

The `bitset` package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/09/28 v1.0

Abstract

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

Contents

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

| | | |
|----------|---|-----------|
| 2.8.2 | <code>\bitsetAndNot</code> | 23 |
| 2.8.3 | <code>\bitsetOr</code> | 24 |
| 2.8.4 | <code>\bitsetXor</code> | 25 |
| 2.8.5 | Shifting | 26 |
| 2.8.6 | <code>\bitsetShiftLeft</code> | 26 |
| 2.8.7 | <code>\bitsetShiftRight</code> | 26 |
| 2.9 | Bit manipulation | 27 |
| 2.9.1 | Clear operation | 28 |
| 2.9.2 | Set operation | 29 |
| 2.9.3 | Flip operation | 29 |
| 2.9.4 | Range operators | 30 |
| 2.10 | Bit retrieval | 32 |
| 2.10.1 | <code>\bitsetGet</code> | 32 |
| 2.10.2 | <code>\bitsetNextClearBit</code> , <code>\bitsetNextSetBit</code> | 32 |
| 2.10.3 | <code>\bitsetGetSetBitList</code> | 35 |
| 2.11 | Bit set properties | 35 |
| 2.12 | Queries | 36 |
| 3 | Test | 38 |
| 3.1 | Catcode checks for loading | 38 |
| 3.2 | Macro tests | 39 |
| 3.2.1 | Preamble | 39 |
| 3.2.2 | Time | 40 |
| 3.2.3 | Detection of unwanted space | 40 |
| 3.2.4 | Test macros | 40 |
| 3.2.5 | Test sets | 42 |
| 4 | Installation | 57 |
| 4.1 | Download | 57 |
| 4.2 | Bundle installation | 57 |
| 4.3 | Package installation | 57 |
| 4.4 | Refresh file name databases | 58 |
| 4.5 | Some details for the interested | 58 |
| 5 | History | 58 |
| | [2007/09/28 v1.0] | 58 |
| 6 | Index | 58 |

1 Documentation

1.1 Introduction

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

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

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

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

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

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

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

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

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

1.2 Glossary

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

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

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

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

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

1.3 Design principles

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

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

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

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

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

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

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

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

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

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

1.4 Operator overview

Miscellaneous (section 1.6.1)

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

Import (section 1.6.2)

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

Export^a (section 1.6.3)

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

Logical operators (section 1.6.4)

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

Shifting (section 1.6.5)

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

Bit manipulation (section 1.6.6)

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

Bit retrieval^a (section 1.6.7)

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

Bit set properties (section 1.6.8)

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

Queries^b (section 1.6.9)

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

^aMacros are expandable, full expansion by two steps.

^bMacros are expandable.

1.5 Package loading

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

```
\usepackage{bitset}
```

Also plain-T_EX is supported:

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

1.6 Operators

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

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

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

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

1.6.1 Miscellaneous

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

`\bitsetReset {\langle BitSet \rangle}`

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

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

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

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

1.6.2 Import

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

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

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

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

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

1.6.3 Export

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

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

```

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

```

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

`\bitsetGetDec {⟨BitSet⟩}`

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

1.6.4 Logical operators

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

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

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

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

`\bitsetOr {⟨BitSet A⟩} {⟨BitSet B⟩}`

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

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

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

1.6.5 Shifting

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

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

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

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

1.6.6 Bit manipulation

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

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

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

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

1.6.7 Bit retrieval

`\bitsetGet {⟨BitSet⟩} {⟨Index⟩}`

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

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

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

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

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

`\bitsetGetSetBitList {⟨BitSet⟩}`

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

1.6.8 Bit set properties

`\bitsetSize {⟨BitSet⟩}`

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

`\bitsetCardinality {⟨BitSet⟩}`

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

1.6.9 Queries

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

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

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

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

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

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

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

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

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

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

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

2 Implementation

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

```
0|[01]*1
1 ⟨*package⟩
```

2.1 Reload check and package identification

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

```
2 \begingroup
3 \catcode44 12 % ,
4 \catcode45 12 % -
5 \catcode46 12 % .
6 \catcode58 12 % :
7 \catcode64 11 % @
8 \catcode123 1 % {
9 \catcode125 2 % }
10 \expandafter\let\expandafter\x\csname ver@bitset.sty\endcsname
11 \ifx\x\relax % plain-TeX, first loading
12 \else
13 \def\empty{}%
14 \ifx\x\empty % LaTeX, first loading,
15 % variable is initialized, but \ProvidesPackage not yet seen
16 \else
17 \catcode35 6 % #
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     \aftergroup\endinput
27   \fi
28 \fi
29 \endgroup

```

Package identification:

```

30 \begingroup
31   \catcode35 6 % #
32   \catcode40 12 % (
33   \catcode41 12 % )
34   \catcode44 12 % ,
35   \catcode45 12 % -
36   \catcode46 12 % .
37   \catcode47 12 % /
38   \catcode58 12 % :
39   \catcode64 11 % @
40   \catcode91 12 % [
41   \catcode93 12 % ]
42   \catcode123 1 % {
43   \catcode125 2 % }
44   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
45     \def\x#1#2#3[#4]{\endgroup
46       \immediate\write-1{Package: #3 #4}%
47       \xdef#1{#4}%
48     }%
49   \else
50     \def\x#1#2[#3]{\endgroup
51       #2[#{#3}]%
52       \ifx#1\@undefined
53         \xdef#1{#3}%
54       \fi
55       \ifx#1\relax
56         \xdef#1{#3}%
57       \fi
58     }%
59   \fi
60 \expandafter\x\csname ver@bitset.sty\endcsname
61 \ProvidesPackage{bitset}%
62 [2007/09/28 v1.0 Data type bit set (H0)]

```

2.2 Catcodes

```

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

```

```

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

```

2.3 Package loading

```

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

```

2.4 Help macros

2.4.1 Number constant

```

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

```

2.4.2 General basic macros

```

\BitSet@Empty
119 \def\BitSet@Empty{}

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

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

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

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

```

\BitSet@Space

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

\BitSet@ZapSpace

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

2.4.3 Tail recursion

\BitSet@Fi

```
133 \let\BitSet@Fi\fi
```

\BitSet@AfterFi

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

\BitSet@AfterFiFi

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

\BitSet@AfterFiFiFi

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

2.4.4 Check macros

\BitSet@IfUndefined

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

\BitSet@CheckIndex

#1: continuation code

#2: BitSet

#3: Index

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

\BitSet@@CheckIndex

#1: plain Index

#2: BitSet

#3: continuation code

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

```

159      #3{#2}{#1}%
160    }%
161    \BitSet@Fi
162 }

```

2.5 Miscellaneous

\bitsetReset

```

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

```

\bitsetLet

```

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

```

2.6 Import

2.6.1 From binary number

\bitsetSetBin

```

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

```

\BitSet@KillZeros

```

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

```

\BitSet@Reverse

```

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

```

```

204     }%
205     \BitSet@Fi
206 }

```

2.6.2 From octal/hex number

\bitsetSetOct

```

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

```

\bitsetSetHex

```

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

```

\BitSet@SetOctHex

```

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

```

\BitSet@FromFirstOct

```

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

```

\BitSet@FromOct

```

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

```

\BitSet@FromFirstHex

```

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

```

\BitSet@FromHex

```

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

```

\BitSet@Hex[0..F]

```

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

```

```

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

```

2.6.3 From decimal number

\bitsetSetDec

```

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

```

\BitSet@SetDecBig

```

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

```



```

365 \BitSet@Fi
366 }

\BitSet@SetDec

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

```

2.7 Export

2.7.1 To binary number

```

\bitsetGetBin

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

\BitSet@@GetBin

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

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

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

```

```

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

```

```

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

```

2.7.2 To octal/hexadecimal number

```

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

```

```

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

```

```

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

```

\BitSet@Oct[000..111]

```

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

```

```

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

```

\BitSet@Hex[0000..1111]

```

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

```

```

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

```

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

```

\BitSet@@GetOctHex #1: plain min size

#2: digits * (bits per digit)

#3: bits per digit - 1

#4: bits per digit

```

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

```

2.7.3 To decimal number

\bitsetGetDec

```

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

```

\BitSet@GetDec

```

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

```

```

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

```

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

```

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

```

```

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

\BitSet@N1073741824

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

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

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

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

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

```

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 |

```

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

```

```

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

```

\BitSet@And

```

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

```

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 |

```

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

```

\BitSet@AndNot

```

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

```

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 |

```

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

```

\BitSet@Or

```

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

```



```

752     }%
753     \fi
754     \BitSet@Fi
755 }

```

2.8.4 \bitsetXor

\bitsetXor Decision table for \bitsetXor:

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

```

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

```

\BitSet@Xor

```

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

```

```

802 \fi
803 \BitSet@Fi
804 }

```

2.8.5 Shifting

2.8.6 \bitsetShiftLeft

\bitsetShiftLeft

```

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

```

\BitSet@ShiftLeft

```

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

```

2.8.7 \bitsetShiftRight

\bitsetShiftRight

```

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

```

\BitSet@ShiftRight

```

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

```

```

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

```

\BitSet@Kill

```

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

```

2.9 Bit manipulation

\bitsetClear

```

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

```

\bitsetSet

```

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

```

\bitsetFlip

```

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

```

\bitsetSetValue

```

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

```

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

```

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

```

```

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

```

```

\BitSet@ErrorInvalidBitValue #1: Wrong bit value

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

```

2.9.1 Clear operation

```

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

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

```

```

\BitSet@@Clear

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

```

2.9.2 Set operation

```

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

\BitSet@@Set

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

\BitSet@@@Set

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

```

2.9.3 Flip operation

```

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

\BitSet@@Flip

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

```

```

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

```

2.9.4 Range operators

\bitsetClearRange

```

1018 \def\bitsetClearRange{%
1019 \BitSet@Range\BitSet@Clear
1020 }

```

\bitsetSetRange

```

1021 \def\bitsetSetRange{%
1022 \BitSet@Range\BitSet@Set
1023 }

```

\bitsetFlipRange

```

1024 \def\bitsetFlipRange{%
1025 \BitSet@Range\BitSet@Flip
1026 }

```

\bitsetSetValueRange

```

1027 \def\bitsetSetValueRange#1#2#3#4{%
1028 \expandafter\expandafter\expandafter\BitSet@SetValueRange
1029 \intcalcNum{#4}!{#1}{#2}{#3}%
1030 }

```

\BitSet@SetValueRange

```

1031 \def\BitSet@SetValueRange#1!#2#3#4{%
1032 \ifcase#1 %
1033 \BitSet@Range\BitSet@Clear{#2}{#3}{#4}%
1034 \or
1035 \BitSet@Range\BitSet@Set{#2}{#3}{#4}%
1036 \else
1037 \BitSet@ErrorInvalidBitValue{#1}%
1038 \fi
1039 }

```

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

```
1040 \def\BitSet@Range#1#2#3#4{%
1041   \edef\BitSet@Temp{%
1042     \noexpand\BitSet@@Range\noexpand#1{#2}%
1043     \intcalNum{#3}!\intcalNum{#4}!%
1044   }%
1045   \BitSet@Temp
1046 }
```

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

```
1047 \def\BitSet@@Range#1#2#3!#4!{%
1048   \ifnum#3<0 %
1049     \BitSet@NegativeIndex#1{#2}#3!#4!0!#4!%
1050   \else
1051     \ifnum#4<0 %
1052       \BitSet@NegativeIndex#1{#2}#3!#4!#3!0!%
1053     \else
1054       \ifcase\intcalCmp{#3}{#4} %
1055       \or
1056         \@PackageError{bitset}{%
1057           Wrong index numbers in range [#3..#4]\MessageBreak% hash-ok
1058           for clear/set/flip on bit set '#2'.\MessageBreak
1059           The lower index exceeds the upper index.\MessageBreak
1060           Canceling the operation as error recovery%
1061         }\@ehc
1062       \else
1063         \BitSet@@@Range#3!#4!#1{#2}%
1064       \fi
1065     \fi
1066   \fi
1067 }
```

\BitSet@NegativeIndex

```
1068 \def\BitSet@NegativeIndex#1#2#3!#4!#5!#6!{%
1069   \@PackageError{bitset}{%
1070     Negative index in range [#3..#4]\MessageBreak % hash-ok
1071     for \string\bitset
1072     \ifx#1\BitSet@Clear
1073       Clear%
1074     \else
1075       \ifx#1\BitSet@Set
1076         Set%
1077       \else
1078         Flip%
1079       \fi
1080     \fi
1081     Range on bit set '#2'.\MessageBreak
1082     Using [#5..#6] as error recovery% hash-ok
1083   }\@ehc
1084   \BitSet@@@Range#1{#2}#5!#6!%
1085 }
```

\BitSet@@@Range

```
1086 \def\BitSet@@@Range#1!#2!#3#4{%
1087   \ifnum#1<#2 %
```

```

1088     #3{#4}{#1}%
1089     \BitSet@AfterFi{%
1090         \expandafter\expandafter\expandafter\BitSet@@@Range
1091         \IntCalcInc#1!!#2!#3{#4}%
1092     }%
1093     \BitSet@Fi
1094 }

```

2.10 Bit retrieval

2.10.1 \bitsetGet

\bitsetGet

```

1095 \def\bitsetGet#1#2{%
1096     \number
1097     \expandafter\expandafter\expandafter\BitSet@Get
1098     \intcalcNum{#2}!{#1}%
1099 }

```

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

```

1100 \def\BitSet@Get#1!#2{%
1101     \ifnum#1<0 %
1102         \BitSet@AfterFi{%
1103             0 \BitSetError:NegativeIndex%
1104         }%
1105     \else
1106         \BitSet@IfUndefined{#2}{0}{%
1107             \expandafter\expandafter\expandafter\BitSet@@Get
1108             \csname BS@#2\expandafter\endcsname
1109             \expandafter!\expandafter=%
1110             \expandafter{\expandafter}\expandafter{\expandafter}%
1111             \romannumeral\intcalcNum{#1}000!%
1112         }%
1113         \expandafter\BitSet@Space
1114     \BitSet@Fi
1115 }

```

\BitSet@@Get

```

1116 \def\BitSet@@Get#1#2=#3#4#5{%
1117     #3#4%
1118     \ifx#5!%
1119         \ifx#1!%
1120             0%
1121         \else
1122             #1%
1123         \fi
1124     \else
1125         \ifx#1!%
1126             0%
1127             \BitSet@AfterFiFi\BitSet@Cleanup
1128         \else
1129             \BitSet@@Get#2=%
1130         \fi
1131     \BitSet@Fi
1132 }

```

2.10.2 \bitsetNextClearBit, \bitsetNextSetBit

\bitsetNextClearBit

```

1133 \def\bitsetNextClearBit#1#2{%

```



```

1134 \number
1135 \expandafter\expandafter\expandafter\BitSet@NextClearBit
1136 \intcalcNum{#2}!{#1} %
1137 }

\BitSet@NextClearBit #1: Index
#2: BitSet
1138 \def\BitSet@NextClearBit#1!#2{%
1139 \ifnum#1<0 %
1140 \BitSet@NextClearBit0!{#2}%
1141 \BitSet@AfterFi{%
1142 \expandafter\BitSet@Space
1143 \expandafter\BitSetError:NegativeIndex\romannumeral0%
1144 }%
1145 \else
1146 \bitsetIsEmpty{#2}{#1}{%
1147 \expandafter\BitSet@Skip
1148 \number#1\expandafter\expandafter\expandafter!%
1149 \csname BS@#2\endcsname!!!!!!!!!=%
1150 {\BitSet@@NextClearBit#1!}%
1151 }%
1152 \BitSet@Fi
1153 }

\BitSet@@NextClearBit #1: index for next bit in #2
#2: next bit
1154 \def\BitSet@@NextClearBit#1!#2{%
1155 \ifx#2!%
1156 #1%
1157 \else
1158 \ifx#20%
1159 #1%
1160 \BitSet@AfterFiFi\BitSet@Cleanup
1161 \else
1162 \BitSet@AfterFiFi{%
1163 \expandafter\expandafter\expandafter\BitSet@@NextClearBit
1164 \IntCalcInc#1!!%
1165 }%
1166 \fi
1167 \BitSet@Fi
1168 }

\bitsetNextSetBit
1169 \def\bitsetNextSetBit#1#2{%
1170 \number
1171 \expandafter\expandafter\expandafter\BitSet@NextSetBit
1172 \intcalcNum{#2}!{#1} %
1173 }

\BitSet@NextSetBit #1: Index
#2: BitSet
1174 \def\BitSet@NextSetBit#1!#2{%
1175 \ifnum#1<0 %
1176 \BitSet@NextSetBit0!{#2}%
1177 \BitSet@AfterFi{%
1178 \expandafter\BitSet@Space
1179 \expandafter\BitSetError:NegativeIndex\romannumeral0%
1180 }%
1181 \else
1182 \bitsetIsEmpty{#2}{-1}{%
1183 \expandafter\BitSet@Skip
1184 \number#1\expandafter\expandafter\expandafter!%

```

```

1185     \csname BS@#2\endcsname!!!!!!!!!=%
1186     {\BitSet@@NextSetBit#1!}%
1187 }%
1188 \BitSet@Fi
1189 }

\BitSet@@NextSetBit #1: index for next bit in #2
#2: next bit
1190 \def\BitSet@@NextSetBit#1!#2{%
1191   \ifx#2!%
1192     -1%
1193   \else
1194     \ifx#21%
1195       #1%
1196       \BitSet@AfterFiFi\BitSet@Cleanup
1197     \else
1198       \BitSet@AfterFiFi{%
1199         \expandafter\expandafter\expandafter\BitSet@@NextSetBit
1200         \IntCalcInc#1!!%
1201       }%
1202     \fi
1203   \BitSet@Fi
1204 }

\BitSet@Cleanup
1205 \def\BitSet@Cleanup#1!{%

\BitSet@Skip #1: number of bits to skip
#2: bits
#3: continuation code
1206 \def\BitSet@Skip#1!#2{%
1207   \ifx#2!%
1208     \BitSet@AfterFi{%
1209       \BitSet@SkipContinue%
1210     }%
1211   \else
1212     \ifcase#1 %
1213       \BitSet@AfterFiFi{%
1214         \BitSet@SkipContinue#2%
1215       }%
1216     \or
1217       \BitSet@AfterFiFi\BitSet@SkipContinue
1218     \or
1219       \BitSet@AfterFiFi{%
1220         \expandafter\BitSet@SkipContinue\BitSet@Gobble
1221       }%
1222     \else
1223       \ifnum#1>8 %
1224         \BitSet@AfterFiFiFi{%
1225           \expandafter\BitSet@Skip
1226           \number\IntCalcSub#1!8!\expandafter!%
1227           \BitSet@GobbleSeven
1228         }%
1229       \else
1230         \BitSet@AfterFiFiFi{%
1231           \expandafter\expandafter\expandafter\BitSet@Skip
1232           \IntCalcDec#1!!%
1233         }%
1234       \fi
1235     \fi
1236   \BitSet@Fi
1237 }

```

```

\BitSet@SkipContinue #1: remaining bits
#2: continuation code
1238 \def\BitSet@SkipContinue#1!#2=#3{%
1239   #3#1!%
1240 }

\BitSet@GobbleSeven
1241 \def\BitSet@GobbleSeven#1#2#3#4#5#6#7{}
```

2.10.3 \bitsetGetSetBitList

```

\bitsetGetSetBitList It's just a wrapper for \bitsetNextSetBit.
1242 \def\bitsetGetSetBitList#1{%
1243   \romannumeral0%
1244   \bitsetIsEmpty{#1}{ }{%
1245     \expandafter\BitSet@GetSetBitList
1246     \number\BitSet@NextSetBit0!{#1}!{#1}!{%
1247   }%
1248 }

\BitSet@GetSetBitList #1: found index
#2: BitSet
#3: comma #4: result
1249 \def\BitSet@GetSetBitList#1!#2#3#4!{%
1250   \ifnum#1<0 %
1251     \BitSet@AfterFi{ #4}%
1252   \else
1253     \BitSet@AfterFi{%
1254       \expandafter\BitSet@GetSetBitList\number
1255       \expandafter\expandafter\expandafter\BitSet@NextSetBit
1256       \IntCalcInc#1!!{#2}!{#2},#4#3#1!%
1257     }%
1258     \BitSet@Fi
1259 }
```

2.11 Bit set properties

```

\bitsetSize
1260 \def\bitsetSize#1{%
1261   \number
1262   \BitSet@IfUndefined{#1}{0 }{%
1263     \expandafter\expandafter\expandafter\BitSet@Size
1264     \expandafter\expandafter\expandafter1%
1265     \expandafter\expandafter\expandafter!%
1266     \csname BS@#1\endcsname!0!%
1267   }%
1268 }

\BitSet@Size #1: counter
#2#3: bits
#4: result
1269 \def\BitSet@Size#1!#2#3!#4!{%
1270   \ifx#21%
1271     \ifx\#3\%
1272       \BitSet@AfterFiFi{#1 }%
1273     \else
1274       \BitSet@AfterFiFi{%
1275         \expandafter\expandafter\expandafter\BitSet@Size
1276         \IntCalcInc#1!!#3!#1!%
1277       }%

```

```

1278     \fi
1279   \else
1280     \ifx\\#3\\%
1281       \BitSet@AfterFiFi{#4 }%
1282     \else
1283       \BitSet@AfterFiFi{%
1284         \expandafter\expandafter\expandafter\BitSet@Size
1285         \IntCalcInc#1!!#3!#4!%
1286       }%
1287     \fi
1288   \fi
1289   \BitSet@Fi
1290 }

```

\bitsetCardinality

```

1291 \def\bitsetCardinality#1{%
1292   \number
1293   \BitSet@IfUndefined{#1}{0 }{%
1294     \expandafter\expandafter\expandafter\BitSet@Cardinality
1295     \expandafter\expandafter\expandafter0%
1296     \expandafter\expandafter\expandafter!%
1297     \csname BS@#1\endcsname!%
1298   }%
1299 }

```

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

```

1300 \def\BitSet@Cardinality#1!#2#3!{%
1301   \ifx#21%
1302     \ifx\\#3\\%
1303       \BitSet@AfterFiFi{\IntCalcInc#1! }%
1304     \else
1305       \BitSet@AfterFiFi{%
1306         \expandafter\expandafter\expandafter\BitSet@Cardinality
1307         \IntCalcInc#1!!#3!%
1308       }%
1309     \fi
1310   \else
1311     \ifx\\#3\\%
1312       \BitSet@AfterFiFi{#1 }%
1313     \else
1314       \BitSet@AfterFiFi{%
1315         \BitSet@Cardinality#1!#3!%
1316       }%
1317     \fi
1318   \fi
1319   \BitSet@Fi
1320 }

```

2.12 Queries

\bitsetIsDefined

```

1321 \def\bitsetIsDefined#1{%
1322   \BitSet@IfUndefined{#1}%
1323   \BitSet@SecondOfTwo
1324   \BitSet@FirstOfTwo
1325 }

```

\bitsetIsEmpty

```

1326 \def\bitsetIsEmpty#1{%
1327   \BitSet@IfUndefined{#1}\BitSet@FirstOfTwo{%

```

```

1328     \expandafter\ifx\csname BS@#1\endcsname\BitSet@Zero
1329     \expandafter\BitSet@FirstOfTwo
1330     \else
1331     \expandafter\BitSet@SecondOfTwo
1332     \fi
1333 }%
1334 }

\BitSet@Zero

1335 \def\BitSet@Zero{0}

\bitsetQuery

1336 \def\bitsetQuery#1#2{%
1337     \ifnum\bitsetGet{#1}{#2}=1 %
1338     \expandafter\BitSet@FirstOfTwo
1339     \else
1340     \expandafter\BitSet@SecondOfTwo
1341     \fi
1342 }

\bitsetEquals

1343 \def\bitsetEquals#1#2{%
1344     \BitSet@IfUndefined{#1}{%
1345     \BitSet@IfUndefined{#2}\BitSet@FirstOfTwo\BitSet@SecondOfTwo
1346     }{%
1347     \BitSet@IfUndefined{#2}\BitSet@SecondOfTwo{%
1348     \expandafter\ifx\csname BS@#1\endcsname\expandafter\endcsname
1349     \csname BS@#2\endcsname
1350     \expandafter\BitSet@FirstOfTwo
1351     \else
1352     \expandafter\BitSet@SecondOfTwo
1353     \fi
1354     }%
1355 }%
1356 }

\bitsetIntersects

1357 \def\bitsetIntersects#1#2{%
1358     \bitsetIsEmpty{#1}\BitSet@SecondOfTwo{%
1359     \bitsetIsEmpty{#2}\BitSet@SecondOfTwo{%
1360     \expandafter\expandafter\expandafter\BitSet@Intersects
1361     \csname BS@#1\endcsname\expandafter\expandafter\expandafter\endcsname
1362     \expandafter\expandafter\expandafter!%
1363     \csname BS@#2\endcsname!%
1364     }%
1365 }%
1366 }

\BitSet@Intersects

1367 \def\BitSet@Intersects#1#2!#3#4!{%
1368     \ifnum#1#3=11 %
1369     \BitSet@AfterFi\BitSet@FirstOfTwo
1370     \else
1371     \ifx\#2\%
1372     \BitSet@AfterFiFi\BitSet@SecondOfTwo
1373     \else
1374     \ifx\#4\%
1375     \BitSet@AfterFiFiFi\BitSet@SecondOfTwo
1376     \else
1377     \BitSet@AfterFiFiFiFi%
1378     \BitSet@Intersects#2!#4!%

```

```

1379        }%
1380     \fi
1381 \fi
1382 \BitSet@Fi
1383 }

1384 \BitSet@AtEnd
1385 \endpackage

```

3 Test

3.1 Catcode checks for loading

```

1386 \test1

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

```

```

1435 \ifnum\count@<#2\relax
1436 \advance\count@ 1 %
1437 \repeat
1438 }
1439 \expandafter\ifx\csname LoadCommand\endcsname\relax
1440 \def\LoadCommand{\input bitset.sty\relax}%
1441 \fi
1442 \def\Test{%
1443 \RangeCatcodeInvalid{0}{47}%
1444 \RangeCatcodeInvalid{58}{64}%
1445 \RangeCatcodeInvalid{91}{96}%
1446 \RangeCatcodeInvalid{123}{255}%
1447 \catcode'\@=12 %
1448 \catcode'\=0 %
1449 \catcode'\{=1 %
1450 \catcode'\}=2 %
1451 \catcode'\#=6 %
1452 \catcode'\[=12 %
1453 \catcode'\]=12 %
1454 \catcode'\%=14 %
1455 \catcode'\ =10 %
1456 \catcode13=5 %
1457 \LoadCommand
1458 \RestoreCatcodes
1459 }
1460 \Test
1461 \csname @@end\endcsname
1462 \end
1463 </test1>

```

3.2 Macro tests

3.2.1 Preamble

```

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

```

```

1493 \IncludeTests{*}
1494 \LogTests{log}{*}{*}
1495 \makeatletter

```

3.2.2 Time

```

1496 \begin{group}\expandafter\expandafter\expandafter\end{group}
1497 \expandafter\ifx\csname pdfresettimer\endcsname\relax
1498 \else
1499   \newcount\SummaryTime
1500   \newcount\TestTime
1501   \SummaryTime=\z@
1502   \newcommand*{\PrintTime}[2]{%
1503     \typeout{%
1504       [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
1505     }%
1506   }%
1507   \newcommand*{\StartTime}[1]{%
1508     \renewcommand*{\TimeDescription}{#1}%
1509     \pdfresettimer
1510   }%
1511   \newcommand*{\TimeDescription}{}%
1512   \newcommand*{\StopTime}{%
1513     \TestTime=\pdfelapsedtime
1514     \global\advance\SummaryTime\TestTime
1515     \PrintTime\TimeDescription\TestTime
1516   }%
1517   \let\saved@qstest\qstest
1518   \let\saved@endqstest\endqstest
1519   \def\qstest#1#2{%
1520     \saved@qstest{#1}{#2}%
1521     \StartTime{#1}%
1522   }%
1523   \def\endqstest{%
1524     \StopTime
1525     \saved@endqstest
1526   }%
1527   \AtEndDocument{%
1528     \PrintTime{summary}\SummaryTime
1529   }%
1530 \fi

```

3.2.3 Detection of unwanted space

```

1531 \let\orig@qstest\qstest
1532 \let\orig@endqstest\endqstest
1533 \def\qstest#1#2{%
1534   \orig@qstest{#1}{#2}%
1535   \setbox0\hbox\bgroup\begin{group}\ignorespaces
1536 }
1537 \def\endqstest{%
1538   \endgroup\egroup
1539   \Expect*{\the\wd0}{0.0pt}%
1540   \orig@endqstest
1541 }

```

3.2.4 Test macros

```

1542 \newcounter{Test}
1543
1544 \def\TestError#1#2{%
1545   \begin{group}
1546     \setcounter{Test}{0}%
1547     \sbox0{%
1548       \def\@PackageError##1##2##3{%

```



```

1549         \stepcounter{Test}%
1550         \begingroup
1551         \let\MessageBreak\relax
1552 <*noetex>
1553         \ETEXEnable
1554 </noetex>
1555         \Expect{##1}{bitset}%
1556         \Expect*{##2}*{#1}%
1557         \endgroup
1558     }%
1559 <*noetex>
1560         \ETEXDisable
1561 </noetex>
1562     #2%
1563 }%
1564 \Expect*{\theTest}{1}%
1565 \Expect*{\the\wd0}{0.0pt}%
1566 \endgroup
1567 }
1568
1569 \def\TestErrorNegativeIndex#1#2{%
1570 \TestError{Invalid negative index (#1)}{#2}%
1571 }
1572
1573 \def\TestGetterUndefined#1{%
1574 \CheckUndef{dummy}%
1575 \expandafter\expandafter\expandafter\Expect
1576 \expandafter\expandafter\expandafter{#1{dummy}}{0}%
1577 }
1578
1579 \def\ExpectBitSet#1#2{%
1580 \expandafter\expandafter\expandafter\Expect
1581 \expandafter\expandafter\expandafter
1582 {\csname BS@#1\endcsname}*{#2}%
1583 }
1584 \def\Check#1#2{%
1585 \ExpectBitSet{#1}{#2}%
1586 }
1587 \def\CheckUndef#1{%
1588 \begingroup
1589 \Expect*{%
1590 \expandafter
1591 \ifx\csname BS@#1\endcsname\relax true\else false\fi
1592 }{true}%
1593 \endgroup
1594 }
1595 \def\RevCheck#1#2{%
1596 \ExpectBitSet{#1}{\Reverse#2!!}%
1597 }
1598 \def\Set#1#2{%
1599 \expandafter\def\csname BS@#1\endcsname{#2}%
1600 }
1601 \def\RevSet#1#2{%
1602 \expandafter\edef\csname BS@#1\endcsname{%
1603 \Reverse#2!!%
1604 }%
1605 }
1606 \def\Reverse#1#2!#3!{%
1607 \ifx\#2\%
1608 #1#3%
1609 \expandafter\@gobble
1610 \else

```

```

1611     \expandafter\@firstofone
1612 \fi
1613 {\Reverse#2!#1#3!}%
1614 }

```

3.2.5 Test sets

```

1615 \begin{qstest}{Let}{Let}
1616   \CheckUndef{abc}%
1617   \CheckUndef{xyz}%
1618   \bitsetLet{xyz}{abc}%
1619   \CheckUndef{abc}%
1620   \Check{xyz}{0}%
1621   \Set{abc}{1}%
1622   \Check{abc}{1}%
1623   \Check{xyz}{0}%
1624   \bitsetLet{xyz}{abc}%
1625   \Check{abc}{1}%
1626   \Check{xyz}{1}%
1627   \Set{xyz}{11}%
1628   \Check{abc}{1}%
1629   \Check{xyz}{11}%
1630 \end{qstest}
1631
1632 \begin{qstest}{Reset}{Reset}
1633   \bitsetReset{xyz}%
1634   \Check{xyz}{0}%
1635   \bitsetReset{abc}%
1636   \Check{abc}{0}%
1637   \Set{abc}{10101}%
1638   \bitsetReset{abc}%
1639   \Check{abc}{0}%
1640 \end{qstest}
1641
1642 \begin{qstest}{Get/Query}{Get/Query}
1643   \expandafter\expandafter\expandafter\Expect
1644   \expandafter\expandafter\expandafter{%
1645     \bitsetGet{dummy}{0}%
1646   }{0}%
1647   \begingroup
1648     \expandafter\def\csname BitSetError:NegativeIndex\endcsname{}%
1649     \Set{abc}{1}%
1650     \Expect*{\bitsetQuery{abc}{-1}{true}{false}}{false}%
1651   \endgroup
1652   \def\Test#1#2#3{%
1653     \Set{abc}{#1}%
1654     \expandafter\expandafter\expandafter\Expect
1655     \expandafter\expandafter\expandafter{\bitsetGet{abc}{#2}}{#3}%
1656     \Expect*{\bitsetQuery{abc}{#2}{true}{false}}%
1657     *{\ifcase#3 false\or true\else error\fi}%
1658   }%
1659   \Test{1}{100}{0}%
1660   \Test{0}{0}{0}%
1661   \Test{1}{0}{1}%
1662   \Test{11}{1}{1}%
1663   \Test{111}{1}{1}%
1664   \Test{101}{1}{0}%
1665   \Test{101}{2}{1}%
1666   \Test{10100110011}{10}{1}%
1667 \end{qstest}
1668
1669 \begin{qstest}{Size}{Size}
1670   \TestGetterUndefined\bitsetSize
1671   \def\Test#1#2{%

```

```

1672 \Set{abc}{#1}%
1673 \expandafter\expandafter\expandafter\Expect
1674 \expandafter\expandafter\expandafter{\bitsetSize{abc}}{#2}%
1675 }%
1676 \Test{0}{0}%
1677 \Test{1}{1}%
1678 \Test{00}{0}%
1679 \Test{0000000}{0}%
1680 \Test{10}{1}%
1681 \Test{01}{2}%
1682 \Test{11}{2}%
1683 \Test{010}{2}%
1684 \Test{011}{3}%
1685 \Test{100110011}{9}%
1686 \Test{0000011111000001111100000}{20}%
1687 \Test{00000000000000000000000001111111111111111111}{45}%
1688 \end{qstest}
1689
1690 \begin{qstest}{Cardinality}{Cardinality}
1691 \TestGetterUndefined\bitsetCardinality
1692 \def\Test#1#2{%
1693     \Set{abc}{#1}%
1694     \expandafter\expandafter\expandafter\Expect
1695     \expandafter\expandafter\expandafter{%
1696         \bitsetCardinality{abc}%
1697     }{#2}%
1698 }%
1699 \Test{0}{0}%
1700 \Test{1}{1}%
1701 \Test{00}{0}%
1702 \Test{0000000}{0}%
1703 \Test{10}{1}%
1704 \Test{01}{1}%
1705 \Test{11}{2}%
1706 \Test{010}{1}%
1707 \Test{011}{2}%
1708 \Test{100110011}{5}%
1709 \Test{0000011111000001111100000}{10}%
1710 \Test{00000000000000000000000001111111111111111111}{20}%
1711 \end{qstest}
1712
1713 \begin{qstest}{NextClearBit/NextSetBit}{NextClearBit/NextSetBit}
1714 \def\Test#1#2{%
1715     \expandafter\expandafter\expandafter\Expect
1716     \expandafter\expandafter\expandafter{%
1717         \TestOp{abc}{#1}%
1718     }{#2}%
1719 }%
1720 \def\CLEAR{\let\TestOp\bitsetNextClearBit}%
1721 \def\SET{\let\TestOp\bitsetNextSetBit}%
1722 \begingroup
1723 \catcode'\:=11 %
1724 \bitsetSetBin{abc}{1}%
1725 \CLEAR
1726 \Test{-1}{1\BitsetError:NegativeIndex}%
1727 \SET
1728 \Test{-1}{0\BitsetError:NegativeIndex}%
1729 \endgroup
1730 \let\BS@abc \@undefined
1731 \CLEAR
1732 \Test{0}{0}%
1733 \Test{1}{1}%

```

```

1734 \Test{2}{2}%
1735 \Test{100}{100}%
1736 \Set
1737 \Test{0}{-1}%
1738 \Test{1}{-1}%
1739 \Test{100}{-1}%
1740 \bitsetReset{abc}%
1741 \Clear
1742 \Test{0}{0}%
1743 \Test{1}{1}%
1744 \Test{2}{2}%
1745 \Test{100}{100}%
1746 \Set
1747 \Test{0}{-1}%
1748 \Test{1}{-1}%
1749 \Test{100}{-1}%
1750 \bitsetSetBin{abc}{1}%
1751 \Clear
1752 \Test{0}{1}%
1753 \Test{1}{1}%
1754 \Test{2}{2}%
1755 \Test{100}{100}%
1756 \Set
1757 \Test{0}{0}%
1758 \Test{1}{-1}%
1759 \Test{100}{-1}%
1760 \bitsetSetBin{abc}{111000111000111000111}%
1761 \Clear
1762 \Test{0}{3}%
1763 \Test{1}{3}%
1764 \Test{2}{3}%
1765 \Test{3}{3}%
1766 \Test{4}{4}%
1767 \Test{5}{5}%
1768 \Test{6}{9}%
1769 \Test{7}{9}%
1770 \Test{8}{9}%
1771 \Test{9}{9}%
1772 \Test{10}{10}%
1773 \Test{11}{11}%
1774 \Test{12}{15}%
1775 \Test{13}{15}%
1776 \Test{14}{15}%
1777 \Test{15}{15}%
1778 \Test{16}{16}%
1779 \Test{17}{17}%
1780 \Test{18}{21}%
1781 \Test{19}{21}%
1782 \Test{20}{21}%
1783 \Test{21}{21}%
1784 \Test{22}{22}%
1785 \Test{100}{100}%
1786 \Set
1787 \Test{0}{0}%
1788 \Test{1}{1}%
1789 \Test{2}{2}%
1790 \Test{3}{6}%
1791 \Test{4}{6}%
1792 \Test{5}{6}%
1793 \Test{6}{6}%
1794 \Test{7}{7}%
1795 \Test{8}{8}%

```

```

1796 \Test{9}{12}%
1797 \Test{10}{12}%
1798 \Test{11}{12}%
1799 \Test{12}{12}%
1800 \Test{13}{13}%
1801 \Test{14}{14}%
1802 \Test{15}{18}%
1803 \Test{16}{18}%
1804 \Test{17}{18}%
1805 \Test{18}{18}%
1806 \Test{19}{19}%
1807 \Test{20}{20}%
1808 \Test{21}{-1}%
1809 \Test{22}{-1}%
1810 \Test{100}{-1}%
1811 \bitsetSetBin{abc}{1111111}%
1812 \Clear
1813 \Test{6}{7}%
1814 \Test{7}{7}%
1815 \Test{8}{8}%
1816 \Test{100}{100}%
1817 \Set
1818 \Test{6}{6}%
1819 \Test{7}{-1}%
1820 \Test{8}{-1}%
1821 \Test{100}{-1}%
1822 \bitsetSetBin{abc}{11111111}%
1823 \Clear
1824 \Test{7}{8}%
1825 \Test{8}{8}%
1826 \Test{9}{9}%
1827 \Test{100}{100}%
1828 \Set
1829 \Test{7}{7}%
1830 \Test{8}{-1}%
1831 \Test{9}{-1}%
1832 \Test{100}{-1}%
1833 \bitsetSetBin{abc}{111111111}%
1834 \Clear
1835 \Test{8}{9}%
1836 \Test{9}{9}%
1837 \Test{10}{10}%
1838 \Test{100}{100}%
1839 \Set
1840 \Test{8}{8}%
1841 \Test{9}{-1}%
1842 \Test{10}{-1}%
1843 \Test{100}{-1}%
1844 \bitsetSetBin{abc}{1111111111}%
1845 \Clear
1846 \Test{9}{10}%
1847 \Test{10}{10}%
1848 \Test{11}{11}%
1849 \Test{100}{100}%
1850 \Set
1851 \Test{9}{9}%
1852 \Test{10}{-1}%
1853 \Test{11}{-1}%
1854 \Test{100}{-1}%
1855 \end{qstest}
1856
1857 \begin{qstest}{GetSetBitList}{GetSetBitList}

```

```

1858 \let\BS@abc\@undefined
1859 \expandafter\expandafter\expandafter\Expect
1860 \expandafter\expandafter\expandafter{%
1861   \bitsetGetSetBitList{abc}%
1862 }{}%
1863 \def\Test#1#2{%
1864   \bitsetSetBin{abc}{#1}%
1865   \expandafter\expandafter\expandafter\Expect
1866   \expandafter\expandafter\expandafter{%
1867     \bitsetGetSetBitList{abc}%
1868   }{#2}%
1869 }%
1870 \Test{0}{}%
1871 \Test{1}{0}%
1872 \Test{10}{1}%
1873 \Test{11}{0,1}%
1874 \Test{10110100}{2,4,5,7}%
1875 \Test{101101001010011}{0,1,4,6,9,11,12,14}%
1876 \end{qstest}
1877
1878 \begin{qstest}{GetDec}{GetDec}
1879   \TestGetterUndefined\bitsetGetDec
1880   \def\Test#1#2{%
1881     \RevSet{abc}{#1}%
1882   }*noetex
1883   \begin{group}\expandafter\expandafter\expandafter\endgroup
1884   \noetex
1885   \expandafter\expandafter\expandafter\Expect
1886   \expandafter\expandafter\expandafter{%
1887     \bitsetGetDec{abc}%
1888   }{#2}%
1889 }%
1890 \Test{0}{0}%
1891 \Test{1}{1}%
1892 \Test{10}{2}%
1893 \Test{11}{3}%
1894 \Test{100}{4}%
1895 \Test{101}{5}%
1896 \Test{110}{6}%
1897 \Test{111}{7}%
1898 \Test{1000}{8}%
1899 \Test{000111}{7}%
1900 \Test{1111111111111111}%
1901   1111111111111111}{2147483647}%
1902 \Test{0001111111111111}%
1903   1111111111111111}{2147483647}%
1904 \Test{1000000000000000}%
1905   0000000000000000}{2147483648}%
1906 \Test{1000000000000000}%
1907   0000000000000000}{4294967296}%
1908 \Test{0001000000000000}%
1909   0000000000000000}{4294967296}%
1910 \Test{1100000000000000}%
1911   0000000000000011}{6442450947}%
1912 \end{qstest}
1913
1914 \begin{qstest}{Clear}{Clear}
1915   \def\Test#1#2#3{%
1916     \RevSet{abc}{#1}%
1917     \bitsetClear{abc}{#2}%
1918     \Expect*{\BS@abc}*{\Reverse#3!!}%
1919   }%

```

```

1920 \bitsetClear{abc}{2}%
1921 \RevCheck{abc}{0}%
1922 \TestErrorNegativeIndex{-1}{\bitsetClear{abc}{-1}}%
1923 \RevCheck{abc}{0}%
1924 \Test{0}{0}{0}%
1925 \Test{1}{0}{0}%
1926 \Test{111}{1}{101}%
1927 \Test{111}{30}{111}%
1928 \Test{0000111}{5}{0000111}% 111 would also be ok
1929 \Test{10000111}{5}{10000111}%
1930 \Test{1001001}{3}{1000001}%
1931 \end{qstest}
1932
1933 \begin{qstest}{Set}{Set}
1934 \def\Test#1#2#3{%
1935   \RevSet{abc}{#1}%
1936   \bitsetSet{abc}{#2}%
1937   \Expect*{\BS@abc}*{\Reverse#3!!}%
1938 }%
1939 \bitsetSet{abc}{2}%
1940 \RevCheck{abc}{100}%
1941 \TestErrorNegativeIndex{-1}{\bitsetSet{abc}{-1}}%
1942 \RevCheck{abc}{100}%
1943 \Test{0}{0}{1}%
1944 \Test{1}{0}{1}%
1945 \Test{100}{1}{110}%
1946 \Test{111}{1}{111}%
1947 \Test{11}{1}{11}%
1948 \Test{11}{2}{111}%
1949 \Test{11}{3}{1011}%
1950 \Test{111}{10}{1000000111}%
1951 \Test{0000111}{5}{0100111}% 100111 would also be ok
1952 \Test{10000111}{5}{10100111}%
1953 \Test{1000001}{3}{1001001}%
1954 \Test{1001001}{3}{1001001}%
1955 \end{qstest}
1956
1957 \begin{qstest}{Flip}{Flip}
1958 \def\Test#1#2#3{%
1959   \RevSet{abc}{#1}%
1960   \bitsetFlip{abc}{#2}%
1961   \Expect*{\BS@abc}*{\Reverse#3!!}%
1962 }%
1963 \bitsetFlip{abc}{2}%
1964 \RevCheck{abc}{100}%
1965 \TestErrorNegativeIndex{-1}{\bitsetFlip{abc}{-1}}%
1966 \RevCheck{abc}{100}%
1967 \Test{0}{0}{1}%
1968 \Test{1}{0}{0}%
1969 \Test{0}{2}{100}%
1970 \Test{100}{1}{110}%
1971 \Test{111}{1}{101}%
1972 \Test{11}{1}{1}%
1973 \Test{11}{2}{111}%
1974 \Test{11}{3}{1011}%
1975 \Test{111}{10}{1000000111}%
1976 \Test{0000111}{5}{0100111}% 100111 would also be ok
1977 \Test{10000111}{5}{10100111}%
1978 \Test{1000001}{3}{1001001}%
1979 \Test{1001001}{3}{1000001}%
1980 \Test{11111}{2}{11011}%
1981 \end{qstest}

```

```

1982
1983 \begin{qstest}{SetValue}{SetValue}
1984   \def\Test#1#2{%
1985     \TestError{Invalid bit value (#2) not in range 0..1}{%
1986       \bitsetSetValue{abc}{#1}{#2}%
1987     }%
1988   }%
1989   \Test{0}{-1}%
1990   \Test{0}{2}%
1991   \Test{0}{10}%
1992   \def\Test#1#2#3{%
1993     \let\BS@abc\@undefined
1994     \bitsetSetValue{abc}{#1}{#2}%
1995     \bitsetSetBin{result}{#3}%
1996     \Expect*{\BS@abc}*{\BS@result}%
1997   }%
1998   \Test{0}{0}{0}%
1999   \Test{0}{1}{1}%
2000   \Test{1}{0}{0}%
2001   \Test{1}{1}{10}%
2002   \def\Test#1#2#3#4{%
2003     \bitsetSetBin{abc}{#1}%
2004     \bitsetSetBin{result}{#4}%
2005     \bitsetSetValue{abc}{#2}{#3}%
2006     \Expect*{\BS@abc}*{\BS@result}%
2007   }%
2008   \Test{0}{0}{0}{0}%
2009   \Test{0}{0}{0}{0}%
2010   \Test{0}{0}{1}{1}%
2011   \Test{0}{1}{0}{0}%
2012   \Test{0}{1}{1}{10}%
2013   \Test{1010}{2}{1}{1110}%
2014   \Test{1010}{4}{1}{11010}%
2015   \Test{1010}{6}{1}{1001010}%
2016   \Test{1010}{1}{0}{1000}%
2017   \Test{1010}{2}{0}{1010}%
2018   \Test{1010}{3}{0}{10}%
2019   \Test{1010}{4}{0}{1010}%
2020   \Test{1010}{6}{0}{1010}%
2021   \Test{1010}{2}{\csname iffalse\endcsname 0\else 1\fi}{1110}%
2022   \Test{1010}{1}{\csname iffalse\endcsname 1\else 0\fi}{1000}%
2023 \end{qstest}
2024
2025 \begin{qstest}{IsDefined}{IsDefined}
2026   \let\BS@abc\@undefined
2027   \Expect*{\bitsetIsDefined{abc}{true}{false}}{false}%
2028   \bitsetReset{abc}%
2029   \Expect*{\bitsetIsDefined{abc}{true}{false}}{true}%
2030 \end{qstest}
2031
2032 \begin{qstest}{IsEmpty}{IsEmpty}
2033   \let\BS@abc\@undefined
2034   \Expect*{\bitsetIsEmpty{abc}{true}{false}}{true}%
2035   \bitsetReset{abc}%
2036   \Expect*{\bitsetIsEmpty{abc}{true}{false}}{true}%
2037   \bitsetSet{abc}{1}%
2038   \Expect*{\bitsetIsEmpty{abc}{true}{false}}{false}%
2039 \end{qstest}
2040
2041 \begin{qstest}{Equals}{Equals}
2042   \def\Test#1#2#3{%
2043     \Expect*{\bitsetEquals{#1}{#2}{true}{false}}{#3}%

```



```

2044 }%
2045 \let\BS@abc\@undefined
2046 \Test{abc}{abc}{true}%
2047 \Test{abc}{foo}{true}%
2048 \Test{foo}{abc}{true}%
2049 \bitsetReset{abc}%
2050 \Test{abc}{abc}{true}%
2051 \Test{abc}{foo}{false}%
2052 \Test{foo}{abc}{false}%
2053 \bitsetReset{foo}%
2054 \Test{abc}{foo}{true}%
2055 \Test{foo}{abc}{true}%
2056 \bitsetSet{abc}{4}%
2057 \Test{abc}{foo}{false}%
2058 \Test{foo}{abc}{false}%
2059 \bitsetFlip{foo}{4}%
2060 \Test{abc}{foo}{true}%
2061 \Test{foo}{abc}{true}%
2062 \end{qstest}
2063
2064 \begin{qstest}{Intersects}{Intersects}
2065   \def\Test#1{%
2066     \Expect*{\bitsetIntersects{abc}{foo}{true}{false}}{#1}%
2067   }%
2068   \let\BS@abc\@undefined
2069   \let\BS@foo\@undefined
2070   \Test{false}%
2071   \Set{abc}{0}%
2072   \Test{false}%
2073   \Set{foo}{0}%
2074   \Test{false}%
2075   \let\BS@abc\@undefined
2076   \Test{false}%
2077   \Set{foo}{1}%
2078   \Test{false}%
2079   \Set{abc}{0}%
2080   \Test{false}%
2081   \Set{abc}{1}%
2082   \Test{true}%
2083   \let\BS@foo\@undefined
2084   \Test{false}%
2085   \Set{foo}{0}%
2086   \Test{false}%
2087   \def\Test#1#2#3{%
2088     \bitsetSetBin{abc}{#1}%
2089     \bitsetSetBin{foo}{#2}%
2090     \Expect*{\bitsetIntersects{abc}{foo}{true}{false}}{#3}%
2091   }%
2092   \Test{1010}{0101}{false}%
2093   \Test{0}{10}{false}%
2094   \Test{1}{11}{true}%
2095   \Test{11}{1}{true}%
2096   \Test{10}{1}{false}%
2097 \end{qstest}
2098
2099 \begin{qstest}{And/AndNot/Or/Xor}{And/AndNot/Or/Xor}
2100   \def\@Test#1#2#3#4#5{%
2101     \begingroup
2102       #5%
2103     \begingroup
2104       \let\BS@foo\@undefined
2105       \csname bitset#1\endcsname{abc}{foo}%

```

```

2106         \CheckUndef{foo}%
2107         \Check{abc}{#2}%
2108     \endgroup
2109     \begingroup
2110         \bitsetReset{foo}%
2111         \csname bitset#1\endcsname{abc}{foo}%
2112         \Check{foo}{0}%
2113         \Check{abc}{#3}%
2114     \endgroup
2115     \begingroup
2116         \def\BS@foo{0101}%
2117         \csname bitset#1\endcsname{abc}{foo}%
2118         \Check{foo}{0101}%
2119         \Check{abc}{#4}%
2120     \endgroup
2121 \endgroup
2122 }%
2123 \def\Test#1{%
2124     \def\Op{#1}%
2125     \Test@
2126 }%
2127 \def\Test@#1#2#3#4#5#6#7#8#9{%
2128     \@Test\Op{#1}{#2}{#3}{%
2129         \let\BS@abc\@undefined
2130     }%
2131     \@Test\Op{#4}{#5}{#6}{%
2132         \bitsetReset{abc}%
2133     }%
2134     \@Test\Op{#7}{#8}{#9}{%
2135         \def\BS@abc{1001}%
2136     }%
2137 }%
2138 \Test{And}%
2139     {0}{0}{0}%
2140     {0}{0}{0}%
2141     {0}{0}{0001}%
2142 \Test{AndNot}%
2143     {0}{0}{0}%
2144     {0}{0}{0}%
2145     {1001}{1001}{1}%
2146 \Test{Or}%
2147     {0}{0}{0101}%
2148     {0}{0}{0101}%
2149     {1001}{1001}{1101}%
2150 \Test{Xor}%
2151     {0}{0}{0101}%
2152     {0}{0}{0101}%
2153     {1001}{1001}{11}%
2154 \def\Test#1#2#3{%
2155     \bitsetSetBin{abc}{#1}%
2156     \bitsetSetBin{foo}{#2}%
2157     \csname bitset\Op\endcsname{abc}{foo}%
2158     \RevCheck{foo}{#2}%
2159     \RevCheck{abc}{#3}%
2160 }%
2161 \def\Op{And}%
2162 \Test{1}{111}{1}%
2163 \Test{111}{1}{1}%
2164 \Test{10}{111}{10}%
2165 \Test{111}{10}{10}%
2166 \Test{111}{1000}{0}%
2167 \Test{1000}{111}{0}%

```

```

2168 \def\Op{AndNot}%
2169 \Test{1010}{11}{1000}%
2170 \Test{100}{100}{0}%
2171 \Test{111}{1111}{0}%
2172 \Test{100}{111}{0}%
2173 \def\Op{Or}%
2174 \Test{0}{0}{0}%
2175 \Test{1}{0}{1}%
2176 \Test{0}{1}{1}%
2177 \Test{1}{1}{1}%
2178 \Test{1000}{10}{1010}%
2179 \Test{10}{1000}{1010}%
2180 \def\Op{Xor}%
2181 \Test{0}{0}{0}%
2182 \Test{1}{0}{1}%
2183 \Test{0}{1}{1}%
2184 \Test{1}{1}{0}%
2185 \Test{1000}{10}{1010}%
2186 \Test{10}{1000}{1010}%
2187 \Test {110011001100}%
2188 {111000111000111}%
2189 {111110100001011}%
2190 \Test{111000111000111}%
2191 {110011001100}%
2192 {111110100001011}%
2193 \end{qstest}
2194
2195 \begin{qstest}{\GetUndef}{\GetUndef, GetBin, GetOct, GetHex}
2196 \def\TestUndef#1#2{%
2197 \let\BS@abc\@undefined
2198 \expandafter\expandafter\expandafter\Expect
2199 \expandafter\expandafter\expandafter{%
2200 \x{abc}{#1}%
2201 }{#2}%
2202 }%
2203 \let\x\bitsetGetBin
2204 \TestUndef{-1}{0}%
2205 \TestUndef{0}{0}%
2206 \TestUndef{1}{0}%
2207 \TestUndef{2}{00}%
2208 \TestUndef{8}{00000000}%
2209 \let\x\bitsetGetOct
2210 \TestUndef{-1}{0}%
2211 \TestUndef{0}{0}%
2212 \TestUndef{1}{0}%
2213 \TestUndef{2}{0}%
2214 \TestUndef{3}{0}%
2215 \TestUndef{4}{00}%
2216 \TestUndef{5}{00}%
2217 \TestUndef{6}{00}%
2218 \TestUndef{7}{000}%
2219 \TestUndef{8}{000}%
2220 \TestUndef{9}{000}%
2221 \TestUndef{10}{0000}%
2222 \let\x\bitsetGetHex
2223 \TestUndef{-1}{0}%
2224 \TestUndef{0}{0}%
2225 \TestUndef{1}{0}%
2226 \TestUndef{2}{0}%
2227 \TestUndef{3}{0}%
2228 \TestUndef{4}{0}%
2229 \TestUndef{5}{00}%

```

```

2230 \TestUndef{6}{00}%
2231 \TestUndef{7}{00}%
2232 \TestUndef{8}{00}%
2233 \TestUndef{9}{000}%
2234 \TestUndef{10}{000}%
2235 \TestUndef{12}{000}%
2236 \TestUndef{13}{0000}%
2237 \TestUndef{16}{0000}%
2238 \TestUndef{17}{00000}%
2239 \end{qstest}
2240
2241 \begin{qstest}{SetBin}{SetBin}
2242 \def\Test#1#2{%
2243 \let\BS@abc\@undefined
2244 \bitsetSetBin{abc}{#1}%
2245 \expandafter\Expect\expandafter{\BS@abc}{#2}%
2246 }%
2247 \Test{}{0}%
2248 \Test{0}{0}%
2249 \Test{1}{1}%
2250 \Test{10}{01}%
2251 \Test{11}{11}%
2252 \Test{010}{01}%
2253 \Test{011}{11}%
2254 \Test{0010}{01}%
2255 \Test{1010}{0101}%
2256 \end{qstest}
2257
2258 \begin{qstest}{SetOct}{SetOct}
2259 \def\Test#1#2{%
2260 \bitsetSetOct{abc}{#1}%
2261 \expandafter\Expect\expandafter{\BS@abc}{#2}%
2262 }%
2263 \Test{}{0}%
2264 \Test{0}{0}%
2265 \Test{000}{0}%
2266 \Test{1}{1}%
2267 \Test{001}{1}%
2268 \Test{010}{0001}%
2269 \Test{020}{00001}%
2270 \Test{42}{010001}%
2271 \Test{377}{11111111}%
2272 \Test{0377}{11111111}%
2273 \Test{76543210}{000100010110001101011111}%
2274 \Test{ 0 7 0 7 1 }{100111000111}%
2275 \end{qstest}
2276
2277 \begin{qstest}{SetHex}{SetHex}
2278 \def\Test#1#2{%
2279 \bitsetSetHex{abc}{#1}%
2280 \expandafter\Expect\expandafter{\BS@abc}{#2}%
2281 }%
2282 \Test{}{0}%
2283 \Test{0}{0}%
2284 \Test{000}{0}%
2285 \Test{1}{1}%
2286 \Test{001}{1}%
2287 \Test{010}{00001}%
2288 \Test{020}{000001}%
2289 \Test{42}{0100001}%
2290 \Test{3F}{111111}%
2291 \Test{03F}{111111}%

```

```

2292 \Test{43210}{0000100001001100001}%
2293 \Test{98765}{10100110111000011001}%
2294 \Test{FEDCBA}{010111010011101101111111}%
2295 \Test{ 0 F 0 F 1 }{1000111100001111}%
2296 \end{qstest}
2297
2298 \begin{qstest}{SetDec}{SetDec}
2299 \def\Test#1#2{%
2300 \bitsetSetDec{abc}{#1}%
2301 \expandafter\Expect\expandafter{\BS@abc}{#2}%
2302 }%
2303 \Test{}{0}%
2304 \Test{0}{0}%
2305 \Test{000}{0}%
2306 \Test{1}{1}%
2307 \Test{7}{111}%
2308 \Test{8}{0001}%
2309 \Test{001}{1}%
2310 \Test{010}{0101}%
2311 \Test{020}{00101}%
2312 \Test{53}{101011}%
2313 \Test{255}{11111111}%
2314 \Test{256}{000000001}%
2315 \Test{999999999}{11111111001001101011001110111}%
2316 \Test{1000000000}{000000000101001101011001110111}%
2317 \Test{4210987654}{01100001010010010111111101011111}%
2318 \Test{2147483647}{11111111111111111111111111111111}%
2319 \Test{2147483648}{00000000000000000000000000000001}%
2320 \end{qstest}
2321
2322 \begin{qstest}{GetBin}{GetBin}
2323 \def\TestUndef#1#2{%
2324 \let\BS@abc\@undefined
2325 \expandafter\expandafter\expandafter\Expect
2326 \expandafter\expandafter\expandafter{%
2327 \bitsetGetBin{abc}{#1}%
2328 }{#2}%
2329 }%
2330 \TestUndef{-1}{0}%
2331 \TestUndef{0}{0}%
2332 \TestUndef{1}{0}%
2333 \TestUndef{2}{00}%
2334 \TestUndef{8}{00000000}%
2335 \def\Test#1#2{%
2336 \bitsetSetBin{abc}{#2}%
2337 \expandafter\expandafter\expandafter\Expect
2338 \expandafter\expandafter\expandafter{%
2339 \bitsetGetBin{abc}{#1}%
2340 }{#2}%
2341 }%
2342 \Test{-1}{0}%
2343 \Test{0}{0}%
2344 \Test{1}{0}%
2345 \Test{1}{1}%
2346 \Test{2}{01}%
2347 \Test{2}{10}%
2348 \Test{3}{010}%
2349 \Test{2}{00}%
2350 \Test{2}{01}%
2351 \Test{8}{00101100}%
2352 \Test{2}{10101}%
2353 \Test{-100}{11011}%

```

```

2354 \end{qstest}
2355
2356 \begin{qstest}{GetOct}{GetOct}
2357   \def\Test#1#2#3{%
2358     \edef\x{\zap@space#1 \@empty}%
2359     \edef\x{\noexpand\bitsetSetBin{abc}{\x}}%
2360     \x
2361     \expandafter\expandafter\expandafter\Expect
2362     \expandafter\expandafter\expandafter{%
2363       \bitsetGetOct{abc}{#2}%
2364     }{#3}%
2365   }%
2366   \Test{111 110 101 100 011 010 001 000}{0}{76543210}%
2367   \Test{000 111}{0}{7}%
2368   \Test{101 000}{-1}{50}%
2369   \Test{111}{-1}{7}%
2370   \Test{111}{0}{7}%
2371   \Test{111}{1}{7}%
2372   \Test{111}{3}{7}%
2373   \Test{111}{4}{07}%
2374   \Test{111}{6}{07}%
2375   \Test{111}{7}{007}%
2376   \Test{111 010}{6}{72}%
2377   \Test{111 010}{7}{072}%
2378   \Test{011 111}{0}{37}%
2379   \Test{011 111}{6}{37}%
2380   \Test{011 111}{7}{037}%
2381   \Test{001 111}{0}{17}%
2382   \Test{001 111}{6}{17}%
2383   \Test{001 111}{7}{017}%
2384 \end{qstest}
2385
2386 \begin{qstest}{GetHex}{GetHex}
2387   \def\Test#1#2#3{%
2388     \bitsetSetBin{abc}{#1}%
2389     \expandafter\expandafter\expandafter\Expect
2390     \expandafter\expandafter\expandafter{%
2391       \bitsetGetHex{abc}{#2}%
2392     }{#3}%
2393   }%
2394   \Test{1111 1110 1101 1100 1011 1010 1001 1000}{0}{FEDCBA98}%
2395   \Test{0111 0110 0101 0100 0011 0010 0001 0000}{0}{76543210}%
2396   \Test{0000 1111}{0}{F}%
2397   \Test{0101 0000}{-1}{50}%
2398   \Test{1111}{-1}{F}%
2399   \Test{1111}{0}{F}%
2400   \Test{1111}{1}{F}%
2401   \Test{1111}{4}{F}%
2402   \Test{1111}{5}{0F}%
2403   \Test{1111}{8}{0F}%
2404   \Test{1111}{9}{00F}%
2405   \Test{1111 0010}{8}{F2}%
2406   \Test{1111 0010}{9}{0F2}%
2407   \Test{0111 1111}{0}{7F}%
2408   \Test{0111 1111}{8}{7F}%
2409   \Test{0111 1111}{9}{07F}%
2410   \Test{0011 1111}{0}{3F}%
2411   \Test{0011 1111}{8}{3F}%
2412   \Test{0011 1111}{9}{03F}%
2413   \Test{0001 1111}{0}{1F}%
2414   \Test{0001 1111}{8}{1F}%
2415   \Test{0001 1111}{9}{01F}%

```

```

2416 \end{qstest}
2417
2418 \begin{qstest}{Range}{Range}
2419   \TestError{%
2420     Wrong index numbers in range [9..8]\MessageBreak% hash-ok
2421     for clear/set/flip on bit set 'abc'.\MessageBreak
2422     The lower index exceeds the upper index.\MessageBreak
2423     Canceling the operation as error recovery%
2424   }{%
2425     \bitsetSetRange{abc}{9}{8}%
2426   }%
2427 \def\TestErrorNegInd#1#2#3#4#5#6{%
2428   \TestError{%
2429     Negative index in range [#2..#3]\MessageBreak % hash-ok
2430     for \string\bitset #1Range on bit set 'abc'.\MessageBreak
2431     Using [#4..#5] as error recovery% hash-ok
2432   }{%
2433     \csname bitset#1Range\endcsname{abc}{#2}{#3}%
2434     \global\let\BS@global\BS@abc
2435   }%
2436   \Check{global}{#6}%
2437 }%
2438 \Set{abc}{111}%
2439 \TestErrorNegInd{Clear}{-1}{0}{0}{0}{111}%
2440 \TestErrorNegInd{Clear}{0}{-1}{0}{0}{111}%
2441 \TestErrorNegInd{Clear}{-2}{2}{0}{2}{001}%
2442 \bitsetReset{abc}%
2443 \TestErrorNegInd{Set}{-1}{0}{0}{0}{0}%
2444 \TestErrorNegInd{Set}{0}{-1}{0}{0}{0}%
2445 \TestErrorNegInd{Set}{-2}{2}{0}{2}{11}%
2446 \Set{abc}{101}%
2447 \TestErrorNegInd{Flip}{-1}{0}{0}{0}{101}%
2448 \TestErrorNegInd{Flip}{0}{-1}{0}{0}{101}%
2449 \TestErrorNegInd{Flip}{-2}{2}{0}{2}{011}%
2450 \def\Test#1#2#3#4{%
2451   \bitsetSetBin{abc}{#1}%
2452   \csname bitset\TestOp Range\endcsname{abc}{#2}{#3}%
2453   \Expect*{\bitsetGetBin{abc}{0}}{#4}%
2454 }%
2455 \def\TestOp{Clear}%
2456 \Test{0}{0}{1}{0}%
2457 \Test{1111}{1}{2}{1101}%
2458 \Test{1111}{1}{3}{1001}%
2459 \Test{1111111100000000}{12}{14}{1100111100000000}%
2460 \def\TestOp{Set}%
2461 \Test{0}{0}{1}{1}%
2462 \Test{1000}{1}{2}{1010}%
2463 \Test{0}{1}{2}{10}%
2464 \Test{1}{12}{15}{111000000000001}%
2465 \Test{1111}{1}{3}{1111}%
2466 \Test{1000000000000000}{12}{14}{1011000000000000}%
2467 \def\TestOp{Flip}%
2468 \Test{0}{0}{1}{1}%
2469 \Test{1}{0}{1}{0}%
2470 \Test{10101010}{1}{5}{10110100}%
2471 \def\Test#1#2#3#4#5{%
2472   \bitsetSetBin{abc}{#1}%
2473   \bitsetSetValueRange{abc}{#2}{#3}{#4}%
2474   \Expect*{\bitsetGetBin{abc}{0}}{#5}%
2475 }%
2476 \Test{0}{0}{1}{0}{0}%
2477 \Test{0}{0}{1}{1}{1}%

```

[illegible]


```
2540 \end{document}
2541 \</test2>
```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

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

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

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

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

4.2 Bundle installation

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

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

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

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

4.3 Package installation

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

```
tex bitset.dtx
```

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

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

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

¹<http://ftp.ctan.org/tex-archive/>

4.4 Refresh file name databases

If your \TeX distribution ($\text{te}\text{\TeX}$, $\text{mik}\text{\TeX}$, ...) relies on file name databases, you must refresh these. For example, $\text{te}\text{\TeX}$ users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

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

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

Unpacking with \LaTeX . The `.dtx` chooses its action depending on the format:

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

\LaTeX : Generate the documentation.

If you insist on using \LaTeX for `docstrip` (really, `docstrip` does not need \LaTeX), then inform the autodetect routine about your intention:

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

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

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

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

An example follows how to generate the documentation with $\text{pdf}\text{\LaTeX}$:

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

5 History

[2007/09/28 v1.0]

- First version.

6 Index

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

| Symbols | | 2128, 2131, 2134, 2483, 2489, 2490 |
|-----------------------------|------------------------------------|--|
| <code>\#</code> | 1389, 1451 | <code>\@ehc</code> 155, 347, 903, 1061, 1083 |
| <code>\%</code> | 1454 | <code>\@empty</code> 2358 |
| <code>\:</code> | 1723 | <code>\@firstofone</code> 1398, 1401, 1611 |
| <code>\@</code> | 1390, 1447 | <code>\@gobble</code> 1395, 1403, 1609 |
| <code>\@PackageError</code> | | <code>\@undefined</code> .. 52, 1474, 1475, 1476, |
| | .. 153, 345, 901, 1056, 1069, 1548 | 1730, 1858, 1993, 2026, 2033, |
| <code>\@Test</code> | 2100, | 2045, 2068, 2069, 2075, 2083, |

| | |
|---|---|
| 2104, 2129, 2197, 2243, 2324, 2484 | \BitSet@AfterFiFiFi 136, 674, 678, |
| \[..... 1452 | 714, 718, 793, 798, 933, 938, |
| \\ 199, 352, 415, 548, 557, | 1006, 1011, 1224, 1230, 1375, 1377 |
| 619, 666, 669, 704, 707, 744, | \BitSet@And 654, <u>665</u> |
| 747, 777, 779, 787, 1271, 1280, | \BitSet@AndNot 692, <u>703</u> |
| 1302, 1311, 1371, 1374, 1448, 1607 | \BitSet@AtEnd 80, 81, 1384 |
| \{ 1387, 1449 | \BitSet@Cardinality 1294, <u>1300</u> |
| \} 1388, 1450 | \BitSet@CheckIndex 144, 875, 878, 881, 888 |
| \] 1453 | \BitSet@Cleanup 868, 930, 1127, 1160, 1196, <u>1205</u> |
| _ 1455 | \BitSet@Clear 875, 890, <u>905</u> , 1019, 1033, 1072 |
| A | \BitSet@Empty 119, 127, |
| \advance 1428, 1436, 1514 | 178, 181, 183, 217, 220, 222, |
| \aftergroup 26 | 228, 324, 327, 329, 447, 461, |
| \AtEndDocument 1527 | 465, 490, 659, 697, 770, 848, |
| | 860, 866, 909, 913, 921, 923, |
| B | 929, 949, 959, 980, 984, 993, 1001 |
| \begin 1615, 1632, 1642, 1669, | \BitSet@ErrorInvalidBitValue ... 894, <u>900</u> , 1037 |
| 1690, 1713, 1857, 1878, 1914, | \BitSet@Fi 133, 134, 135, 136, 161, |
| 1933, 1957, 1983, 2025, 2032, | 205, 251, 282, 365, 380, 396, |
| 2041, 2064, 2099, 2195, 2241, | 412, 424, 434, 476, 501, 538, |
| 2258, 2277, 2298, 2322, 2356, | 554, 578, 640, 683, 723, 754, |
| 2386, 2418, 2482, 2530, 2535, 2539 | 803, 872, 943, 966, 1016, 1093, |
| \BigIntCalcAdd 623, 632 | 1114, 1131, 1152, 1167, 1188, |
| \bigintcalcCmp 335 | 1203, 1236, 1258, 1289, 1319, 1382 |
| \BigIntCalcOdd 355 | \BitSet@Fill 391, <u>404</u> , 429, 531 |
| \bigintcalcSgn 332 | \BitSet@FirstOfOne 120 |
| \BigIntCalcShl 637, 644 | \BitSet@FirstOfTwo 122, 139, 1324, |
| \BigIntCalcShr 363 | 1327, 1329, 1338, 1345, 1350, 1369 |
| \bitset 1071, 2430 | \BitSet@Flip 881, <u>976</u> , 1025 |
| \BitSet@@@Range 1063, 1086, 1090 | \BitSet@FromFirstHex 211, <u>269</u> |
| \BitSet@@@Set 961, <u>968</u> , 1003 | \BitSet@FromFirstOct 208, <u>237</u> |
| \BitSet@@@CheckIndex 146, <u>150</u> | \BitSet@FromHex 281, <u>284</u> |
| \BitSet@@@Clear 907, <u>919</u> | \BitSet@FromOct 250, <u>253</u> |
| \BitSet@@@Flip 978, <u>990</u> | \BitSet@Get 1097, <u>1100</u> |
| \BitSet@@@Get 1107, <u>1116</u> | \BitSet@GetDec 543, <u>547</u> |
| \BitSet@@@GetBin 384, <u>387</u> | \BitSet@GetDecBig 616, <u>618</u> , 643 |
| \BitSet@@@GetDec 552, <u>556</u> , 582 | \BitSet@GetOctHex 467, 492, <u>522</u> |
| \BitSet@@@GetDecBig 631, <u>642</u> | \BitSet@GetSetBitList ... 1245, <u>1249</u> |
| \BitSet@@@GetHex 456, <u>489</u> | \BitSet@Gobble 121, 829, 854, 895, 896, 1220 |
| \BitSet@@@GetOct 442, <u>464</u> | \BitSet@GobbleSeven 1227, <u>1241</u> |
| \BitSet@@@GetOctHex . 439, 453, 523, <u>527</u> | \BitSet@Hex[0..F] 295 |
| \BitSet@@@NextClearBit ... 1150, <u>1154</u> | \BitSet@Hex[0000..1111] 503 |
| \BitSet@@@NextSetBit 1186, <u>1190</u> | \BitSet@IfUndefined 137, 145, 167, 388, 542, |
| \BitSet@@@Range . 1042, <u>1047</u> , 1084, <u>1086</u> | 727, 758, 806, 833, 1106, 1262, |
| \BitSet@@@Set 947, <u>954</u> | 1293, 1322, 1327, 1344, 1345, 1347 |
| \BitSet@@@TestMode 107 | \BitSet@Intersects 1360, <u>1367</u> |
| \BitSet@AfterFi 134, 152, 158, 202, 361, 376, | \BitSet@Kill 847, <u>857</u> |
| 390, 395, 406, 411, 416, 420, | \BitSet@KillZeros 181, <u>191</u> , 220, 278, 327 |
| 428, 433, 466, 471, 491, 496, | \BitSet@MaxSize 118, 335 |
| 529, 537, 549, 551, 1089, 1102, | \BitSet@N1073741824 615 |
| 1141, 1177, 1208, 1251, 1253, 1369 | \BitSet@N[1,2,4,...] 580 |
| \BitSet@AfterFiFi 568, 574, 621, 626, 630, 635, | \BitSet@NegativeIndex 1049, 1052, <u>1068</u> |
| 750, 868, 930, 961, 1003, 1127, | \BitSet@NextClearBit 1135, <u>1138</u> |
| 1160, 1162, 1196, 1198, 1213, | \BitSet@NextSetBit 1171, <u>1174</u> , 1246, 1255 |
| 1217, 1219, 1272, 1274, 1281, | |
| 1283, 1303, 1305, 1312, 1314, 1372 | |

| | | | |
|-----------------------------|--|----------------------------|--|
| \BitSet@NumBinFill | 417, <u>426</u> | \bitsetGetBin | 6, <u>382</u> , |
| \BitSet@NumBinRev | 398, <u>414</u> | | 2203, 2327, 2339, 2453, 2474, 2501 |
| \BitSet@Oct[000..111] | <u>478</u> | \bitsetGetDec | 7, <u>540</u> , 1879, 1887 |
| \BitSet@Or | <u>734</u> , <u>742</u> | \bitsetGetHex | <u>450</u> , 2222, 2391 |
| \BitSet@Range | | \bitsetGetOct | <u>436</u> , 2209, 2363 |
| | 1019, 1022, 1025, 1033, 1035, <u>1040</u> | \bitsetGetSetBitList | |
| \BitSet@Reverse | 187, <u>198</u> , 232 | | 8, <u>1242</u> , 1861, 1867 |
| \BitSet@SecondOfTwo | <u>123</u> , | \bitsetIntersects | 9, <u>1357</u> , 2066, 2090 |
| | 141, 1323, 1331, 1340, 1345, | \bitsetIsDefined | 8, <u>1321</u> , 2027, 2029 |
| | 1347, 1352, 1358, 1359, 1372, 1375 | \bitsetIsEmpty | 9, 438, 452, 647, |
| \BitSet@Set | | | 650, 686, 689, 726, 729, 757, |
| | 878, 892, <u>945</u> , 1022, 1035, 1075 | | 760, 809, 836, 1146, 1182, 1244, |
| \BitSet@SetDec | 341, 353, <u>367</u> | | <u>1326</u> , 1358, 1359, 2034, 2036, 2038 |
| \BitSet@SetDecBig | 337, <u>351</u> | \bitsetLet | 6, <u>166</u> , 1618, 1624 |
| \BitSet@SetOctHex | 208, 211, <u>213</u> | \bitsetNextClearBit | 8, <u>1133</u> , 1720 |
| \BitSet@SetValue | 884, <u>887</u> | \bitsetNextSetBit | 8, <u>1169</u> , 1721 |
| \BitSet@SetValueRange | 1028, <u>1031</u> | \bitsetOr | 7, <u>725</u> |
| \BitSet@ShiftLeft | 811, <u>816</u> , 854 | \bitsetQuery | 9, <u>1336</u> , 1650, 1656 |
| \BitSet@ShiftRight | 829, 838, <u>843</u> | \bitsetReset | 6, 145, |
| \BitSet@Size | 1263, <u>1269</u> | | <u>163</u> , 168, 648, 651, 660, 687, |
| \BitSet@Skip | 1147, 1183, <u>1206</u> | | 698, 727, 758, 771, 807, 834, |
| \BitSet@SkipContinue | | | 1633, 1635, 1638, 1740, 2028, |
| | 1209, 1214, 1217, 1220, <u>1238</u> | | 2035, 2049, 2053, 2110, 2132, 2442 |
| \BitSet@Space | <u>124</u> , 178, 217, | \bitsetSet | 877, |
| | 324, 560, 622, 824, 1113, 1142, 1178 | | 1936, 1939, 1941, 2037, 2056, 2531 |
| \BitSet@Temp | | \bitsetSetBin | 6, <u>174</u> , 1724, 1750, |
| | 175, 176, 178, 180, 181, 183, | | 1760, 1811, 1822, 1833, 1844, |
| | 187, 214, 215, 217, 219, 220, | | 1864, 1995, 2003, 2004, 2088, |
| | 222, 225, 226, 228, 232, 295, | | 2089, 2155, 2156, 2244, 2336, |
| | 298, 299, 300, 301, 302, 303, | | 2359, 2388, 2451, 2472, 2498, 2499 |
| | 304, 305, 306, 307, 308, 309, | \bitsetSetDec | 6, <u>320</u> , 2300 |
| | 310, 311, 312, 313, 314, 315, | \bitsetSetHex | <u>210</u> , 2279 |
| | 316, 317, 318, 319, 321, 322, | \bitsetSetOct | <u>207</u> , 2260 |
| | 324, 326, 327, 329, 332, 335, | \bitsetSetRange | <u>1021</u> , 2425 |
| | 337, 341, 478, 481, 482, 483, | \bitsetSetValue | 8, <u>883</u> , 1986, 1994, 2005 |
| | 484, 485, 486, 487, 488, 503, | \bitsetSetValueRange | <u>1027</u> , 2473 |
| | 506, 507, 508, 509, 510, 511, | \bitsetShiftLeft | 7, <u>805</u> |
| | 512, 513, 514, 515, 516, 517, | \bitsetShiftRight | <u>832</u> |
| | 518, 519, 520, 521, 580, 585, | \bitsetSize | 8, <u>1260</u> , 1670, 1674 |
| | 586, 587, 588, 589, 590, 591, | \bitsetXor | 7, <u>756</u> |
| | 592, 593, 594, 595, 596, 597, | \body | 1407, 1411 |
| | 598, 599, 600, 601, 602, 603, | \BS@abc | 1730, 1858, 1918, 1937, 1961, |
| | 604, 605, 606, 607, 608, 609, | | 1993, 1996, 2006, 2026, 2033, |
| | 610, 611, 612, 613, 614, 906, | | 2045, 2068, 2075, 2129, 2135, |
| | 913, 916, 977, 984, 987, 1041, 1045 | | 2197, 2243, 2245, 2261, 2280, |
| \BitSet@TestMode | 107, <u>1481</u> | | 2301, 2324, 2434, 2484, 2486, 2532 |
| \BitSet@Xor | 765, <u>776</u> | \BS@foo | 2069, 2083, 2104, 2116 |
| \BitSet@ZapSpace | <u>125</u> , <u>177</u> , 216, 323 | \BS@global | 2434, 2532 |
| \BitSet@Zero | 184, 223, | \BS@result | 1996, 2006 |
| | 229, 330, 333, 914, 985, 1328, <u>1335</u> | | |
| \bitsetAnd | 7, <u>646</u> | | |
| \bitsetAndNot | 7, <u>685</u> | | |
| \bitsetCardinality | 8, <u>1291</u> , 1691, 1696 | | |
| \bitsetClear | 7, <u>874</u> , 1917, 1920, 1922 | | |
| \bitsetClearRange | <u>1018</u> | | |
| \bitsetEquals | 9, <u>1343</u> , 2043 | | |
| \BitSetError | 248, 264, 276, 288, | | |
| | 359, 1103, 1143, 1179, 1726, 1728 | | |
| \bitsetFlip | <u>880</u> , 1960, 1963, 1965, 2059 | | |
| \bitsetFlipRange | <u>1024</u> | | |
| \bitsetGet | | | |
| | 8, <u>1095</u> , 1337, 1645, 1655, 2536 | | |

| C | |
|----------------|-------------------------------------|
| \catcode | 3, 4, 5, 6, 7, 8, 9, 17, 31, |
| | 32, 33, 34, 35, 36, 37, 38, 39, 40, |
| | 41, 42, 43, 64, 65, 68, 69, 70, 71, |
| | 75, 76, 77, 78, 82, 84, 105, 1387, |
| | 1388, 1389, 1390, 1425, 1434, |
| | 1447, 1448, 1449, 1450, 1451, |
| | 1452, 1453, 1454, 1455, 1456, 1723 |
| \chardef | 1481 |
| \Check | 1584, |
| | 1620, 1622, 1623, 1625, 1626, |
| | 1628, 1629, 1634, 1636, 1639, |
| | 2107, 2112, 2113, 2118, 2119, 2436 |

| | | |
|---|---|---|
| <code>\CheckUndef</code> | 1574, 1587, 1616, 1617, 1619, 2106 | 1918, 1937, 1961, 1996, 2006, 2027, 2029, 2034, 2036, 2038, 2043, 2066, 2090, 2198, 2245, 2261, 2280, 2301, 2325, 2337, 2361, 2389, 2453, 2474, 2486, 2501 |
| <code>\Clear</code> | 1720, 1725, 1731, 1741, 1751, 1761, 1812, 1823, 1834, 1845 | <code>\ExpectBitSet</code> 1579, 1585, 1596 |
| <code>\count@</code> | 1392, 1421, 1425, 1427, 1428, 1432, 1434, 1435, 1436 | |
| <code>\countdef</code> | 1392 | |
| <code>\csname</code> 10, 18, 44, 60, 67, 103, 109, 138, 164, 170, 171, 184, 186, 223, 229, 231, 275, 279, 287, 290, 296, 330, 333, 336, 340, 401, 446, 460, 474, 479, 499, 504, 544, 569, 575, 581, 615, 653, 655, 657, 659, 691, 693, 695, 697, 730, 731, 733, 735, 737, 761, 762, 764, 766, 768, 770, 822, 825, 846, 848, 908, 912, 946, 948, 979, 983, 1108, 1149, 1185, 1266, 1297, 1328, 1348, 1349, 1361, 1363, 1391, 1394, 1397, 1400, 1439, 1461, 1497, 1582, 1591, 1599, 1602, 1648, 2021, 2022, 2105, 2111, 2117, 2157, 2433, 2452, 2485, 2500 | | H |
| <code>\currentgrouplevel</code> .. | 1472, 1476, 1488 | <code>\hbox</code> 1535 |
| | | I |
| D | | <code>\ifcase</code> 240, 256, 332, 355, 368, 817, 844, 889, 1032, 1054, 1212, 1657 |
| <code>\dimexpr</code> | 1504 | <code>\ifcsname</code> 1471, 1474, 1487 |
| <code>\documentclass</code> | 1467 | <code>\ifnum</code> 151, 335, 389, 405, 427, 528, 667, 670, 672, 705, 709, 712, 743, 1048, 1051, 1087, 1101, 1139, 1175, 1223, 1250, 1337, 1368, 1427, 1435 |
| E | | <code>\ifodd</code> 371 |
| <code>\empty</code> | 13, 14 | <code>\ifx</code> 11, 14, 18, 44, 52, 55, 103, 109, 127, 138, 183, 192, 199, 222, 228, 238, 254, 270, 272, 275, 285, 287, 329, 352, 415, 465, 490, 548, 557, 558, 567, 619, 620, 629, 659, 666, 669, 697, 704, 707, 744, 747, 770, 777, 778, 779, 787, 789, 792, 859, 860, 866, 913, 920, 921, 923, 929, 932, 956, 959, 969, 984, 991, 992, 993, 1001, 1005, 1072, 1075, 1118, 1119, 1125, 1155, 1158, 1191, 1194, 1207, 1270, 1271, 1280, 1301, 1302, 1311, 1328, 1348, 1371, 1374, 1391, 1394, 1397, 1400, 1439, 1497, 1591, 1607 |
| <code>\end</code> .. | 1462, 1630, 1640, 1667, 1688, 1711, 1855, 1876, 1912, 1931, 1955, 1981, 2023, 2030, 2039, 2062, 2097, 2193, 2239, 2256, 2275, 2296, 2320, 2354, 2384, 2416, 2480, 2528, 2533, 2537, 2540 | <code>\ignorespaces</code> 1535 |
| <code>\endcsname</code> 10, 18, 44, 60, 67, 103, 109, 138, 164, 170, 171, 184, 186, 223, 229, 231, 275, 279, 287, 290, 296, 330, 333, 336, 340, 401, 446, 460, 474, 479, 499, 504, 544, 570, 575, 581, 615, 653, 655, 657, 659, 691, 693, 695, 697, 730, 731, 733, 735, 737, 761, 762, 764, 766, 768, 770, 822, 825, 846, 848, 908, 912, 946, 948, 979, 983, 1108, 1149, 1185, 1266, 1297, 1328, 1348, 1349, 1361, 1363, 1391, 1394, 1397, 1400, 1439, 1461, 1497, 1582, 1591, 1599, 1602, 1648, 2021, 2022, 2105, 2111, 2117, 2157, 2433, 2452, 2485, 2500 | <code>\immediate</code> 20, 46 | <code>\IncludeTests</code> 1493 |
| <code>\endinginput</code> | 26 | <code>\input</code> 110, 111, 112, 1440 |
| <code>\endqstest</code> | 1518, 1523, 1532, 1537 | <code>\IntCalcAdd</code> 533, 561, 571 |
| <code>\ETeXDisable</code> | 1473, 1478, 1560 | <code>\intcalcCmp</code> 1054 |
| <code>\ETeXEnable</code> | 1485, 1490, 1553 | <code>\IntCalcDec</code> ... 392, 408, 468, 493, 1232 |
| <code>\Expect</code> | 1539, 1555, 1556, 1564, 1565, 1575, 1580, 1589, 1643, 1650, 1654, 1656, 1673, 1694, 1715, 1859, 1865, 1885, | <code>\IntCalcDiv</code> 532 |
| | | <code>\IntCalcInc</code> 422, 473, 498, 1091, 1164, 1200, 1256, 1276, 1285, 1303, 1307 |
| | | <code>\IntCalcMul</code> 525 |
| | | <code>\intcalcNum</code> 147, 385, 440, 454, 524, 812, 839, 885, 1029, 1043, 1098, 1111, 1136, 1172 |
| | | <code>\intcalcSgn</code> 817, 844 |
| | | <code>\IntCalcShr</code> 378 |
| | | <code>\IntCalcSub</code> 430, 534, 1226 |
| | | <code>\iterate</code> 1408, 1410, 1412 |
| | | L |
| | | <code>\LoadCommand</code> 1440, 1457 |
| | | <code>\LogTests</code> 1494 |
| | | <code>\loop</code> 1406, 1422, 1433 |
| | | M |
| | | <code>\makeatletter</code> 1468, 1480, 1495 |
| | | <code>\makeatother</code> 1482 |

| | |
|---|--|
| <code>\MessageBreak</code> | <code>\StartTime</code> 1507, 1521 |
| . 1057, 1058, 1059, 1070, 1081, | <code>\stepcounter</code> 1549 |
| 1551, 2420, 2421, 2422, 2429, 2430 | <code>\StopTime</code> 1512, 1524 |
| | <code>\strip@pt</code> 1504 |
| | <code>\SummaryTime</code> . . . 1499, 1501, 1514, 1528 |
| N | |
| <code>\NeedsTeXFormat</code> 1465 | |
| <code>\newcommand</code> 1502, 1507, 1511, 1512 | T |
| <code>\newcount</code> 1499, 1500 | <code>\Test</code> 1442, 1460, 1652, 1659, |
| <code>\newcounter</code> 1542 | 1660, 1661, 1662, 1663, 1664, |
| <code>\next</code> 1412, 1414, 1416 | 1665, 1666, 1671, 1676, 1677, |
| <code>\nofiles</code> 1466 | 1678, 1679, 1680, 1681, 1682, |
| <code>\number</code> 473, 498, 524, 532, | 1683, 1684, 1685, 1686, 1687, |
| 1096, 1134, 1148, 1170, 1184, | 1692, 1699, 1700, 1701, 1702, |
| 1226, 1246, 1254, 1261, 1292, 1504 | 1703, 1704, 1705, 1706, 1707, |
| <code>\numexpr</code> 1470, 1475, 1486 | 1708, 1709, 1710, 1714, 1726, |
| | 1728, 1732, 1733, 1734, 1735, |
| O | 1737, 1738, 1739, 1742, 1743, |
| <code>\Op</code> 2124, 2128, 2131, 2134, 2157, 2161, | 1744, 1745, 1747, 1748, 1749, |
| 2168, 2173, 2180, 2500, 2503, 2514 | 1752, 1753, 1754, 1755, 1757, |
| <code>\orig@endqstest</code> 1532, 1540 | 1758, 1759, 1762, 1763, 1764, |
| <code>\orig@qstest</code> 1531, 1534 | 1765, 1766, 1767, 1768, 1769, |
| | 1770, 1771, 1772, 1773, 1774, |
| | 1775, 1776, 1777, 1778, 1779, |
| P | 1780, 1781, 1782, 1783, 1784, |
| <code>\PackageInfo</code> 23 | 1785, 1787, 1788, 1789, 1790, |
| <code>\pdfelapsedtime</code> 1513 | 1791, 1792, 1793, 1794, 1795, |
| <code>\pdfresettimer</code> 1509 | 1796, 1797, 1798, 1799, 1800, |
| <code>\PrintTime</code> 1502, 1515, 1528 | 1801, 1802, 1803, 1804, 1805, |
| <code>\ProvidesPackage</code> 15, 61 | 1806, 1807, 1808, 1809, 1810, |
| | 1813, 1814, 1815, 1816, 1818, |
| | 1819, 1820, 1821, 1824, 1825, |
| Q | 1826, 1827, 1829, 1830, 1831, |
| <code>\qstest</code> 1517, 1519, 1531, 1533 | 1832, 1835, 1836, 1837, 1838, |
| | 1840, 1841, 1842, 1843, 1846, |
| | 1847, 1848, 1849, 1851, 1852, |
| R | 1853, 1854, 1863, 1870, 1871, |
| <code>\RangeCatcodeInvalid</code> | 1872, 1873, 1874, 1875, 1880, |
| 1431, 1443, 1444, 1445, 1446 | 1890, 1891, 1892, 1893, 1894, |
| <code>\renewcommand</code> 1508 | 1895, 1896, 1897, 1898, 1899, |
| <code>\repeat</code> 1406, 1418, 1429, 1437 | 1900, 1902, 1904, 1906, 1908, |
| <code>\RequirePackage</code> 114, 115, 116 | 1910, 1915, 1924, 1925, 1926, |
| <code>\RestoreCatcodes</code> 1420, 1423, 1424, 1458 | 1927, 1928, 1929, 1930, 1934, |
| <code>\RevCheck</code> 1595, 1921, 1923, | 1943, 1944, 1945, 1946, 1947, |
| 1940, 1942, 1964, 1966, 2158, 2159 | 1948, 1949, 1950, 1951, 1952, |
| <code>\Reverse</code> 1596, | 1953, 1954, 1958, 1967, 1968, |
| 1603, 1606, 1613, 1918, 1937, 1961 | 1969, 1970, 1971, 1972, 1973, |
| <code>\RevSet</code> . . 1601, 1881, 1916, 1935, 1959 | 1974, 1975, 1976, 1977, 1978, |
| <code>\romannumeral</code> 383, | 1979, 1980, 1984, 1989, 1990, |
| 437, 451, 541, 824, 851, 910, | 1991, 1992, 1998, 1999, 2000, |
| 951, 981, 1111, 1143, 1179, 1243 | 2001, 2002, 2008, 2009, 2010, |
| | 2011, 2012, 2013, 2014, 2015, |
| S | 2016, 2017, 2018, 2019, 2020, |
| <code>\saved@endqstest</code> 1518, 1525 | 2021, 2022, 2042, 2046, 2047, |
| <code>\saved@qstest</code> 1517, 1520 | 2048, 2050, 2051, 2052, 2054, |
| <code>\SavedCurrentgrouplevel</code> . 1472, 1488 | 2055, 2057, 2058, 2060, 2061, |
| <code>\SavedIfcsname</code> 1471, 1487 | 2065, 2070, 2072, 2074, 2076, |
| <code>\SavedNumexpr</code> 1470, 1486 | 2078, 2080, 2082, 2084, 2086, |
| <code>\sbox</code> 1547 | 2087, 2092, 2093, 2094, 2095, |
| <code>\Set</code> . . 1598, 1621, 1627, 1637, 1649, | 2096, 2123, 2138, 2142, 2146, |
| 1653, 1672, 1693, 1721, 1727, | 2150, 2154, 2162, 2163, 2164, |
| 1736, 1746, 1756, 1786, 1817, | 2165, 2166, 2167, 2169, 2170, |
| 1828, 1839, 1850, 2071, 2073, | 2171, 2172, 2174, 2175, 2176, |
| 2077, 2079, 2081, 2085, 2438, 2446 | 2177, 2178, 2179, 2181, 2182, |
| <code>\setbox</code> 1535 | |
| <code>\setcounter</code> 1546 | |
| <code>\space</code> 1504 | |

| | |
|---|---|
| 2183, 2184, 2185, 2186, 2187, | 2443, 2444, 2445, 2447, 2448, 2449 |
| 2190, 2242, 2247, 2248, 2249, | \TestGetterUndefined |
| 2250, 2251, 2252, 2253, 2254, | 1573, 1670, 1691, 1879 |
| 2255, 2259, 2263, 2264, 2265, | \TestOp 1717, |
| 2266, 2267, 2268, 2269, 2270, | 1720, 1721, 2452, 2455, 2460, 2467 |
| 2271, 2272, 2273, 2274, 2278, | \TestTime 1500, 1513, 1514, 1515 |
| 2282, 2283, 2284, 2285, 2286, | \TestUndef 2196, 2204, 2205, 2206, |
| 2287, 2288, 2289, 2290, 2291, | 2207, 2208, 2210, 2211, 2212, |
| 2292, 2293, 2294, 2295, 2299, | 2213, 2214, 2215, 2216, 2217, |
| 2303, 2304, 2305, 2306, 2307, | 2218, 2219, 2220, 2221, 2223, |
| 2308, 2309, 2310, 2311, 2312, | 2224, 2225, 2226, 2227, 2228, |
| 2313, 2314, 2315, 2316, 2317, | 2229, 2230, 2231, 2232, 2233, |
| 2318, 2319, 2335, 2342, 2343, | 2234, 2235, 2236, 2237, 2238, |
| 2344, 2345, 2346, 2347, 2348, | 2323, 2330, 2331, 2332, 2333, 2334 |
| 2349, 2350, 2351, 2352, 2353, | \the . 68, 69, 70, 71, 82, 1425, 1539, 1565 |
| 2357, 2366, 2367, 2368, 2369, | \theTest 1564 |
| 2370, 2371, 2372, 2373, 2374, | \TimeDescription . . . 1508, 1511, 1515 |
| 2375, 2376, 2377, 2378, 2379, | \TMP@EnsureCode 79, |
| 2380, 2381, 2382, 2383, 2387, | 86, 87, 88, 89, 90, 91, 92, 93, |
| 2394, 2395, 2396, 2397, 2398, | 94, 95, 96, 97, 98, 99, 100, 101 |
| 2399, 2400, 2401, 2402, 2403, | \typeout 1503 |
| 2404, 2405, 2406, 2407, 2408, | |
| 2409, 2410, 2411, 2412, 2413, | U |
| 2414, 2415, 2450, 2456, 2457, | \uccode 820 |
| 2458, 2459, 2461, 2462, 2463, | \uppercase 821 |
| 2464, 2465, 2466, 2468, 2469, | \usepackage 1483, 1492 |
| 2470, 2471, 2476, 2477, 2478, | |
| 2479, 2488, 2492, 2493, 2494, | W |
| 2495, 2496, 2497, 2504, 2505, | \wd 1539, 1565 |
| 2506, 2507, 2508, 2509, 2510, | \write 20, 46 |
| 2511, 2512, 2513, 2515, 2516, | |
| 2517, 2518, 2519, 2520, 2521, | X |
| 2522, 2523, 2524, 2525, 2526, 2527 | \x 10, 11, 14, 19, 23, 25, |
| \Test@ 2125, 2127 | 45, 50, 60, 66, 74, 2200, 2203, |
| \TestError 1544, 1570, 1985, 2419, 2428 | 2209, 2222, 2358, 2359, 2360, 2536 |
| \TestErrorNegativeIndex | |
| 1569, 1922, 1941, 1965 | Z |
| \TestErrorNegInd | \z@ 1501 |
| 2427, 2439, 2440, 2441, | \zap@space 2358 |