

---

# PHP BitArray

*Release 1.0.7*

**Ch. Demko**

**Feb 10, 2024**



# CONTENTS

<b>1</b>	<b>PHP BitArray</b>	<b>1</b>
1.1	Instructions . . . . .	1
1.2	Documentation . . . . .	2
1.3	Citation . . . . .	2
<b>2</b>	<b>API documentation</b>	<b>3</b>
2.1	BitArray . . . . .	3
<b>3</b>	<b>Indices and tables</b>	<b>11</b>



## PHP BITARRAY

BitArray for PHP.

This project manipulates compact array of bit values stored internally as strings.

The bit arrays may have variable length specified when an object is created using either:

- a specific size;
- a traversable collection;
- a string representation of bits;
- a json representation of bits;
- a slice from another bit array;
- a concatenation from two others bit arrays.

The project provides methods to get and set bits values using PHP natural syntax as well as the iterator facility offered by the PHP `foreach` language construct. It also provides methods for bitwise logical operations between two bit arrays `and`, `or`, `xor` and the `not` operation.

This project uses:

- [PHP Code Sniffer](#) for checking PHP code style
- [PHPUnit](#) for unit test (100% covered)
- [Sphinx](#) and [Doxygen](#) for the [documentation](#)

### 1.1 Instructions

Using composer: either

```
$ composer create-project chdemko/bitarray:1.2.x-dev --dev; cd bitarray
```

or create a `composer.json` file containing

```
{
  "require": {
    "chedemko/bitarray": "1.2.x-dev"
  }
}
```

and run

```
$ composer install
```

Create a `test.php` file containing

```
<?php

require __DIR__ . '/vendor/autoload.php';

use chdemko\BitArray\BitArray;

$bits = BitArray::fromTraversable([true,false,false,true]);
echo $bits . PHP_EOL;
```

This should print

```
1001
```

See the [examples](#) folder for more information.

## 1.2 Documentation

Run

```
$ sudo apt install doxygen python3-pip python3-virtualenv
$ virtualenv venv
$ venv/bin/activate
(venv) $ pip install -r docs/requirements.txt
(venv) $ sphinx-build -b html docs/ html/
(venv) $ deactivate
$
```

if you want to create local documentation with Sphinx.

## 1.3 Citation

If you are using this project including publication in research activities, you have to cite it using ([BibTeX format](#)). You are also pleased to send me an email to [chdemko@gmail.com](mailto:chdemko@gmail.com).

- authors: Christophe Demko
- title: php-bitarray: a PHP library for handling bit arrays
- year: 2014
- how published: <https://packagist.org/packages/chdemko/bitarray/>

## API DOCUMENTATION

## 2.1 BitArray

**BitArray()** → :BitArray : public ArrayAccess, public Countable, public IteratorAggregate, public JsonSerializable

### Public Functions

**chdemko\BitArray\BitArray::\_\_clone()**

Clone a BitArray

**Since**

1.0.0

**return**

void

**chdemko\BitArray\BitArray::\_\_toString()**

Convert the object to a string

**Since**

1.0.0

**return**

string String representation of this object

**chdemko\BitArray\BitArray::\_\_get( \$property)**

Magic get method

**Since**

1.0.0

**param \$property**

The property

**throws RuntimeException**

If the property does not exist

**return**

mixed The value associated to the property

**chdemko\BitArray\BitArray::offsetExists( \$offset)**

Test the existence of an index

**Since**

1.0.0

**param \$offset**

The offset

**return**

boolean The truth value

**chdemko\BitArray\BitArray::offsetGet( \$offset)**

Get the truth value for an index

**Since**

1.0.0

**param \$offset**

The offset

**throws OutOfRangeException**

Argument index must be an positive integer lesser than the size

**return**

boolean The truth value

**chdemko\BitArray\BitArray::offsetSet( \$offset, \$value)**

Set the truth value for an index

**Since**

1.0.0

**param \$offset**

The offset

**param \$value**

The truth value

**throws OutOfRangeException**

Argument index must be an positive integer lesser than the size

**return**

void

**chdemko\BitArray\BitArray::offsetUnset( \$offset)**

Unset the existence of an index

**Since**

1.0.0



**param \$offset**

The index

**throws RuntimeException**

Values cannot be unset

**return**

void

**chdemko\BitArray\BitArray::count()**

Return the number of true bits

**Since**

1.0.0

**return**

integer The number of true bits

**chdemko\BitArray\BitArray::toArray()**

Transform the object to an array

**Since**

1.1.0

**return**

array Array of values

**chdemko\BitArray\BitArray::jsonSerialize()**

Serialize the object

**Since**

1.0.0

**return**

array Array of values

**chdemko\BitArray\BitArray::getIterator()**

Get an iterator

**Since**

1.0.0

**return**

Iterator Iterator

**chdemko\BitArray\BitArray::size()**

Return the size

**Since**

1.0.0

**return**

integer The size

**chdemko\BitArray\BitArray::directCopy(BitArray \$bits, \$index = 0, \$offset = 0, \$size = 0)**

Copy bits directly from a BitArray

**Since**

1.1.0

**param \$bits**

A BitArray to copy

**param \$index**

Starting index for destination

**param \$offset**

Starting index for copying

**param \$size**

Copy size

**throws OutOfRangeException**

Argument index must be an positive integer lesser than the size

**return**

BitArray This object for chaining

**chdemko\BitArray\BitArray::copy(BitArray \$bits, \$index = 0, \$offset = 0, \$size = null)**

Copy bits from a BitArray

- if index is non-negative, the index parameter is used as it is, keeping its real value between 0 and size-1;
- if index is negative, the index parameter starts from the end, keeping its real value between 0 and size-1.
- if offset is non-negative, the offset parameter is used as it is, keeping its positive value between 0 and size-1;
- if offset is negative, the offset parameter starts from the end, keeping its real value between 0 and size-1.
- if size is given and is positive, then the copy will copy size elements.
- if the bits argument is shorter than the size, then only the available elements will be copied.
- if size is given and is negative, then the copy starts from the end.
- if size is omitted, then the copy will have everything from offset up until the end of the bits argument.

**Since**

1.1.0

**param \$bits**

A BitArray to copy

**param \$index**

Starting index for destination.

**param \$offset**

Starting index for copying.

**param \$size**

Copy size.

**return**

BitArray This object for chaining

**chdemko\BitArray\BitArray::applyComplement()**

Complement the bit array

**Since**

1.0.0

**return**

BitArray This object for chaining

**chdemko\BitArray\BitArray::applyOr(BitArray \$bits)**

Or with an another bit array

**Since**

1.0.0

**param \$bits**

A bit array

**throws InvalidArgumentException**

Argument must be of equal size

**return**

BitArray This object for chaining

**chdemko\BitArray\BitArray::applyAnd(BitArray \$bits)**

And with an another bit array

**Since**

1.0.0

**param \$bits**

A bit array

**throws InvalidArgumentException**

Argument must be of equal size

**return**

BitArray This object for chaining

**chdemko\BitArray\BitArray::applyXor(BitArray \$bits)**

Xor with an another bit array

**Since**

1.0.0

**param \$bits**

A bit array

**throws InvalidArgumentException**

Argument must be of equal size

**return**

BitArray This object for chaining

**chdemko\BitArray\BitArray::shift( \$size = 1, \$value = false)**

Shift a bit array.

Negative value means the shifting is done right to left while positive value means the shifting is done left to right.

**Since**

1.2.0

**param \$size**

Size to shift.

**param \$value**

Value to shift

**return**

BitArray \$this for chaining

## Public Static Functions

**static chdemko\BitArray\BitArray::fromInteger( \$size, \$default = false)**

Create a new BitArray from an integer

**Since**

1.0.0

**param \$size**

Size of the BitArray

**param \$default**

The default value for bits

**return**

BitArray A new BitArray

**static chdemko\BitArray\BitArray::fromDecimal( \$size, \$values = 0)**

Create a new BitArray from a sequence of bits.

**Since**

1.2.0

**param \$size**

Size of the BitArray

**param \$values**

The values for the bits

**return**

BitArray A new BitArray

**static chdemko\BitArray\BitArray::fromTraversable( \$traversable)**

Create a new BitArray from a traversable

**Since**

1.0.0

**param \$traversable**

A traversable and countable

**return**

BitArray A new BitArray

**static chdemko\BitArray\BitArray::fromString( \$string)**

Create a new BitArray from a bit string

**Since**

1.0.0

**param \$string**

A bit string

**return**

BitArray A new BitArray

**static chdemko\BitArray\BitArray::fromJson( \$json)**

Create a new BitArray from json

**Since**

1.0.0

**param \$json**

A json encoded value

**return**

BitArray A new BitArray

**static chdemko\BitArray\BitArray::fromSlice(BitArray \$bits, \$offset = 0, \$size = null)**

Create a new BitArray using a slice

- if offset is non-negative, the slice will start at that offset in the bits argument.
- if offset is negative, the slice will start from the end of the bits argument.
- if size is given and is positive, then the slice will have up to that many elements in it.
- if the bits argument is shorter than the size, then only the available elements will be present.
- if size is given and is negative, then the slice will stop that many elements from the end of the bits argument.
- if size is omitted, then the slice will have everything from offset up until the end of the bits argument.

**Since**

1.1.0

**param \$bits**

A BitArray to get the slice

**param \$offset**

The offset

**param \$size**

The size

**return**

BitArray A new BitArray

**static chdemko\BitArray\BitArray::fromConcat(BitArray \$bits1, BitArray \$bits2)**

Create a new BitArray using the concat operation

**Since**

1.1.0

**param \$bits1**

A BitArray

**param \$bits2**

A BitArray

**return**

BitArray A new BitArray

## INDICES AND TABLES

- `genindex`