIndex#1{#2}#3!#4!#3!0!%
1033     \else
1034       \ifcase\intcalcCmp{#3}-{#4} %
1035       \or
1036         \@PackageError{bitset}{%
1037           Wrong index numbers in range [#3..#4]\MessageBreak% hash-ok
1038           for clear/set/flip on bit set '#2'.\MessageBreak
1039           The lower index exceeds the upper index.\MessageBreak
1040           Canceling the operation as error recovery%
1041         }\@ehc
1042       \else
1043         \BitSet@@Range#3!#4!#1{#2}%
1044       \fi
1045     \fi
1046   \fi
1047 }

```

\BitSet@NegativeIndex

```

1048 \def\BitSet@NegativeIndex#1#2#3!#4!#5!#6!{%
1049   \@PackageError{bitset}{%
1050     Negative index in range [#3..#4]\MessageBreak % hash-ok
1051     for \string\bitset
1052     \ifx#1\BitSet@Clear
1053       Clear%
1054     \else
1055       \ifx#1\BitSet@Set
1056         Set%
1057       \else
1058         Flip%
1059       \fi
1060     \fi
1061     Range on bit set '#2'.\MessageBreak
1062     Using [#5..#6] as error recovery% hash-ok
1063   }\@ehc
1064   \BitSet@@Range#1{#2}#5!#6!%
1065 }

```

\BitSet@@Range

```

1066 \def\BitSet@@Range#1!#2!#3#4{%
1067   \ifnum#1<#2 %
1068     #3{#4}{#1}%
1069     \BitSet@AfterFi{%
1070       \expandafter\expandafter\expandafter\BitSet@@Range
1071       \IntCalcInc#1!!#2!#3{#4}%
1072     }%
1073   \BitSet@Fi
1074 }

```

## 2.10 Bit retrieval

### 2.10.1 \bitsetGet

\bitsetGet

```

1075 \def\bitsetGet#1#2{%
1076   \number

```

```

1077 \expandafter\expandafter\expandafter\BitSet@Get
1078 \intcalculus{#2}!{#1}%
1079 }

\BitSet@Get #1: plain index
#2: BitSet
1080 \def\BitSet@Get#1!#2{%
1081 \ifnum#1<0 %
1082 \BitSet@AfterFi{%
1083 0 \BitSetError:NegativeIndex%
1084 }%
1085 \else
1086 \BitSet@IfUndefined{#2}{0}{%
1087 \expandafter\expandafter\expandafter\BitSet@@Get
1088 \csname BS@#2\expandafter\endcsname
1089 \expandafter!\expandafter=%
1090 \expandafter{\expandafter}\expandafter{\expandafter}%
1091 \romannumeral\intcalculus{#1}000!%
1092 }%
1093 \expandafter\BitSet@Space
1094 \BitSet@Fi
1095 }

\BitSet@@@Get
1096 \def\BitSet@@@Get#1#2=#3#4#5{%
1097 #3#4%
1098 \ifx#5!%
1099 \ifx#1!%
1100 0%
1101 \else
1102 #1%
1103 \fi
1104 \else
1105 \ifx#1!%
1106 0%
1107 \BitSet@AfterFiFi\BitSet@Cleanup
1108 \else
1109 \BitSet@@@Get#2=%
1110 \fi
1111 \BitSet@Fi
1112 }

```

## 2.10.2 \bitsetNextClearBit, \bitsetNextSetBit

\bitsetNextClearBit

```

1113 \def\bitsetNextClearBit#1#2{%
1114 \number
1115 \expandafter\expandafter\expandafter\BitSet@NextClearBit
1116 \intcalculus{#2}!{#1} %
1117 }

```

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

```

1118 \def\BitSet@NextClearBit#1!#2{%
1119 \ifnum#1<0 %
1120 \BitSet@NextClearBit0!{#2}%
1121 \BitSet@AfterFi{%
1122 \expandafter\BitSet@Space
1123 \expandafter\BitSetError:NegativeIndex\romannumeral0%
1124 }%
1125 \else
1126 \bitsetIsEmpty{#2}{#1}{%

```



```

1127     \expandafter\BitSet@Skip
1128     \number#1\expandafter\expandafter\expandafter!%
1129     \csname BS@#2\endcsname!!!!!!!!!=%
1130     {\BitSet@@NextClearBit#1!}%
1131 }%
1132 \BitSet@Fi
1133 }

\BitSet@@NextClearBit #1: index for next bit in #2
#2: next bit
1134 \def\BitSet@@NextClearBit#1!#2{%
1135     \ifx#2!%
1136         #1%
1137     \else
1138         \ifx#20%
1139             #1%
1140             \BitSet@AfterFiFi\BitSet@Cleanup
1141         \else
1142             \BitSet@AfterFiFi{%
1143                 \expandafter\expandafter\expandafter\BitSet@@NextClearBit
1144                 \IntCalcInc#1!!%
1145             }%
1146         \fi
1147     \BitSet@Fi
1148 }

\bitsetNextSetBit
1149 \def\bitsetNextSetBit#1#2{%
1150     \number
1151     \expandafter\expandafter\expandafter\BitSet@NextSetBit
1152     \intcalcNum{#2}!{#1} %
1153 }

\BitSet@NextSetBit #1: Index
#2: BitSet
1154 \def\BitSet@NextSetBit#1!#2{%
1155     \ifnum#1<0 %
1156         \BitSet@NextSetBit0!{#2}%
1157         \BitSet@AfterFi{%
1158             \expandafter\BitSet@Space
1159             \expandafter\BitSetError:NegativeIndex\romannumeral0%
1160         }%
1161     \else
1162         \bitsetIsEmpty{#2}{-1}{%
1163             \expandafter\BitSet@Skip
1164             \number#1\expandafter\expandafter\expandafter!%
1165             \csname BS@#2\endcsname!!!!!!!!!=%
1166             {\BitSet@@NextSetBit#1!}%
1167         }%
1168     \BitSet@Fi
1169 }

\BitSet@@NextSetBit #1: index for next bit in #2
#2: next bit
1170 \def\BitSet@@NextSetBit#1!#2{%
1171     \ifx#2!%
1172         -1%
1173     \else
1174         \ifx#21%
1175             #1%
1176             \BitSet@AfterFiFi\BitSet@Cleanup
1177         \else

```

```

1178     \BitSet@AfterFiFi{%
1179     \expandafter\expandafter\expandafter\BitSet@@NextSetBit
1180     \IntCalcInc#1!!%
1181     }%
1182     \fi
1183     \BitSet@Fi
1184 }

\BitSet@Cleanup
1185 \def\BitSet@Cleanup#1!{ }

\BitSet@Skip #1: number of bits to skip
#2: bits
#3: continuation code
1186 \def\BitSet@Skip#1!#2{%
1187     \ifx#2!%
1188         \BitSet@AfterFiFi{%
1189         \BitSet@SkipContinue%
1190         }%
1191     \else
1192         \ifcase#1 %
1193             \BitSet@AfterFiFi{%
1194             \BitSet@SkipContinue#2%
1195             }%
1196         \or
1197             \BitSet@AfterFiFi\BitSet@SkipContinue
1198         \or
1199             \BitSet@AfterFiFi{%
1200             \expandafter\BitSet@SkipContinue\BitSet@Gobble
1201             }%
1202         \else
1203             \ifnum#1>8 %
1204                 \BitSet@AfterFiFiFiFi{%
1205                 \expandafter\BitSet@Skip
1206                 \number\IntCalcSub#1!8!\expandafter!%
1207                 \BitSet@GobbleSeven
1208                 }%
1209             \else
1210                 \BitSet@AfterFiFiFiFi{%
1211                 \expandafter\expandafter\expandafter\BitSet@Skip
1212                 \IntCalcDec#1!!%
1213                 }%
1214             \fi
1215         \fi
1216         \BitSet@Fi
1217 }

\BitSet@SkipContinue #1: remaining bits
#2: continuation code
1218 \def\BitSet@SkipContinue#1!#2=#3{%
1219     #3#1!%
1220 }

\BitSet@GobbleSeven
1221 \def\BitSet@GobbleSeven#1#2#3#4#5#6#7{ }

```

### 2.10.3 \bitsetGetSetBitList

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

```

1222 \def\bitsetGetSetBitList#1{%
1223     \romannumeral0%

```

```

1224 \bitsetIsEmpty{#1}{ }{%
1225   \expandafter\BitSet@GetSetBitList
1226   \number\BitSet@NextSetBit0!{#1}!{#1}{ }!%
1227 }%
1228 }

```

```

\BitSet@GetSetBitList #1: found index
#2: BitSet
#3: comma #4: result
1229 \def\BitSet@GetSetBitList#1!#2#3#4!{%
1230   \ifnum#1<0 %
1231     \BitSet@AfterFi{ #4}%
1232   \else
1233     \BitSet@AfterFi{%
1234       \expandafter\BitSet@GetSetBitList\number
1235       \expandafter\expandafter\expandafter\BitSet@NextSetBit
1236       \IntCalcInc#1!!{#2}!{#2},#4#3#1!%
1237     }%
1238   \BitSet@Fi
1239 }

```

## 2.11 Bit set properties

```

\bitsetSize
1240 \def\bitsetSize#1{%
1241   \number
1242   \BitSet@IfUndefined{#1}{0 }{%
1243     \expandafter\expandafter\expandafter\BitSet@Size
1244     \expandafter\expandafter\expandafter1%
1245     \expandafter\expandafter\expandafter!%
1246     \csname BS@#1\endcsname!0!%
1247   }%
1248 }

```

```

\BitSet@Size #1: counter
#2#3: bits
#4: result
1249 \def\BitSet@Size#1!#2#3!#4!{%
1250   \ifx#21%
1251     \ifx\#3\%
1252       \BitSet@AfterFiFi{#1 }%
1253     \else
1254       \BitSet@AfterFiFi{%
1255         \expandafter\expandafter\expandafter\BitSet@Size
1256         \IntCalcInc#1!!#3!#1!%
1257       }%
1258     \fi
1259   \else
1260     \ifx\#3\%
1261       \BitSet@AfterFiFi{#4 }%
1262     \else
1263       \BitSet@AfterFiFi{%
1264         \expandafter\expandafter\expandafter\BitSet@Size
1265         \IntCalcInc#1!!#3!#4!%
1266       }%
1267     \fi
1268   \fi
1269   \BitSet@Fi
1270 }

```

```

\bitsetCardinality

```

```

1271 \def\bitsetCardinality#1{%
1272   \number
1273   \BitSet@IfUndefined{#1}{0 }{%
1274     \expandafter\expandafter\expandafter\BitSet@Cardinality
1275     \expandafter\expandafter\expandafter0%
1276     \expandafter\expandafter\expandafter!%
1277     \csname BS@#1\endcsname!%
1278   }%
1279 }

```

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

```

1280 \def\BitSet@Cardinality#1!#2#3!{%
1281   \ifx#21%
1282     \ifx\#3\%
1283       \BitSet@AfterFiFi{\IntCalcInc#1! }%
1284     \else
1285       \BitSet@AfterFiFi{%
1286         \expandafter\expandafter\expandafter\BitSet@Cardinality
1287         \IntCalcInc#1!#3!%
1288       }%
1289     \fi
1290   \else
1291     \ifx\#3\%
1292       \BitSet@AfterFiFi{#1 }%
1293     \else
1294       \BitSet@AfterFiFi{%
1295         \BitSet@Cardinality#1!#3!%
1296       }%
1297     \fi
1298   \fi
1299   \BitSet@Fi
1300 }

```

## 2.12 Queries

\bitsetIsDefined

```

1301 \def\bitsetIsDefined#1{%
1302   \BitSet@IfUndefined{#1}%
1303   \BitSet@SecondOfTwo
1304   \BitSet@FirstOfTwo
1305 }

```

\bitsetIsEmpty

```

1306 \def\bitsetIsEmpty#1{%
1307   \BitSet@IfUndefined{#1}\BitSet@FirstOfTwo{%
1308     \expandafter\ifx\csname BS@#1\endcsname\BitSet@Zero
1309     \expandafter\BitSet@FirstOfTwo
1310   \else
1311     \expandafter\BitSet@SecondOfTwo
1312   \fi
1313 }%
1314 }

```

\BitSet@Zero

```

1315 \def\BitSet@Zero{0}

```

\bitsetQuery

```

1316 \def\bitsetQuery#1#2{%
1317   \ifnum\bitsetGet{#1}{#2}=1 %
1318     \expandafter\BitSet@FirstOfTwo

```

```

1319 \else
1320 \expandafter\BitSet@SecondOfTwo
1321 \fi
1322 }

```

\bitsetEquals

```

1323 \def\bitsetEquals#1#2{%
1324 \BitSet@IfUndefined{#1}{%
1325 \BitSet@IfUndefined{#2}\BitSet@FirstOfTwo\BitSet@SecondOfTwo
1326 }{%
1327 \BitSet@IfUndefined{#2}\BitSet@SecondOfTwo{%
1328 \expandafter\ifx\csname BS@#1\expandafter\endcsname
1329 \csname BS@#2\endcsname
1330 \expandafter\BitSet@FirstOfTwo
1331 \else
1332 \expandafter\BitSet@SecondOfTwo
1333 \fi
1334 }%
1335 }%
1336 }

```

\bitsetIntersects

```

1337 \def\bitsetIntersects#1#2{%
1338 \bitsetIsEmpty{#1}\BitSet@SecondOfTwo{%
1339 \bitsetIsEmpty{#2}\BitSet@SecondOfTwo{%
1340 \expandafter\expandafter\expandafter\BitSet@Intersects
1341 \csname BS@#1\expandafter\expandafter\expandafter\endcsname
1342 \expandafter\expandafter\expandafter!%
1343 \csname BS@#2\endcsname!%
1344 }%
1345 }%
1346 }

```

\BitSet@Intersects

```

1347 \def\BitSet@Intersects#1#2!#3#4!{%
1348 \ifnum#1#3=11 %
1349 \BitSet@AfterFi\BitSet@FirstOfTwo
1350 \else
1351 \ifx\\#2\\%
1352 \BitSet@AfterFiFi\BitSet@SecondOfTwo
1353 \else
1354 \ifx\\#4\\%
1355 \BitSet@AfterFiFiFi\BitSet@SecondOfTwo
1356 \else
1357 \BitSet@AfterFiFiFiFi{%
1358 \BitSet@Intersects#2!#4!%
1359 }%
1360 \fi
1361 \fi
1362 \BitSet@Fi
1363 }

```

```

1364 \BitSet@AtEnd
1365 </package>

```

## 3 Test

### 3.1 Catcode checks for loading

```

1366 <*test1>
1367 \catcode'\@=11 %

```

```

1368 \def\RestoreCatcodes{}
1369 \count@=0 %
1370 \loop
1371   \edef\RestoreCatcodes{%
1372     \RestoreCatcodes
1373     \catcode\the\count@=\the\catcode\count@\relax
1374   }%
1375 \ifnum\count@<255 %
1376   \advance\count@\@ne
1377 \repeat
1378
1379 \def\RangeCatcodeInvalid#1#2{%
1380   \count@=#1\relax
1381   \loop
1382     \catcode\count@=15 %
1383   \ifnum\count@<#2\relax
1384     \advance\count@\@ne
1385   \repeat
1386 }
1387 \def\Test{%
1388   \RangeCatcodeInvalid{0}{47}%
1389   \RangeCatcodeInvalid{58}{64}%
1390   \RangeCatcodeInvalid{91}{96}%
1391   \RangeCatcodeInvalid{123}{255}%
1392   \catcode'\@=12 %
1393   \catcode'\=0 %
1394   \catcode'\{=1 %
1395   \catcode'\}=2 %
1396   \catcode'\#=6 %
1397   \catcode'\[=12 %
1398   \catcode'\]=12 %
1399   \catcode'\%=14 %
1400   \catcode'\ =10 %
1401   \catcode13=5 %
1402   \input bitset.sty\relax
1403   \RestoreCatcodes
1404 }
1405 \Test
1406 \csname @@end\endcsname
1407 \end
1408 </test1>

```

## 3.2 Macro tests

### 3.2.1 Preamble

```

1409 <*test2>
1410 \NeedsTeXFormat{LaTeX2e}
1411 \nofiles
1412 \documentclass{article}
1413 \makeatletter
1414 <*noetex>
1415 \let\SavedNumexpr\numexpr
1416 \let\SavedIfcsname\ifcsname
1417 \let\SavedCurrentgrouplevel\currentgrouplevel
1418 \def\ETeXDisable{%
1419   \let\ifcsname\@undefined
1420   \let\numexpr\@undefined
1421   \let\currentgrouplevel\@undefined
1422 }
1423 \ETeXDisable
1424 </noetex>
1425 \makeatletter

```

```

1426 \chardef\BitSet@TestMode=1 %
1427 \makeatother
1428 \usepackage{bitset}[2007/09/28]
1429 \noetex
1430 \def\ETeXEnable{%
1431   \let\numexpr\SavedNumexpr
1432   \let\ifcename\SavedIfcename
1433   \let\currentgrouplevel\SavedCurrentgrouplevel
1434 }
1435 \ETeXEnable
1436 \noetex
1437 \usepackage{qstest}
1438 \IncludeTests{*}
1439 \LogTests{log}{*}{*}
1440 \makeatletter

```

### 3.2.2 Time

```

1441 \begingroup\expandafter\expandafter\expandafter\endgroup
1442 \expandafter\ifx\cename pdfresettimer\endcename\relax
1443 \else
1444   \newcount\SummaryTime
1445   \newcount\TestTime
1446   \SummaryTime=\z@
1447   \newcommand*\PrintTime}[2]{%
1448     \typeout{%
1449       [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
1450     }%
1451   }%
1452   \newcommand*\StartTime#[1]{%
1453     \renewcommand*\TimeDescription}{#1}%
1454     \pdfresettimer
1455   }%
1456   \newcommand*\TimeDescription{}%
1457   \newcommand*\StopTime{%
1458     \TestTime=\pdfelapsedtime
1459     \global\advance\SummaryTime\TestTime
1460     \PrintTime\TimeDescription\TestTime
1461   }%
1462   \let\saved@qstest\qstest
1463   \let\saved@endqstest\endqstest
1464   \def\qstest#1#2{%
1465     \saved@qstest{#1}{#2}%
1466     \StartTime{#1}%
1467   }%
1468   \def\endqstest{%
1469     \StopTime
1470     \saved@endqstest
1471   }%
1472   \AtEndDocument{%
1473     \PrintTime{summary}\SummaryTime
1474   }%
1475 \fi

```

### 3.2.3 Detection of unwanted space

```

1476 \let\orig@qstest\qstest
1477 \let\orig@endqstest\endqstest
1478 \def\qstest#1#2{%
1479   \orig@qstest{#1}{#2}%
1480   \setbox0\hbox\bgroup\begingroup\ignorespaces
1481 }
1482 \def\endqstest{%
1483   \endgroup\egroup
1484   \Expect*{\the\wd0}{0.0pt}%

```

```

1485 \orig@endqstest
1486 }

```

### 3.2.4 Test macros

```

1487 \newcounter{Test}
1488
1489 \def\TestError#1#2{%
1490   \begingroup
1491     \setcounter{Test}{0}%
1492     \sbox0{%
1493       \def\@PackageError##1##2##3{%
1494         \stepcounter{Test}%
1495         \begingroup
1496           \let\MessageBreak\relax
1497 <*noetex>
1498           \ETEXEnable
1499 </noetex>
1500           \Expect{##1}{bitset}%
1501           \Expect*{##2}*{#1}%
1502         \endgroup
1503       }%
1504 <*noetex>
1505       \ETEXDisable
1506 </noetex>
1507       #2%
1508     }%
1509     \Expect*\theTest}{1}%
1510     \Expect*\the\wd0}{0.0pt}%
1511   \endgroup
1512 }
1513
1514 \def\TestErrorNegativeIndex#1#2{%
1515   \TestError{Invalid negative index (#1)}{#2}%
1516 }
1517
1518 \def\TestGetterUndefined#1{%
1519   \CheckUndef{dummy}%
1520   \expandafter\expandafter\expandafter\Expect
1521   \expandafter\expandafter\expandafter{#1{dummy}}{0}%
1522 }
1523
1524 \def\ExpectBitSet#1#2{%
1525   \expandafter\expandafter\expandafter\Expect
1526   \expandafter\expandafter\expandafter
1527   {\csname BS@#1\endcsname}*{#2}%
1528 }
1529 \def\Check#1#2{%
1530   \ExpectBitSet{#1}{#2}%
1531 }
1532 \def\CheckUndef#1{%
1533   \begingroup
1534     \Expect*{%
1535       \expandafter
1536       \ifx\csname BS@#1\endcsname\relax true\else false\fi
1537     }{true}%
1538   \endgroup
1539 }
1540 \def\RevCheck#1#2{%
1541   \ExpectBitSet{#1}{\Reverse#2!!}%
1542 }
1543 \def\Set#1#2{%
1544   \expandafter\def\csname BS@#1\endcsname{#2}%
1545 }

```



```

1546 \def\RevSet#1#2{%
1547   \expandafter\edef\csname BS@#1\endcsname{%
1548     \Reverse#2!!%
1549   }%
1550 }
1551 \def\Reverse#1#2!#3!{%
1552   \ifx\#2\%
1553     #1#3%
1554     \expandafter\@gobble
1555   \else
1556     \expandafter\@firstofone
1557   \fi
1558   {\Reverse#2!#1#3!}%
1559 }

```

### 3.2.5 Test sets

```

1560 \begin{qstest}{Let}{Let}
1561   \CheckUndef{abc}%
1562   \CheckUndef{xyz}%
1563   \bitsetLet{xyz}{abc}%
1564   \CheckUndef{abc}%
1565   \Check{xyz}{0}%
1566   \Set{abc}{1}%
1567   \Check{abc}{1}%
1568   \Check{xyz}{0}%
1569   \bitsetLet{xyz}{abc}%
1570   \Check{abc}{1}%
1571   \Check{xyz}{1}%
1572   \Set{xyz}{11}%
1573   \Check{abc}{1}%
1574   \Check{xyz}{11}%
1575 \end{qstest}
1576
1577 \begin{qstest}{Reset}{Reset}
1578   \bitsetReset{xyz}%
1579   \Check{xyz}{0}%
1580   \bitsetReset{abc}%
1581   \Check{abc}{0}%
1582   \Set{abc}{10101}%
1583   \bitsetReset{abc}%
1584   \Check{abc}{0}%
1585 \end{qstest}
1586
1587 \begin{qstest}{Get/Query}{Get/Query}
1588   \expandafter\expandafter\expandafter\Expect
1589   \expandafter\expandafter\expandafter{%
1590     \bitsetGet{dummy}{0}%
1591   }{0}%
1592   \begingroup
1593     \expandafter\def\csname BitSetError:NegativeIndex\endcsname{}%
1594     \Set{abc}{1}%
1595     \Expect*{\bitsetQuery{abc}{-1}{true}{false}}{false}%
1596   \endgroup
1597   \def\Test#1#2#3{%
1598     \Set{abc}{#1}%
1599     \expandafter\expandafter\expandafter\Expect
1600     \expandafter\expandafter\expandafter{\bitsetGet{abc}{#2}}{#3}%
1601     \Expect*{\bitsetQuery{abc}{#2}{true}{false}}%
1602     *{\ifcase#3 false\or true\else error\fi}%
1603   }%
1604   \Test{1}{100}{0}%
1605   \Test{0}{0}{0}%
1606   \Test{1}{0}{1}%

```

```

1607 \Test{11}{1}{1}%
1608 \Test{111}{1}{1}%
1609 \Test{101}{1}{0}%
1610 \Test{101}{2}{1}%
1611 \Test{10100110011}{10}{1}%
1612 \end{qstest}
1613
1614 \begin{qstest}{Size}{Size}
1615 \TestGetterUndefined\bitsetSize
1616 \def\Test#1#2{%
1617   \Set{abc}{#1}%
1618   \expandafter\expandafter\expandafter\Expect
1619   \expandafter\expandafter\expandafter{\bitsetSize{abc}}{#2}%
1620 }%
1621 \Test{0}{0}%
1622 \Test{1}{1}%
1623 \Test{00}{0}%
1624 \Test{0000000}{0}%
1625 \Test{10}{1}%
1626 \Test{01}{2}%
1627 \Test{11}{2}%
1628 \Test{010}{2}%
1629 \Test{011}{3}%
1630 \Test{100110011}{9}%
1631 \Test{0000011111000001111100000}{20}%
1632 \Test{00000000000000000000000011111111111111111}{45}%
1633 \end{qstest}
1634
1635 \begin{qstest}{Cardinality}{Cardinality}
1636 \TestGetterUndefined\bitsetCardinality
1637 \def\Test#1#2{%
1638   \Set{abc}{#1}%
1639   \expandafter\expandafter\expandafter\Expect
1640   \expandafter\expandafter\expandafter{%
1641     \bitsetCardinality{abc}%
1642   }{#2}%
1643 }%
1644 \Test{0}{0}%
1645 \Test{1}{1}%
1646 \Test{00}{0}%
1647 \Test{0000000}{0}%
1648 \Test{10}{1}%
1649 \Test{01}{1}%
1650 \Test{11}{2}%
1651 \Test{010}{1}%
1652 \Test{011}{2}%
1653 \Test{100110011}{5}%
1654 \Test{0000011111000001111100000}{10}%
1655 \Test{0000000000000000000000000000000011111111111111111111}{20}%
1656 \end{qstest}
1657
1658 \begin{qstest}{NextClearBit/NextSetBit}{NextClearBit/NextSetBit}
1659 \def\Test#1#2{%
1660   \expandafter\expandafter\expandafter\Expect
1661   \expandafter\expandafter\expandafter{%
1662     \TestOp{abc}{#1}%
1663   }{#2}%
1664 }%
1665 \def\Clear{\let\TestOp\bitsetNextClearBit}%
1666 \def/Set{\let\TestOp\bitsetNextSetBit}%
1667 \begingroup
1668 \catcode'\:=11 %

```

```

1669     \bitsetSetBin{abc}{1}%
1670     \Clear
1671     \Test{-1}{1\BitSetError:NegativeIndex}%
1672     \Set
1673     \Test{-1}{0\BitSetError:NegativeIndex}%
1674 \endgroup
1675 \let\BS@abc\undefined
1676 \Clear
1677 \Test{0}{0}%
1678 \Test{1}{1}%
1679 \Test{2}{2}%
1680 \Test{100}{100}%
1681 \Set
1682 \Test{0}{-1}%
1683 \Test{1}{-1}%
1684 \Test{100}{-1}%
1685 \bitsetReset{abc}%
1686 \Clear
1687 \Test{0}{0}%
1688 \Test{1}{1}%
1689 \Test{2}{2}%
1690 \Test{100}{100}%
1691 \Set
1692 \Test{0}{-1}%
1693 \Test{1}{-1}%
1694 \Test{100}{-1}%
1695 \bitsetSetBin{abc}{1}%
1696 \Clear
1697 \Test{0}{1}%
1698 \Test{1}{1}%
1699 \Test{2}{2}%
1700 \Test{100}{100}%
1701 \Set
1702 \Test{0}{0}%
1703 \Test{1}{-1}%
1704 \Test{100}{-1}%
1705 \bitsetSetBin{abc}{111000111000111000111}%
1706 \Clear
1707 \Test{0}{3}%
1708 \Test{1}{3}%
1709 \Test{2}{3}%
1710 \Test{3}{3}%
1711 \Test{4}{4}%
1712 \Test{5}{5}%
1713 \Test{6}{9}%
1714 \Test{7}{9}%
1715 \Test{8}{9}%
1716 \Test{9}{9}%
1717 \Test{10}{10}%
1718 \Test{11}{11}%
1719 \Test{12}{15}%
1720 \Test{13}{15}%
1721 \Test{14}{15}%
1722 \Test{15}{15}%
1723 \Test{16}{16}%
1724 \Test{17}{17}%
1725 \Test{18}{21}%
1726 \Test{19}{21}%
1727 \Test{20}{21}%
1728 \Test{21}{21}%
1729 \Test{22}{22}%
1730 \Test{100}{100}%

```

```

1731 \Set
1732 \Test{0}{0}%
1733 \Test{1}{1}%
1734 \Test{2}{2}%
1735 \Test{3}{6}%
1736 \Test{4}{6}%
1737 \Test{5}{6}%
1738 \Test{6}{6}%
1739 \Test{7}{7}%
1740 \Test{8}{8}%
1741 \Test{9}{12}%
1742 \Test{10}{12}%
1743 \Test{11}{12}%
1744 \Test{12}{12}%
1745 \Test{13}{13}%
1746 \Test{14}{14}%
1747 \Test{15}{18}%
1748 \Test{16}{18}%
1749 \Test{17}{18}%
1750 \Test{18}{18}%
1751 \Test{19}{19}%
1752 \Test{20}{20}%
1753 \Test{21}{-1}%
1754 \Test{22}{-1}%
1755 \Test{100}{-1}%
1756 \bitsetSetBin{abc}{1111111}%
1757 \Clear
1758 \Test{6}{7}%
1759 \Test{7}{7}%
1760 \Test{8}{8}%
1761 \Test{100}{100}%
1762 \Set
1763 \Test{6}{6}%
1764 \Test{7}{-1}%
1765 \Test{8}{-1}%
1766 \Test{100}{-1}%
1767 \bitsetSetBin{abc}{11111111}%
1768 \Clear
1769 \Test{7}{8}%
1770 \Test{8}{8}%
1771 \Test{9}{9}%
1772 \Test{100}{100}%
1773 \Set
1774 \Test{7}{7}%
1775 \Test{8}{-1}%
1776 \Test{9}{-1}%
1777 \Test{100}{-1}%
1778 \bitsetSetBin{abc}{111111111}%
1779 \Clear
1780 \Test{8}{9}%
1781 \Test{9}{9}%
1782 \Test{10}{10}%
1783 \Test{100}{100}%
1784 \Set
1785 \Test{8}{8}%
1786 \Test{9}{-1}%
1787 \Test{10}{-1}%
1788 \Test{100}{-1}%
1789 \bitsetSetBin{abc}{1111111111}%
1790 \Clear
1791 \Test{9}{10}%
1792 \Test{10}{10}%

```

```

1793 \Test{11}{11}%
1794 \Test{100}{100}%
1795 \Set
1796 \Test{9}{9}%
1797 \Test{10}{-1}%
1798 \Test{11}{-1}%
1799 \Test{100}{-1}%
1800 \end{qstest}
1801
1802 \begin{qstest}{GetSetBitList}{GetSetBitList}
1803 \let\BS@abc\@undefined
1804 \expandafter\expandafter\expandafter\Expect
1805 \expandafter\expandafter\expandafter{%
1806 \bitsetGetSetBitList{abc}%
1807 }{}%
1808 \def\Test#1#2{%
1809 \bitsetSetBin{abc}{#1}%
1810 \expandafter\expandafter\expandafter\Expect
1811 \expandafter\expandafter\expandafter{%
1812 \bitsetGetSetBitList{abc}%
1813 }{#2}%
1814 }%
1815 \Test{0}{}%
1816 \Test{1}{0}%
1817 \Test{10}{1}%
1818 \Test{11}{0,1}%
1819 \Test{10110100}{2,4,5,7}%
1820 \Test{101101001010011}{0,1,4,6,9,11,12,14}%
1821 \end{qstest}
1822
1823 \begin{qstest}{GetDec}{GetDec}
1824 \TestGetterUndefined\bitsetGetDec
1825 \def\Test#1#2{%
1826 \RevSet{abc}{#1}%
1827 \noetex
1828 \begin{group}\expandafter\expandafter\expandafter\endgroup
1829 \noetex
1830 \expandafter\expandafter\expandafter\Expect
1831 \expandafter\expandafter\expandafter{%
1832 \bitsetGetDec{abc}%
1833 }{#2}%
1834 }%
1835 \Test{0}{0}%
1836 \Test{1}{1}%
1837 \Test{10}{2}%
1838 \Test{11}{3}%
1839 \Test{100}{4}%
1840 \Test{101}{5}%
1841 \Test{110}{6}%
1842 \Test{111}{7}%
1843 \Test{1000}{8}%
1844 \Test{000111}{7}%
1845 \Test{1111111111111111%
1846 1111111111111111}{2147483647}%
1847 \Test{000111111111111111%
1848 1111111111111111}{2147483647}%
1849 \Test{1000000000000000%
1850 0000000000000000}{2147483648}%
1851 \Test{1000000000000000%
1852 0000000000000000}{4294967296}%
1853 \Test{000100000000000000%
1854 0000000000000000}{4294967296}%

```

```

1855 \Test{1100000000000000%
1856      00000000000000011}{6442450947}%
1857 \end{qstest}
1858
1859 \begin{qstest}{Clear}{Clear}
1860 \def\Test#1#2#3{%
1861     \RevSet{abc}{#1}%
1862     \bitsetClear{abc}{#2}%
1863     \Expect*{\BS@abc}*{\Reverse#3!!}%
1864 }%
1865 \bitsetClear{abc}{2}%
1866 \RevCheck{abc}{0}%
1867 \TestErrorNegativeIndex{-1}{\bitsetClear{abc}{-1}}%
1868 \RevCheck{abc}{0}%
1869 \Test{0}{0}{0}%
1870 \Test{1}{0}{0}%
1871 \Test{111}{1}{101}%
1872 \Test{111}{30}{111}%
1873 \Test{0000111}{5}{0000111}% 111 would also be ok
1874 \Test{10000111}{5}{10000111}%
1875 \Test{1001001}{3}{1000001}%
1876 \end{qstest}
1877
1878 \begin{qstest}{Set}{Set}
1879 \def\Test#1#2#3{%
1880     \RevSet{abc}{#1}%
1881     \bitsetSet{abc}{#2}%
1882     \Expect*{\BS@abc}*{\Reverse#3!!}%
1883 }%
1884 \bitsetSet{abc}{2}%
1885 \RevCheck{abc}{100}%
1886 \TestErrorNegativeIndex{-1}{\bitsetSet{abc}{-1}}%
1887 \RevCheck{abc}{100}%
1888 \Test{0}{0}{1}%
1889 \Test{1}{0}{1}%
1890 \Test{100}{1}{110}%
1891 \Test{111}{1}{111}%
1892 \Test{11}{1}{11}%
1893 \Test{11}{2}{111}%
1894 \Test{11}{3}{1011}%
1895 \Test{111}{10}{1000000111}%
1896 \Test{0000111}{5}{0100111}% 100111 would also be ok
1897 \Test{10000111}{5}{10100111}%
1898 \Test{1000001}{3}{1001001}%
1899 \Test{1001001}{3}{1001001}%
1900 \end{qstest}
1901
1902 \begin{qstest}{Flip}{Flip}
1903 \def\Test#1#2#3{%
1904     \RevSet{abc}{#1}%
1905     \bitsetFlip{abc}{#2}%
1906     \Expect*{\BS@abc}*{\Reverse#3!!}%
1907 }%
1908 \bitsetFlip{abc}{2}%
1909 \RevCheck{abc}{100}%
1910 \TestErrorNegativeIndex{-1}{\bitsetFlip{abc}{-1}}%
1911 \RevCheck{abc}{100}%
1912 \Test{0}{0}{1}%
1913 \Test{1}{0}{0}%
1914 \Test{0}{2}{100}%
1915 \Test{100}{1}{110}%
1916 \Test{111}{1}{101}%

```

```

1917 \Test{11}{1}{1}%
1918 \Test{11}{2}{111}%
1919 \Test{11}{3}{1011}%
1920 \Test{111}{10}{1000000111}%
1921 \Test{0000111}{5}{0100111}% 100111 would also be ok
1922 \Test{10000111}{5}{10100111}%
1923 \Test{1000001}{3}{1001001}%
1924 \Test{1001001}{3}{1000001}%
1925 \Test{11111}{2}{11011}%
1926 \end{qstest}
1927
1928 \begin{qstest}{SetValue}{SetValue}
1929 \def\Test#1#2{%
1930   \TestError{Invalid bit value (#2) not in range 0..1}{%
1931     \bitsetSetValue{abc}{#1}{#2}%
1932   }%
1933 }%
1934 \Test{0}{-1}%
1935 \Test{0}{2}%
1936 \Test{0}{10}%
1937 \def\Test#1#2#3{%
1938   \let\BS@abc\@undefined
1939   \bitsetSetValue{abc}{#1}{#2}%
1940   \bitsetSetBin{result}{#3}%
1941   \Expect*{\BS@abc}*{\BS@result}%
1942 }%
1943 \Test{0}{0}{0}%
1944 \Test{0}{1}{1}%
1945 \Test{1}{0}{0}%
1946 \Test{1}{1}{10}%
1947 \def\Test#1#2#3#4{%
1948   \bitsetSetBin{abc}{#1}%
1949   \bitsetSetBin{result}{#4}%
1950   \bitsetSetValue{abc}{#2}{#3}%
1951   \Expect*{\BS@abc}*{\BS@result}%
1952 }%
1953 \Test{0}{0}{0}{0}%
1954 \Test{0}{0}{0}{0}%
1955 \Test{0}{0}{1}{1}%
1956 \Test{0}{1}{0}{0}%
1957 \Test{0}{1}{1}{10}%
1958 \Test{1010}{2}{1}{1110}%
1959 \Test{1010}{4}{1}{11010}%
1960 \Test{1010}{6}{1}{1001010}%
1961 \Test{1010}{1}{0}{1000}%
1962 \Test{1010}{2}{0}{1010}%
1963 \Test{1010}{3}{0}{10}%
1964 \Test{1010}{4}{0}{1010}%
1965 \Test{1010}{6}{0}{1010}%
1966 \Test{1010}{2}{\csname iffalse\endcsname 0\else 1\fi}{1110}%
1967 \Test{1010}{1}{\csname iffalse\endcsname 1\else 0\fi}{1000}%
1968 \end{qstest}
1969
1970 \begin{qstest}{IsDefined}{IsDefined}
1971   \let\BS@abc\@undefined
1972   \Expect*{\bitsetIsDefined{abc}{true}{false}}{false}%
1973   \bitsetReset{abc}%
1974   \Expect*{\bitsetIsDefined{abc}{true}{false}}{true}%
1975 \end{qstest}
1976
1977 \begin{qstest}{IsEmpty}{IsEmpty}
1978   \let\BS@abc\@undefined

```

```

1979 \Expect*{\bitsetIsEmpty{abc}{true}{false}}{true}%
1980 \bitsetReset{abc}%
1981 \Expect*{\bitsetIsEmpty{abc}{true}{false}}{true}%
1982 \bitsetSet{abc}{1}%
1983 \Expect*{\bitsetIsEmpty{abc}{true}{false}}{false}%
1984 \end{qstest}
1985
1986 \begin{qstest}{Equals}{Equals}
1987   \def\Test#1#2#3{%
1988     \Expect*{\bitsetEquals{#1}{#2}{true}{false}}{#3}%
1989   }%
1990   \let\BS@abc\@undefined
1991   \Test{abc}{abc}{true}%
1992   \Test{abc}{foo}{true}%
1993   \Test{foo}{abc}{true}%
1994   \bitsetReset{abc}%
1995   \Test{abc}{abc}{true}%
1996   \Test{abc}{foo}{false}%
1997   \Test{foo}{abc}{false}%
1998   \bitsetReset{foo}%
1999   \Test{abc}{foo}{true}%
2000   \Test{foo}{abc}{true}%
2001   \bitsetSet{abc}{4}%
2002   \Test{abc}{foo}{false}%
2003   \Test{foo}{abc}{false}%
2004   \bitsetFlip{foo}{4}%
2005   \Test{abc}{foo}{true}%
2006   \Test{foo}{abc}{true}%
2007 \end{qstest}
2008
2009 \begin{qstest}{Intersects}{Intersects}
2010   \def\Test#1{%
2011     \Expect*{\bitsetIntersects{abc}{foo}{true}{false}}{#1}%
2012   }%
2013   \let\BS@abc\@undefined
2014   \let\BS@foo\@undefined
2015   \Test{false}%
2016   \Set{abc}{0}%
2017   \Test{false}%
2018   \Set{foo}{0}%
2019   \Test{false}%
2020   \let\BS@abc\@undefined
2021   \Test{false}%
2022   \Set{foo}{1}%
2023   \Test{false}%
2024   \Set{abc}{0}%
2025   \Test{false}%
2026   \Set{abc}{1}%
2027   \Test{true}%
2028   \let\BS@foo\@undefined
2029   \Test{false}%
2030   \Set{foo}{0}%
2031   \Test{false}%
2032   \def\Test#1#2#3{%
2033     \bitsetSetBin{abc}{#1}%
2034     \bitsetSetBin{foo}{#2}%
2035     \Expect*{\bitsetIntersects{abc}{foo}{true}{false}}{#3}%
2036   }%
2037   \Test{1010}{0101}{false}%
2038   \Test{0}{10}{false}%
2039   \Test{1}{11}{true}%
2040   \Test{11}{11}{true}%

```



```

2041 \Test{10}{1}{false}%
2042 \end{qstest}
2043
2044 \begin{qstest}{And/AndNot/Or/Xor}{And/AndNot/Or/Xor}
2045 \def\@Test#1#2#3#4#5{%
2046 \begin{group
2047 #5%
2048 \begin{group
2049 \let\BS@foo\@undefined
2050 \csname bitset#1\endcsname{abc}{foo}%
2051 \CheckUndef{foo}%
2052 \Check{abc}{#2}%
2053 \end{group
2054 \begin{group
2055 \bitsetReset{foo}%
2056 \csname bitset#1\endcsname{abc}{foo}%
2057 \Check{foo}{0}%
2058 \Check{abc}{#3}%
2059 \end{group
2060 \begin{group
2061 \def\BS@foo{0101}%
2062 \csname bitset#1\endcsname{abc}{foo}%
2063 \Check{foo}{0101}%
2064 \Check{abc}{#4}%
2065 \end{group
2066 \end{group
2067 }%
2068 \def\Test#1{%
2069 \def\Op{#1}%
2070 \Test@
2071 }%
2072 \def\Test@#1#2#3#4#5#6#7#8#9{%
2073 \@Test\Op{#1}{#2}{#3}{%
2074 \let\BS@abc\@undefined
2075 }%
2076 \@Test\Op{#4}{#5}{#6}{%
2077 \bitsetReset{abc}%
2078 }%
2079 \@Test\Op{#7}{#8}{#9}{%
2080 \def\BS@abc{1001}%
2081 }%
2082 }%
2083 \Test{And}%
2084 {0}{0}{0}%
2085 {0}{0}{0}%
2086 {0}{0}{0001}%
2087 \Test{AndNot}%
2088 {0}{0}{0}%
2089 {0}{0}{0}%
2090 {1001}{1001}{1}%
2091 \Test{Or}%
2092 {0}{0}{0101}%
2093 {0}{0}{0101}%
2094 {1001}{1001}{1101}%
2095 \Test{Xor}%
2096 {0}{0}{0101}%
2097 {0}{0}{0101}%
2098 {1001}{1001}{11}%
2099 \def\Test#1#2#3{%
2100 \bitsetSetBin{abc}{#1}%
2101 \bitsetSetBin{foo}{#2}%
2102 \csname bitset\Op\endcsname{abc}{foo}%

```

```

2103     \RevCheck{foo}{#2}%
2104     \RevCheck{abc}{#3}%
2105 }%
2106 \def\Op{And}%
2107 \Test{1}{111}{1}%
2108 \Test{111}{1}{1}%
2109 \Test{10}{111}{10}%
2110 \Test{111}{10}{10}%
2111 \Test{111}{1000}{0}%
2112 \Test{1000}{111}{0}%
2113 \def\Op{AndNot}%
2114 \Test{1010}{11}{1000}%
2115 \Test{100}{100}{0}%
2116 \Test{111}{1111}{0}%
2117 \Test{100}{111}{0}%
2118 \def\Op{Or}%
2119 \Test{0}{0}{0}%
2120 \Test{1}{0}{1}%
2121 \Test{0}{1}{1}%
2122 \Test{1}{1}{1}%
2123 \Test{1000}{10}{1010}%
2124 \Test{10}{1000}{1010}%
2125 \def\Op{Xor}%
2126 \Test{0}{0}{0}%
2127 \Test{1}{0}{1}%
2128 \Test{0}{1}{1}%
2129 \Test{1}{1}{0}%
2130 \Test{1000}{10}{1010}%
2131 \Test{10}{1000}{1010}%
2132 \Test    {110011001100}%
2133         {111000111000111}%
2134         {111110100001011}%
2135 \Test{111000111000111}%
2136         {110011001100}%
2137         {111110100001011}%
2138 \end{qstest}
2139
2140 \begin{qstest}{GetUndef}{GetUndef, GetBin, GetOct, GetHex}
2141   \def\TestUndef#1#2{%
2142     \let\BS@abc\@undefined
2143     \expandafter\expandafter\expandafter\Expect
2144     \expandafter\expandafter\expandafter{%
2145       \x{abc}{#1}%
2146     }{#2}%
2147   }%
2148   \let\x\bitsetGetBin
2149   \TestUndef{-1}{0}%
2150   \TestUndef{0}{0}%
2151   \TestUndef{1}{0}%
2152   \TestUndef{2}{00}%
2153   \TestUndef{8}{00000000}%
2154   \let\x\bitsetGetOct
2155   \TestUndef{-1}{0}%
2156   \TestUndef{0}{0}%
2157   \TestUndef{1}{0}%
2158   \TestUndef{2}{0}%
2159   \TestUndef{3}{0}%
2160   \TestUndef{4}{00}%
2161   \TestUndef{5}{00}%
2162   \TestUndef{6}{00}%
2163   \TestUndef{7}{000}%
2164   \TestUndef{8}{000}%

```

```

2165 \TestUndef{9}{000}%
2166 \TestUndef{10}{0000}%
2167 \let\x\bitsetGetHex
2168 \TestUndef{-1}{0}%
2169 \TestUndef{0}{0}%
2170 \TestUndef{1}{0}%
2171 \TestUndef{2}{0}%
2172 \TestUndef{3}{0}%
2173 \TestUndef{4}{0}%
2174 \TestUndef{5}{00}%
2175 \TestUndef{6}{00}%
2176 \TestUndef{7}{00}%
2177 \TestUndef{8}{00}%
2178 \TestUndef{9}{000}%
2179 \TestUndef{10}{000}%
2180 \TestUndef{12}{000}%
2181 \TestUndef{13}{0000}%
2182 \TestUndef{16}{0000}%
2183 \TestUndef{17}{00000}%
2184 \end{qstest}
2185
2186 \begin{qstest}{SetBin}{SetBin}
2187 \def\Test#1#2{%
2188 \let\BS@abc\@undefined
2189 \bitsetSetBin{abc}{#1}%
2190 \expandafter\Expect\expandafter{\BS@abc}{#2}%
2191 }%
2192 \Test{}{0}%
2193 \Test{0}{0}%
2194 \Test{1}{1}%
2195 \Test{10}{01}%
2196 \Test{11}{11}%
2197 \Test{010}{01}%
2198 \Test{011}{11}%
2199 \Test{0010}{01}%
2200 \Test{1010}{0101}%
2201 \end{qstest}
2202
2203 \begin{qstest}{SetOct}{SetOct}
2204 \def\Test#1#2{%
2205 \bitsetSetOct{abc}{#1}%
2206 \expandafter\Expect\expandafter{\BS@abc}{#2}%
2207 }%
2208 \Test{}{0}%
2209 \Test{0}{0}%
2210 \Test{000}{0}%
2211 \Test{1}{1}%
2212 \Test{001}{1}%
2213 \Test{010}{0001}%
2214 \Test{020}{00001}%
2215 \Test{42}{010001}%
2216 \Test{377}{11111111}%
2217 \Test{0377}{11111111}%
2218 \Test{76543210}{000100010110001101011111}%
2219 \Test{ 0 7 0 7 1 }{100111000111}%
2220 \end{qstest}
2221
2222 \begin{qstest}{SetHex}{SetHex}
2223 \def\Test#1#2{%
2224 \bitsetSetHex{abc}{#1}%
2225 \expandafter\Expect\expandafter{\BS@abc}{#2}%
2226 }%

```

```

2227 \Test{}{0}%
2228 \Test{0}{0}%
2229 \Test{000}{0}%
2230 \Test{1}{1}%
2231 \Test{001}{1}%
2232 \Test{010}{00001}%
2233 \Test{020}{000001}%
2234 \Test{42}{0100001}%
2235 \Test{3F}{111111}%
2236 \Test{03F}{111111}%
2237 \Test{43210}{0000100001001100001}%
2238 \Test{98765}{10100110111000011001}%
2239 \Test{FEDCBA}{010111010011101101111111}%
2240 \Test{ O F O F 1 }{1000111100001111}%
2241 \end{qstest}
2242
2243 \begin{qstest}{SetDec}{SetDec}
2244 \def\Test#1#2{%
2245     \bitsetSetDec{abc}{#1}%
2246     \expandafter\Expect\expandafter{\BS@abc}{#2}%
2247 }%
2248 \Test{}{0}%
2249 \Test{0}{0}%
2250 \Test{000}{0}%
2251 \Test{1}{1}%
2252 \Test{7}{111}%
2253 \Test{8}{0001}%
2254 \Test{001}{1}%
2255 \Test{010}{0101}%
2256 \Test{020}{00101}%
2257 \Test{53}{101011}%
2258 \Test{255}{11111111}%
2259 \Test{256}{000000001}%
2260 \Test{999999999}{11111111001001101011001110111}%
2261 \Test{1000000000}{000000000101001101011001110111}%
2262 \Test{4210987654}{0110000101001001011111101011111}%
2263 \Test{2147483647}{1111111111111111111111111111111}%
2264 \Test{2147483648}{00000000000000000000000000000001}%
2265 \end{qstest}
2266
2267 \begin{qstest}{GetBin}{GetBin}
2268 \def\TestUndef#1#2{%
2269     \let\BS@abc\@undefined
2270     \expandafter\expandafter\expandafter\Expect
2271     \expandafter\expandafter\expandafter{%
2272         \bitsetGetBin{abc}{#1}%
2273     }{#2}%
2274 }%
2275 \TestUndef{-1}{0}%
2276 \TestUndef{0}{0}%
2277 \TestUndef{1}{0}%
2278 \TestUndef{2}{00}%
2279 \TestUndef{8}{00000000}%
2280 \def\Test#1#2{%
2281     \bitsetSetBin{abc}{#2}%
2282     \expandafter\expandafter\expandafter\Expect
2283     \expandafter\expandafter\expandafter{%
2284         \bitsetGetBin{abc}{#1}%
2285     }{#2}%
2286 }%
2287 \Test{-1}{0}%
2288 \Test{0}{0}%

```

```

2289 \Test{1}{0}%
2290 \Test{1}{1}%
2291 \Test{2}{01}%
2292 \Test{2}{10}%
2293 \Test{3}{010}%
2294 \Test{2}{00}%
2295 \Test{2}{01}%
2296 \Test{8}{00101100}%
2297 \Test{2}{10101}%
2298 \Test{-100}{11011}%
2299 \end{qstest}
2300
2301 \begin{qstest}{GetOct}{GetOct}
2302 \def\Test#1#2#3{%
2303   \edef\x{\zap@space#1 \@empty}%
2304   \edef\x{\noexpand\bitsetSetBin{abc}{\x}}%
2305   \x
2306   \expandafter\expandafter\expandafter\Expect
2307   \expandafter\expandafter\expandafter{%
2308     \bitsetGetOct{abc}{#2}%
2309   }{#3}%
2310 }%
2311 \Test{111 110 101 100 011 010 001 000}{0}{76543210}%
2312 \Test{000 111}{0}{7}%
2313 \Test{101 000}{-1}{50}%
2314 \Test{111}{-1}{7}%
2315 \Test{111}{0}{7}%
2316 \Test{111}{1}{7}%
2317 \Test{111}{3}{7}%
2318 \Test{111}{4}{07}%
2319 \Test{111}{6}{07}%
2320 \Test{111}{7}{007}%
2321 \Test{111 010}{6}{72}%
2322 \Test{111 010}{7}{072}%
2323 \Test{011 111}{0}{37}%
2324 \Test{011 111}{6}{37}%
2325 \Test{011 111}{7}{037}%
2326 \Test{001 111}{0}{17}%
2327 \Test{001 111}{6}{17}%
2328 \Test{001 111}{7}{017}%
2329 \end{qstest}
2330
2331 \begin{qstest}{GetHex}{GetHex}
2332 \def\Test#1#2#3{%
2333   \bitsetSetBin{abc}{#1}%
2334   \expandafter\expandafter\expandafter\Expect
2335   \expandafter\expandafter\expandafter{%
2336     \bitsetGetHex{abc}{#2}%
2337   }{#3}%
2338 }%
2339 \Test{1111 1110 1101 1100 1011 1010 1001 1000}{0}{FEDCBA98}%
2340 \Test{0111 0110 0101 0100 0011 0010 0001 0000}{0}{76543210}%
2341 \Test{0000 1111}{0}{F}%
2342 \Test{0101 0000}{-1}{50}%
2343 \Test{1111}{-1}{F}%
2344 \Test{1111}{0}{F}%
2345 \Test{1111}{1}{F}%
2346 \Test{1111}{4}{F}%
2347 \Test{1111}{5}{0F}%
2348 \Test{1111}{8}{0F}%
2349 \Test{1111}{9}{00F}%
2350 \Test{1111 0010}{8}{F2}%

```

```

2351 \Test{1111 0010}{9}{0F2}%
2352 \Test{0111 1111}{0}{7F}%
2353 \Test{0111 1111}{8}{7F}%
2354 \Test{0111 1111}{9}{07F}%
2355 \Test{0011 1111}{0}{3F}%
2356 \Test{0011 1111}{8}{3F}%
2357 \Test{0011 1111}{9}{03F}%
2358 \Test{0001 1111}{0}{1F}%
2359 \Test{0001 1111}{8}{1F}%
2360 \Test{0001 1111}{9}{01F}%
2361 \end{qstest}
2362
2363 \begin{qstest}{Range}{Range}
2364 \TestError{%
2365     Wrong index numbers in range [9..8]\MessageBreak% hash-ok
2366     for clear/set/flip on bit set 'abc'.\MessageBreak
2367     The lower index exceeds the upper index.\MessageBreak
2368     Canceling the operation as error recovery%
2369 }{%
2370     \bitsetSetRange{abc}{9}{8}%
2371 }%
2372 \def\TestErrorNegInd#1#2#3#4#5#6{%
2373     \TestError{%
2374         Negative index in range [#2..#3]\MessageBreak % hash-ok
2375         for \string\bitset #1Range on bit set 'abc'.\MessageBreak
2376         Using [#4..#5] as error recovery% hash-ok
2377     }{%
2378         \csname bitset#1Range\endcsname{abc}{#2}{#3}%
2379         \global\let\BS@global\BS@abc
2380     }%
2381     \Check{global}{#6}%
2382 }%
2383 \Set{abc}{111}%
2384 \TestErrorNegInd{Clear}{-1}{0}{0}{0}{111}%
2385 \TestErrorNegInd{Clear}{0}{-1}{0}{0}{111}%
2386 \TestErrorNegInd{Clear}{-2}{2}{0}{2}{001}%
2387 \bitsetReset{abc}%
2388 \TestErrorNegInd{Set}{-1}{0}{0}{0}{0}%
2389 \TestErrorNegInd{Set}{0}{-1}{0}{0}{0}%
2390 \TestErrorNegInd{Set}{-2}{2}{0}{2}{11}%
2391 \Set{abc}{101}%
2392 \TestErrorNegInd{Flip}{-1}{0}{0}{0}{101}%
2393 \TestErrorNegInd{Flip}{0}{-1}{0}{0}{101}%
2394 \TestErrorNegInd{Flip}{-2}{2}{0}{2}{011}%
2395 \def\Test#1#2#3#4{%
2396     \bitsetSetBin{abc}{#1}%
2397     \csname bitset\TestOp Range\endcsname{abc}{#2}{#3}%
2398     \Expect*{\bitsetGetBin{abc}{0}}{#4}%
2399 }%
2400 \def\TestOp{Clear}%
2401 \Test{0}{0}{1}{0}%
2402 \Test{1111}{1}{2}{1101}%
2403 \Test{1111}{1}{3}{1001}%
2404 \Test{1111111100000000}{12}{14}{1100111100000000}%
2405 \def\TestOp{Set}%
2406 \Test{0}{0}{1}{1}%
2407 \Test{1000}{1}{2}{1010}%
2408 \Test{0}{1}{2}{10}%
2409 \Test{1}{12}{15}{111000000000001}%
2410 \Test{1111}{1}{3}{1111}%
2411 \Test{1000000000000000}{12}{14}{1011000000000000}%
2412 \def\TestOp{Flip}%

```



```

2475 \begin{qstest}{Profile: Set}{Profile: Set}
2476 \bitsetSet{abc}{4095}%
2477 \global\let\BS@global\BS@abc
2478 \end{qstest}
2479
2480 \begin{qstest}{Profile: Get}{Profile: Get}
2481 \edef\x{\bitsetGet{global}{4095}}%
2482 \end{qstest}
2483
2484 \begin{document}
2485 \end{document}
2486 </test2>

```

## 4 Installation

### 4.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

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

*TDS* refers to the standard “A Directory Structure for  $\text{\TeX}$  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- $\text{\TeX}$ :

```
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):

---

<sup>1</sup><http://ftp.ctan.org/tex-archive/>



```

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

```

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

## 4.4 Refresh file name databases

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

**plain- $\text{\TeX}$ :** Run `docstrip` and extract the files.

**$\text{\LaTeX}$ :** Generate the documentation.

If you insist on using  $\text{\LaTeX}$  for `docstrip` (really, `docstrip` does not need  $\text{\LaTeX}$ ), 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 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	
\#	1396
\%	1399
\:	1668
\@	1367, 1392
\@PackageError	133, 325, 881, 1036, 1049, 1493
\@Test	2045, 2073, 2076, 2079, 2428, 2434, 2435
\@ehc	135, 327, 883, 1041, 1063
\@empty	2303
\@firstofone	1556
\@gobble	1554
\@ne	1376, 1384
\@undefined	1419, 1420, 1421, 1675, 1803, 1938, 1971, 1978, 1990, 2013, 2014, 2020, 2028, 2049, 2074, 2142, 2188, 2269, 2429
\[	1397
\[	179, 332, 395, 528, 537, 599, 646, 649, 684, 687, 724, 727, 757, 759, 767, 1251, 1260, 1282, 1291, 1351, 1354, 1393, 1552
\{	1394
\}	1395
\]	1398
\_	1400
A	
\advance	1376, 1384, 1459
\AtEndDocument	1472
B	
\begin	1560, 1577, 1587, 1614, 1635, 1658, 1802, 1823, 1859, 1878, 1902, 1928, 1970, 1977, 1986, 2009, 2044, 2140, 2186, 2203, 2222, 2243, 2267, 2301, 2331, 2363, 2427, 2475, 2480, 2484
\BigIntCalcAdd	603, 612
\bigintcalcCmp	315
\BigIntCalcOdd	335
\bigintcalcSgn	312
\BigIntCalcShl	617, 624
\BigIntCalcShr	343
\bitset	1051, 2375
\BitSet@@@Range	1043, 1066, 1070
\BitSet@@@Set	941, 948, 983
\BitSet@@@CheckIndex	126, 130
\BitSet@@@Clear	887, 899
\BitSet@@@Flip	958, 970
\BitSet@@@Get	1087, 1096
\BitSet@@@GetBin	364, 367
\BitSet@@@GetDec	532, 536, 562
\BitSet@@@GetDecBig	611, 622
\BitSet@@@GetHex	436, 469
\BitSet@@@GetOct	422, 444
\BitSet@@@GetOctHex	419, 433, 503, 507
\BitSet@@@NextClearBit	1130, 1134
\BitSet@@@NextSetBit	1166, 1170
\BitSet@@@Range	1022, 1027, 1064, 1066
\BitSet@@@Set	927, 934
\BitSet@@@TestMode	87
\BitSet@AfterFi	114, 132, 138, 182, 341, 356, 370, 375, 386, 391, 396, 400, 408, 413, 446, 451, 471, 476, 509, 517, 529, 531, 1069, 1082, 1121, 1157, 1188, 1231, 1233, 1349
\BitSet@AfterFiFi	115, 220, 253, 539, 544, 548, 554, 601, 606, 610, 615, 730, 848, 910, 941, 983, 1107, 1140, 1142, 1176, 1178, 1193, 1197, 1199, 1252, 1254, 1261, 1263, 1283, 1285, 1292, 1294, 1352
\BitSet@AfterFiFiFi	116, 654, 658, 694, 698, 773, 778, 913, 918, 986, 991, 1204, 1210, 1355, 1357
\BitSet@And	634, 645
\BitSet@AndNot	672, 683
\BitSet@AtEnd	60, 61, 1364
\BitSet@Cardinality	1274, 1280
\BitSet@CheckIndex	124, 855, 858, 861, 868
\BitSet@Cleanup	848, 910, 1107, 1140, 1176, 1185
\BitSet@Clear	855, 870, 885, 999, 1013, 1052
\BitSet@Empty	99, 107, 158, 161, 163, 197, 200, 202, 208, 304, 307, 309, 427, 441, 445, 470, 639, 677, 750, 828, 840, 846, 889, 893, 901, 903, 909, 929, 939, 960, 964, 973, 981
\BitSet@ErrorInvalidBitValue	874, 880, 1017
\BitSet@Fi	113, 114, 115, 116, 141, 185, 231, 262, 345, 360, 376, 392, 404, 414, 456, 481, 518, 534, 558, 620, 663, 703, 734, 783, 852, 923, 946, 996, 1073, 1094, 1111, 1132, 1147, 1168, 1183, 1216, 1238, 1269, 1299, 1362
\BitSet@Fill	371, 384, 409, 511
\BitSet@FirstOfOne	100
\BitSet@FirstOfTwo	102, 119, 1304, 1307, 1309, 1318, 1325, 1330, 1349
\BitSet@Flip	861, 956, 1005
\BitSet@FromFirstHex	191, 249
\BitSet@FromFirstOct	188, 217

<code>\BitSet@FromHex</code> .....	261, 264	464, 465, 466, 467, 468, 483,
<code>\BitSet@FromOct</code> .....	230, 233	486, 487, 488, 489, 490, 491,
<code>\BitSet@Get</code> .....	1077, 1080	492, 493, 494, 495, 496, 497,
<code>\BitSet@GetDec</code> .....	523, 527	498, 499, 500, 501, 560, 565,
<code>\BitSet@GetDecBig</code> ....	596, 598, 623	566, 567, 568, 569, 570, 571,
<code>\BitSet@GetOctHex</code> ....	447, 472, 502	572, 573, 574, 575, 576, 577,
<code>\BitSet@GetSetBitList</code> ...	1225, 1229	578, 579, 580, 581, 582, 583,
<code>\BitSet@Gobble</code> .....		584, 585, 586, 587, 588, 589,
....	101, 809, 834, 875, 876, 1200	590, 591, 592, 593, 594, 886,
<code>\BitSet@GobbleSeven</code> ....	1207, 1221	893, 896, 957, 964, 967, 1021, 1025
<code>\BitSet@Hex[O..F]</code> .....	275	<code>\BitSet@TestMode</code> .....
<code>\BitSet@Hex[0000..1111]</code> .....	483	87, 1426
<code>\BitSet@IfUndefined</code> .....		<code>\BitSet@Xor</code> .....
....	117, 125, 147, 368, 522,	745, 756
707, 738, 786, 813, 1086, 1242,		<code>\BitSet@ZapSpace</code> ..
1273, 1302, 1307, 1324, 1325, 1327		105, 157, 196, 303
<code>\BitSet@Intersects</code> .....	1340, 1347	<code>\BitSet@Zero</code> .....
<code>\BitSet@Kill</code> .....	827, 837	164, 203,
<code>\BitSet@KillZeros</code> .....		209, 310, 313, 894, 965, 1308, 1315
....	161, 171, 200, 258, 307	<code>\bitsetAnd</code> .....
<code>\BitSet@MaxSize</code> .....	98, 315	7, 626
<code>\BitSet@N1073741824</code> .....	595	<code>\bitsetAndNot</code> .....
<code>\BitSet@N[1,2,4,...]</code> .....	560	7, 665
<code>\BitSet@NegativeIndex</code>	1029, 1032, 1048	<code>\bitsetCardinality</code>
<code>\BitSet@NextClearBit</code> ....	1115, 1118	8, 1271, 1636, 1641
<code>\BitSet@NextSetBit</code> .....		<code>\bitsetClear</code> ..
....	1151, 1154, 1226, 1235	7, 854, 1862, 1865, 1867
<code>\BitSet@NumBinFill</code> .....	397, 406	<code>\bitsetClearRange</code> .....
<code>\BitSet@NumBinRev</code> .....	378, 394	998
<code>\BitSet@Oct[000..111]</code> .....	458	<code>\bitsetEquals</code> .....
<code>\BitSet@Or</code> .....	714, 722	9, 1323, 1988
<code>\BitSet@Range</code> .....		<code>\BitSetError</code> ...
999, 1002, 1005, 1013, 1015, 1020		228, 244, 256, 268,
<code>\BitSet@Reverse</code> .....	167, 178, 212	339, 1083, 1123, 1159, 1671, 1673
<code>\BitSet@SecondOfTwo</code> .....	103,	<code>\bitsetFlip</code>
121, 1303, 1311, 1320, 1325,		860, 1905, 1908, 1910, 2004
1327, 1332, 1338, 1339, 1352, 1355		<code>\bitsetFlipRange</code> .....
<code>\BitSet@Set</code> .....		1004
..	858, 872, 925, 1002, 1015, 1055	<code>\bitsetGet</code> .....
<code>\BitSet@SetDec</code> .....	321, 333, 347	..
<code>\BitSet@SetDecBig</code> .....	317, 331	8, 1075, 1317, 1590, 1600, 2481
<code>\BitSet@SetOctHex</code> ....	188, 191, 193	<code>\bitsetGetBin</code> .....
<code>\BitSet@SetValue</code> .....	864, 867	6, 362,
<code>\BitSet@SetValueRange</code> ...	1008, 1011	2148, 2272, 2284, 2398, 2419, 2446
<code>\BitSet@ShiftLeft</code> ....	791, 796, 834	<code>\bitsetGetDec</code> .....
<code>\BitSet@ShiftRight</code> ....	809, 818, 823	7, 520, 1824, 1832
<code>\BitSet@Size</code> .....	1243, 1249	<code>\bitsetGetHex</code> .....
<code>\BitSet@Skip</code> .....	1127, 1163, 1186	430, 2167, 2336
<code>\BitSet@SkipContinue</code> .....		<code>\bitsetGetOct</code> .....
....	1189, 1194, 1197, 1200, 1218	416, 2154, 2308
<code>\BitSet@Space</code> ....	104, 158, 197,	<code>\bitsetGetSetBitList</code> .....
304, 540, 602, 804, 1093, 1122, 1158		..
<code>\BitSet@Temp</code> .....		8, 1222, 1806, 1812
..	155, 156, 158, 160, 161, 163,	<code>\bitsetIntersects</code>
167, 194, 195, 197, 199, 200,		9, 1337, 2011, 2035
202, 205, 206, 208, 212, 275,		<code>\bitsetIsDefined</code> ..
278, 279, 280, 281, 282, 283,		8, 1301, 1972, 1974
284, 285, 286, 287, 288, 289,		<code>\bitsetIsEmpty</code> ..
290, 291, 292, 293, 294, 295,		9, 418, 432, 627,
296, 297, 298, 299, 301, 302,		630, 666, 669, 706, 709, 737,
304, 306, 307, 309, 312, 315,		740, 789, 816, 1126, 1162, 1224,
317, 321, 458, 461, 462, 463,		1306, 1338, 1339, 1979, 1981, 1983
		<code>\bitsetLet</code> .....
		6, 146, 1563, 1569
		<code>\bitsetNextClearBit</code> ...
		8, 1113, 1665
		<code>\bitsetNextSetBit</code> ....
		8, 1149, 1666
		<code>\bitsetOr</code> .....
		7, 705
		<code>\bitsetQuery</code> .....
		9, 1316, 1595, 1601
		<code>\bitsetReset</code> .....
		6, 125,
		143, 148, 628, 631, 640, 667,
		678, 707, 738, 751, 787, 814,
		1578, 1580, 1583, 1685, 1973,
		1980, 1994, 1998, 2055, 2077, 2387
		<code>\bitsetSet</code> .....
		857,
		1881, 1884, 1886, 1982, 2001, 2476
		<code>\bitsetSetBin</code> ..
		6, 154, 1669, 1695,
		1705, 1756, 1767, 1778, 1789,
		1809, 1940, 1948, 1949, 2033,
		2034, 2100, 2101, 2189, 2281,
		2304, 2333, 2396, 2417, 2443, 2444
		<code>\bitsetSetDec</code> .....
		6, 300, 2245
		<code>\bitsetSetHex</code> .....
		190, 2224
		<code>\bitsetSetOct</code> .....
		187, 2205
		<code>\bitsetSetRange</code> .....
		1001, 2370
		<code>\bitsetSetValue</code>
		8, 863, 1931, 1939, 1950

<code>\bitsetSetValueRange</code> . . . . .	1007, 2418	203, 209, 211, 255, 259, 267,
<code>\bitsetShiftLeft</code> . . . . .	7, 785	270, 276, 310, 313, 316, 320,
<code>\bitsetShiftRight</code> . . . . .	812	381, 426, 440, 454, 459, 479,
<code>\bitsetSize</code> . . . . .	8, 1240, 1615, 1619	484, 524, 550, 555, 561, 595,
<code>\bitsetXor</code> . . . . .	7, 736	633, 635, 637, 639, 671, 673,
<code>\BS@abc</code> 1675, 1803, 1863, 1882, 1906,		675, 677, 710, 711, 713, 715,
1938, 1941, 1951, 1971, 1978,		717, 741, 742, 744, 746, 748,
1990, 2013, 2020, 2074, 2080,		750, 802, 805, 826, 828, 888,
2142, 2188, 2190, 2206, 2225,		892, 926, 928, 959, 963, 1088,
2246, 2269, 2379, 2429, 2431, 2477		1129, 1165, 1246, 1277, 1308,
<code>\BS@foo</code> . . . . .	2014, 2028, 2049, 2061	1328, 1329, 1341, 1343, 1406,
<code>\BS@global</code> . . . . .	2379, 2477	1442, 1527, 1536, 1544, 1547,
<code>\BS@result</code> . . . . .	1941, 1951	1593, 1966, 1967, 2050, 2056,
		2062, 2102, 2378, 2397, 2430, 2445
<b>C</b>		
<code>\catcode</code> . . . . .	3, 4, 5, 6, 7,	<code>\endinput</code> . . . . .
31, 32, 33, 34, 35, 36, 37, 38, 56,		27
58, 62, 64, 85, 1367, 1373, 1382,		<code>\endqstest</code> . . . . .
1392, 1393, 1394, 1395, 1396,		1463, 1468, 1477, 1482
1397, 1398, 1399, 1400, 1401, 1668		<code>\ETeXDisable</code> . . . . .
<code>\chardef</code> . . . . .	1426	1418, 1423, 1505
<code>\Check</code> . . . . .	1529,	<code>\ETeXEnable</code> . . . . .
1565, 1567, 1568, 1570, 1571,		1430, 1435, 1498
1573, 1574, 1579, 1581, 1584,		<code>\Expect</code> . . . . .
2052, 2057, 2058, 2063, 2064, 2381		1484, 1500, 1501,
<code>\CheckUndef</code> . . . . .		1509, 1510, 1520, 1525, 1534,
1519, 1532, 1561, 1562, 1564, 2051		1588, 1595, 1599, 1601, 1618,
<code>\Clear</code> . . . . .	1665, 1670, 1676, 1686,	1639, 1660, 1804, 1810, 1830,
1696, 1706, 1757, 1768, 1779, 1790		1863, 1882, 1906, 1941, 1951,
<code>\count@</code> . . . . .	1369, 1373,	1972, 1974, 1979, 1981, 1983,
1375, 1376, 1380, 1382, 1383, 1384		1988, 2011, 2035, 2143, 2190,
<code>\csname</code> . . . . .	8, 18, 39, 52, 55, 83,	2206, 2225, 2246, 2270, 2282,
89, 118, 144, 150, 151, 164, 166,		2306, 2334, 2398, 2419, 2431, 2446
203, 209, 211, 255, 259, 267,		<code>\ExpectBitSet</code> . . . . .
270, 276, 310, 313, 316, 320,		1524, 1530, 1541
381, 426, 440, 454, 459, 479,		
484, 524, 549, 555, 561, 595,		<b>H</b>
633, 635, 637, 639, 671, 673,		<code>\hbox</code> . . . . .
675, 677, 710, 711, 713, 715,		1480
717, 741, 742, 744, 746, 748,		
750, 802, 805, 826, 828, 888,		<b>I</b>
892, 926, 928, 959, 963, 1088,		<code>\ifcase</code> 9, 220, 236, 312, 335, 348, 797,
1129, 1165, 1246, 1277, 1308,		824, 869, 1012, 1034, 1192, 1602
1328, 1329, 1341, 1343, 1406,		<code>\ifcsname</code> . . . . .
1442, 1527, 1536, 1544, 1547,		1416, 1419, 1432
1593, 1966, 1967, 2050, 2056,		<code>\ifnum</code> . . . . .
2062, 2102, 2378, 2397, 2430, 2445		131, 315,
<code>\currentgrouplevel</code> . . . . .	1417, 1421, 1433	369, 385, 407, 508, 647, 650,
		652, 685, 689, 692, 723, 1028,
<b>D</b>		1031, 1067, 1081, 1119, 1155,
<code>\dimexpr</code> . . . . .	1449	1203, 1230, 1317, 1348, 1375, 1383
<code>\documentclass</code> . . . . .	1412	<code>\ifodd</code> . . . . .
		351
<b>E</b>		<code>\ifx</code> . . . . .
<code>\empty</code> . . . . .	12	10, 12, 18, 39, 47, 83, 89, 107,
<code>\end</code> . . . . .	1407, 1575, 1585, 1612, 1633,	118, 163, 172, 179, 202, 208,
1656, 1800, 1821, 1857, 1876,		218, 234, 250, 252, 255, 265,
1900, 1926, 1968, 1975, 1984,		267, 309, 332, 395, 445, 470,
2007, 2042, 2138, 2184, 2201,		528, 537, 538, 547, 599, 600,
2220, 2241, 2265, 2299, 2329,		609, 639, 646, 649, 677, 684,
2361, 2425, 2473, 2478, 2482, 2485		687, 724, 727, 750, 757, 758,
<code>\endcsname</code> . . . . .	8, 18, 39, 52, 55, 83,	759, 767, 769, 772, 839, 840,
89, 118, 144, 150, 151, 164, 166,		846, 893, 900, 901, 903, 909,
		912, 936, 939, 949, 964, 971,
		972, 973, 981, 985, 1052, 1055,
		1098, 1099, 1105, 1135, 1138,
		1171, 1174, 1187, 1250, 1251,
		1260, 1281, 1282, 1291, 1308,
		1328, 1351, 1354, 1442, 1536, 1552
		<code>\ignorespaces</code> . . . . .
		1480
		<code>\immediate</code> . . . . .
		20, 41
		<code>\IncludeTests</code> . . . . .
		1438
		<code>\input</code> . . . . .
		90, 91, 92, 1402
		<code>\IntCalcAdd</code> . . . . .
		513, 541, 551
		<code>\intcalcCmp</code> . . . . .
		1034



1898, 1899, 1903, 1912, 1913,	2451, 2452, 2453, 2454, 2455,
1914, 1915, 1916, 1917, 1918,	2456, 2457, 2458, 2460, 2461,
1919, 1920, 1921, 1922, 1923,	2462, 2463, 2464, 2465, 2466,
1924, 1925, 1929, 1934, 1935,	2467, 2468, 2469, 2470, 2471, 2472
1936, 1937, 1943, 1944, 1945,	\Test@ ..... 2070, 2072
1946, 1947, 1953, 1954, 1955,	\TestError 1489, 1515, 1930, 2364, 2373
1956, 1957, 1958, 1959, 1960,	\TestErrorNegativeIndex .....
1961, 1962, 1963, 1964, 1965,	..... 1514, 1867, 1886, 1910
1966, 1967, 1987, 1991, 1992,	\TestErrorNegInd .....
1993, 1995, 1996, 1997, 1999,	..... 2372, 2384, 2385, 2386,
2000, 2002, 2003, 2005, 2006,	2388, 2389, 2390, 2392, 2393, 2394
2010, 2015, 2017, 2019, 2021,	\TestGetterUndefined .....
2023, 2025, 2027, 2029, 2031,	..... 1518, 1615, 1636, 1824
2032, 2037, 2038, 2039, 2040,	\TestOp ..... 1662,
2041, 2068, 2083, 2087, 2091,	1665, 1666, 2397, 2400, 2405, 2412
2095, 2099, 2107, 2108, 2109,	\TestTime ..... 1445, 1458, 1459, 1460
2110, 2111, 2112, 2114, 2115,	\TestUndef 2141, 2149, 2150, 2151,
2116, 2117, 2119, 2120, 2121,	2152, 2153, 2155, 2156, 2157,
2122, 2123, 2124, 2126, 2127,	2158, 2159, 2160, 2161, 2162,
2128, 2129, 2130, 2131, 2132,	2163, 2164, 2165, 2166, 2168,
2135, 2187, 2192, 2193, 2194,	2169, 2170, 2171, 2172, 2173,
2195, 2196, 2197, 2198, 2199,	2174, 2175, 2176, 2177, 2178,
2200, 2204, 2208, 2209, 2210,	2179, 2180, 2181, 2182, 2183,
2211, 2212, 2213, 2214, 2215,	2268, 2275, 2276, 2277, 2278, 2279
2216, 2217, 2218, 2219, 2223,	\the ..... 56, 62, 1373, 1484, 1510
2227, 2228, 2229, 2230, 2231,	\theTest ..... 1509
2232, 2233, 2234, 2235, 2236,	\TimeDescription ... 1453, 1456, 1460
2237, 2238, 2239, 2240, 2244,	\TMP@EnsureCode .....
2248, 2249, 2250, 2251, 2252,	..... 59, 66, 67, 68, 69, 70, 71,
2253, 2254, 2255, 2256, 2257,	72, 73, 74, 75, 76, 77, 78, 79, 80, 81
2258, 2259, 2260, 2261, 2262,	\typeout ..... 1448
2263, 2264, 2280, 2287, 2288,	
2289, 2290, 2291, 2292, 2293,	U
2294, 2295, 2296, 2297, 2298,	\uccode ..... 800
2302, 2311, 2312, 2313, 2314,	\uppercase ..... 801
2315, 2316, 2317, 2318, 2319,	\usepackage ..... 1428, 1437
2320, 2321, 2322, 2323, 2324,	
2325, 2326, 2327, 2328, 2332,	W
2339, 2340, 2341, 2342, 2343,	\wd ..... 1484, 1510
2344, 2345, 2346, 2347, 2348,	\write ..... 20, 41
2349, 2350, 2351, 2352, 2353,	
2354, 2355, 2356, 2357, 2358,	X
2359, 2360, 2395, 2401, 2402,	\x ..... 8, 10, 12, 19,
2403, 2404, 2406, 2407, 2408,	23, 25, 40, 45, 52, 2145, 2148,
2409, 2410, 2411, 2413, 2414,	2154, 2167, 2303, 2304, 2305, 2481
2415, 2416, 2421, 2422, 2423,	
2424, 2433, 2437, 2438, 2439,	Z
2440, 2441, 2442, 2449, 2450,	\z@ ..... 1446
	\zap@space ..... 2303