The **bitset** package

Heiko Oberdiek*

<heiko.oberdiek at googlemail.com>

2016/05/16 v1.2

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

---

*Please report any issues at https://github.com/ho-tex/oberdiek/issues

1 Documentation

1.1 Introduction

Annotations in the PDF format know entries whose values are integers. This 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 snipsets 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 to 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 consists 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 enoded as `0` for cleared/disabled or `1` for set/enabled.

1.3   Design principles

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

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

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

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

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

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

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

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

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

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

1.4 Operator overview

Miscellaneous

\bitset**Reset** ⟨*BitSet*⟩

\bitset**Let** ⟨*BitSet A*⟩ ⟨*BitSet B*⟩

Import

\bitset**SetBin**, \bitsetSetOct, \bitset**SetHex** ⟨*BitSet*⟩ ⟨*Value*⟩

\bitset**SetDec** ⟨*BitSet*⟩ ⟨*Value*⟩

Export[a]

\bitset**GetBin**, \bitset**GetOct**, \bitset**GetHex** ⟨*BitSet*⟩ ⟨*MinSize*⟩

\bitset**GetDec** ⟨*BitSet*⟩

Logical operators

\bitset**And**, \bitset**AndNot** ⟨*BitSet A*⟩ ⟨*BitSet B*⟩

\bitset**Or**, \bitset**Xor** ⟨*BitSet A*⟩ ⟨*BitSet B*⟩

Shifting

\bitset**ShiftLeft**, \bitset**ShiftRight** ⟨*BitSet*⟩ ⟨*ShiftAmount*⟩

Bit manipulation

\bitset**Clear**, \bitset**Set**, \bitset**Flip** ⟨*BitSet*⟩ ⟨*Index*⟩

\bitset**SetValue** ⟨*BitSet*⟩ ⟨*Index*⟩ ⟨*Value*⟩

\bitset**ClearRange**, \bitset**SetRange**, \bitset**FlipRange**

⟨*BitSet*⟩ ⟨*IndexFrom*⟩ ⟨*IndexTo*⟩

\bitset**SetValueRange** ⟨*BitSet*⟩ ⟨*IndexFrom*⟩ ⟨*IndexTo*⟩

Bit retrieval[a]

\bitset**Get** ⟨*BitSet*⟩ ⟨*Index*⟩

\bitset**NextClearBit**, \bitset**NextSetBit** ⟨*BitSet*⟩ ⟨*Index*⟩

\bitset**GetSetBitList** ⟨*BitSet*⟩

Bit set properties

\bitset**Size**, \bitset**Cardinality** ⟨*BitSet*⟩

Queries[b]

\bitset**IsDefined**, \bitset**IsEmpty** ⟨*BitSet*⟩ ⟨*Then*⟩ ⟨*Else*⟩

\bitset**Equals**, \bitset**Intersects** ⟨*BitSet A*⟩ ⟨*BitSet B*⟩ ⟨*Then*⟩ ⟨*Else*⟩

\bitset**Query** ⟨*BitSet*⟩ ⟨*Index*⟩ ⟨*Then*⟩ ⟨*Else*⟩

---

[a]Macros are expandable, full expansion by two steps.

[b]Macros are expandable.

1.5 Package loading

The package can be used as normal LATEX package:

    \usepackage{bitset}

Also plain TEX is supported:

    \input bitset.sty\relax

1.6 Operators

The following macros work on and with bit sets. A bit set ⟨*BitSet*⟩ 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 LATEX.

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 {⟨BitSet⟩}`

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 {⟨BitSet A⟩} {⟨BitSet B⟩}`

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

Note: If ⟨*BitSet A*⟩ exists, it will be overwritten.

1.6.2 Import

`\bitsetSetBin {⟨BitSet⟩} {⟨BinaryNumber⟩}`
`\bitsetSetOct {⟨BitSet⟩} {⟨OctalNumber⟩}`
`\bitsetSetHex {⟨BitSet⟩} {⟨HexadecimalNumber⟩}`

The numbers are interpreted as bit vectors and the flags in the bit ⟨*BitSet*⟩ set are set accordingly. These numeric arguments are the only arguments where spaces are allowed. Then the numbers are easier to read.

`\bitsetSetDec {⟨BitSet⟩} {⟨DecimalNumber⟩}`

Macro `\bitsetSetDec` uses ⟨*DecimalNumber*⟩ to set the bit set ⟨*BitSet*⟩. 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 {⟨BitSet⟩} {⟨MinSize⟩}`
`\bitsetGetOct {⟨BitSet⟩} {⟨MinSize⟩}`
`\bitsetGetHex {⟨BitSet⟩} {⟨MinSize⟩}`

These macros returns the bit set as binary, octal or hexadecimal number. If the bit size is smaller than ⟨*MinSize*⟩ the gap is filled with leading zeros. Example:

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

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

`\bitsetGetDec {⟨BitSet⟩}`

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

1.6.4 Logical operators

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

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

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

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

`\bitsetOr {⟨BitSet A⟩} {⟨BitSet B⟩}`

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

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

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

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, movint the bits to lower positions. The number will become smaller, the lowest bits are lost.

If the ⟨ShiftAmount⟩ is negative, it reverts the meaning of the shift operation. A left shift becomes a right shift. A ⟨ShiftAmount⟩ of zero is ignored.

1.6.6 Bit manipulation

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

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

$\boxed{\texttt{\textbackslash bitsetSetValue }\{\langle \textit{BitSet}\rangle\} \; \{\langle \textit{Index}\rangle\} \; \{\langle \textit{Bit}\rangle\}}$

Macro `\bitsetSetValue` puts bit $\langle \textit{Bit}\rangle$ at position $\langle \textit{Index}\rangle$ in bit set $\langle \textit{BitSet}\rangle$. $\langle \textit{Bit}\rangle$ must be a valid TeX number equals to zero (disabled/cleared) or one (enabled/set).

1.6.7 Bit retrieval

$\boxed{\texttt{\textbackslash bitsetGet }\{\langle \textit{BitSet}\rangle\} \; \{\langle \textit{Index}\rangle\}}$

Macro `\bitsetGet` extracts the status of the bit at position $\langle \textit{Index}\rangle$ in bit set $\langle \textit{BitSet}\rangle$. 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.

$\boxed{\texttt{\textbackslash bitsetNextClearBit }\{\langle \textit{BitSet}\rangle\} \; \{\langle \textit{Index}\rangle\}}$

Starting at position $\langle \textit{Index}\rangle$ (inclusive) the bits are inspected. The first position without a set bit is returned. Possible results are decimal numbers: $\langle \textit{Index}\rangle$, $\langle \textit{Index}\rangle + 1$, …, (∞)

$\boxed{\texttt{\textbackslash bitsetNextSetBit }\{\langle \textit{BitSet}\rangle\} \; \{\langle \textit{Index}\rangle\}}$

Starting at position $\langle \textit{Index}\rangle$ (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, $\langle \textit{Index}\rangle$, $\langle \textit{Index}\rangle + 1$, …, (∞)

$\boxed{\texttt{\textbackslash bitsetGetSetBitList }\{\langle \textit{BitSet}\rangle\}}$

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

$\boxed{\texttt{\textbackslash bitsetSize }\{\langle \textit{BitSet}\rangle\}}$

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.

$\boxed{\texttt{\textbackslash bitsetCardinality }\{\langle \textit{BitSet}\rangle\}}$

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.

$\boxed{\texttt{\textbackslash bitsetIsDefined }\{\langle \textit{BitSet}\rangle\} \; \{\langle \textit{Then}\rangle\} \; \{\langle \textit{Else}\rangle\}}$

If the bit set with the name $\langle \textit{BitSet}\rangle$ exists the code given in $\langle \textit{Then}\rangle$ is executed, otherwise $\langle \textit{Else}\rangle$ is used.

$\boxed{\texttt{\textbackslash bitsetIsEmpty} \ \{\langle BitSet\rangle\} \ \{\langle Then\rangle\} \ \{\langle Else\rangle\}}$

If the bit set ⟨*BitSet*⟩ exists and at least one bit is set/enabled, the code in ⟨*Then*⟩ is executed, ⟨*Else*⟩ otherwise.

$\boxed{\texttt{\textbackslash bitsetEquals} \ \{\langle BitSet\ A\rangle\} \ \{\langle BitSet\ B\rangle\} \ \{\langle Then\rangle\} \ \{\langle Else\rangle\}}$

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, symmeric, and transitive, enough for an equivalent relation.

$\boxed{\texttt{\textbackslash bitsetIntersects} \ \{\langle BitSet\ A\rangle\} \ \{\langle BitSet\ B\rangle\} \ \{\langle Then\rangle\} \ \{\langle Else\rangle\}}$

If and only if ⟨*BitSet A*⟩ and ⟨*BitSet B*⟩ have at least one bit at the same position that is set, then code part ⟨*Then*⟩ is executed.

$\boxed{\texttt{\textbackslash bitsetQuery} \ \{\langle BitSet\rangle\} \ \{\langle Index\rangle\} \ \{\langle Then\rangle\} \ \{\langle Else\rangle\}}$

It's just a wrapper for `\bitsetGet`. If the bit at position ⟨*Index*⟩ is enabled, code ⟨*Then*⟩ 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 LaTeX.

```
2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3   \catcode13=5 % ^^M
4   \endlinechar=13 %
5   \catcode35=6 % #
6   \catcode39=12 % '
7   \catcode44=12 % ,
8   \catcode45=12 % -
9   \catcode46=12 % .
10  \catcode58=12 % :
11  \catcode64=11 % @
12  \catcode123=1 % {
13  \catcode125=2 % }
14  \expandafter\let\expandafter\x\csname ver@bitset.sty\endcsname
15  \ifx\x\relax % plain-TeX, first loading
16  \else
17    \def\empty{}%
18    \ifx\x\empty % LaTeX, first loading,
19      % variable is initialized, but \ProvidesPackage not yet seen
20    \else
21      \expandafter\ifx\csname PackageInfo\endcsname\relax
22        \def\x#1#2{%
23          \immediate\write-1{Package #1 Info: #2.}%
24        }%
```

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

Package identification:

```
33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34  \catcode13=5 % ^^M
35  \endlinechar=13 %
36  \catcode35=6 % #
37  \catcode39=12 % '
38  \catcode40=12 % (
39  \catcode41=12 % )
40  \catcode44=12 % ,
41  \catcode45=12 % -
42  \catcode46=12 % .
43  \catcode47=12 % /
44  \catcode58=12 % :
45  \catcode64=11 % @
46  \catcode91=12 % [
47  \catcode93=12 % ]
48  \catcode123=1 % {
49  \catcode125=2 % }
50  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51    \def\x#1#2#3[#4]{\endgroup
52      \immediate\write-1{Package: #3 #4}%
53      \xdef#1{#4}%
54    }%
55  \else
56    \def\x#1#2[#3]{\endgroup
57      #2[{#3}]%
58      \ifx#1\@undefined
59        \xdef#1{#3}%
60      \fi
61      \ifx#1\relax
62        \xdef#1{#3}%
63      \fi
64    }%
65  \fi
66 \expandafter\x\csname ver@bitset.sty\endcsname
67 \ProvidesPackage{bitset}%
68   [2016/05/16 v1.2 Handle bit-vector datatype (HO)]%
```

2.2 Catcodes

```
69 \begingroup\catcode61\catcode48\catcode32=10\relax%
70  \catcode13=5 % ^^M
71  \endlinechar=13 %
72  \catcode123=1 % {
73  \catcode125=2 % }
74  \catcode64=11 % @
75  \def\x{\endgroup
76    \expandafter\edef\csname BitSet@AtEnd\endcsname{%
77      \endlinechar=\the\endlinechar\relax
78      \catcode13=\the\catcode13\relax
79      \catcode32=\the\catcode32\relax
80      \catcode35=\the\catcode35\relax
81      \catcode61=\the\catcode61\relax
82      \catcode64=\the\catcode64\relax
```

```
83    \catcode123=\the\catcode123\relax
84    \catcode125=\the\catcode125\relax
85   }%
86  }%
87 \x\catcode61\catcode48\catcode32=10\relax%
88 \catcode13=5 % ^^M
89 \endlinechar=13 %
90 \catcode35=6 % #
91 \catcode64=11 % @
92 \catcode123=1 % {
93 \catcode125=2 % }
94 \def\TMP@EnsureCode#1#2{%
95  \edef\BitSet@AtEnd{%
96    \BitSet@AtEnd
97    \catcode#1=\the\catcode#1\relax
98  }%
99  \catcode#1=#2\relax
100 }
101 \TMP@EnsureCode{33}{12}% !
102 \TMP@EnsureCode{39}{12}% '
103 \TMP@EnsureCode{40}{12}% (
104 \TMP@EnsureCode{41}{12}% )
105 \TMP@EnsureCode{42}{12}% *
106 \TMP@EnsureCode{43}{12}% +
107 \TMP@EnsureCode{44}{12}% ,
108 \TMP@EnsureCode{45}{12}% -
109 \TMP@EnsureCode{46}{12}% .
110 \TMP@EnsureCode{47}{12}% /
111 \TMP@EnsureCode{58}{11}% : (letter!)
112 \TMP@EnsureCode{60}{12}% <
113 \TMP@EnsureCode{62}{12}% >
114 \TMP@EnsureCode{63}{14}% ? (comment!)
115 \TMP@EnsureCode{91}{12}% [
116 \TMP@EnsureCode{93}{12}% ]
117 \TMP@EnsureCode{96}{12}% `
118 \edef\BitSet@AtEnd{\BitSet@AtEnd\noexpand\endinput}
119 \begingroup\expandafter\expandafter\expandafter\endgroup
120 \expandafter\ifx\csname BitSet@TestMode\endcsname\relax
121 \else
122   \catcode63=9 % ? (ignore)
123 \fi
124 ? \let\BitSet@@TestMode\BitSet@TestMode
```

2.3 Package loading

```
125 \begingroup\expandafter\expandafter\expandafter\endgroup
126 \expandafter\ifx\csname RequirePackage\endcsname\relax
127  \def\TMP@RequirePackage#1[#2]{%
128    \begingroup\expandafter\expandafter\expandafter\endgroup
129    \expandafter\ifx\csname ver@#1.sty\endcsname\relax
130      \input #1.sty\relax
131    \fi
132  }%
133  \TMP@RequirePackage{infwarerr}[2007/09/09]%
134  \TMP@RequirePackage{intcalc}[2007/09/27]%
135  \TMP@RequirePackage{bigintcalc}[2007/09/27]%
136 \else
137  \RequirePackage{infwarerr}[2007/09/09]%
138  \RequirePackage{intcalc}[2007/09/27]%
139  \RequirePackage{bigintcalc}[2007/09/27]%
140 \fi
```

2.4 Help macros

2.4.1 Number constant

\BitSet@MaxSize

```
141 \def\BitSet@MaxSize{2147483647}%
```

2.4.2 General basic macros

\BitSet@Empty

```
142 \def\BitSet@Empty{}
```

\BitSet@FirstOfOne

```
143 \def\BitSet@FirstOfOne#1{#1}
```

\BitSet@Gobble

```
144 \def\BitSet@Gobble#1{}
```

\BitSet@FirstOfTwo

```
145 \def\BitSet@FirstOfTwo#1#2{#1}
```

\BitSet@SecondOfTwo

```
146 \def\BitSet@SecondOfTwo#1#2{#2}
```

\BitSet@Space

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

\BitSet@ZapSpace

```
148 \def\BitSet@ZapSpace#1 #2{%
149   #1%
150   \ifx\BitSet@Empty#2%
151   \else
152     \expandafter\BitSet@ZapSpace
153   \fi
154   #2%
155 }
```

2.4.3 Tail recursion

\BitSet@Fi

```
156 \let\BitSet@Fi\fi
```

\BitSet@AfterFi

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

\BitSet@AfterFiFi

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

\BitSet@AfterFiFiFi

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

2.4.4 Check macros

\BitSet@IfUndefined

```
160 \def\BitSet@IfUndefined#1{%
161   \expandafter\ifx\csname BS@#1\endcsname\relax
162     \expandafter\BitSet@FirstOfTwo
163   \else
164     \expandafter\BitSet@SecondOfTwo
165   \fi
166 }
```

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

```
167 \def\BitSet@CheckIndex#1#2#3{%
168   \BitSet@IfUndefined{#2}{\bitsetReset{#2}}{}%
169   \expandafter\expandafter\expandafter\BitSet@@CheckIndex
170   \intcalcNum{#3}!%
171   {#2}{#1}%
172 }
```

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

```
173 \def\BitSet@@CheckIndex#1!#2#3{%
174   \ifnum#1<0 %
175     \BitSet@AfterFi{%
176       \@PackageError{bitset}{%
177         Invalid negative index (#1)%
178       }\@ehc
179     }%
180   \else
181     \BitSet@AfterFi{%
182       #3{#2}{#1}%
183     }%
184   \BitSet@Fi
185 }
```

2.5   Miscellaneous

\bitsetReset

```
186 \def\bitsetReset#1{%
187   \expandafter\def\csname BS@#1\endcsname{0}%
188 }
```

\bitsetLet

```
189 \def\bitsetLet#1#2{%
190   \BitSet@IfUndefined{#2}{%
191     \bitsetReset{#1}%
192   }{%
193     \expandafter\let\csname BS@#1\expandafter\endcsname
194             \csname BS@#2\endcsname
195   }%
196 }
```

2.6   Import

2.6.1   From binary number

\bitsetSetBin

```
197 \def\bitsetSetBin#1#2{%
198   \edef\BitSet@Temp{#2}%
199   \edef\BitSet@Temp{%
200     \expandafter\expandafter\expandafter\BitSet@ZapSpace
201     \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
202   }%
203   \edef\BitSet@Temp{%
204     \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
205   }%
206   \ifx\BitSet@Temp\BitSet@Empty
207     \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
208   \else
```

```
209    \expandafter\edef\csname BS@#1\endcsname{%
210      \expandafter\BitSet@Reverse\BitSet@Temp!%
211    }%
212  \fi
213 }
```

\BitSet@KillZeros

```
214 \def\BitSet@KillZeros#1{%
215  \ifx#10%
216    \expandafter\BitSet@KillZeros
217  \else
218    #1%
219  \fi
220 }
```

\BitSet@Reverse

```
221 \def\BitSet@Reverse#1#2!{%
222  \ifx\\#2\\%
223    #1%
224  \else
225    \BitSet@AfterFi{%
226      \BitSet@Reverse#2!#1%
227    }%
228  \BitSet@Fi
229 }
```

2.6.2 From octal/hex number

\bitsetSetOct

```
230 \def\bitsetSetOct{%
231  \BitSet@SetOctHex\BitSet@FromFirstOct
232 }
```

\bitsetSetHex

```
233 \def\bitsetSetHex{%
234  \BitSet@SetOctHex\BitSet@FromFirstHex
235 }
```

\BitSet@SetOctHex

```
236 \def\BitSet@SetOctHex#1#2#3{%
237  \edef\BitSet@Temp{#3}%
238  \edef\BitSet@Temp{%
239    \expandafter\expandafter\expandafter\BitSet@ZapSpace
240    \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
241  }%
242  \edef\BitSet@Temp{%
243    \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
244  }%
245  \ifx\BitSet@Temp\BitSet@Empty
246    \expandafter\let\csname BS@#2\endcsname\BitSet@Zero
247  \else
248    \edef\BitSet@Temp{%
249      \expandafter#1\BitSet@Temp!%
250    }%
251    \ifx\BitSet@Temp\BitSet@Empty
252      \expandafter\let\csname BS@#2\endcsname\BitSet@Zero
253    \else
254      \expandafter\edef\csname BS@#2\endcsname{%
255        \expandafter\BitSet@Reverse\BitSet@Temp!%
256      }%
257    \fi
258  \fi
259 }
```

\BitSet@FromFirstOct

```
260 \def\BitSet@FromFirstOct#1{%
261   \ifx#1!%
262   \else
263     \ifcase#1 \BitSet@AfterFiFi\BitSet@FromFirstOct
264     \or 1%
265     \or 10%
266     \or 11%
267     \or 100%
268     \or 101%
269     \or 110%
270     \or 111%
271     \else \BitSetError:WrongOctalDigit%
272     \fi
273     \expandafter\BitSet@FromOct
274   \BitSet@Fi
275 }
```

\BitSet@FromOct

```
276 \def\BitSet@FromOct#1{%
277   \ifx#1!%
278   \else
279     \ifcase#1 000%
280     \or 001%
281     \or 010%
282     \or 011%
283     \or 100%
284     \or 101%
285     \or 110%
286     \or 111%
287     \else \BitSetError:WrongOctalDigit%
288     \fi
289     \expandafter\BitSet@FromOct
290   \fi
291 }
```

\BitSet@FromFirstHex

```
292 \def\BitSet@FromFirstHex#1{%
293   \ifx#1!%
294   \else
295     \ifx#10%
296       \BitSet@AfterFiFi\BitSet@FromFirstHex
297     \fi
298     \expandafter\ifx\csname BitSet@Hex#1\endcsname\relax
299       \BitSetError:InvalidHexDigit%
300     \else
301       \expandafter\expandafter\expandafter\BitSet@KillZeros
302       \csname BitSet@Hex#1\endcsname
303     \fi
304     \expandafter\BitSet@FromHex
305   \BitSet@Fi
306 }
```

\BitSet@FromHex

```
307 \def\BitSet@FromHex#1{%
308   \ifx#1!%
309   \else
310     \expandafter\ifx\csname BitSet@Hex#1\endcsname\relax
311       \BitSetError:InvalidHexDigit%
312     \else
313       \csname BitSet@Hex#1\endcsname
314     \fi
```

15

```
315   \expandafter\BitSet@FromHex
316  \fi
317 }
```

\BitSet@Hex[0..F]

```
318 \def\BitSet@Temp#1{%
319   \expandafter\def\csname BitSet@Hex#1\endcsname
320 }
321 \BitSet@Temp 0{0000}%
322 \BitSet@Temp 1{0001}%
323 \BitSet@Temp 2{0010}%
324 \BitSet@Temp 3{0011}%
325 \BitSet@Temp 4{0100}%
326 \BitSet@Temp 5{0101}%
327 \BitSet@Temp 6{0110}%
328 \BitSet@Temp 7{0111}%
329 \BitSet@Temp 8{1000}%
330 \BitSet@Temp 9{1001}%
331 \BitSet@Temp A{1010}%
332 \BitSet@Temp B{1011}%
333 \BitSet@Temp C{1100}%
334 \BitSet@Temp D{1101}%
335 \BitSet@Temp E{1110}%
336 \BitSet@Temp F{1111}%
337 \BitSet@Temp a{1010}%
338 \BitSet@Temp b{1011}%
339 \BitSet@Temp c{1100}%
340 \BitSet@Temp d{1101}%
341 \BitSet@Temp e{1110}%
342 \BitSet@Temp f{1111}%
```

2.6.3   From decimal number

\bitsetSetDec

```
343 \def\bitsetSetDec#1#2{%
344  \edef\BitSet@Temp{#2}%
345  \edef\BitSet@Temp{%
346   \expandafter\expandafter\expandafter\BitSet@ZapSpace
347   \expandafter\BitSet@Temp\BitSet@Space\BitSet@Empty
348  }%
349  \edef\BitSet@Temp{%
350   \expandafter\BitSet@KillZeros\BitSet@Temp\BitSet@Empty
351  }%
352  \ifx\BitSet@Temp\BitSet@Empty
353   \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
354  \else
355   \ifcase\bigintcalcSgn{\BitSet@Temp} %
356    \expandafter\let\csname BS@#1\endcsname\BitSet@Zero
357   \or
358    \ifnum\bigintcalcCmp\BitSet@Temp\BitSet@MaxSize>0 %
359     \expandafter\edef\csname BS@#1\endcsname{%
360      \expandafter\BitSet@SetDecBig\BitSet@Temp!%
361     }%
362    \else
363     \expandafter\edef\csname BS@#1\endcsname{%
364      \expandafter\BitSet@SetDec\BitSet@Temp!%
365     }%
366    \fi
367   \else
368    \@PackageError{bitset}{%
369     Bit sets cannot be negative%
370    }\@ehc
```

```
371      \fi
372   \fi
373 }
```

\BitSet@SetDecBig

```
374 \def\BitSet@SetDecBig#1#2#3#4#5#6#7#8#9!{%
375   \ifx\\#9\\%
376     \BitSet@SetDec#1#2#3#4#5#6#7#8!%
377   \else
378     \ifcase\BigIntCalcOdd#1#2#4#5#6#7#8#9! %
379       0%
380     \or
381       1%
382 ?   \else\BitSetError:ThisCannotHappen%
383     \fi
384     \BitSet@AfterFi{%
385       \expandafter\expandafter\expandafter\BitSet@SetDecBig
386       \BigIntCalcShr#1#2#3#4#5#6#7#8#9!!%
387     }%
388   \BitSet@Fi
389 }
```

\BitSet@SetDec

```
390 \def\BitSet@SetDec#1!{%
391   \ifcase#1 %
392   \or 1%
393   \else
394     \ifodd#1 %
395       1%
396     \else
397       0%
398     \fi
399     \BitSet@AfterFi{%
400       \expandafter\expandafter\expandafter\BitSet@SetDec
401       \IntCalcShr#1!!%
402     }%
403   \BitSet@Fi
404 }
```

2.7 Export

2.7.1 To binary number

\bitsetGetBin

```
405 \def\bitsetGetBin#1#2{%
406   \romannumeral0%
407   \expandafter\expandafter\expandafter\BitSet@@GetBin
408   \intcalcNum{#2}!{#1}%
409 }
```

\BitSet@@GetBin

```
410 \def\BitSet@@GetBin#1!#2{%
411   \BitSet@IfUndefined{#2}{%
412     \ifnum#1>1 %
413       \BitSet@AfterFi{%
414         \expandafter\expandafter\expandafter\BitSet@Fill
415         \IntCalcDec#1!!0%
416       }%
417     \else
418       \BitSet@AfterFi{ 0}%
419     \BitSet@Fi
420   }{%
```

17

```
421      \expandafter\expandafter\expandafter\BitSet@NumBinRev
422      \expandafter\expandafter\expandafter1%
423      \expandafter\expandafter\expandafter!%
424      \csname BS@#2\endcsname!!#1!%
425    }%
426 }
```

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

```
427 \def\BitSet@Fill#1!{%
428   \ifnum#1>0 %
429     \BitSet@AfterFi{%
430       \expandafter\expandafter\expandafter\BitSet@Fill
431       \IntCalcDec#1!!0%
432     }%
433   \else
434     \BitSet@AfterFi{ }%
435   \BitSet@Fi
436 }
```

\BitSet@NumBinRev #1: bit counter (including #2)
                  #2#3: reverted number
                  #4: result
                  #5: min size

```
437 \def\BitSet@NumBinRev#1!#2#3!{%
438   \ifx\\#3\\%
439     \BitSet@AfterFi{%
440       \BitSet@NumBinFill#1!#2%
441     }%
442   \else
443     \BitSet@AfterFi{%
444       \expandafter\expandafter\expandafter\BitSet@NumBinRev
445       \IntCalcInc#1!!#3!#2%
446     }%
447   \BitSet@Fi
448 }
```

\BitSet@NumBinFill

```
449 \def\BitSet@NumBinFill#1!#2!#3!{%
450   \ifnum#3>#1 %
451     \BitSet@AfterFi{%
452       \expandafter\expandafter\expandafter\BitSet@Fill
453       \IntCalcSub#3!#1!!#2%
454     }%
455   \else
456     \BitSet@AfterFi{ #2}%
457   \BitSet@Fi
458 }
```

2.7.2 To octal/hexadecimal number

\bitsetGetOct

```
459 \def\bitsetGetOct#1#2{%
460   \romannumeral0%
461   \bitsetIsEmpty{#1}{%
462     \expandafter\expandafter\expandafter\BitSet@@GetOctHex
463     \intcalcNum{#2}!3!230%
464   }{%
465     \expandafter\expandafter\expandafter\BitSet@@GetOct
466     \expandafter\expandafter\expandafter1%
467     \expandafter\expandafter\expandafter!%
468     \expandafter\expandafter\expandafter!%
```

```
469    \csname BS@#1\endcsname00%
470    \BitSet@Empty\BitSet@Empty\BitSet@Empty!{#2}%
471  }%
472 }
```

\bitsetGetHex

```
473 \def\bitsetGetHex#1#2{%
474   \romannumeral0%
475   \bitsetIsEmpty{#1}{%
476     \expandafter\expandafter\expandafter\BitSet@@GetOctHex
477     \intcalcNum{#2}!4!340%
478   }{%
479     \expandafter\expandafter\expandafter\BitSet@@GetHex
480     \expandafter\expandafter\expandafter1%
481     \expandafter\expandafter\expandafter!%
482     \expandafter\expandafter\expandafter!%
483     \csname BS@#1\endcsname000%
484     \BitSet@Empty\BitSet@Empty\BitSet@Empty\BitSet@Empty!{#2}%
485   }%
486 }
```

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

```
487 \def\BitSet@@GetOct#1!#2!#3#4#5{%
488   \ifx#5\BitSet@Empty
489     \BitSet@AfterFi{%
490       \expandafter\expandafter\expandafter\BitSet@GetOctHex
491       \IntCalcDec#1!!#2!23%
492     }%
493   \else
494     \BitSet@AfterFi{%
495       \expandafter\expandafter\expandafter\BitSet@@GetOct
496       \number\IntCalcInc#1!\expandafter\expandafter\expandafter!%
497       \csname BitSet@Oct#5#4#3\endcsname#2!%
498     }%
499   \BitSet@Fi
500 }
```

\BitSet@Oct[000..111]

```
501 \def\BitSet@Temp#1#2#3#4{%
502   \expandafter\def\csname BitSet@Oct#1#2#3\endcsname{#4}%
503 }
504 \BitSet@Temp0000%
505 \BitSet@Temp0011%
506 \BitSet@Temp0102%
507 \BitSet@Temp0113%
508 \BitSet@Temp1004%
509 \BitSet@Temp1015%
510 \BitSet@Temp1106%
511 \BitSet@Temp1117%
```

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

```
512 \def\BitSet@@GetHex#1!#2!#3#4#5#6{%
513   \ifx#6\BitSet@Empty
514     \BitSet@AfterFi{%
515       \expandafter\expandafter\expandafter\BitSet@GetOctHex
516       \IntCalcDec#1!!#2!34%
517     }%
518   \else
```

```
519    \BitSet@AfterFi{%
520      \expandafter\expandafter\expandafter\BitSet@@GetHex
521      \number\IntCalcInc#1!\expandafter\expandafter\expandafter!%
522      \csname BitSet@Hex#6#5#4#3\endcsname#2!%
523    }%
524  \BitSet@Fi
525 }
```

\BitSet@Hex[0000..1111]

```
526 \def\BitSet@Temp#1#2#3#4#5{%
527  \expandafter\def\csname BitSet@Hex#1#2#3#4\endcsname{#5}%
528 }
529 \BitSet@Temp00000%
530 \BitSet@Temp00011%
531 \BitSet@Temp00102%
532 \BitSet@Temp00113%
533 \BitSet@Temp01004%
534 \BitSet@Temp01015%
535 \BitSet@Temp01106%
536 \BitSet@Temp01117%
537 \BitSet@Temp10008%
538 \BitSet@Temp10019%
539 \BitSet@Temp1010A%
540 \BitSet@Temp1011B%
541 \BitSet@Temp1100C%
542 \BitSet@Temp1101D%
543 \BitSet@Temp1110E%
544 \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

```
545 \def\BitSet@GetOctHex#1!#2!#3#4#5!#6{%
546  \expandafter\BitSet@@GetOctHex
547  \number\intcalcNum{#6}\expandafter\expandafter\expandafter!%
548  \IntCalcMul#1!#4!!#3#4#2%
549 }
```

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

```
550 \def\BitSet@@GetOctHex#1!#2!#3#4{%
551  \ifnum#1>#2 %
552    \BitSet@AfterFi{%
553      \expandafter\expandafter\expandafter\expandafter
554      \expandafter\expandafter\expandafter\BitSet@Fill
555      \expandafter\IntCalcDiv\number
556      \expandafter\expandafter\expandafter\IntCalcAdd
557      \IntCalcSub#1!#2!!#3!!#4!!%
558    }%
559  \else
560    \BitSet@AfterFi{ }%
561  \BitSet@Fi
562 }
```

2.7.3   To decimal number

\bitsetGetDec

```
563 \def\bitsetGetDec#1{%
564   \romannumeral0%
565   \BitSet@IfUndefined{#1}{ 0}{%
566     \expandafter\expandafter\expandafter\BitSet@GetDec
567     \csname BS@#1\endcsname!%
568   }%
569 }
```

\BitSet@GetDec

```
570 \def\BitSet@GetDec#1#2!{%
571   \ifx\\#2\\%
572     \BitSet@AfterFi{ #1}%
573   \else
574     \BitSet@AfterFi{%
575       \BitSet@@GetDec2!#1!#2!%
576     }%
577   \BitSet@Fi
578 }
```

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

```
579 \def\BitSet@@GetDec#1!#2!#3#4!{%
580   \ifx\\#4\\%
581     \ifx#31%
582       \BitSet@AfterFiFi{%
583         \expandafter\expandafter\expandafter\BitSet@Space
584         \IntCalcAdd#1!#2!%
585       }%
586     \else
587       \BitSet@AfterFiFi{ #2}%
588     \fi
589   \else
590     \ifx#31%
591       \BitSet@AfterFiFi{%
592         \csname BitSet@N#1%
593         \expandafter\expandafter\expandafter\endcsname
594         \IntCalcAdd#1!#2!!#4!%
595       }%
596     \else
597       \BitSet@AfterFiFi{%
598         \csname BitSet@N#1\endcsname#2!#4!%
599       }%
600     \fi
601   \BitSet@Fi
602 }
```

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

```
603 \def\BitSet@Temp#1#2{%
604   \expandafter\def\csname BitSet@N#1\endcsname{%
605     \BitSet@@GetDec#2!%
606   }%
607 }
608 \BitSet@Temp{1}{2}
609 \BitSet@Temp{2}{4}
610 \BitSet@Temp{4}{8}
611 \BitSet@Temp{8}{16}
612 \BitSet@Temp{16}{32}
613 \BitSet@Temp{32}{64}
614 \BitSet@Temp{64}{128}
615 \BitSet@Temp{128}{256}
616 \BitSet@Temp{256}{512}
```

```
617 \BitSet@Temp{512}{1024}
618 \BitSet@Temp{1024}{2048}
619 \BitSet@Temp{2048}{4096}
620 \BitSet@Temp{4096}{8192}
621 \BitSet@Temp{8192}{16384}
622 \BitSet@Temp{16384}{32768}
623 \BitSet@Temp{32768}{65536}
624 \BitSet@Temp{65536}{131072}
625 \BitSet@Temp{131072}{262144}
626 \BitSet@Temp{262144}{524288}
627 \BitSet@Temp{524288}{1048576}
628 \BitSet@Temp{1048576}{2097152}
629 \BitSet@Temp{2097152}{4194304}
630 \BitSet@Temp{4194304}{8388608}
631 \BitSet@Temp{8388608}{16777216}
632 \BitSet@Temp{16777216}{33554432}
633 \BitSet@Temp{33554432}{67108864}
634 \BitSet@Temp{67108864}{134217728}
635 \BitSet@Temp{134217728}{268435456}
636 \BitSet@Temp{268435456}{536870912}
637 \BitSet@Temp{536870912}{1073741824}
```

\BitSet@N1073741824

```
638 \expandafter\def\csname BitSet@N1073741824\endcsname{%
639   \BitSet@GetDecBig2147483648!%
640 }%
```

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

```
641 \def\BitSet@GetDecBig#1!#2!#3#4!{%
642   \ifx\\#4\\%
643     \ifx#31%
644       \BitSet@AfterFiFi{%
645         \expandafter\expandafter\expandafter\BitSet@Space
646         \BigIntCalcAdd#1!#2!%
647       }%
648     \else
649       \BitSet@AfterFiFi{ #2}%
650     \fi
651   \else
652     \ifx#31%
653       \BitSet@AfterFiFi{%
654         \expandafter\expandafter\expandafter\BitSet@@GetDecBig
655         \BigIntCalcAdd#1!#2!!#1!#4!%
656       }%
657     \else
658       \BitSet@AfterFiFi{%
659         \expandafter\expandafter\expandafter\BitSet@GetDecBig
660         \BigIntCalcShl#1!!#2!#4!%
661       }%
662     \fi
663   \BitSet@Fi
664 }
```

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

```
665 \def\BitSet@@GetDecBig#1!#2!{%
666   \expandafter\expandafter\expandafter\BitSet@GetDecBig
667   \BigIntCalcShl#2!!#1!%
668 }
```

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

```
669 \def\bitsetAnd#1#2{%
670   \bitsetIsEmpty{#1}{%
671     \bitsetReset{#1}%
672   }{%
673     \bitsetIsEmpty{#2}{%
674       \bitsetReset{#1}%
675     }{%
676       \expandafter\edef\csname BS@#1\endcsname{%
677         \expandafter\expandafter\expandafter\BitSet@And
678         \csname BS@#1\expandafter\expandafter\expandafter\endcsname
679         \expandafter\expandafter\expandafter!%
680         \csname BS@#2\endcsname!!%
681       }%
682       \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
683         \bitsetReset{#1}%
684       \fi
685     }%
686   }%
687 }
```

\BitSet@And

```
688 \def\BitSet@And#1#2!#3#4!#5!{%
689   \ifx\\#2\\%
690     \ifnum#1#3=11 #51\fi
691   \else
692     \ifx\\#4\\%
693       \ifnum#1#3=11 #51\fi
694     \else
695       \ifnum#1#3=11 %
696         #51%
697         \BitSet@AfterFiFiFi{%
698           \BitSet@And#2!#4!!%
699         }%
700       \else
701         \BitSet@AfterFiFiFi{%
702           \BitSet@And#2!#4!#50!%
703         }%
704       \fi
705     \fi
706   \BitSet@Fi
707 }
```

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

```
708 \def\bitsetAndNot#1#2{%
```

```
709   \bitsetIsEmpty{#1}{%
710     \bitsetReset{#1}%
711   }{%
712     \bitsetIsEmpty{#2}{%
713     }{%
714       \expandafter\edef\csname BS@#1\endcsname{%
715         \expandafter\expandafter\expandafter\BitSet@AndNot
716         \csname BS@#1\expandafter\expandafter\expandafter\endcsname
717         \expandafter\expandafter\expandafter!%
718         \csname BS@#2\endcsname!!%
719       }%
720       \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
721         \bitsetReset{#1}%
722       \fi
723     }%
724   }%
725 }
```

```
726 \def\BitSet@AndNot#1#2!#3#4!#5!{%
727   \ifx\\#2\\%
728     \ifnum#1#3=10 #51\fi
729   \else
730     \ifx\\#4\\%
731       #5%
732       \ifnum#1#3=10 1\else 0\fi
733       #2%
734     \else
735       \ifnum#1#3=10 %
736         #51%
737         \BitSet@AfterFiFiFi{%
738           \BitSet@AndNot#2!#4!!%
739         }%
740       \else
741         \BitSet@AfterFiFiFi{%
742           \BitSet@AndNot#2!#4!#50!%
743         }%
744       \fi
745     \fi
746   \BitSet@Fi
747 }
```

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

```
748 \def\bitsetOr#1#2{%
749   \bitsetIsEmpty{#2}{%
750     \BitSet@IfUndefined{#1}{\bitsetReset{#1}}{}%
751   }{%
752     \bitsetIsEmpty{#1}{%
753       \expandafter\let\csname BS@#1\expandafter\endcsname
754                 \csname BS@#2\endcsname
755     }{%
756       \expandafter\edef\csname BS@#1\endcsname{%
757         \expandafter\expandafter\expandafter\BitSet@Or
758         \csname BS@#1\expandafter\expandafter\expandafter\endcsname
759         \expandafter\expandafter\expandafter!%
```

```
760     \csname BS@#2\endcsname!%
761    }%
762   }%
763  }%
764 }
```

```
765 \def\BitSet@Or#1#2!#3#4!{%
766  \ifnum#1#3>0 1\else 0\fi
767  \ifx\\#2\\%
768    #4%
769  \else
770   \ifx\\#4\\%
771     #2%
772   \else
773     \BitSet@AfterFiFi{%
774       \BitSet@Or#2!#4!%
775     }%
776   \fi
777  \BitSet@Fi
778 }
```

2.8.4 \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 $\widehat{}$= B

```
779 \def\bitsetXor#1#2{%
780  \bitsetIsEmpty{#2}{%
781    \BitSet@IfUndefined{#1}{\bitsetReset{#1}}{}%
782 }{%
783   \bitsetIsEmpty{#1}{%
784     \expandafter\let\csname BS@#1\expandafter\endcsname
785               \csname BS@#2\endcsname
786   }{%
787     \expandafter\edef\csname BS@#1\endcsname{%
788       \expandafter\expandafter\expandafter\BitSet@Xor
789       \csname BS@#1\expandafter\expandafter\expandafter\endcsname
790       \expandafter\expandafter\expandafter!%
791       \csname BS@#2\endcsname!!%
792     }%
793     \expandafter\ifx\csname BS@#1\endcsname\BitSet@Empty
794       \bitsetReset{#1}%
795     \fi
796   }%
797  }%
798 }
```

```
799 \def\BitSet@Xor#1#2!#3#4!#5!{%
800  \ifx\\#2\\%
801   \ifx#1#3%
802    \ifx\\#4\\%
803    \else
804      #50#4%
805    \fi
806   \else
807     #51#4%
```

```
808    \fi
809  \else
810    \ifx\\#4\\%
811      #5%
812      \ifx#1#30\else 1\fi
813      #2%
814    \else
815      \ifx#1#3%
816        \BitSet@AfterFiFiFi{%
817          \BitSet@Xor#2!#4!#50!%
818        }%
819      \else
820        #51%
821        \BitSet@AfterFiFiFi{%
822          \BitSet@Xor#2!#4!!%
823        }%
824      \fi
825    \fi
826  \BitSet@Fi
827 }
```

2.8.5 Shifting

2.8.6 \bitsetShiftLeft

\bitsetShiftLeft

```
828 \def\bitsetShiftLeft#1#2{%
829  \BitSet@IfUndefined{#1}{%
830    \bitsetReset{#1}%
831 }{%
832    \bitsetIsEmpty{#1}{%
833    }{%
834      \expandafter\expandafter\expandafter\BitSet@ShiftLeft
835      \intcalcNum{#2}!{#1}%
836    }%
837 }%
838 }
```

\BitSet@ShiftLeft

```
839 \def\BitSet@ShiftLeft#1!#2{%
840  \ifcase\intcalcSgn{#1} %
841  \or
842    \begingroup
843      \uccode`m=`0 %
844      \uppercase\expandafter{\expandafter\endgroup
845      \expandafter\edef\csname BS@#2\expandafter\endcsname
846      \expandafter{%
847        \romannumeral#1000\expandafter\BitSet@Space
848        \csname BS@#2\endcsname
849      }%
850    }%
851  \else
852    \expandafter\BitSet@ShiftRight\BitSet@Gobble#1!{#2}%
853  \fi
854 }
```

2.8.7 \bitsetShiftRight

\bitsetShiftRight

```
855 \def\bitsetShiftRight#1#2{%
856  \BitSet@IfUndefined{#1}{%
857    \bitsetReset{#1}%
```

```
858    }{%
859      \bitsetIsEmpty{#1}{%
860      }{%
861        \expandafter\expandafter\expandafter\BitSet@ShiftRight
862        \intcalcNum{#2}!{#1}%
863      }%
864    }%
865  }
```

\BitSet@ShiftRight

```
866  \def\BitSet@ShiftRight#1!#2{%
867    \ifcase\intcalcSgn{#1} %
868    \or
869      \expandafter\edef\csname BS@#2\endcsname{%
870        \expandafter\expandafter\expandafter\BitSet@Kill
871        \csname BS@#2\expandafter\endcsname\expandafter\BitSet@Empty
872        \expandafter=%
873        \expandafter{\expandafter}\expandafter{\expandafter}%
874        \romannumeral#1000!%
875      }%
876    \else
877      \expandafter\BitSet@ShiftLeft\BitSet@Gobble#1!{#2}%
878    \fi
879  }
```

\BitSet@Kill

```
880  \def\BitSet@Kill#1#2=#3#4#5{%
881    #3#4%
882    \ifx#5!%
883      \ifx#1\BitSet@Empty
884        0%
885      \else
886        #1#2%
887      \fi
888    \else
889      \ifx#1\BitSet@Empty
890        0%
891        \BitSet@AfterFiFi\BitSet@Cleanup
892      \else
893        \BitSet@Kill#2=%
894      \fi
895    \BitSet@Fi
896  }
```

2.9 Bit manipulation

\bitsetClear

```
897  \def\bitsetClear{%
898    \BitSet@CheckIndex\BitSet@Clear
899  }
```

\bitsetSet

```
900  \def\bitsetSet{%
901    \BitSet@CheckIndex\BitSet@Set
902  }
```

\bitsetFlip

```
903  \def\bitsetFlip{%
904    \BitSet@CheckIndex\BitSet@Flip
905  }
```

```
906 \def\bitsetSetValue#1#2#3{%
907   \expandafter\expandafter\expandafter\BitSet@SetValue
908   \intcalcNum{#3}!{#1}{#2}%
909 }
```

#1: plain value
#2: BitSet
#3: Index

```
910 \def\BitSet@SetValue#1!{%
911   \BitSet@CheckIndex{%
912     \ifcase#1 %
913       \expandafter\BitSet@Clear
914     \or
915       \expandafter\BitSet@Set
916     \else
917       \BitSet@ErrorInvalidBitValue{#1}%
918       \expandafter\expandafter\expandafter\BitSet@Gobble
919       \expandafter\BitSet@Gobble
920     \fi
921   }%
922 }
```

#1: Wrong bit value

```
923 \def\BitSet@ErrorInvalidBitValue#1{%
924   \@PackageError{bitset}{%
925     Invalid bit value (#1) not in range 0..1%
926   }\@ehc
927 }
```

2.9.1 Clear operation

#1: BitSet
#2: plain and checked index

```
928 \def\BitSet@Clear#1#2{%
929   \edef\BitSet@Temp{%
930     \expandafter\expandafter\expandafter\BitSet@@Clear
931     \csname BS@#1\expandafter\endcsname
932     \expandafter\BitSet@Empty\expandafter=\expandafter!%
933     \romannumeral#2000!%
934   }%
935   \expandafter\let\csname BS@#1\expandafter\endcsname
936   \ifx\BitSet@Temp\BitSet@Empty
937     \BitSet@Zero
938   \else
939     \BitSet@Temp
940   \fi
941 }
```

```
942 \def\BitSet@@Clear#1#2=#3!#4{%
943   \ifx#4!%
944   \ifx#1\BitSet@Empty
945   \else
946     \ifx\BitSet@Empty#2%
947     \else
948       #30#2%
949     \fi
950   \fi
951   \else
952     \ifx#1\BitSet@Empty
```

```
953     \BitSet@AfterFiFi\BitSet@Cleanup
954   \else
955     \ifx#10%
956       \BitSet@AfterFiFiFi{%
957         \BitSet@@Clear#2=#30!%
958       }%
959     \else
960       #31%
961       \BitSet@AfterFiFiFi{%
962         \BitSet@@Clear#2=!%
963       }%
964     \fi
965   \fi
966   \BitSet@Fi
967 }
```

2.9.2 Set operation

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

```
968 \def\BitSet@Set#1#2{%
969   \expandafter\edef\csname BS@#1\endcsname{%
970     \expandafter\expandafter\expandafter\BitSet@@Set
971     \csname BS@#1\expandafter\endcsname
972     \expandafter\BitSet@Empty\expandafter=%
973     \expandafter{\expandafter}\expandafter{\expandafter}%
974     \romannumeral#2000!%
975   }%
976 }
```

\BitSet@@Set

```
977 \def\BitSet@@Set#1#2=#3#4#5{%
978   #3#4%
979   \ifx#5!%
980     1#2%
981   \else
982   \ifx#1\BitSet@Empty
983     0%
984     \BitSet@AfterFiFi\BitSet@@@Set
985   \else
986     #1%
987     \BitSet@@Set#2=%
988   \fi
989   \BitSet@Fi
990 }
```

\BitSet@@@Set

```
991 \def\BitSet@@@Set#1{%
992   \ifx#1!%
993     1%
994   \else
995     0%
996     \expandafter\BitSet@@@Set
997   \fi
998 }
```

2.9.3 Flip operation

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

```
999  \def\BitSet@Flip#1#2{%
1000   \edef\BitSet@Temp{%
```

```
1001    \expandafter\expandafter\expandafter\BitSet@@Flip
1002    \csname BS@#1\expandafter\endcsname
1003    \expandafter\BitSet@Empty\expandafter=\expandafter!%
1004    \romannumeral#2000!%
1005 }%
1006 \expandafter\let\csname BS@#1\expandafter\endcsname
1007 \ifx\BitSet@Temp\BitSet@Empty
1008    \BitSet@Zero
1009 \else
1010    \BitSet@Temp
1011 \fi
1012 }
```

```
1013 \def\BitSet@@Flip#1#2=#3!#4{%
1014 \ifx#4!%
1015    \ifx#11%
1016      \ifx\BitSet@Empty#2%
1017      \else
1018        #30#2%
1019      \fi
1020    \else
1021      #31#2%
1022    \fi
1023 \else
1024    \ifx#1\BitSet@Empty
1025      #30%
1026      \BitSet@AfterFiFi\BitSet@@@Set
1027    \else
1028      \ifx#10%
1029        \BitSet@AfterFiFiFi{%
1030          \BitSet@@Flip#2=#30!%
1031        }%
1032      \else
1033        #31%
1034        \BitSet@AfterFiFiFi{%
1035          \BitSet@@Flip#2=!%
1036        }%
1037      \fi
1038    \fi
1039 \BitSet@Fi
1040 }
```

2.9.4 Range operators

```
1041 \def\bitsetClearRange{%
1042   \BitSet@Range\BitSet@Clear
1043 }
```

```
1044 \def\bitsetSetRange{%
1045   \BitSet@Range\BitSet@Set
1046 }
```

```
1047 \def\bitsetFlipRange{%
1048   \BitSet@Range\BitSet@Flip
1049 }
```

```
1050 \def\bitsetSetValueRange#1#2#3#4{%
```

```
1051  \expandafter\expandafter\expandafter\BitSet@SetValueRange
1052  \intcalcNum{#4}!{#1}{#2}{#3}%
1053 }
```

\BitSet@SetValueRange

```
1054 \def\BitSet@SetValueRange#1!#2#3#4{%
1055  \ifcase#1 %
1056    \BitSet@Range\BitSet@Clear{#2}{#3}{#4}%
1057  \or
1058    \BitSet@Range\BitSet@Set{#2}{#3}{#4}%
1059  \else
1060    \BitSet@ErrorInvalidBitValue{#1}%
1061  \fi
1062 }
```

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

```
1063 \def\BitSet@Range#1#2#3#4{%
1064  \edef\BitSet@Temp{%
1065    \noexpand\BitSet@@Range\noexpand#1{#2}%
1066    \intcalcNum{#3}!\intcalcNum{#4}!%
1067  }%
1068  \BitSet@Temp
1069 }
```

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

```
1070 \def\BitSet@@Range#1#2#3!#4!{%
1071  \ifnum#3<0 %
1072    \BitSet@NegativeIndex#1{#2}#3!#4!0!#4!%
1073  \else
1074    \ifnum#4<0 %
1075      \BitSet@NegativeIndex#1{#2}#3!#4!#3!0!%
1076    \else
1077      \ifcase\intcalcCmp{#3}{#4} %
1078      \or
1079        \@PackageError{bitset}{%
1080          Wrong index numbers in range [#3..#4]\MessageBreak% hash-ok
1081          for clear/set/flip on bit set `#2'.\MessageBreak
1082          The lower index exceeds the upper index.\MessageBreak
1083          Canceling the operation as error recovery%
1084        }\@ehc
1085      \else
1086        \BitSet@@@Range#3!#4!#1{#2}%
1087      \fi
1088    \fi
1089  \fi
1090 }
```

\BitSet@NegativeIndex

```
1091 \def\BitSet@NegativeIndex#1#2#3!#4!#5!#6!{%
1092  \@PackageError{bitset}{%
1093    Negative index in range [#3..#4]\MessageBreak % hash-ok
1094    for \string\bitset
1095    \ifx#1\BitSet@Clear
1096      Clear%
1097    \else
1098      \ifx#1\BitSet@Set
```

```
1099        Set%
1100      \else
1101        Flip%
1102      \fi
1103    \fi
1104    Range on bit set '#2'.\MessageBreak
1105    Using [#5..#6] as error recovery% hash-ok
1106  }\@ehc
1107  \BitSet@@Range#1{#2}#5!#6!%
1108 }
```

\BitSet@@Range

```
1109 \def\BitSet@@Range#1!#2!#3#4{%
1110   \ifnum#1<#2 %
1111     #3{#4}{#1}%
1112     \BitSet@AfterFi{%
1113       \expandafter\expandafter\expandafter\BitSet@@Range
1114       \IntCalcInc#1!!#2!#3{#4}%
1115     }%
1116   \BitSet@Fi
1117 }
```

2.10 Bit retrieval

2.10.1 \bitsetGet

\bitsetGet

```
1118 \def\bitsetGet#1#2{%
1119   \number
1120   \expandafter\expandafter\expandafter\BitSet@Get
1121   \intcalcNum{#2}!{#1}%
1122 }
```

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

```
1123 \def\BitSet@Get#1!#2{%
1124   \ifnum#1<0 %
1125     \BitSet@AfterFi{%
1126       0 \BitSetError:NegativeIndex%
1127     }%
1128   \else
1129     \BitSet@IfUndefined{#2}{0}{%
1130       \expandafter\expandafter\expandafter\BitSet@@Get
1131       \csname BS@#2\expandafter\endcsname
1132       \expandafter!\expandafter=%
1133       \expandafter{\expandafter}\expandafter{\expandafter}%
1134       \romannumeral\intcalcNum{#1}000!%
1135     }%
1136     \expandafter\BitSet@Space
1137   \BitSet@Fi
1138 }
```

\BitSet@@Get

```
1139 \def\BitSet@@Get#1#2=#3#4#5{%
1140   #3#4%
1141   \ifx#5!%
1142     \ifx#1!%
1143       0%
1144     \else
1145       #1%
1146     \fi
1147   \else
```

```
1148    \ifx#1!%
1149      0%
1150      \BitSet@AfterFiFi\BitSet@Cleanup
1151    \else
1152      \BitSet@@Get#2=%
1153    \fi
1154  \BitSet@Fi
1155 }
```

2.10.2 \bitsetNextClearBit, \bitsetNextSetBit

\bitsetNextClearBit

```
1156 \def\bitsetNextClearBit#1#2{%
1157   \number
1158   \expandafter\expandafter\expandafter\BitSet@NextClearBit
1159   \intcalcNum{#2}!{#1} %
1160 }
```

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

```
1161 \def\BitSet@NextClearBit#1!#2{%
1162   \ifnum#1<0 %
1163     \BitSet@NextClearBit0!{#2}%
1164     \BitSet@AfterFi{%
1165       \expandafter\BitSet@Space
1166       \expandafter\BitSetError:NegativeIndex\romannumeral0%
1167     }%
1168   \else
1169     \bitsetIsEmpty{#2}{#1}{%
1170       \expandafter\BitSet@Skip
1171       \number#1\expandafter\expandafter\expandafter!%
1172       \csname BS@#2\endcsname!!!!!!!!!!=%
1173       {\BitSet@@NextClearBit#1!}%
1174     }%
1175   \BitSet@Fi
1176 }
```

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

```
1177 \def\BitSet@@NextClearBit#1!#2{%
1178   \ifx#2!%
1179     #1%
1180   \else
1181     \ifx#20%
1182       #1%
1183       \BitSet@AfterFiFi\BitSet@Cleanup
1184     \else
1185       \BitSet@AfterFiFi{%
1186         \expandafter\expandafter\expandafter\BitSet@@NextClearBit
1187         \IntCalcInc#1!!%
1188       }%
1189     \fi
1190   \BitSet@Fi
1191 }
```

\bitsetNextSetBit

```
1192 \def\bitsetNextSetBit#1#2{%
1193   \number
1194   \expandafter\expandafter\expandafter\BitSet@NextSetBit
1195   \intcalcNum{#2}!{#1} %
1196 }
```

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

```
1197 \def\BitSet@NextSetBit#1!#2{%
1198   \ifnum#1<0 %
1199     \BitSet@NextSetBit0!{#2}%
1200     \BitSet@AfterFi{%
1201       \expandafter\BitSet@Space
1202       \expandafter\BitSetError:NegativeIndex\romannumeral0%
1203     }%
1204   \else
1205     \bitsetIsEmpty{#2}{-1}{%
1206       \expandafter\BitSet@Skip
1207       \number#1\expandafter\expandafter\expandafter!%
1208       \csname BS@#2\endcsname!!!!!!!!!=%
1209       {\BitSet@@NextSetBit#1!}%
1210     }%
1211   \BitSet@Fi
1212 }
```

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

```
1213 \def\BitSet@@NextSetBit#1!#2{%
1214   \ifx#2!%
1215     -1%
1216   \else
1217     \ifx#21%
1218       #1%
1219       \BitSet@AfterFiFi\BitSet@Cleanup
1220     \else
1221       \BitSet@AfterFiFi{%
1222         \expandafter\expandafter\expandafter\BitSet@@NextSetBit
1223         \IntCalcInc#1!!%
1224       }%
1225     \fi
1226   \BitSet@Fi
1227 }
```

\BitSet@Cleanup	

```
1228 \def\BitSet@Cleanup#1!{}
```

\BitSet@Skip	#1: number of bits to skip
	#2: bits
	#3: continuation code

```
1229 \def\BitSet@Skip#1!#2{%
1230   \ifx#2!%
1231     \BitSet@AfterFi{%
1232       \BitSet@SkipContinue%
1233     }%
1234   \else
1235     \ifcase#1 %
1236       \BitSet@AfterFiFi{%
1237         \BitSet@SkipContinue#2%
1238       }%
1239     \or
1240       \BitSet@AfterFiFi\BitSet@SkipContinue
1241     \or
1242       \BitSet@AfterFiFi{%
1243         \expandafter\BitSet@SkipContinue\BitSet@Gobble
1244       }%
1245     \else
1246       \ifnum#1>8 %
1247         \BitSet@AfterFiFiFi{%
```

34

```
1248        \expandafter\BitSet@Skip
1249        \number\IntCalcSub#1!8!\expandafter!%
1250        \BitSet@GobbleSeven
1251      }%
1252    \else
1253      \BitSet@AfterFiFiFi{%
1254        \expandafter\expandafter\expandafter\BitSet@Skip
1255        \IntCalcDec#1!!%
1256      }%
1257    \fi
1258    \fi
1259  \BitSet@Fi
1260 }
```

\BitSet@SkipContinue | #1: remaining bits
#2: continuation code

```
1261 \def\BitSet@SkipContinue#1!#2=#3{%
1262   #3#1!%
1263 }
```

\BitSet@GobbleSeven

```
1264 \def\BitSet@GobbleSeven#1#2#3#4#5#6#7{}
```

2.10.3 \bitsetGetSetBitList

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

```
1265 \def\bitsetGetSetBitList#1{%
1266   \romannumeral0%
1267   \bitsetIsEmpty{#1}{ }{%
1268     \expandafter\BitSet@GetSetBitList
1269     \number\BitSet@NextSetBit0!{#1}!{#1}{}!%
1270   }%
1271 }
```

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

```
1272 \def\BitSet@GetSetBitList#1!#2#3#4!{%
1273   \ifnum#1<0 %
1274     \BitSet@AfterFi{ #4}%
1275   \else
1276     \BitSet@AfterFi{%
1277       \expandafter\BitSet@GetSetBitList\number
1278       \expandafter\expandafter\expandafter\BitSet@NextSetBit
1279       \IntCalcInc#1!!{#2}!{#2},#4#3#1!%
1280     }%
1281   \BitSet@Fi
1282 }
```

2.11 Bit set properties

\bitsetSize

```
1283 \def\bitsetSize#1{%
1284   \number
1285   \BitSet@IfUndefined{#1}{0 }{%
1286     \expandafter\expandafter\expandafter\BitSet@Size
1287     \expandafter\expandafter\expandafter1%
1288     \expandafter\expandafter\expandafter!%
1289     \csname BS@#1\endcsname!0!%
1290   }%
1291 }
```

```
\BitSet@Size    #1: counter
                #2#3: bits
                #4: result
         1292 \def\BitSet@Size#1!#2#3!#4!{%
         1293   \ifx#21%
         1294    \ifx\\#3\\%
         1295      \BitSet@AfterFiFi{#1 }%
         1296    \else
         1297      \BitSet@AfterFiFi{%
         1298        \expandafter\expandafter\expandafter\BitSet@Size
         1299        \IntCalcInc#1!!#3!#1!%
         1300      }%
         1301    \fi
         1302   \else
         1303    \ifx\\#3\\%
         1304      \BitSet@AfterFiFi{#4 }%
         1305    \else
         1306      \BitSet@AfterFiFi{%
         1307        \expandafter\expandafter\expandafter\BitSet@Size
         1308        \IntCalcInc#1!!#3!#4!%
         1309      }%
         1310    \fi
         1311   \fi
         1312   \BitSet@Fi
         1313 }

\bitsetCardinality
         1314 \def\bitsetCardinality#1{%
         1315   \number
         1316   \BitSet@IfUndefined{#1}{0 }{%
         1317    \expandafter\expandafter\expandafter\BitSet@Cardinality
         1318    \expandafter\expandafter\expandafter0%
         1319    \expandafter\expandafter\expandafter!%
         1320    \csname BS@#1\endcsname!%
         1321   }%
         1322 }

\BitSet@Cardinality   #1: result
                      #2#3: bits
         1323 \def\BitSet@Cardinality#1!#2#3!{%
         1324   \ifx#21%
         1325    \ifx\\#3\\%
         1326      \BitSet@AfterFiFi{\IntCalcInc#1! }%
         1327    \else
         1328      \BitSet@AfterFiFi{%
         1329        \expandafter\expandafter\expandafter\BitSet@Cardinality
         1330        \IntCalcInc#1!!#3!%
         1331      }%
         1332    \fi
         1333   \else
         1334    \ifx\\#3\\%
         1335      \BitSet@AfterFiFi{#1 }%
         1336    \else
         1337      \BitSet@AfterFiFi{%
         1338        \BitSet@Cardinality#1!#3!%
         1339      }%
         1340    \fi
         1341   \fi
         1342   \BitSet@Fi
         1343 }
```

2.12 Queries

\bitsetIsDefined

```
1344 \def\bitsetIsDefined#1{%
1345   \BitSet@IfUndefined{#1}%
1346   \BitSet@SecondOfTwo
1347   \BitSet@FirstOfTwo
1348 }
```

\bitsetIsEmpty

```
1349 \def\bitsetIsEmpty#1{%
1350   \BitSet@IfUndefined{#1}\BitSet@FirstOfTwo{%
1351     \expandafter\ifx\csname BS@#1\endcsname\BitSet@Zero
1352       \expandafter\BitSet@FirstOfTwo
1353     \else
1354       \expandafter\BitSet@SecondOfTwo
1355     \fi
1356   }%
1357 }
```

\BitSet@Zero

```
1358 \def\BitSet@Zero{0}
```

\bitsetQuery

```
1359 \def\bitsetQuery#1#2{%
1360   \ifnum\bitsetGet{#1}{#2}=1 %
1361     \expandafter\BitSet@FirstOfTwo
1362   \else
1363     \expandafter\BitSet@SecondOfTwo
1364   \fi
1365 }
```

\bitsetEquals

```
1366 \def\bitsetEquals#1#2{%
1367   \BitSet@IfUndefined{#1}{%
1368     \BitSet@IfUndefined{#2}\BitSet@FirstOfTwo\BitSet@SecondOfTwo
1369   }{%
1370     \BitSet@IfUndefined{#2}\BitSet@SecondOfTwo{%
1371       \expandafter\ifx\csname BS@#1\expandafter\endcsname
1372               \csname BS@#2\endcsname
1373         \expandafter\BitSet@FirstOfTwo
1374       \else
1375         \expandafter\BitSet@SecondOfTwo
1376       \fi
1377     }%
1378   }%
1379 }
```

\bitsetIntersects

```
1380 \def\bitsetIntersects#1#2{%
1381   \bitsetIsEmpty{#1}\BitSet@SecondOfTwo{%
1382     \bitsetIsEmpty{#2}\BitSet@SecondOfTwo{%
1383       \expandafter\expandafter\expandafter\BitSet@Intersects
1384       \csname BS@#1\expandafter\expandafter\expandafter\endcsname
1385       \expandafter\expandafter\expandafter!%
1386       \csname BS@#2\endcsname!%
1387     }%
1388   }%
1389 }
```

\BitSet@Intersects

```
1390 \def\BitSet@Intersects#1#2!#3#4!{%
```

```
1391  \ifnum#1#3=11 %
1392    \BitSet@AfterFi\BitSet@FirstOfTwo
1393  \else
1394    \ifx\\#2\\%
1395      \BitSet@AfterFiFi\BitSet@SecondOfTwo
1396    \else
1397      \ifx\\#4\\%
1398        \BitSet@AfterFiFiFi\BitSet@SecondOfTwo
1399      \else
1400        \BitSet@AfterFiFiFi{%
1401          \BitSet@Intersects#2!#4!%
1402        }%
1403      \fi
1404    \fi
1405  \BitSet@Fi
1406 }

1407 \BitSet@AtEnd%
1408 ⟨/package⟩
```

3 Test

3.1 Catcode checks for loading

```
1409 ⟨*test1⟩
```

```
1410 \catcode`\{=1 %
1411 \catcode`\}=2 %
1412 \catcode`\#=6 %
1413 \catcode`\@=11 %
1414 \expandafter\ifx\csname count@\endcsname\relax
1415   \countdef\count@=255 %
1416 \fi
1417 \expandafter\ifx\csname @gobble\endcsname\relax
1418   \long\def\@gobble#1{}%
1419 \fi
1420 \expandafter\ifx\csname @firstofone\endcsname\relax
1421   \long\def\@firstofone#1{#1}%
1422 \fi
1423 \expandafter\ifx\csname loop\endcsname\relax
1424   \expandafter\@firstofone
1425 \else
1426   \expandafter\@gobble
1427 \fi
1428 {%
1429   \def\loop#1\repeat{%
1430     \def\body{#1}%
1431     \iterate
1432   }%
1433   \def\iterate{%
1434     \body
1435       \let\next\iterate
1436     \else
1437       \let\next\relax
1438     \fi
1439     \next
1440   }%
1441   \let\repeat=\fi
1442 }%
1443 \def\RestoreCatcodes{}
1444 \count@=0 %
1445 \loop
1446   \edef\RestoreCatcodes{%
```

```
1447     \RestoreCatcodes
1448     \catcode\the\count@=\the\catcode\count@\relax
1449   }%
1450 \ifnum\count@<255 %
1451   \advance\count@ 1 %
1452 \repeat
1453
1454 \def\RangeCatcodeInvalid#1#2{%
1455   \count@=#1\relax
1456   \loop
1457     \catcode\count@=15 %
1458   \ifnum\count@<#2\relax
1459     \advance\count@ 1 %
1460   \repeat
1461 }
1462 \def\RangeCatcodeCheck#1#2#3{%
1463   \count@=#1\relax
1464   \loop
1465     \ifnum#3=\catcode\count@
1466     \else
1467       \errmessage{%
1468         Character \the\count@\space
1469         with wrong catcode \the\catcode\count@\space
1470         instead of \number#3%
1471       }%
1472     \fi
1473   \ifnum\count@<#2\relax
1474     \advance\count@ 1 %
1475   \repeat
1476 }
1477 \def\space{ }
1478 \expandafter\ifx\csname LoadCommand\endcsname\relax
1479   \def\LoadCommand{\input bitset.sty\relax}%
1480 \fi
1481 \def\Test{%
1482   \RangeCatcodeInvalid{0}{47}%
1483   \RangeCatcodeInvalid{58}{64}%
1484   \RangeCatcodeInvalid{91}{96}%
1485   \RangeCatcodeInvalid{123}{255}%
1486   \catcode`\@=12 %
1487   \catcode`\\=0 %
1488   \catcode`\%=14 %
1489   \LoadCommand
1490   \RangeCatcodeCheck{0}{36}{15}%
1491   \RangeCatcodeCheck{37}{37}{14}%
1492   \RangeCatcodeCheck{38}{47}{15}%
1493   \RangeCatcodeCheck{48}{57}{12}%
1494   \RangeCatcodeCheck{58}{63}{15}%
1495   \RangeCatcodeCheck{64}{64}{12}%
1496   \RangeCatcodeCheck{65}{90}{11}%
1497   \RangeCatcodeCheck{91}{91}{15}%
1498   \RangeCatcodeCheck{92}{92}{0}%
1499   \RangeCatcodeCheck{93}{96}{15}%
1500   \RangeCatcodeCheck{97}{122}{11}%
1501   \RangeCatcodeCheck{123}{255}{15}%
1502   \RestoreCatcodes
1503 }
1504 \Test
1505 \csname @@end\endcsname
1506 \end
1507 ⟨/test1⟩
```

3.2 Macro tests

3.2.1 Preamble

```
1508 ⟨*test2⟩
1509 \NeedsTeXFormat{LaTeX2e}
1510 \nofiles
1511 \documentclass{article}
1512 \makeatletter
1513 ⟨*noetex⟩
1514 \let\SavedNumexpr\numexpr
1515 \let\SavedIfcsname\ifcsname
1516 \let\SavedCurrentgrouplevel\currentgrouplevel
1517 \def\ETeXDisable{%
1518  \let\ifcsname\@undefined
1519  \let\numexpr\@undefined
1520  \let\currentgrouplevel\@undefined
1521 }
1522 \ETeXDisable
1523 ⟨/noetex⟩
1524 \makeatletter
1525 \chardef\BitSet@TestMode=1 %
1526 \makeatother
1527 \usepackage{bitset}[2016/05/16]
1528 ⟨*noetex⟩
1529 \def\ETeXEnable{%
1530  \let\numexpr\SavedNumexpr
1531  \let\ifcsname\SavedIfcsname
1532  \let\currentgrouplevel\SavedCurrentgrouplevel
1533 }
1534 \ETeXEnable
1535 ⟨/noetex⟩
1536 \usepackage{qstest}
1537 \IncludeTests{*}
1538 \LogTests{log}{*}{*}
1539 \makeatletter
```

3.2.2 Time

```
1540 \begingroup\expandafter\expandafter\expandafter\endgroup
1541 \expandafter\ifx\csname pdfresettimer\endcsname\relax
1542 \else
1543  \newcount\SummaryTime
1544  \newcount\TestTime
1545  \SummaryTime=\z@
1546  \newcommand*{\PrintTime}[2]{%
1547   \typeout{%
1548     [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
1549   }%
1550 }%
1551  \newcommand*{\StartTime}[1]{%
1552   \renewcommand*{\TimeDescription}{#1}%
1553   \pdfresettimer
1554 }%
1555  \newcommand*{\TimeDescription}{}%
1556  \newcommand*{\StopTime}{%
1557   \TestTime=\pdfelapsedtime
1558   \global\advance\SummaryTime\TestTime
1559   \PrintTime\TimeDescription\TestTime
1560 }%
1561  \let\saved@qstest\qstest
1562  \let\saved@endqstest\endqstest
1563  \def\qstest#1#2{%
1564   \saved@qstest{#1}{#2}%
```

```
1565    \StartTime{#1}%
1566  }%
1567  \def\endqstest{%
1568    \StopTime
1569    \saved@endqstest
1570  }%
1571  \AtEndDocument{%
1572    \PrintTime{summary}\SummaryTime
1573  }%
1574 \fi
```

3.2.3   Detection of unwanted space

```
1575 \let\orig@qstest\qstest
1576 \let\orig@endqstest\endqstest
1577 \def\qstest#1#2{%
1578   \orig@qstest{#1}{#2}%
1579   \setbox0\hbox\bgroup\begingroup\ignorespaces
1580 }
1581 \def\endqstest{%
1582   \endgroup\egroup
1583   \Expect*{\the\wd0}{0.0pt}%
1584   \orig@endqstest
1585 }
```

3.2.4   Test macros

```
1586 \newcounter{Test}
1587
1588 \def\TestError#1#2{%
1589   \begingroup
1590     \setcounter{Test}{0}%
1591     \sbox0{%
1592       \def\@PackageError##1##2##3{%
1593         \stepcounter{Test}%
1594         \begingroup
1595           \let\MessageBreak\relax
1596 ⟨*noetex⟩
1597          \ETeXEnable
1598 ⟨/noetex⟩
1599          \Expect{##1}{bitset}%
1600          \Expect*{##2}*{#1}%
1601        \endgroup
1602      }%
1603 ⟨*noetex⟩
1604        \ETeXDisable
1605 ⟨/noetex⟩
1606      #2%
1607    }%
1608    \Expect*{\theTest}{1}%
1609    \Expect*{\the\wd0}{0.0pt}%
1610  \endgroup
1611 }
1612
1613 \def\TestErrorNegativeIndex#1#2{%
1614   \TestError{Invalid negative index (#1)}{#2}%
1615 }
1616
1617 \def\TestGetterUndefined#1{%
1618   \CheckUndef{dummy}%
1619   \expandafter\expandafter\expandafter\Expect
1620   \expandafter\expandafter\expandafter{#1{dummy}}{0}%
1621 }
1622
1623 \def\ExpectBitSet#1#2{%
```

41

```
1624  \expandafter\expandafter\expandafter\Expect
1625  \expandafter\expandafter\expandafter
1626  {\csname BS@#1\endcsname}*{#2}%
1627 }
1628 \def\Check#1#2{%
1629  \ExpectBitSet{#1}{#2}%
1630 }
1631 \def\CheckUndef#1{%
1632  \begingroup
1633   \Expect*{%
1634     \expandafter
1635     \ifx\csname BS@#1\endcsname\relax true\else false\fi
1636   }{true}%
1637  \endgroup
1638 }
1639 \def\RevCheck#1#2{%
1640  \ExpectBitSet{#1}{\Reverse#2!!}%
1641 }
1642 \def\Set#1#2{%
1643  \expandafter\def\csname BS@#1\endcsname{#2}%
1644 }
1645 \def\RevSet#1#2{%
1646  \expandafter\edef\csname BS@#1\endcsname{%
1647    \Reverse#2!!%
1648  }%
1649 }
1650 \def\Reverse#1#2!#3!{%
1651  \ifx\\#2\\%
1652    #1#3%
1653    \expandafter\@gobble
1654  \else
1655    \expandafter\@firstofone
1656  \fi
1657  {\Reverse#2!#1#3!}%
1658 }
```

3.2.5 Test sets

```
1659 \begin{qstest}{Let}{Let}
1660  \CheckUndef{abc}%
1661  \CheckUndef{xyz}%
1662  \bitsetLet{xyz}{abc}%
1663  \CheckUndef{abc}%
1664  \Check{xyz}{0}%
1665  \Set{abc}{1}%
1666  \Check{abc}{1}%
1667  \Check{xyz}{0}%
1668  \bitsetLet{xyz}{abc}%
1669  \Check{abc}{1}%
1670  \Check{xyz}{1}%
1671  \Set{xyz}{11}%
1672  \Check{abc}{1}%
1673  \Check{xyz}{11}%
1674 \end{qstest}
1675
1676 \begin{qstest}{Reset}{Reset}
1677  \bitsetReset{xyz}%
1678  \Check{xyz}{0}%
1679  \bitsetReset{abc}%
1680  \Check{abc}{0}%
1681  \Set{abc}{10101}%
1682  \bitsetReset{abc}%
1683  \Check{abc}{0}%
1684 \end{qstest}
```

```
1685
1686  \begin{qstest}{Get/Query}{Get/Query}
1687  \expandafter\expandafter\expandafter\Expect
1688  \expandafter\expandafter\expandafter{%
1689    \bitsetGet{dummy}{0}%
1690  }{0}%
1691  \begingroup
1692    \expandafter\def\csname BitSetError:NegativeIndex\endcsname{}%
1693    \Set{abc}{1}%
1694    \Expect*{\bitsetQuery{abc}{-1}{true}{false}}{false}%
1695  \endgroup
1696  \def\Test#1#2#3{%
1697    \Set{abc}{#1}%
1698    \expandafter\expandafter\expandafter\Expect
1699    \expandafter\expandafter\expandafter{\bitsetGet{abc}{#2}}{#3}%
1700    \Expect*{\bitsetQuery{abc}{#2}{true}{false}}%
1701        *{\ifcase#3 false\or true\else error\fi}%
1702  }%
1703  \Test{1}{100}{0}%
1704  \Test{0}{0}{0}%
1705  \Test{1}{0}{1}%
1706  \Test{11}{1}{1}%
1707  \Test{111}{1}{1}%
1708  \Test{101}{1}{0}%
1709  \Test{101}{2}{1}%
1710  \Test{10100110011}{10}{1}%
1711  \end{qstest}
1712
1713  \begin{qstest}{Size}{Size}
1714  \TestGetterUndefined\bitsetSize
1715  \def\Test#1#2{%
1716    \Set{abc}{#1}%
1717    \expandafter\expandafter\expandafter\Expect
1718    \expandafter\expandafter\expandafter{\bitsetSize{abc}}{#2}%
1719  }%
1720  \Test{0}{0}%
1721  \Test{1}{1}%
1722  \Test{00}{0}%
1723  \Test{0000000}{0}%
1724  \Test{10}{1}%
1725  \Test{01}{2}%
1726  \Test{11}{2}%
1727  \Test{010}{2}%
1728  \Test{011}{3}%
1729  \Test{100110011}{9}%
1730  \Test{00000111110000011111100000}{20}%
1731  \Test{000000000000000000000000011111111111111111111}{45}%
1732  \end{qstest}
1733
1734  \begin{qstest}{Cardinality}{Cardinality}
1735  \TestGetterUndefined\bitsetCardinality
1736  \def\Test#1#2{%
1737    \Set{abc}{#1}%
1738    \expandafter\expandafter\expandafter\Expect
1739    \expandafter\expandafter\expandafter{%
1740      \bitsetCardinality{abc}%
1741    }{#2}%
1742  }%
1743  \Test{0}{0}%
1744  \Test{1}{1}%
1745  \Test{00}{0}%
1746  \Test{0000000}{0}%
```

```
1747  \Test{10}{1}%
1748  \Test{01}{1}%
1749  \Test{11}{2}%
1750  \Test{010}{1}%
1751  \Test{011}{2}%
1752  \Test{100110011}{5}%
1753  \Test{0000011111000001111100000}{10}%
1754  \Test{00000000000000000000000011111111111111111111}{20}%
1755  \end{qstest}
1756
1757  \begin{qstest}{NextClearBit/NextSetBit}{NextClearBit/NextSetBit}
1758  \def\Test#1#2{%
1759    \expandafter\expandafter\expandafter\Expect
1760    \expandafter\expandafter\expandafter{%
1761      \TestOp{abc}{#1}%
1762    }{#2}%
1763  }%
1764  \def\Clear{\let\TestOp\bitsetNextClearBit}%
1765  \def\Set{\let\TestOp\bitsetNextSetBit}%
1766  \begingroup
1767    \catcode`\:=11 %
1768    \bitsetSetBin{abc}{1}%
1769    \Clear
1770    \Test{-1}{1\BitSetError:NegativeIndex}%
1771    \Set
1772    \Test{-1}{0\BitSetError:NegativeIndex}%
1773  \endgroup
1774  \let\BS@abc\@undefined
1775  \Clear
1776  \Test{0}{0}%
1777  \Test{1}{1}%
1778  \Test{2}{2}%
1779  \Test{100}{100}%
1780  \Set
1781  \Test{0}{-1}%
1782  \Test{1}{-1}%
1783  \Test{100}{-1}%
1784  \bitsetReset{abc}%
1785  \Clear
1786  \Test{0}{0}%
1787  \Test{1}{1}%
1788  \Test{2}{2}%
1789  \Test{100}{100}%
1790  \Set
1791  \Test{0}{-1}%
1792  \Test{1}{-1}%
1793  \Test{100}{-1}%
1794  \bitsetSetBin{abc}{1}%
1795  \Clear
1796  \Test{0}{1}%
1797  \Test{1}{1}%
1798  \Test{2}{2}%
1799  \Test{100}{100}%
1800  \Set
1801  \Test{0}{0}%
1802  \Test{1}{-1}%
1803  \Test{100}{-1}%
1804  \bitsetSetBin{abc}{111000111000111000111}%
1805  \Clear
1806  \Test{0}{3}%
1807  \Test{1}{3}%
1808  \Test{2}{3}%
```

```
1809  \Test{3}{3}%
1810  \Test{4}{4}%
1811  \Test{5}{5}%
1812  \Test{6}{9}%
1813  \Test{7}{9}%
1814  \Test{8}{9}%
1815  \Test{9}{9}%
1816  \Test{10}{10}%
1817  \Test{11}{11}%
1818  \Test{12}{15}%
1819  \Test{13}{15}%
1820  \Test{14}{15}%
1821  \Test{15}{15}%
1822  \Test{16}{16}%
1823  \Test{17}{17}%
1824  \Test{18}{21}%
1825  \Test{19}{21}%
1826  \Test{20}{21}%
1827  \Test{21}{21}%
1828  \Test{22}{22}%
1829  \Test{100}{100}%
1830  \Set
1831  \Test{0}{0}%
1832  \Test{1}{1}%
1833  \Test{2}{2}%
1834  \Test{3}{6}%
1835  \Test{4}{6}%
1836  \Test{5}{6}%
1837  \Test{6}{6}%
1838  \Test{7}{7}%
1839  \Test{8}{8}%
1840  \Test{9}{12}%
1841  \Test{10}{12}%
1842  \Test{11}{12}%
1843  \Test{12}{12}%
1844  \Test{13}{13}%
1845  \Test{14}{14}%
1846  \Test{15}{18}%
1847  \Test{16}{18}%
1848  \Test{17}{18}%
1849  \Test{18}{18}%
1850  \Test{19}{19}%
1851  \Test{20}{20}%
1852  \Test{21}{-1}%
1853  \Test{22}{-1}%
1854  \Test{100}{-1}%
1855  \bitsetSetBin{abc}{1111111}%
1856  \Clear
1857  \Test{6}{7}%
1858  \Test{7}{7}%
1859  \Test{8}{8}%
1860  \Test{100}{100}%
1861  \Set
1862  \Test{6}{6}%
1863  \Test{7}{-1}%
1864  \Test{8}{-1}%
1865  \Test{100}{-1}%
1866  \bitsetSetBin{abc}{11111111}%
1867  \Clear
1868  \Test{7}{8}%
1869  \Test{8}{8}%
1870  \Test{9}{9}%
```

```
1871  \Test{100}{100}%
1872  \Set
1873  \Test{7}{7}%
1874  \Test{8}{-1}%
1875  \Test{9}{-1}%
1876  \Test{100}{-1}%
1877  \bitsetSetBin{abc}{111111111}%
1878  \Clear
1879  \Test{8}{9}%
1880  \Test{9}{9}%
1881  \Test{10}{10}%
1882  \Test{100}{100}%
1883  \Set
1884  \Test{8}{8}%
1885  \Test{9}{-1}%
1886  \Test{10}{-1}%
1887  \Test{100}{-1}%
1888  \bitsetSetBin{abc}{1111111111}%
1889  \Clear
1890  \Test{9}{10}%
1891  \Test{10}{10}%
1892  \Test{11}{11}%
1893  \Test{100}{100}%
1894  \Set
1895  \Test{9}{9}%
1896  \Test{10}{-1}%
1897  \Test{11}{-1}%
1898  \Test{100}{-1}%
1899  \end{qstest}
1900
1901  \begin{qstest}{GetSetBitList}{GetSetBitList}
1902  \let\BS@abc\@undefined
1903  \expandafter\expandafter\expandafter\Expect
1904  \expandafter\expandafter\expandafter{%
1905    \bitsetGetSetBitList{abc}%
1906  }{}%
1907  \def\Test#1#2{%
1908    \bitsetSetBin{abc}{#1}%
1909    \expandafter\expandafter\expandafter\Expect
1910    \expandafter\expandafter\expandafter{%
1911      \bitsetGetSetBitList{abc}%
1912    }{#2}%
1913  }%
1914  \Test{0}{}%
1915  \Test{1}{0}%
1916  \Test{10}{1}%
1917  \Test{11}{0,1}%
1918  \Test{10110100}{2,4,5,7}%
1919  \Test{101101001010011}{0,1,4,6,9,11,12,14}%
1920  \end{qstest}
1921
1922  \begin{qstest}{GetDec}{GetDec}
1923  \TestGetterUndefined\bitsetGetDec
1924  \def\Test#1#2{%
1925    \RevSet{abc}{#1}%
1926  ⟨*noetex⟩
1927    \begingroup\expandafter\expandafter\expandafter\endgroup
1928  ⟨/noetex⟩
1929    \expandafter\expandafter\expandafter\Expect
1930    \expandafter\expandafter\expandafter{%
1931      \bitsetGetDec{abc}%
1932    }{#2}%
```

46

```
1933  }%
1934  \Test{0}{0}%
1935  \Test{1}{1}%
1936  \Test{10}{2}%
1937  \Test{11}{3}%
1938  \Test{100}{4}%
1939  \Test{101}{5}%
1940  \Test{110}{6}%
1941  \Test{111}{7}%
1942  \Test{1000}{8}%
1943  \Test{000111}{7}%
1944  \Test{1111111111111111%
1945      1111111111111111}{2147483647}%
1946  \Test{0001111111111111111%
1947      1111111111111111}{2147483647}%
1948  \Test{1000000000000000%
1949      0000000000000000}{2147483648}%
1950  \Test{1000000000000000%
1951      00000000000000000}{4294967296}%
1952  \Test{00010000000000000000%
1953      00000000000000000}{4294967296}%
1954  \Test{1100000000000000%
1955      00000000000000011}{6442450947}%
1956  \end{qstest}
1957
1958  \begin{qstest}{Clear}{Clear}
1959  \def\Test#1#2#3{%
1960    \RevSet{abc}{#1}%
1961    \bitsetClear{abc}{#2}%
1962    \Expect*{\BS@abc}*{\Reverse#3!!}%
1963  }%
1964  \bitsetClear{abc}{2}%
1965  \RevCheck{abc}{0}%
1966  \TestErrorNegativeIndex{-1}{\bitsetClear{abc}{-1}}%
1967  \RevCheck{abc}{0}%
1968  \Test{0}{0}{0}%
1969  \Test{1}{0}{0}%
1970  \Test{111}{1}{101}%
1971  \Test{111}{30}{111}%
1972  \Test{0000111}{5}{0000111}% 111 would also be ok
1973  \Test{10000111}{5}{10000111}%
1974  \Test{1001001}{3}{1000001}%
1975  \end{qstest}
1976
1977  \begin{qstest}{Set}{Set}
1978  \def\Test#1#2#3{%
1979    \RevSet{abc}{#1}%
1980    \bitsetSet{abc}{#2}%
1981    \Expect*{\BS@abc}*{\Reverse#3!!}%
1982  }%
1983  \bitsetSet{abc}{2}%
1984  \RevCheck{abc}{100}%
1985  \TestErrorNegativeIndex{-1}{\bitsetSet{abc}{-1}}%
1986  \RevCheck{abc}{100}%
1987  \Test{0}{0}{1}%
1988  \Test{1}{0}{1}%
1989  \Test{100}{1}{110}%
1990  \Test{111}{1}{111}%
1991  \Test{11}{1}{11}%
1992  \Test{11}{2}{111}%
1993  \Test{11}{3}{1011}%
1994  \Test{111}{10}{10000000111}%
```

```
1995  \Test{0000111}{5}{0100111}% 100111 would also be ok
1996  \Test{10000111}{5}{10100111}%
1997  \Test{1000001}{3}{1001001}%
1998  \Test{1001001}{3}{1001001}%
1999 \end{qstest}
2000
2001 \begin{qstest}{Flip}{Flip}
2002  \def\Test#1#2#3{%
2003    \RevSet{abc}{#1}%
2004    \bitsetFlip{abc}{#2}%
2005    \Expect*{\BS@abc}*{\Reverse#3!!}%
2006  }%
2007  \bitsetFlip{abc}{2}%
2008  \RevCheck{abc}{100}%
2009  \TestErrorNegativeIndex{-1}{\bitsetFlip{abc}{-1}}%
2010  \RevCheck{abc}{100}%
2011  \Test{0}{0}{1}%
2012  \Test{1}{0}{0}%
2013  \Test{0}{2}{100}%
2014  \Test{100}{1}{110}%
2015  \Test{111}{1}{101}%
2016  \Test{11}{1}{1}%
2017  \Test{11}{2}{111}%
2018  \Test{11}{3}{1011}%
2019  \Test{111}{10}{10000000111}%
2020  \Test{0000111}{5}{0100111}% 100111 would also be ok
2021  \Test{10000111}{5}{10100111}%
2022  \Test{1000001}{3}{1001001}%
2023  \Test{1001001}{3}{1000001}%
2024  \Test{11111}{2}{11011}%
2025 \end{qstest}
2026
2027 \begin{qstest}{SetValue}{SetValue}
2028  \def\Test#1#2{%
2029    \TestError{Invalid bit value (#2) not in range 0..1}{%
2030      \bitsetSetValue{abc}{#1}{#2}%
2031    }%
2032  }%
2033  \Test{0}{-1}%
2034  \Test{0}{2}%
2035  \Test{0}{10}%
2036  \def\Test#1#2#3{%
2037    \let\BS@abc\@undefined
2038    \bitsetSetValue{abc}{#1}{#2}%
2039    \bitsetSetBin{result}{#3}%
2040    \Expect*{\BS@abc}*{\BS@result}%
2041  }%
2042  \Test{0}{0}{0}%
2043  \Test{0}{1}{1}%
2044  \Test{1}{0}{0}%
2045  \Test{1}{1}{10}%
2046  \def\Test#1#2#3#4{%
2047    \bitsetSetBin{abc}{#1}%
2048    \bitsetSetBin{result}{#4}%
2049    \bitsetSetValue{abc}{#2}{#3}%
2050    \Expect*{\BS@abc}*{\BS@result}%
2051  }%
2052  \Test{0}{0}{0}{0}%
2053  \Test{0}{0}{0}{0}%
2054  \Test{0}{0}{1}{1}%
2055  \Test{0}{1}{0}{0}%
2056  \Test{0}{1}{1}{10}%
```

48

```
2057  \Test{1010}{2}{1}{1110}%
2058  \Test{1010}{4}{1}{11010}%
2059  \Test{1010}{6}{1}{1001010}%
2060  \Test{1010}{1}{0}{1000}%
2061  \Test{1010}{2}{0}{1010}%
2062  \Test{1010}{3}{0}{10}%
2063  \Test{1010}{4}{0}{1010}%
2064  \Test{1010}{6}{0}{1010}%
2065  \Test{1010}{2}{\csname iffalse\endcsname 0\else 1\fi}{1110}%
2066  \Test{1010}{1}{\csname iffalse\endcsname 1\else 0\fi}{1000}%
2067  \end{qstest}
2068
2069  \begin{qstest}{IsDefined}{IsDefined}
2070  \let\BS@abc\@undefined
2071  \Expect*{\bitsetIsDefined{abc}{true}{false}}{false}%
2072  \bitsetReset{abc}%
2073  \Expect*{\bitsetIsDefined{abc}{true}{false}}{true}%
2074  \end{qstest}
2075
2076  \begin{qstest}{IsEmpty}{IsEmpty}
2077  \let\BS@abc\@undefined
2078  \Expect*{\bitsetIsEmpty{abc}{true}{false}}{true}%
2079  \bitsetReset{abc}%
2080  \Expect*{\bitsetIsEmpty{abc}{true}{false}}{true}%
2081  \bitsetSet{abc}{1}%
2082  \Expect*{\bitsetIsEmpty{abc}{true}{false}}{false}%
2083  \end{qstest}
2084
2085  \begin{qstest}{Equals}{Equals}
2086  \def\Test#1#2#3{%
2087    \Expect*{\bitsetEquals{#1}{#2}{true}{false}}{#3}%
2088  }%
2089  \let\BS@abc\@undefined
2090  \Test{abc}{abc}{true}%
2091  \Test{abc}{foo}{true}%
2092  \Test{foo}{abc}{true}%
2093  \bitsetReset{abc}%
2094  \Test{abc}{abc}{true}%
2095  \Test{abc}{foo}{false}%
2096  \Test{foo}{abc}{false}%
2097  \bitsetReset{foo}%
2098  \Test{abc}{foo}{true}%
2099  \Test{foo}{abc}{true}%
2100  \bitsetSet{abc}{4}%
2101  \Test{abc}{foo}{false}%
2102  \Test{foo}{abc}{false}%
2103  \bitsetFlip{foo}{4}%
2104  \Test{abc}{foo}{true}%
2105  \Test{foo}{abc}{true}%
2106  \end{qstest}
2107
2108  \begin{qstest}{Intersects}{Intersects}
2109  \def\Test#1{%
2110    \Expect*{\bitsetIntersects{abc}{foo}{true}{false}}{#1}%
2111  }%
2112  \let\BS@abc\@undefined
2113  \let\BS@foo\@undefined
2114  \Test{false}%
2115  \Set{abc}{0}%
2116  \Test{false}%
2117  \Set{foo}{0}%
2118  \Test{false}%
```

```
2119  \let\BS@abc\@undefined
2120  \Test{false}%
2121  \Set{foo}{1}%
2122  \Test{false}%
2123  \Set{abc}{0}%
2124  \Test{false}%
2125  \Set{abc}{1}%
2126  \Test{true}%
2127  \let\BS@foo\@undefined
2128  \Test{false}%
2129  \Set{foo}{0}%
2130  \Test{false}%
2131  \def\Test#1#2#3{%
2132    \bitsetSetBin{abc}{#1}%
2133    \bitsetSetBin{foo}{#2}%
2134    \Expect*{\bitsetIntersects{abc}{foo}{true}{false}}{#3}%
2135  }%
2136  \Test{1010}{0101}{false}%
2137  \Test{0}{10}{false}%
2138  \Test{1}{11}{true}%
2139  \Test{11}{1}{true}%
2140  \Test{10}{1}{false}%
2141  \end{qstest}
2142
2143  \begin{qstest}{And/AndNot/Or/Xor}{And/AndNot/Or/Xor}
2144  \def\@Test#1#2#3#4#5{%
2145    \begingroup
2146      #5%
2147      \begingroup
2148        \let\BS@foo\@undefined
2149        \csname bitset#1\endcsname{abc}{foo}%
2150        \CheckUndef{foo}%
2151        \Check{abc}{#2}%
2152      \endgroup
2153      \begingroup
2154        \bitsetReset{foo}%
2155        \csname bitset#1\endcsname{abc}{foo}%
2156        \Check{foo}{0}%
2157        \Check{abc}{#3}%
2158      \endgroup
2159      \begingroup
2160        \def\BS@foo{0101}%
2161        \csname bitset#1\endcsname{abc}{foo}%
2162        \Check{foo}{0101}%
2163        \Check{abc}{#4}%
2164      \endgroup
2165    \endgroup
2166  }%
2167  \def\Test#1{%
2168    \def\Op{#1}%
2169    \Test@
2170  }%
2171  \def\Test@#1#2#3#4#5#6#7#8#9{%
2172    \@Test\Op{#1}{#2}{#3}{%
2173      \let\BS@abc\@undefined
2174    }%
2175    \@Test\Op{#4}{#5}{#6}{%
2176      \bitsetReset{abc}%
2177    }%
2178    \@Test\Op{#7}{#8}{#9}{%
2179      \def\BS@abc{1001}%
2180    }%
```

```
2181  }%
2182  \Test{And}%
2183      {0}{0}{0}%
2184      {0}{0}{0}%
2185      {0}{0}{0001}%
2186  \Test{AndNot}%
2187      {0}{0}{0}%
2188      {0}{0}{0}%
2189      {1001}{1001}{1}%
2190  \Test{Or}%
2191      {0}{0}{0101}%
2192      {0}{0}{0101}%
2193      {1001}{1001}{1101}%
2194  \Test{Xor}%
2195      {0}{0}{0101}%
2196      {0}{0}{0101}%
2197      {1001}{1001}{11}%
2198  \def\Test#1#2#3{%
2199    \bitsetSetBin{abc}{#1}%
2200    \bitsetSetBin{foo}{#2}%
2201    \csname bitset\Op\endcsname{abc}{foo}%
2202    \RevCheck{foo}{#2}%
2203    \RevCheck{abc}{#3}%
2204  }%
2205  \def\Op{And}%
2206  \Test{1}{111}{1}%
2207  \Test{111}{1}{1}%
2208  \Test{10}{111}{10}%
2209  \Test{111}{10}{10}%
2210  \Test{111}{1000}{0}%
2211  \Test{1000}{111}{0}%
2212  \def\Op{AndNot}%
2213  \Test{1010}{11}{1000}%
2214  \Test{100}{100}{0}%
2215  \Test{111}{1111}{0}%
2216  \Test{100}{111}{0}%
2217  \def\Op{Or}%
2218  \Test{0}{0}{0}%
2219  \Test{1}{0}{1}%
2220  \Test{0}{1}{1}%
2221  \Test{1}{1}{1}%
2222  \Test{1000}{10}{1010}%
2223  \Test{10}{1000}{1010}%
2224  \def\Op{Xor}%
2225  \Test{0}{0}{0}%
2226  \Test{1}{0}{1}%
2227  \Test{0}{1}{1}%
2228  \Test{1}{1}{0}%
2229  \Test{1000}{10}{1010}%
2230  \Test{10}{1000}{1010}%
2231  \Test  {110011001100}%
2232      {111000111000111}%
2233      {111110100001011}%
2234  \Test{111000111000111}%
2235        {110011001100}%
2236      {111110100001011}%
2237  \end{qstest}
2238
2239  \begin{qstest}{GetUndef}{GetUndef, GetBin, GetOct, GetHex}
2240    \def\TestUndef#1#2{%
2241    \let\BS@abc\@undefined
2242    \expandafter\expandafter\expandafter\Expect
```

```
2243      \expandafter\expandafter\expandafter{%
2244        \x{abc}{#1}%
2245      }{#2}%
2246    }%
2247    \let\x\bitsetGetBin
2248    \TestUndef{-1}{0}%
2249    \TestUndef{0}{0}%
2250    \TestUndef{1}{0}%
2251    \TestUndef{2}{00}%
2252    \TestUndef{8}{00000000}%
2253    \let\x\bitsetGetOct
2254    \TestUndef{-1}{0}%
2255    \TestUndef{0}{0}%
2256    \TestUndef{1}{0}%
2257    \TestUndef{2}{0}%
2258    \TestUndef{3}{0}%
2259    \TestUndef{4}{00}%
2260    \TestUndef{5}{00}%
2261    \TestUndef{6}{00}%
2262    \TestUndef{7}{000}%
2263    \TestUndef{8}{000}%
2264    \TestUndef{9}{000}%
2265    \TestUndef{10}{0000}%
2266    \let\x\bitsetGetHex
2267    \TestUndef{-1}{0}%
2268    \TestUndef{0}{0}%
2269    \TestUndef{1}{0}%
2270    \TestUndef{2}{0}%
2271    \TestUndef{3}{0}%
2272    \TestUndef{4}{0}%
2273    \TestUndef{5}{00}%
2274    \TestUndef{6}{00}%
2275    \TestUndef{7}{00}%
2276    \TestUndef{8}{00}%
2277    \TestUndef{9}{000}%
2278    \TestUndef{10}{000}%
2279    \TestUndef{12}{000}%
2280    \TestUndef{13}{0000}%
2281    \TestUndef{16}{0000}%
2282    \TestUndef{17}{00000}%
2283  \end{qstest}
2284
2285  \begin{qstest}{SetBin}{SetBin}
2286    \def\Test#1#2{%
2287      \let\BS@abc\@undefined
2288      \bitsetSetBin{abc}{#1}%
2289      \expandafter\Expect\expandafter{\BS@abc}{#2}%
2290    }%
2291    \Test{}{0}%
2292    \Test{0}{0}%
2293    \Test{1}{1}%
2294    \Test{10}{01}%
2295    \Test{11}{11}%
2296    \Test{010}{01}%
2297    \Test{011}{11}%
2298    \Test{0010}{01}%
2299    \Test{1010}{0101}%
2300  \end{qstest}
2301
2302  \begin{qstest}{SetOct}{SetOct}
2303    \def\Test#1#2{%
2304      \bitsetSetOct{abc}{#1}%
```

```
2305     \expandafter\Expect\expandafter{\BS@abc}{#2}%
2306   }%
2307   \Test{}{0}%
2308   \Test{0}{0}%
2309   \Test{000}{0}%
2310   \Test{1}{1}%
2311   \Test{001}{1}%
2312   \Test{010}{0001}%
2313   \Test{020}{00001}%
2314   \Test{42}{010001}%
2315   \Test{377}{11111111}%
2316   \Test{0377}{11111111}%
2317   \Test{76543210}{00010001011000110100111111}%
2318   \Test{ 0 7 0 7 1 }{100111000111}%
2319 \end{qstest}
2320
2321 \begin{qstest}{SetHex}{SetHex}
2322   \def\Test#1#2{%
2323     \bitsetSetHex{abc}{#1}%
2324     \expandafter\Expect\expandafter{\BS@abc}{#2}%
2325   }%
2326   \Test{}{0}%
2327   \Test{0}{0}%
2328   \Test{000}{0}%
2329   \Test{1}{1}%
2330   \Test{001}{1}%
2331   \Test{010}{00001}%
2332   \Test{020}{000001}%
2333   \Test{42}{0100001}%
2334   \Test{3F}{111111}%
2335   \Test{03F}{111111}%
2336   \Test{43210}{0000100001001100001}%
2337   \Test{98765}{10100110111000011001}%
2338   \Test{FEDCBA}{010111010011101101111111}%
2339   \Test{ 0 F 0 F 1 }{1000111100001111}%
2340 \end{qstest}
2341
2342 \begin{qstest}{SetDec}{SetDec}
2343   \def\Test#1#2{%
2344     \bitsetSetDec{abc}{#1}%
2345     \expandafter\Expect\expandafter{\BS@abc}{#2}%
2346   }%
2347   \Test{}{0}%
2348   \Test{0}{0}%
2349   \Test{000}{0}%
2350   \Test{1}{1}%
2351   \Test{7}{111}%
2352   \Test{8}{0001}%
2353   \Test{001}{1}%
2354   \Test{010}{0101}%
2355   \Test{020}{00101}%
2356   \Test{53}{101011}%
2357   \Test{255}{11111111}%
2358   \Test{256}{000000001}%
2359   \Test{999999999}{11111111100100110101100111011111}%
2360   \Test{1000000000}{00000000010100110101100111011111}%
2361   \Test{4210987654}{0110000101001001011111101011111}%
2362   \Test{2147483647}{1111111111111111111111111111111}%
2363   \Test{2147483648}{00000000000000000000000000000001}%
2364 \end{qstest}
2365
2366 \begin{qstest}{GetBin}{GetBin}
```

```
2367  \def\TestUndef#1#2{%
2368    \let\BS@abc\@undefined
2369    \expandafter\expandafter\expandafter\Expect
2370    \expandafter\expandafter\expandafter{%
2371      \bitsetGetBin{abc}{#1}%
2372    }{#2}%
2373  }%
2374  \TestUndef{-1}{0}%
2375  \TestUndef{0}{0}%
2376  \TestUndef{1}{0}%
2377  \TestUndef{2}{00}%
2378  \TestUndef{8}{00000000}%
2379  \def\Test#1#2{%
2380    \bitsetSetBin{abc}{#2}%
2381    \expandafter\expandafter\expandafter\Expect
2382    \expandafter\expandafter\expandafter{%
2383      \bitsetGetBin{abc}{#1}%
2384    }{#2}%
2385  }%
2386  \Test{-1}{0}%
2387  \Test{0}{0}%
2388  \Test{1}{0}%
2389  \Test{1}{1}%
2390  \Test{2}{01}%
2391  \Test{2}{10}%
2392  \Test{3}{010}%
2393  \Test{2}{00}%
2394  \Test{2}{01}%
2395  \Test{8}{00101100}%
2396  \Test{2}{10101}%
2397  \Test{-100}{11011}%
2398  \end{qstest}
2399
2400  \begin{qstest}{GetOct}{GetOct}
2401    \def\Test#1#2#3{%
2402      \edef\x{\zap@space#1 \@empty}%
2403      \edef\x{\noexpand\bitsetSetBin{abc}{\x}}%
2404      \x
2405      \expandafter\expandafter\expandafter\Expect
2406      \expandafter\expandafter\expandafter{%
2407        \bitsetGetOct{abc}{#2}%
2408      }{#3}%
2409    }%
2410    \Test{111 110 101 100 011 010 001 000}{0}{76543210}%
2411    \Test{000 111}{0}{7}%
2412    \Test{101 000}{-1}{50}%
2413    \Test{111}{-1}{7}%
2414    \Test{111}{0}{7}%
2415    \Test{111}{1}{7}%
2416    \Test{111}{3}{7}%
2417    \Test{111}{4}{07}%
2418    \Test{111}{6}{07}%
2419    \Test{111}{7}{007}%
2420    \Test{111 010}{6}{72}%
2421    \Test{111 010}{7}{072}%
2422    \Test{011 111}{0}{37}%
2423    \Test{011 111}{6}{37}%
2424    \Test{011 111}{7}{037}%
2425    \Test{001 111}{0}{17}%
2426    \Test{001 111}{6}{17}%
2427    \Test{001 111}{7}{017}%
2428  \end{qstest}
```

```
2429
2430 \begin{qstest}{GetHex}{GetHex}
2431   \def\Test#1#2#3{%
2432     \bitsetSetBin{abc}{#1}%
2433     \expandafter\expandafter\expandafter\Expect
2434     \expandafter\expandafter\expandafter{%
2435       \bitsetGetHex{abc}{#2}%
2436     }{#3}%
2437   }%
2438   \Test{1111 1110 1101 1100 1011 1010 1001 1000}{0}{FEDCBA98}%
2439   \Test{0111 0110 0101 0100 0011 0010 0001 0000}{0}{76543210}%
2440   \Test{0000 1111}{0}{F}%
2441   \Test{0101 0000}{-1}{50}%
2442   \Test{1111}{-1}{F}%
2443   \Test{1111}{0}{F}%
2444   \Test{1111}{1}{F}%
2445   \Test{1111}{4}{F}%
2446   \Test{1111}{5}{0F}%
2447   \Test{1111}{8}{0F}%
2448   \Test{1111}{9}{00F}%
2449   \Test{1111 0010}{8}{F2}%
2450   \Test{1111 0010}{9}{0F2}%
2451   \Test{0111 1111}{0}{7F}%
2452   \Test{0111 1111}{8}{7F}%
2453   \Test{0111 1111}{9}{07F}%
2454   \Test{0011 1111}{0}{3F}%
2455   \Test{0011 1111}{8}{3F}%
2456   \Test{0011 1111}{9}{03F}%
2457   \Test{0001 1111}{0}{1F}%
2458   \Test{0001 1111}{8}{1F}%
2459   \Test{0001 1111}{9}{01F}%
2460 \end{qstest}
2461
2462 \begin{qstest}{Range}{Range}
2463   \TestError{%
2464     Wrong index numbers in range [9..8]\MessageBreak% hash-ok
2465     for clear/set/flip on bit set 'abc'.\MessageBreak
2466     The lower index exceeds the upper index.\MessageBreak
2467     Canceling the operation as error recovery%
2468   }{%
2469     \bitsetSetRange{abc}{9}{8}%
2470   }%
2471   \def\TestErrorNegInd#1#2#3#4#5#6{%
2472     \TestError{%
2473       Negative index in range [#2..#3]\MessageBreak % hash-ok
2474       for \string\bitset #1Range on bit set 'abc'.\MessageBreak
2475       Using [#4..#5] as error recovery% hash-ok
2476     }{%
2477       \csname bitset#1Range\endcsname{abc}{#2}{#3}%
2478       \global\let\BS@global\BS@abc
2479     }%
2480     \Check{global}{#6}%
2481   }%
2482   \Set{abc}{111}%
2483   \TestErrorNegInd{Clear}{-1}{0}{0}{0}{111}%
2484   \TestErrorNegInd{Clear}{0}{-1}{0}{0}{111}%
2485   \TestErrorNegInd{Clear}{-2}{2}{0}{2}{001}%
2486   \bitsetReset{abc}%
2487   \TestErrorNegInd{Set}{-1}{0}{0}{0}{0}%
2488   \TestErrorNegInd{Set}{0}{-1}{0}{0}{0}%
2489   \TestErrorNegInd{Set}{-2}{2}{0}{2}{11}%
2490   \Set{abc}{101}%
```

```
2491  \TestErrorNegInd{Flip}{-1}{0}{0}{0}{101}%
2492  \TestErrorNegInd{Flip}{0}{-1}{0}{0}{101}%
2493  \TestErrorNegInd{Flip}{-2}{2}{0}{2}{011}%
2494  \def\Test#1#2#3#4{%
2495    \bitsetSetBin{abc}{#1}%
2496    \csname bitset\TestOp Range\endcsname{abc}{#2}{#3}%
2497    \Expect*{\bitsetGetBin{abc}{0}}{#4}%
2498  }%
2499  \def\TestOp{Clear}%
2500  \Test{0}{0}{1}{0}%
2501  \Test{1111}{1}{2}{1101}%
2502  \Test{1111}{1}{3}{1001}%
2503  \Test{1111111100000000}{12}{14}{1100111100000000}%
2504  \def\TestOp{Set}%
2505  \Test{0}{0}{1}{1}%
2506  \Test{1000}{1}{2}{1010}%
2507  \Test{0}{1}{2}{10}%
2508  \Test{1}{12}{15}{111000000000001}%
2509  \Test{1111}{1}{3}{1111}%
2510  \Test{1000000000000000}{12}{14}{1011000000000000}%
2511  \def\TestOp{Flip}%
2512  \Test{0}{0}{1}{1}%
2513  \Test{1}{0}{1}{0}%
2514  \Test{10101010}{1}{5}{10110100}%
2515  \def\Test#1#2#3#4#5{%
2516    \bitsetSetBin{abc}{#1}%
2517    \bitsetSetValueRange{abc}{#2}{#3}{#4}%
2518    \Expect*{\bitsetGetBin{abc}{0}}{#5}%
2519  }%
2520  \Test{0}{0}{1}{0}{0}%
2521  \Test{0}{0}{1}{1}{1}%
2522  \Test{1010}{1}{3}{0}{1000}%
2523  \Test{1010}{1}{3}{1}{1110}%
2524  \end{qstest}
2525
2526  \begin{qstest}{ShiftLeft/ShiftRight}{ShiftLeft/ShiftRight}
2527  \def\@Test#1#2{%
2528    \let\BS@abc\@undefined
2529    \csname bitsetShift#1\endcsname{abc}{#2}%
2530    \Expect*{\BS@abc}{0}%
2531  }%
2532  \def\Test#1{%
2533    \@Test{Left}{#1}%
2534    \@Test{Right}{#1}%
2535  }%
2536  \Test{-16}%
2537  \Test{-1}%
2538  \Test{0}%
2539  \Test{1}%
2540  \Test{16}%
2541  \def\Test#1#2#3{%
2542    \bitsetSetBin{abc}{#1}%
2543    \bitsetSetBin{result}{#3}%
2544    \csname bitsetShift\Op\endcsname{abc}{#2}%
2545    \Expect*{\bitsetGetBin{abc}{0}}*{\bitsetGetBin{result}{0}}%
2546  }%
2547  \def\Op{Left}%
2548  \Test{0}{0}{0}%
2549  \Test{0}{1}{0}%
2550  \Test{0}{-1}{0}%
2551  \Test{1}{0}{1}%
2552  \Test{1}{1}{10}%
```

```
2553  \Test{1}{-1}{0}%
2554  \Test{10}{1}{100}%
2555  \Test{10}{-1}{1}%
2556  \Test{1}{32}{100000000000000000000000000000000}%
2557  \Test{1}{-100}{0}%
2558  \def\Op{Right}%
2559  \Test{0}{0}{0}%
2560  \Test{0}{1}{0}%
2561  \Test{0}{-1}{0}%
2562  \Test{1}{0}{1}%
2563  \Test{1}{1}{0}%
2564  \Test{1}{-1}{10}%
2565  \Test{10}{1}{1}%
2566  \Test{10}{-1}{100}%
2567  \Test{1}{-32}{100000000000000000000000000000000}%
2568  \Test{1}{100}{0}%
2569  \Test{110110110110110}{10}{11011}%
2570  \Test{110110110110110}{100}{0}%
2571  \Test{1}{100000}{0}%
2572 \end{qstest}
2573
2574 \begin{qstest}{Profile: Set}{Profile: Set}
2575  \bitsetSet{abc}{4095}%
2576  \global\let\BS@global\BS@abc
2577 \end{qstest}
2578
2579 \begin{qstest}{Profile: Get}{Profile: Get}
2580  \edef\x{\bitsetGet{global}{4095}}%
2581 \end{qstest}
2582
2583 \begin{document}
2584 \end{document}
2585 ⟨/test2⟩
```

4 Installation

4.1 Download

Package.   This package is available on CTAN[1]:

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 TeX Files" (**CTAN:tds/tds.pdf**). Directories with **texmf** in their name are usually organized this way.

4.2 Bundle installation

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

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

---

[1] http://ctan.org/pkg/bitset

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 TeX:

```
tex bitset.dtx
```

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

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

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

4.4 Refresh file name databases

If your TeX distribution (teTeX, mikTeX, …) relies on file name databases, you must refresh these. For example, teTeX users run `texhash` or `mktexlsr`.

4.5 Some details for the interested

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 pdfLaTeX:

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

5 Catalogue

The following XML file can be used as source for the [TEX Catalogue](). The elements `caption` and `description` are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is `bitset.xml`.

2586 ⟨*catalogue⟩
2587 <?xml version='1.0' encoding='us-ascii'?>
2588 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
2589 <entry datestamp='$Date$' modifier='$Author$' id='bitset'>
2590   <name>bitset</name>
2591   <caption>Handle bit-vector datatype.</caption>
2592   <authorref id='auth:oberdiek'/>
2593   <copyright owner='Heiko Oberdiek' year='2007,2011'/>
2594   <license type='lppl1.3'/>
2595   <version number='1.2'/>
2596   <description>
2597   This package defines and implements the data type bit set,
2598   a vector of bits. The size of the vector may grow dynamically.
2599   Individual bits can be manipulated.
2600   <p/>
2601   The package is part of the <xref refid='oberdiek'>oberdiek</xref> bundle.
2602   </description>
2603   <documentation details='Package documentation'
2604   href='ctan:/macros/latex/contrib/oberdiek/bitset.pdf'/>
2605   <ctan file='true' path='/macros/latex/contrib/oberdiek/bitset.dtx'/>
2606   <miktex location='oberdiek'/>
2607   <texlive location='oberdiek'/>
2608   <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip'/>
2609 </entry>
2610 ⟨/catalogue⟩

6 History

[2007/09/28 v1.0]

- First version.

[2011/01/30 v1.1]

- Already loaded package files are not input in plain TEX.

[2016/05/16 v1.2]

- Documentation updates.

7 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; plain numbers refer to the code lines where the entry is used.

60

61

63