crccheck Documentation

Release 1.3.0

Martin Scharrer

Oct 13, 2022

CONTENTS

1 Classes to calculate CRCs and checksums from binary data **1**

2 crccheck package **3**
- 2.1 Submodules . . . 3
- 2.2 crccheck.base module . . . 3
- 2.3 crccheck.checksum module . . . 6
- 2.4 crccheck.crc module . . . 8
- 2.5 Module contents . . . 27

3 Supported CRCs **29**
- 3.1 Aliases . . . 31

4 How-to **33**
- 4.1 How to quickly calculate a CRC/checksum . . . 33
- 4.2 How to calculate over multiple data blocks . . . 33
- 4.3 How to use a CRC not implemented by the package . . . 33
- 4.4 How to calculate the CRC or checksum of a file . . . 34

5 Changelog **35**
- 5.1 v1.3.0 - 2022-10-13 . . . 35
- 5.2 v1.2.0 - 2022-10-06 . . . 35
- 5.3 v1.1.1 - 2022-10-06 . . . 35
- 5.4 v1.1 - 2021-11-25 . . . 35
- 5.5 v1.0 - 2020-09-21 . . . 35
- 5.6 v0.6 - 2016-04-03 . . . 36
- 5.7 v0.5 - 2016-03-30 . . . 36
- 5.8 v0.4 - 2016-03-29 . . . 36
- 5.9 v0.3 - 2016-03-28 . . . 36
- 5.10 v0.2 - 2016-03-27 . . . 36
- 5.11 v0.1 - 2015-09-23 . . . 37

6 Indices and tables **39**

Python Module Index **41**

Index **43**

CHAPTER

ONE

CLASSES TO CALCULATE CRCS AND CHECKSUMS FROM BINARY DATA

The *crccheck.crc* module implements all CRCs listed in the Catalogue of parametrised CRC algorithms:

CRC-3/GSM, CRC-3/ROHC, CRC-4/G-704, CRC-4/ITU, CRC-4/INTERLAKEN, CRC-5/EPC-C1G2, CRC-5/EPC, CRC-5/G-704, CRC-5/ITU, CRC-5/USB, CRC-6/CDMA2000-A, CRC-6/CDMA2000-B, CRC-6/DARC, CRC-6/G-704, CRC-6/ITU, CRC-6/GSM, CRC-7/MMC, CRC-7, CRC-7/ROHC, CRC-7/UMTS, CRC-8/AUTOSAR, CRC-8/BLUETOOTH, CRC-8/CDMA2000, CRC-8/DARC, CRC-8/DVB-S2, CRC-8/GSM-A, CRC-8/GSM-B, CRC-8/I-432-1, CRC-8/ITU, CRC-8/I-CODE, CRC-8/LTE, CRC-8/MAXIM-DOW, CRC-8/MAXIM, DOW-CRC, CRC-8/MIFARE-MAD, CRC-8/NRSC-5, CRC-8/OPENSAFETY, CRC-8/ROHC, CRC-8/SAE-J1850, CRC-8/SMBUS, CRC-8, CRC-8/TECH-3250, CRC-8/AES, CRC-8/EBU, CRC-8/WCDMA, CRC-10/ATM, CRC-10, CRC-10/I-610, CRC-10/CDMA2000, CRC-10/GSM, CRC-11/FLEXRAY, CRC-11, CRC-11/UMTS, CRC-12/CDMA2000, CRC-12/DECT, CRC-12-X, CRC-12/GSM, CRC-12/UMTS, CRC-12/3GPP, CRC-13/BBC, CRC-14/DARC, CRC-14/GSM, CRC-15/CAN, CRC-15, CRC-15/MPT1327, CRC-16/ARC, ARC, CRC-16/LHA, CRC-IBM, CRC-16/CDMA2000, CRC-16/CMS, CRC-16/DDS-110, CRC-16/DECT-R, R-CRC-16, CRC-16/DECT-X, X-CRC-16, CRC-16/DNP, CRC-16/EN-13757, CRC-16/GENIBUS, CRC-16/DARC, CRC-16/EPC, CRC-16/EPC-C1G2, CRC-16/I-CODE, CRC-16/GSM, CRC-16/IBM-3740, CRC-16/AUTOSAR, CRC-16/CCITT-FALSE, CRC-16/IBM-SDLC, CRC-16/ISO-HDLC, CRC-16/ISO-IEC-14443-3-B, CRC-16/X-25, CRC-B, X-25, CRC-16/ISO-IEC-14443-3-A, CRC-A, CRC-16/KERMIT, CRC-16/CCITT, CRC-16/CCITT-TRUE, CRC-16/V-41-LSB, CRC-CCITT, KERMIT, CRC-16/LJ1200, CRC-16/MAXIM-DOW, CRC-16/MAXIM, CRC-16/MCRF4XX, CRC-16/MODBUS, MODBUS, CRC-16/NRSC-5, CRC-16/OPENSAFETY-A, CRC-16/OPENSAFETY-B, CRC-16/PROFIBUS, CRC-16/IEC-61158-2, CRC-16/RIELLO, CRC-16/SPI-FUJITSU, CRC-16/AUG-CCITT, CRC-16/T10-DIF, CRC-16/TELEDISK, CRC-16/TMS37157, CRC-16/UMTS, CRC-16/BUYPASS, CRC-16/VERIFONE, CRC-16/USB, CRC-16/XMODEM, CRC-16/ACORN, CRC-16/LTE, CRC-16/V-41-MSB, XMODEM, ZMODEM, CRC-17/CAN-FD, CRC-21/CAN-FD, CRC-24/BLE, CRC-24/FLEXRAY-A, CRC-24/FLEXRAY-B, CRC-24/INTERLAKEN, CRC-24/LTE-A, CRC-24/LTE-B, CRC-24/OPENPGP, CRC-24, CRC-24/OS-9, CRC-30/CDMA, CRC-31/PHILIPS, CRC-32/AIXM, CRC-32Q, CRC-32/AUTOSAR, CRC-32/BASE91-D, CRC-32D, CRC-32/BZIP2, CRC-32/AAL5, CRC-32/DECT-B, B-CRC-32, CRC-32/CD-ROM-EDC, CRC-32/CKSUM, CKSUM, CRC-32/POSIX, CRC-32/ISCSI, CRC-32/BASE91-C, CRC-32/CASTAGNOLI, CRC-32/INTERLAKEN, CRC-32C, CRC-32/ISO-HDLC, CRC-32, CRC-32/ADCCP, CRC-32/V-42, CRC-32/XZ, PKZIP, CRC-32/JAMCRC, JAMCRC, CRC-32/MPEG-2, CRC-32/XFER, XFER, CRC-40/GSM, CRC-64/ECMA-182, CRC-64, CRC-64/GO-ISO, CRC-64/WE, CRC-64/XZ, CRC-64/GO-ECMA, CRC-82/DARC.

For the class names simply remove all dashes and slashes from the above names and apply CamelCase, e.g. "CRC-32/MPEG-2" is implemented by *Crc32Mpeg2*. Other CRC can be calculated by using the general class *crccheck.crc.Crc* by providing all required CRC parameters.

The *crccheck.checksum* module implements additive and XOR checksums with 8, 16 and 32 bit: *Checksum8*, *Checksum16*, *Checksum32* and *ChecksumXor8*, *ChecksumXor16*, *ChecksumXor32*.

Usage example:

```
from crccheck.crc import Crc32, CrcXmodem
from crccheck.checksum import Checksum32

# Quick calculation
data = bytearray.fromhex("DEADBEEF")
crc = Crc32.calc(data)
checksum = Checksum32.calc(data)

# Procsss multiple data buffers
data1 = b"Binary string"  # or use .encode(..) on normal sring - Python 3 only
data2 = bytes.fromhex("1234567890")  # Python 3 only, use bytearray for older versions
data3 = (0x0, 255, 12, 99)  # Iterable which returns ints in byte range (0..255)
crcinst = CrcXmodem()
crcinst.process(data1)
crcinst.process(data2)
crcinst.process(data3[1:-1])
crcbytes = crcinst.finalbytes()
crchex = crcinst.finalhex()
crcint = crcinst.final()
```


Contents:

CHAPTER

TWO

CRCCHECK PACKAGE

2.1 Submodules

2.2 crccheck.base module

Base class for CRC and checksum classes.

License:

```
MIT License

Copyright (c) 2015-2022 by Martin Scharrer <martin.scharrer@web.de>

Permission is hereby granted, free of charge, to any person obtaining a copy of this␣
↪software
and associated documentation files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy, modify, merge,␣
↪publish,
distribute, sublicense, and/or sell copies of the Software, and to permit persons to␣
↪whom the
Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or
substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,␣
↪INCLUDING
BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE␣
↪AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY␣
↪CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING␣
↪FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

crccheck.base.**reflectbitorder**(*width*, *value*)

Reflects the bit order of the given value according to the given bit width.

Parameters

- **width** (*int*) – bitwidth

- **value** (*int*) – value to reflect

exception crccheck.base.**CrccheckError**

Bases: Exception

General checksum error exception

class crccheck.base.**CrccheckBase**(*initvalue=None*)

Bases: object

Abstract base class for checksumming classes.

Parameters
initvalue (*int*) – Initial value. If None then the default value for the class is used.

reset(*value=None*)

Reset instance.

Resets the instance state to the initial value. This is not required for a just created instance.

Parameters
value (*int*) – Set internal value. If None then the default initial value for the class is used.

Returns
self

process(*data*)

Process given data.

Parameters
data (*bytes, bytearray or list of ints [0-255]*) – input data to process.

Returns
self

final()

Return final check value. The internal state is not modified by this so further data can be processed afterwards.

Returns
final value

Return type
int

finalhex(*byteorder='big'*)

Return final checksum value as hexadecimal string (without leading "0x"). The hex value is zero padded to bitwidth/8. The internal state is not modified by this so further data can be processed afterwards.

Returns
final value as hex string without leading '0x'.

Return type
str

finalbytes(*byteorder='big'*)

Return final checksum value as bytes. The internal state is not modified by this so further data can be processed afterwards.

Returns
final value as bytes

Return type
bytes

`value`()

Returns current intermediate value. Note that in general final() must be used to get the a final value.

Returns
current value

Return type
int

`classmethod calc`(*data*, *initvalue=None*, ***kwargs*)

Fully calculate CRC/checksum over given data.

Parameters

- **data** (*bytes, bytearray or list of ints [0-255]*) – input data to process.
- **initvalue** (*int*) – Initial value. If None then the default value for the class is used.

Returns
final value

Return type
int

`classmethod calchex`(*data*, *initvalue=None*, *byteorder='big'*, ***kwargs*)

Fully calculate checksum over given data. Return result as hex string.

Parameters

- **data** (*bytes, bytearray or list of ints [0-255]*) – input data to process.
- **initvalue** (*int*) – Initial value. If None then the default value for the class is used.
- **byteorder** (*'big' or 'little'*) – order (endianness) of returned bytes.

Returns
final value as hex string without leading '0x'.

Return type
str

`classmethod calcbytes`(*data*, *initvalue=None*, *byteorder='big'*, ***kwargs*)

Fully calculate checksum over given data. Return result as bytearray.

Parameters

- **data** (*bytes, bytearray or list of ints [0-255]*) – input data to process.
- **initvalue** (*int*) – Initial value. If None then the default value for the class is used.
- **byteorder** (*'big' or 'little'*) – order (endianness) of returned bytes.

Returns
final value as bytes

Return type
bytes

`classmethod selftest`(*data=None*, *expectedresult=None*, ***kwargs*)

Selftest method for automated tests.

Parameters

- **data** (*bytes, bytearray or list of int [0-255]*) – data to process
- **expectedresult** (*int*) – expected result

Raises

CrccheckError – if result is not as expected

2.3 crccheck.checksum module

Classes to calculated additive and XOR checksums.

License:

```
MIT License

Copyright (c) 2015-2022 by Martin Scharrer <martin.scharrer@web.de>

Permission is hereby granted, free of charge, to any person obtaining a copy of this␣
↪software
and associated documentation files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy, modify, merge,␣
↪publish,
distribute, sublicense, and/or sell copies of the Software, and to permit persons to␣
↪whom the
Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or
substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,␣
↪INCLUDING
BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE␣
↪AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY␣
↪CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING␣
↪FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

class crccheck.checksum.**ChecksumBase**(*initvalue=0*, *byteorder='big'*)

Bases: *CrccheckBase*

Base class for all checksum classes.

Parameters

- **initvalue** (*int*) – Initial value. If None then the default value for the class is used.
- **byteorder** (*'big' or 'little'*) – byte order (endianness) used when reading the input bytes.

process(*data*)

Process given data.

Parameters

data (*bytes, bytearray or list of ints [0-255]*) – input data to process.

Returns
self

classmethod selftest(*data=None*, *expectedresult=None*, *byteorder='big'*)

Selftest method for automated tests.

Parameters

- **data** (*bytes, bytearray or list of int [0-255]*) – data to process
- **expectedresult** (*int*) – expected result
- **byteorder** (*'big' or 'little'*) – byte order (endianness) used when reading the input bytes.

Raises
CrccheckError – if result is not as expected

class crccheck.checksum.**Checksum32**(*initvalue=0*, *byteorder='big'*)

Bases: *ChecksumBase*

32-bit checksum.

Calculates 32-bit checksum by adding the input bytes in groups of four. Input data length must be a multiple of four, otherwise the last bytes are not used.

class crccheck.checksum.**Checksum16**(*initvalue=0*, *byteorder='big'*)

Bases: *ChecksumBase*

16-bit checksum.

Calculates 16-bit checksum by adding the input bytes in groups of two. Input data length must be a multiple of two, otherwise the last byte is not used.

class crccheck.checksum.**Checksum8**(*initvalue=0*, *byteorder='big'*)

Bases: *ChecksumBase*

8-bit checksum.

Calculates 8-bit checksum by adding the input bytes.

class crccheck.checksum.**ChecksumXorBase**(*initvalue=0*, *byteorder='big'*)

Bases: *ChecksumBase*

Base class for all XOR checksum classes.

process(*data*)

Process given data.

Parameters
data (*bytes, bytearray or list of ints [0-255]*) – input data to process.

Returns
self

class crccheck.checksum.**ChecksumXor32**(*initvalue=0*, *byteorder='big'*)

Bases: *ChecksumXorBase*

32-bit XOR checksum.

Calculates 32-bit checksum by XOR-ing the input bytes in groups of four. Input data length must be a multiple of four, otherwise the last bytes are not used.

class crccheck.checksum.**ChecksumXor16**(*initvalue=0*, *byteorder='big'*)

Bases: *ChecksumXorBase*

16-bit XOR checksum.

Calculates 16-bit checksum by XOR-ing the input bytes in groups of two. Input data length must be a multiple of two, otherwise the last byte is not used.

class crccheck.checksum.**ChecksumXor8**(*initvalue=0*, *byteorder='big'*)

Bases: *ChecksumXorBase*

8-bit XOR checksum.

Calculates 8-bit checksum by XOR-ing the input bytes.

class crccheck.checksum.**Checksum**(*width*, *initvalue=0*, *byteorder='big'*)

Bases: *ChecksumBase*

General additive checksum.

Parameters

- **width** (*int*) – bit width of checksum. Must be positive and a multiple of 8.
- **initvalue** (*int*) – Initial value. If None then the default value for the class is used.
- **byteorder** (*'big' or 'little'*) – byte order (endianness) used when reading the input bytes.

class crccheck.checksum.**ChecksumXor**(*width*, *initvalue=0*, *byteorder='big'*)

Bases: *ChecksumXorBase*

General XOR checksum.

Parameters

- **width** (*int*) – bit width of checksum. Must be positive and a multiple of 8.
- **initvalue** (*int*) – Initial value. If None then the default value for the class is used.
- **byteorder** (*'big' or 'little'*) – byte order (endianness) used when reading the input bytes.

2.4 crccheck.crc module

Classes to calculate CRCs (Cyclic Redundancy Check).

License:

```
MIT License

Copyright (c) 2015-2022 by Martin Scharrer <martin.scharrer@web.de>

Permission is hereby granted, free of charge, to any person obtaining a copy of this␣
↪software
and associated documentation files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use, copy, modify, merge,␣
↪publish,
distribute, sublicense, and/or sell copies of the Software, and to permit persons to␣
↪whom the
```

(continues on next page)

(continued from previous page)

```
Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or
substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,␣
↪INCLUDING
BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE␣
↪AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY␣
↪CLAIM,
DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING␣
↪FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

class crccheck.crc.**CrcBase**(*initvalue=None*)

Bases: *CrccheckBase*

Abstract base class for all Cyclic Redundancy Checks (CRC) checksums

process(*data*)

Process given data.

Parameters
data (*bytes, bytearray or list of ints [0-255]*) – input data to process.

Returns
self

final()

Return final CRC value.

Returns
final CRC value

Return type
int

crccheck.crc.**find**(*classes=None*, *width=None*, *poly=None*, *initvalue=None*, *reflect_input=None*, *reflect_output=None*, *xor_output=None*, *check_result=None*, *residue=None*)

Find CRC classes which the matching properties.

Parameters

- **classes** (*None or list*) – List of classes to search in. If None the list ALLCRCCLASSES will be used.
- **width** (*None or int*) – number of bits of the CRC classes to find
- **poly** (*None or int*) – polygon to find
- **initvalue** (*None or int*) – initvalue to find
- **reflect_input** (*None or bool*) – reflect_input to find
- **reflect_output** (*None or bool*) – reflect_output to find
- **xor_output** (*None or int*) – xor_output to find
- **check_result** (*None or int*) – check_result to find

- **residue** (*None or int*) – residue to find

Returns
List of CRC classes with the selected properties.

Examples

Find all CRC16 classes:
$ find(width=16)

Find all CRC32 classes with all-1 init value and XOR output:
$ find(width=32, initvalue=0xFFFF, xor_output=0xFFFF)

crccheck.crc.**identify**(*data*, *crc*, *width=None*, *classes=None*, *one=True*)

Identify the used CRC algorithm which was used to calculate the CRC from some data.

This function can be used to identify a suitable CRC class if the exact CRC algorithm/parameters are not known, but a CRC value is known from some data. Note that this function can be quite time consuming on large data, especially if the given width is not known.

Parameters

- **data** (*bytes*) – Data to compare with the *crc*.
- **crc** (*int*) – Known CRC of the given *data*.
- **width** (*int or None*) – Known bit width of given *crc*. Providing the width will speed up the identification of the CRC algorithm.
- **classes** (*iterable or None*) – Listing of classes to check. If None then ALLCRC-CLASSES is used.
- **one** (*bool*) – If True then only the first found CRC class is retunred. Otherwise a list of all suitable CRC classes.

Returns

CRC class which instances produce the given CRC from the given data.
If no CRC class could be found *None* is returned.

If *one* is False:
List of CRC classes which instances produce the given CRC from the given data. The list may be empty.

Return type
If *one* is True

class crccheck.crc.**Crc**(*width*, *poly*, *initvalue=0*, *reflect_input=False*, *reflect_output=False*, *xor_output=0*, *check_result=0*, *residue=0*)

Bases: *CrcBase*

Creates a new general (user-defined) CRC calculator instance.

Parameters

- **width** (*int*) – bit width of CRC.
- **poly** (*int*) – polynomial of CRC with the top bit omitted.
- **initvalue** (*int*) – initial value of internal running CRC value. Usually either 0 or (1<<width)-1, i.e. “all-1s”.

- **reflect_input** (*bool*) – If true the bit order of the input bytes are reflected first. This is to calculate the CRC like least-significant bit first systems will do it.
- **reflect_output** (*bool*) – If true the bit order of the calculation result will be reflected before the XOR output stage.
- **xor_output** (*int*) – The result is bit-wise XOR-ed with this value. Usually 0 (value stays the same) or (1<<width)-1, i.e. "all-1s" (invert value).
- **check_result** (*int*) – The expected result for the check input "123456789" (= [0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39]). This value is used for the selftest() method to verify proper operation.
- **residue** (*int*) – The residue expected after calculating the CRC over the original data followed by the CRC of the original data. With initvalue=0 and xor_output=0 the residue calculates always to 0.

calc(*data*, *initvalue=None*, ***kwargs*)

Fully calculate CRC/checksum over given data.

Parameters

- **data** (*bytes, bytearray or list of ints [0-255]*) – input data to process.
- **initvalue** (*int*) – Initial value. If None then the default value for the class is used.

Returns
final value

Return type
int

calchex(*data*, *initvalue=None*, *byteorder='big'*, ***kwargs*)

Fully calculate checksum over given data. Return result as hex string.

Parameters

- **data** (*bytes, bytearray or list of ints [0-255]*) – input data to process.
- **initvalue** (*int*) – Initial value. If None then the default value for the class is used.
- **byteorder** (*'big' or 'little'*) – order (endianness) of returned bytes.

Returns
final value as hex string without leading '0x'.

Return type
str

calcbytes(*data*, *initvalue=None*, *byteorder='big'*, ***kwargs*)

Fully calculate checksum over given data. Return result as bytearray.

Parameters

- **data** (*bytes, bytearray or list of ints [0-255]*) – input data to process.
- **initvalue** (*int*) – Initial value. If None then the default value for the class is used.
- **byteorder** (*'big' or 'little'*) – order (endianness) of returned bytes.

Returns
final value as bytes

Return type
bytes

selftest(*data=None*, *expectedresult=None*, ***kwargs*)

Selftest method for automated tests.

Parameters

- **data** (*bytes, bytearray or list of int [0-255]*) – data to process
- **expectedresult** (*int*) – expected result

Raises

CrccheckError – if result is not as expected

class crccheck.crc.**Crc8Base**(*initvalue=None*)

Bases: *CrcBase*

CRC-8. Has optimised code for 8-bit CRCs and is used as base class for all other CRC with this width.

process(*data*)

Process given data.

Parameters

data (*bytes, bytearray or list of ints [0-255]*) – input data to process.

Returns

self

class crccheck.crc.**Crc16Base**(*initvalue=None*)

Bases: *CrcBase*

CRC-16. Has optimised code for 16-bit CRCs and is used as base class for all other CRC with this width.

process(*data*)

Process given data.

Parameters

data (*bytes, bytearray or list of ints [0-255]*) – input data to process.

Returns

self

class crccheck.crc.**Crc32Base**(*initvalue=None*)

Bases: *CrcBase*

CRC-32. Has optimised code for 32-bit CRCs and is used as base class for all other CRC with this width.

process(*data*)

Process given data.

Parameters

data (*bytes, bytearray or list of ints [0-255]*) – input data to process.

Returns

self

class crccheck.crc.**Crc3Gsm**(*initvalue=None*)

Bases: *CrcBase*

CRC-3/GSM

class crccheck.crc.**Crc3Rohc**(*initvalue=None*)

Bases: *CrcBase*

CRC-3/ROHC

class crccheck.crc.**Crc4G704**(*initvalue=None*)

Bases: *CrcBase*

CRC-4/G-704

Aliases: CRC-4/ITU

crccheck.crc.**Crc4Itu**

alias of *Crc4G704*

class crccheck.crc.**Crc4Interlaken**(*initvalue=None*)

Bases: *CrcBase*

CRC-4/INTERLAKEN

class crccheck.crc.**Crc5EpcC1G2**(*initvalue=None*)

Bases: *CrcBase*

CRC-5/EPC-C1G2

Aliases: CRC-5/EPC

crccheck.crc.**Crc5Epc**

alias of *Crc5EpcC1G2*

class crccheck.crc.**Crc5G704**(*initvalue=None*)

Bases: *CrcBase*

CRC-5/G-704

Aliases: CRC-5/ITU

crccheck.crc.**Crc5Itu**

alias of *Crc5G704*

class crccheck.crc.**Crc5Usb**(*initvalue=None*)

Bases: *CrcBase*

CRC-5/USB

class crccheck.crc.**Crc6Cdma2000A**(*initvalue=None*)

Bases: *CrcBase*

CRC-6/CDMA2000-A

class crccheck.crc.**Crc6Cdma2000B**(*initvalue=None*)

Bases: *CrcBase*

CRC-6/CDMA2000-B

class crccheck.crc.**Crc6Darc**(*initvalue=None*)

Bases: *CrcBase*

CRC-6/DARC

class crccheck.crc.**Crc6G704**(*initvalue=None*)

Bases: *CrcBase*

CRC-6/G-704

Aliases: CRC-6/ITU

crccheck.crc.**Crc6Itu**

alias of *Crc6G704*

class crccheck.crc.**Crc6Gsm**(*initvalue=None*)

Bases: *CrcBase*

CRC-6/GSM

class crccheck.crc.**Crc7Mmc**(*initvalue=None*)

Bases: *CrcBase*

CRC-7/MMC

Aliases: CRC-7

crccheck.crc.**Crc7**

alias of *Crc7Mmc*

class crccheck.crc.**Crc7Rohc**(*initvalue=None*)

Bases: *CrcBase*

CRC-7/ROHC

class crccheck.crc.**Crc7Umts**(*initvalue=None*)

Bases: *CrcBase*

CRC-7/UMTS

class crccheck.crc.**Crc8Autosar**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/AUTOSAR

class crccheck.crc.**Crc8Bluetooth**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/BLUETOOTH

class crccheck.crc.**Crc8Cdma2000**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/CDMA2000

class crccheck.crc.**Crc8Darc**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/DARC

class crccheck.crc.**Crc8DvbS2**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/DVB-S2

class crccheck.crc.**Crc8GsmA**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/GSM-A

class crccheck.crc.**Crc8GsmB**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/GSM-B

class crccheck.crc.**Crc8Hitag**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/HITAG

class crccheck.crc.**Crc8I4321**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/I-432-1

Aliases: CRC-8/ITU

crccheck.crc.**Crc8Itu**

alias of *Crc8I4321*

class crccheck.crc.**Crc8ICode**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/I-CODE

class crccheck.crc.**Crc8Lte**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/LTE

class crccheck.crc.**Crc8MaximDow**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/MAXIM-DOW

Aliases: CRC-8/MAXIM, DOW-CRC

crccheck.crc.**Crc8Maxim**

alias of *Crc8MaximDow*

crccheck.crc.**CrcDow**

alias of *Crc8MaximDow*

class crccheck.crc.**Crc8MifareMad**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/MIFARE-MAD

class crccheck.crc.**Crc8Nrsc5**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/NRSC-5

class crccheck.crc.**Crc8Opensafety**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/OPENSAFETY

class crccheck.crc.**Crc8Rohc**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/ROHC

class crccheck.crc.**Crc8SaeJ1850**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/SAE-J1850

class crccheck.crc.**Crc8Smbus**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/SMBUS

Aliases: CRC-8

crccheck.crc.**Crc8**

alias of *Crc8Smbus*

class crccheck.crc.**Crc8Tech3250**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/TECH-3250

Aliases: CRC-8/AES, CRC-8/EBU

crccheck.crc.**Crc8Aes**

alias of *Crc8Tech3250*

crccheck.crc.**Crc8Ebu**

alias of *Crc8Tech3250*

class crccheck.crc.**Crc8Wcdma**(*initvalue=None*)

Bases: *Crc8Base*

CRC-8/WCDMA

class crccheck.crc.**Crc10Atm**(*initvalue=None*)

Bases: *CrcBase*

CRC-10/ATM

Aliases: CRC-10, CRC-10/I-610

crccheck.crc.**Crc10**

alias of *Crc10Atm*

crccheck.crc.**Crc10I610**

alias of *Crc10Atm*

class crccheck.crc.**Crc10Cdma2000**(*initvalue=None*)

Bases: *CrcBase*

CRC-10/CDMA2000

class crccheck.crc.**Crc10Gsm**(*initvalue=None*)

Bases: *CrcBase*

CRC-10/GSM

class crccheck.crc.**Crc11Flexray**(*initvalue=None*)

Bases: *CrcBase*

CRC-11/FLEXRAY

Aliases: CRC-11

crccheck.crc.**Crc11**

alias of *Crc11Flexray*

class crccheck.crc.**Crc11Umts**(*initvalue=None*)

Bases: *CrcBase*

CRC-11/UMTS

class crccheck.crc.**Crc12Cdma2000**(*initvalue=None*)

Bases: *CrcBase*

CRC-12/CDMA2000

class crccheck.crc.**Crc12Dect**(*initvalue=None*)

Bases: *CrcBase*

CRC-12/DECT

Aliases: CRC-12-X

crccheck.crc.**Crc12X**

alias of *Crc12Dect*

class crccheck.crc.**Crc12Gsm**(*initvalue=None*)

Bases: *CrcBase*

CRC-12/GSM

class crccheck.crc.**Crc12Umts**(*initvalue=None*)

Bases: *CrcBase*

CRC-12/UMTS

Aliases: CRC-12/3GPP

crccheck.crc.**Crc123Gpp**

alias of *Crc12Umts*

class crccheck.crc.**Crc13Bbc**(*initvalue=None*)

Bases: *CrcBase*

CRC-13/BBC

class crccheck.crc.**Crc14Darc**(*initvalue=None*)

Bases: *CrcBase*

CRC-14/DARC

class crccheck.crc.**Crc14Gsm**(*initvalue=None*)

Bases: *CrcBase*

CRC-14/GSM

class crccheck.crc.**Crc15Can**(*initvalue=None*)

Bases: *CrcBase*

CRC-15/CAN

Aliases: CRC-15

crccheck.crc.**Crc15**

alias of *Crc15Can*

class crccheck.crc.**Crc15Mpt1327**(*initvalue=None*)

Bases: *CrcBase*

CRC-15/MPT1327

class crccheck.crc.**Crc16Arc**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/ARC

Aliases: ARC, CRC-16/LHA, CRC-IBM

crccheck.crc.**CrcArc**

alias of *Crc16Arc*

crccheck.crc.**Crc16Lha**

alias of *Crc16Arc*

crccheck.crc.**CrcIbm**

alias of *Crc16Arc*

class crccheck.crc.**Crc16Cdma2000**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/CDMA2000

class crccheck.crc.**Crc16Cms**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/CMS

class crccheck.crc.**Crc16Dds110**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/DDS-110

class crccheck.crc.**Crc16DectR**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/DECT-R

Aliases: R-CRC-16

crccheck.crc.**Crc16R**

alias of *Crc16DectR*

class crccheck.crc.**Crc16DectX**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/DECT-X

Aliases: X-CRC-16

crccheck.crc.**Crc16X**

alias of *Crc16DectX*

class crccheck.crc.**Crc16Dnp**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/DNP

class crccheck.crc.**Crc16En13757**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/EN-13757

class crccheck.crc.**Crc16Genibus**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/GENIBUS

Aliases: CRC-16/DARC, CRC-16/EPC, CRC-16/EPC-C1G2, CRC-16/I-CODE

crccheck.crc.**Crc16Darc**

alias of *Crc16Genibus*

crccheck.crc.**Crc16Epc**

alias of *Crc16Genibus*

crccheck.crc.**Crc16EpcC1G2**

alias of *Crc16Genibus*

crccheck.crc.**Crc16ICode**

alias of *Crc16Genibus*

class crccheck.crc.**Crc16Gsm**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/GSM

class crccheck.crc.**Crc16Ibm3740**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/IBM-3740

Aliases: CRC-16/AUTOSAR, CRC-16/CCITT-FALSE

crccheck.crc.**Crc16Autosar**

alias of *Crc16Ibm3740*

crccheck.crc.**Crc16CcittFalse**

alias of *Crc16Ibm3740*

class crccheck.crc.**Crc16IbmSdlc**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/IBM-SDLC

Aliases: CRC-16/ISO-HDLC, CRC-16/ISO-IEC-14443-3-B, CRC-16/X-25, CRC-B, X-25

crccheck.crc.**Crc16IsoHdlc**

alias of *Crc16IbmSdlc*

crccheck.crc.**Crc16IsoIec144433B**

alias of *Crc16IbmSdlc*

crccheck.crc.**Crc16X25**

alias of *Crc16IbmSdlc*

crccheck.crc.**CrcB**

alias of *Crc16IbmSdlc*

crccheck.crc.**CrcX25**
alias of *Crc16IbmSdlc*

class crccheck.crc.**Crc16IsoIec144433A**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/ISO-IEC-14443-3-A

Aliases: CRC-A

crccheck.crc.**CrcA**
alias of *Crc16IsoIec144433A*

class crccheck.crc.**Crc16Kermit**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/KERMIT

Aliases: CRC-16/CCITT, CRC-16/CCITT-TRUE, CRC-16/V-41-LSB, CRC-CCITT, KERMIT

crccheck.crc.**Crc16Ccitt**
alias of *Crc16Kermit*

crccheck.crc.**Crc16CcittTrue**
alias of *Crc16Kermit*

crccheck.crc.**Crc16V41Lsb**
alias of *Crc16Kermit*

crccheck.crc.**CrcCcitt**
alias of *Crc16Kermit*

crccheck.crc.**CrcKermit**
alias of *Crc16Kermit*

class crccheck.crc.**Crc16Lj1200**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/LJ1200

class crccheck.crc.**Crc16M17**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/M17

class crccheck.crc.**Crc16MaximDow**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/MAXIM-DOW

Aliases: CRC-16/MAXIM

crccheck.crc.**Crc16Maxim**
alias of *Crc16MaximDow*

class crccheck.crc.**Crc16Mcrf4Xx**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/MCRF4XX

crccheck.crc.**Crc16Mcrf4XX**
alias of *Crc16Mcrf4Xx*

crccheck.crc.**Crcc16Mcrf4xx**
alias of *Crc16Mcrf4Xx*

class crccheck.crc.**Crc16Modbus**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/MODBUS

Aliases: MODBUS

crccheck.crc.**CrcModbus**
alias of *Crc16Modbus*

class crccheck.crc.**Crc16Nrsc5**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/NRSC-5

class crccheck.crc.**Crc16OpensafetyA**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/OPENSAFETY-A

class crccheck.crc.**Crc16OpensafetyB**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/OPENSAFETY-B

class crccheck.crc.**Crc16Profibus**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/PROFIBUS

Aliases: CRC-16/IEC-61158-2

crccheck.crc.**Crc16Iec611582**
alias of *Crc16Profibus*

class crccheck.crc.**Crc16Riello**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/RIELLO

class crccheck.crc.**Crc16SpiFujitsu**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/SPI-FUJITSU

Aliases: CRC-16/AUG-CCITT

crccheck.crc.**Crc16AugCcitt**
alias of *Crc16SpiFujitsu*

class crccheck.crc.**Crc16T10Dif**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/T10-DIF

class crccheck.crc.**Crc16Teledisk**(*initvalue=None*)
Bases: *Crc16Base*

CRC-16/TELEDISK

class crccheck.crc.**Crc16Tms37157**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/TMS37157

class crccheck.crc.**Crc16Umts**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/UMTS

Aliases: CRC-16/BUYPASS, CRC-16/VERIFONE

crccheck.crc.**Crc16Buypass**

alias of *Crc16Umts*

crccheck.crc.**Crc16Verifone**

alias of *Crc16Umts*

class crccheck.crc.**Crc16Usb**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/USB

class crccheck.crc.**Crc16Xmodem**(*initvalue=None*)

Bases: *Crc16Base*

CRC-16/XMODEM

Aliases: CRC-16/ACORN, CRC-16/LTE, CRC-16/V-41-MSB, XMODEM, ZMODEM

crccheck.crc.**Crc16Acorn**

alias of *Crc16Xmodem*

crccheck.crc.**Crc16Lte**

alias of *Crc16Xmodem*

crccheck.crc.**Crc16V41Msb**

alias of *Crc16Xmodem*

crccheck.crc.**CrcXmodem**

alias of *Crc16Xmodem*

crccheck.crc.**CrcZmodem**

alias of *Crc16Xmodem*

crccheck.crc.**Crc16**

alias of *Crc16Xmodem*

class crccheck.crc.**Crc17CanFd**(*initvalue=None*)

Bases: *CrcBase*

CRC-17/CAN-FD

class crccheck.crc.**Crc21CanFd**(*initvalue=None*)

Bases: *CrcBase*

CRC-21/CAN-FD

class crccheck.crc.**Crc24Ble**(*initvalue=None*)

Bases: *CrcBase*

CRC-24/BLE

class crccheck.crc.**Crc24FlexrayA**(*initvalue=None*)

Bases: *CrcBase*

CRC-24/FLEXRAY-A

class crccheck.crc.**Crc24FlexrayB**(*initvalue=None*)

Bases: *CrcBase*

CRC-24/FLEXRAY-B

class crccheck.crc.**Crc24Interlaken**(*initvalue=None*)

Bases: *CrcBase*

CRC-24/INTERLAKEN

class crccheck.crc.**Crc24LteA**(*initvalue=None*)

Bases: *CrcBase*

CRC-24/LTE-A

class crccheck.crc.**Crc24LteB**(*initvalue=None*)

Bases: *CrcBase*

CRC-24/LTE-B

class crccheck.crc.**Crc24Openpgp**(*initvalue=None*)

Bases: *CrcBase*

CRC-24/OPENPGP

Aliases: CRC-24

crccheck.crc.**Crc24**

alias of *Crc24Openpgp*

crccheck.crc.**Crc24OpenPgp**

alias of *Crc24Openpgp*

class crccheck.crc.**Crc24Os9**(*initvalue=None*)

Bases: *CrcBase*

CRC-24/OS-9

class crccheck.crc.**Crc30Cdma**(*initvalue=None*)

Bases: *CrcBase*

CRC-30/CDMA

class crccheck.crc.**Crc31Philips**(*initvalue=None*)

Bases: *CrcBase*

CRC-31/PHILIPS

class crccheck.crc.**Crc32Aixm**(*initvalue=None*)

Bases: *Crc32Base*

CRC-32/AIXM

Aliases: CRC-32Q

crccheck.crc.**Crc32Q**

alias of *Crc32Aixm*

crccheck.crc.**Crc32q**
alias of *Crc32Aixm*

class crccheck.crc.**Crc32Autosar**(*initvalue=None*)
Bases: *Crc32Base*

CRC-32/AUTOSAR

class crccheck.crc.**Crc32Base91D**(*initvalue=None*)
Bases: *Crc32Base*

CRC-32/BASE91-D

Aliases: CRC-32D

crccheck.crc.**Crc32D**
alias of *Crc32Base91D*

crccheck.crc.**Crc32d**
alias of *Crc32Base91D*

class crccheck.crc.**Crc32Bzip2**(*initvalue=None*)
Bases: *Crc32Base*

CRC-32/BZIP2

Aliases: CRC-32/AAL5, CRC-32/DECT-B, B-CRC-32

crccheck.crc.**Crc32Aal5**
alias of *Crc32Bzip2*

crccheck.crc.**Crc32DectB**
alias of *Crc32Bzip2*

crccheck.crc.**Crc32B**
alias of *Crc32Bzip2*

class crccheck.crc.**Crc32CdRomEdc**(*initvalue=None*)
Bases: *Crc32Base*

CRC-32/CD-ROM-EDC

class crccheck.crc.**Crc32Cksum**(*initvalue=None*)
Bases: *Crc32Base*

CRC-32/CKSUM

Aliases: CKSUM, CRC-32/POSIX

crccheck.crc.**CrcCksum**
alias of *Crc32Cksum*

crccheck.crc.**Crc32Posix**
alias of *Crc32Cksum*

class crccheck.crc.**Crc32Iscsi**(*initvalue=None*)
Bases: *Crc32Base*

CRC-32/ISCSI

Aliases: CRC-32/BASE91-C, CRC-32/CASTAGNOLI, CRC-32/INTERLAKEN, CRC-32C

crccheck.crc.**Crc32Base91C**
alias of *Crc32Iscsi*

crccheck.crc.**Crc32Castagnoli**
alias of *Crc32Iscsi*

crccheck.crc.**Crc32Interlaken**
alias of *Crc32Iscsi*

crccheck.crc.**Crc32C**
alias of *Crc32Iscsi*

crccheck.crc.**Crc32c**
alias of *Crc32Iscsi*

class crccheck.crc.**Crc32IsoHdlc**(*initvalue=None*)
Bases: *Crc32Base*

CRC-32/ISO-HDLC

Aliases: CRC-32, CRC-32/ADCCP, CRC-32/V-42, CRC-32/XZ, PKZIP

crccheck.crc.**Crc32**
alias of *Crc32IsoHdlc*

crccheck.crc.**Crc32Adccp**
alias of *Crc32IsoHdlc*

crccheck.crc.**Crc32V42**
alias of *Crc32IsoHdlc*

crccheck.crc.**Crc32Xz**
alias of *Crc32IsoHdlc*

crccheck.crc.**CrcPkzip**
alias of *Crc32IsoHdlc*

class crccheck.crc.**Crc32Jamcrc**(*initvalue=None*)
Bases: *Crc32Base*

CRC-32/JAMCRC

Aliases: JAMCRC

crccheck.crc.**CrcJamcrc**
alias of *Crc32Jamcrc*

class crccheck.crc.**Crc32Mef**(*initvalue=None*)
Bases: *Crc32Base*

CRC-32/MEF

class crccheck.crc.**Crc32Mpeg2**(*initvalue=None*)
Bases: *Crc32Base*

CRC-32/MPEG-2

class crccheck.crc.**Crc32Xfer**(*initvalue=None*)

Bases: *Crc32Base*

CRC-32/XFER

Aliases: XFER

crccheck.crc.**CrcXfer**

alias of *Crc32Xfer*

class crccheck.crc.**Crc40Gsm**(*initvalue=None*)

Bases: *CrcBase*

CRC-40/GSM

class crccheck.crc.**Crc64Ecma182**(*initvalue=None*)

Bases: *CrcBase*

CRC-64/ECMA-182

Aliases: CRC-64

crccheck.crc.**Crc64**

alias of *Crc64Ecma182*

class crccheck.crc.**Crc64GoIso**(*initvalue=None*)

Bases: *CrcBase*

CRC-64/GO-ISO

class crccheck.crc.**Crc64Ms**(*initvalue=None*)

Bases: *CrcBase*

CRC-64/MS

class crccheck.crc.**Crc64Redis**(*initvalue=None*)

Bases: *CrcBase*

CRC-64/REDIS

class crccheck.crc.**Crc64We**(*initvalue=None*)

Bases: *CrcBase*

CRC-64/WE

class crccheck.crc.**Crc64Xz**(*initvalue=None*)

Bases: *CrcBase*

CRC-64/XZ

Aliases: CRC-64/GO-ECMA

crccheck.crc.**Crc64GoEcma**

alias of *Crc64Xz*

class crccheck.crc.**Crc82Darc**(*initvalue=None*)

Bases: *CrcBase*

CRC-82/DARC

2.5 Module contents

2.5.1 Classes to calculate CRCs and checksums from binary data

The *crccheck.crc* module implements all CRCs listed in the Catalogue of parametrised CRC algorithms:

CRC-3/GSM, CRC-3/ROHC, CRC-4/G-704, CRC-4/ITU, CRC-4/INTERLAKEN, CRC-5/EPC-C1G2, CRC-5/EPC, CRC-5/G-704, CRC-5/ITU, CRC-5/USB, CRC-6/CDMA2000-A, CRC-6/CDMA2000-B, CRC-6/DARC, CRC-6/G-704, CRC-6/ITU, CRC-6/GSM, CRC-7/MMC, CRC-7, CRC-7/ROHC, CRC-7/UMTS, CRC-8/AUTOSAR, CRC-8/BLUETOOTH, CRC-8/CDMA2000, CRC-8/DARC, CRC-8/DVB-S2, CRC-8/GSM-A, CRC-8/GSM-B, CRC-8/I-432-1, CRC-8/ITU, CRC-8/I-CODE, CRC-8/LTE, CRC-8/MAXIM-DOW, CRC-8/MAXIM, DOW-CRC, CRC-8/MIFARE-MAD, CRC-8/NRSC-5, CRC-8/OPENSAFETY, CRC-8/ROHC, CRC-8/SAE-J1850, CRC-8/SMBUS, CRC-8, CRC-8/TECH-3250, CRC-8/AES, CRC-8/EBU, CRC-8/WCDMA, CRC-10/ATM, CRC-10, CRC-10/I-610, CRC-10/CDMA2000, CRC-10/GSM, CRC-11/FLEXRAY, CRC-11, CRC-11/UMTS, CRC-12/CDMA2000, CRC-12/DECT, CRC-12-X, CRC-12/GSM, CRC-12/UMTS, CRC-12/3GPP, CRC-13/BBC, CRC-14/DARC, CRC-14/GSM, CRC-15/CAN, CRC-15, CRC-15/MPT1327, CRC-16/ARC, ARC, CRC-16/LHA, CRC-IBM, CRC-16/CDMA2000, CRC-16/CMS, CRC-16/DDS-110, CRC-16/DECT-R, R-CRC-16, CRC-16/DECT-X, X-CRC-16, CRC-16/DNP, CRC-16/EN-13757, CRC-16/GENIBUS, CRC-16/DARC, CRC-16/EPC, CRC-16/EPC-C1G2, CRC-16/I-CODE, CRC-16/GSM, CRC-16/IBM-3740, CRC-16/AUTOSAR, CRC-16/CCITT-FALSE, CRC-16/IBM-SDLC, CRC-16/ISO-HDLC, CRC-16/ISO-IEC-14443-3-B, CRC-16/X-25, CRC-B, X-25, CRC-16/ISO-IEC-14443-3-A, CRC-A, CRC-16/KERMIT, CRC-16/CCITT, CRC-16/CCITT-TRUE, CRC-16/V-41-LSB, CRC-CCITT, KERMIT, CRC-16/LJ1200, CRC-16/MAXIM-DOW, CRC-16/MAXIM, CRC-16/MCRF4XX, CRC-16/MODBUS, MODBUS, CRC-16/NRSC-5, CRC-16/OPENSAFETY-A, CRC-16/OPENSAFETY-B, CRC-16/PROFIBUS, CRC-16/IEC-61158-2, CRC-16/RIELLO, CRC-16/SPI-FUJITSU, CRC-16/AUG-CCITT, CRC-16/T10-DIF, CRC-16/TELEDISK, CRC-16/TMS37157, CRC-16/UMTS, CRC-16/BUYPASS, CRC-16/VERIFONE, CRC-16/USB, CRC-16/XMODEM, CRC-16/ACORN, CRC-16/LTE, CRC-16/V-41-MSB, XMODEM, ZMODEM, CRC-17/CAN-FD, CRC-21/CAN-FD, CRC-24/BLE, CRC-24/FLEXRAY-A, CRC-24/FLEXRAY-B, CRC-24/INTERLAKEN, CRC-24/LTE-A, CRC-24/LTE-B, CRC-24/OPENPGP, CRC-24, CRC-24/OS-9, CRC-30/CDMA, CRC-31/PHILIPS, CRC-32/AIXM, CRC-32Q, CRC-32/AUTOSAR, CRC-32/BASE91-D, CRC-32D, CRC-32/BZIP2, CRC-32/AAL5, CRC-32/DECT-B, B-CRC-32, CRC-32/CD-ROM-EDC, CRC-32/CKSUM, CKSUM, CRC-32/POSIX, CRC-32/ISCSI, CRC-32/BASE91-C, CRC-32/CASTAGNOLI, CRC-32/INTERLAKEN, CRC-32C, CRC-32/ISO-HDLC, CRC-32, CRC-32/ADCCP, CRC-32/V-42, CRC-32/XZ, PKZIP, CRC-32/JAMCRC, JAMCRC, CRC-32/MPEG-2, CRC-32/XFER, XFER, CRC-40/GSM, CRC-64/ECMA-182, CRC-64, CRC-64/GO-ISO, CRC-64/WE, CRC-64/XZ, CRC-64/GO-ECMA, CRC-82/DARC.

For the class names simply remove all dashes and slashes from the above names and apply CamelCase, e.g. "CRC-32/MPEG-2" is implemented by *Crc32Mpeg2*. Other CRC can be calculated by using the general class *crccheck.crc.Crc* by providing all required CRC parameters.

The *crccheck.checksum* module implements additive and XOR checksums with 8, 16 and 32 bit: *Checksum8*, *Checksum16*, *Checksum32* and *ChecksumXor8*, *ChecksumXor16*, *ChecksumXor32*.

Usage example:

```
from crccheck.crc import Crc32, CrcXmodem
from crccheck.checksum import Checksum32

# Quick calculation
data = bytearray.fromhex("DEADBEEF")
crc = Crc32.calc(data)
checksum = Checksum32.calc(data)

# Procsss multiple data buffers
data1 = b"Binary string"  # or use .encode(..) on normal sring - Python 3 only
```

(continues on next page)

(continued from previous page)

```
data2 = bytes.fromhex("1234567890")  # Python 3 only, use bytearray for older versions
data3 = (0x0, 255, 12, 99)  # Iterable which returns ints in byte range (0..255)
crcinst = CrcXmodem()
crcinst.process(data1)
crcinst.process(data2)
crcinst.process(data3[1:-1])
crcbytes = crcinst.finalbytes()
crchex = crcinst.finalhex()
crcint = crcinst.final()
```

License:

CHAPTER

THREE

SUPPORTED CRCS

CRC Name	Class	Bit width	Poly	Initvalue	Refle
CRC-3/GSM	Crc3Gsm	3	0x3	0x0	False
CRC-3/ROHC	Crc3Rohc	3	0x3	0x7	True
CRC-4/G-704	Crc4G704	4	0x3	0x0	True
CRC-4/INTERLAKEN	Crc4Interlaken	4	0x3	0xF	False
CRC-5/EPC-C1G2	Crc5EpcC1G2	5	0x9	0x9	False
CRC-5/G-704	Crc5G704	5	0x15	0x0	True
CRC-5/USB	Crc5Usb	5	0x5	0x1F	True
CRC-6/CDMA2000-A	Crc6Cdma2000A	6	0x27	0x3F	False
CRC-6/CDMA2000-B	Crc6Cdma2000B	6	0x7	0x3F	False
CRC-6/DARC	Crc6Darc	6	0x19	0x0	True
CRC-6/G-704	Crc6G704	6	0x3	0x0	True
CRC-6/GSM	Crc6Gsm	6	0x2F	0x0	False
CRC-7/MMC	Crc7Mmc	7	0x9	0x0	False
CRC-7/ROHC	Crc7Rohc	7	0x4F	0x7F	True
CRC-7/UMTS	Crc7Umts	7	0x45	0x0	False
CRC-8/AUTOSAR	Crc8Autosar	8	0x2F	0xFF	False
CRC-8/BLUETOOTH	Crc8Bluetooth	8	0xA7	0x0	True
CRC-8/CDMA2000	Crc8Cdma2000	8	0x9B	0xFF	False
CRC-8/DARC	Crc8Darc	8	0x39	0x0	True
CRC-8/DVB-S2	Crc8DvbS2	8	0xD5	0x0	False
CRC-8/GSM-A	Crc8GsmA	8	0x1D	0x0	False
CRC-8/GSM-B	Crc8GsmB	8	0x49	0x0	False
CRC-8/HITAG	Crc8Hitag	8	0x1D	0xFF	False
CRC-8/I-432-1	Crc8I4321	8	0x7	0x0	False
CRC-8/I-CODE	Crc8ICode	8	0x1D	0xFD	False
CRC-8/LTE	Crc8Lte	8	0x9B	0x0	False
CRC-8/MAXIM-DOW	Crc8MaximDow	8	0x31	0x0	True
CRC-8/MIFARE-MAD	Crc8MifareMad	8	0x1D	0xC7	False
CRC-8/NRSC-5	Crc8Nrsc5	8	0x31	0xFF	False
CRC-8/OPENSAFETY	Crc8Opensafety	8	0x2F	0x0	False
CRC-8/ROHC	Crc8Rohc	8	0x7	0xFF	True
CRC-8/SAE-J1850	Crc8SaeJ1850	8	0x1D	0xFF	False
CRC-8/SMBUS	Crc8Smbus	8	0x7	0x0	False
CRC-8/TECH-3250	Crc8Tech3250	8	0x1D	0xFF	True
CRC-8/WCDMA	Crc8Wcdma	8	0x9B	0x0	True
CRC-10/ATM	Crc10Atm	10	0x233	0x0	False
CRC-10/CDMA2000	Crc10Cdma2000	10	0x3D9	0x3FF	False

Table 1 – continued from pr

CRC Name	Class	Bit width	Poly	Initvalue	Refle
CRC-10/GSM	Crc10Gsm	10	0x175	0x0	False
CRC-11/FLEXRAY	Crc11Flexray	11	0x385	0x1A	False
CRC-11/UMTS	Crc11Umts	11	0x307	0x0	False
CRC-12/CDMA2000	Crc12Cdma2000	12	0xF13	0xFFF	False
CRC-12/DECT	Crc12Dect	12	0x80F	0x0	False
CRC-12/GSM	Crc12Gsm	12	0xD31	0x0	False
CRC-12/UMTS	Crc12Umts	12	0x80F	0x0	False
CRC-13/BBC	Crc13Bbc	13	0x1CF5	0x0	False
CRC-14/DARC	Crc14Darc	14	0x805	0x0	True
CRC-14/GSM	Crc14Gsm	14	0x202D	0x0	False
CRC-15/CAN	Crc15Can	15	0x4599	0x0	False
CRC-15/MPT1327	Crc15Mpt1327	15	0x6815	0x0	False
CRC-16/ARC	Crc16Arc	16	0x8005	0x0	True
CRC-16/CDMA2000	Crc16Cdma2000	16	0xC867	0xFFFF	False
CRC-16/CMS	Crc16Cms	16	0x8005	0xFFFF	False
CRC-16/DDS-110	Crc16Dds110	16	0x8005	0x800D	False
CRC-16/DECT-R	Crc16DectR	16	0x589	0x0	False
CRC-16/DECT-X	Crc16DectX	16	0x589	0x0	False
CRC-16/DNP	Crc16Dnp	16	0x3D65	0x0	True
CRC-16/EN-13757	Crc16En13757	16	0x3D65	0x0	False
CRC-16/GENIBUS	Crc16Genibus	16	0x1021	0xFFFF	False
CRC-16/GSM	Crc16Gsm	16	0x1021	0x0	False
CRC-16/IBM-3740	Crc16Ibm3740	16	0x1021	0xFFFF	False
CRC-16/IBM-SDLC	Crc16IbmSdlc	16	0x1021	0xFFFF	True
CRC-16/ISO-IEC-14443-3-A	Crc16IsoIec144433A	16	0x1021	0xC6C6	True
CRC-16/KERMIT	Crc16Kermit	16	0x1021	0x0	True
CRC-16/LJ1200	Crc16Lj1200	16	0x6F63	0x0	False
CRC-16/M17	Crc16M17	16	0x5935	0xFFFF	False
CRC-16/MAXIM-DOW	Crc16MaximDow	16	0x8005	0x0	True
CRC-16/MCRF4XX	Crc16Mcrf4Xx	16	0x1021	0xFFFF	True
CRC-16/MODBUS	Crc16Modbus	16	0x8005	0xFFFF	True
CRC-16/NRSC-5	Crc16Nrsc5	16	0x80B	0xFFFF	True
CRC-16/OPENSAFETY-A	Crc16OpensafetyA	16	0x5935	0x0	False
CRC-16/OPENSAFETY-B	Crc16OpensafetyB	16	0x755B	0x0	False
CRC-16/PROFIBUS	Crc16Profibus	16	0x1DCF	0xFFFF	False
CRC-16/RIELLO	Crc16Riello	16	0x1021	0xB2AA	True
CRC-16/SPI-FUJITSU	Crc16SpiFujitsu	16	0x1021	0x1D0F	False
CRC-16/T10-DIF	Crc16T10Dif	16	0x8BB7	0x0	False
CRC-16/TELEDISK	Crc16Teledisk	16	0xA097	0x0	False
CRC-16/TMS37157	Crc16Tms37157	16	0x1021	0x89EC	True
CRC-16/UMTS	Crc16Umts	16	0x8005	0x0	False
CRC-16/USB	Crc16Usb	16	0x8005	0xFFFF	True
CRC-16/XMODEM	Crc16Xmodem	16	0x1021	0x0	False
CRC-17/CAN-FD	Crc17CanFd	17	0x1685B	0x0	False
CRC-21/CAN-FD	Crc21CanFd	21	0x102899	0x0	False
CRC-24/BLE	Crc24Ble	24	0x65B	0x555555	True
CRC-24/FLEXRAY-A	Crc24FlexrayA	24	0x5D6DCB	0xFEDCBA	False
CRC-24/FLEXRAY-B	Crc24FlexrayB	24	0x5D6DCB	0xABCDEF	False
CRC-24/INTERLAKEN	Crc24Interlaken	24	0x328B63	0xFFFFFF	False

Table 1 – continued from pr

CRC Name	Class	Bit width	Poly	Initvalue	Refle
CRC-24/LTE-A	Crc24LteA	24	0x864CFB	0x0	False
CRC-24/LTE-B	Crc24LteB	24	0x800063	0x0	False
CRC-24/OPENPGP	Crc24Openpgp	24	0x864CFB	0xB704CE	False
CRC-24/OS-9	Crc24Os9	24	0x800063	0xFFFFFF	False
CRC-30/CDMA	Crc30Cdma	30	0x2030B9C7	0x3FFFFFFF	False
CRC-31/PHILIPS	Crc31Philips	31	0x4C11DB7	0x7FFFFFFF	False
CRC-32/AIXM	Crc32Aixm	32	0x814141AB	0x0	False
CRC-32/AUTOSAR	Crc32Autosar	32	0xF4ACFB13	0xFFFFFFFF	True
CRC-32/BASE91-D	Crc32Base91D	32	0xA833982B	0xFFFFFFFF	True
CRC-32/BZIP2	Crc32Bzip2	32	0x4C11DB7	0xFFFFFFFF	False
CRC-32/CD-ROM-EDC	Crc32CdRomEdc	32	0x8001801B	0x0	True
CRC-32/CKSUM	Crc32Cksum	32	0x4C11DB7	0x0	False
CRC-32/ISCSI	Crc32Iscsi	32	0x1EDC6F41	0xFFFFFFFF	True
CRC-32/ISO-HDLC	Crc32IsoHdlc	32	0x4C11DB7	0xFFFFFFFF	True
CRC-32/JAMCRC	Crc32Jamcrc	32	0x4C11DB7	0xFFFFFFFF	True
CRC-32/MEF	Crc32Mef	32	0x741B8CD7	0xFFFFFFFF	True
CRC-32/MPEG-2	Crc32Mpeg2	32	0x4C11DB7	0xFFFFFFFF	False
CRC-32/XFER	Crc32Xfer	32	0xAF	0x0	False
CRC-40/GSM	Crc40Gsm	40	0x4820009	0x0	False
CRC-64/ECMA-182	Crc64Ecma182	64	0x42F0E1EBA9EA3693	0x0	False
CRC-64/GO-ISO	Crc64GoIso	64	0x1B	0xFFFFFFFFFFFFFFFF	True
CRC-64/MS	Crc64Ms	64	0x259C84CBA6426349	0xFFFFFFFFFFFFFFFF	True
CRC-64/REDIS	Crc64Redis	64	0xAD93D23594C935A9	0x0	True
CRC-64/WE	Crc64We	64	0x42F0E1EBA9EA3693	0xFFFFFFFFFFFFFFFF	False
CRC-64/XZ	Crc64Xz	64	0x42F0E1EBA9EA3693	0xFFFFFFFFFFFFFFFF	True
CRC-82/DARC	Crc82Darc	82	0x308C0111011401440411	0x0	True

3.1 Aliases

As some CRCs are also known under different names aliases for the CRC classes are defined.

CRC	Class	Alias
CRC-4/G-704	Crc4G704	CRC-4/ITU
CRC-5/EPC-C1G2	Crc5EpcC1G2	CRC-5/EPC
CRC-5/G-704	Crc5G704	CRC-5/ITU
CRC-6/G-704	Crc6G704	CRC-6/ITU
CRC-7/MMC	Crc7Mmc	CRC-7
CRC-8/I-432-1	Crc8I4321	CRC-8/ITU
CRC-8/MAXIM-DOW	Crc8MaximDow	CRC-8/MAXIM, DOW-CRC
CRC-8/SMBUS	Crc8Smbus	CRC-8
CRC-8/TECH-3250	Crc8Tech3250	CRC-8/AES, CRC-8/EBU
CRC-10/ATM	Crc10Atm	CRC-10, CRC-10/I-610
CRC-11/FLEXRAY	Crc11Flexray	CRC-11
CRC-12/DECT	Crc12Dect	CRC-12-X
CRC-12/UMTS	Crc12Umts	CRC-12/3GPP
CRC-15/CAN	Crc15Can	CRC-15
CRC-16/ARC	Crc16Arc	ARC, CRC-16/LHA, CRC-IBM

Table 2 – continued from previous page

CRC	Class	Alias
CRC-16/DECT-R	Crc16DectR	R-CRC-16
CRC-16/DECT-X	Crc16DectX	X-CRC-16
CRC-16/GENIBUS	Crc16Genibus	CRC-16/DARC, CRC-16/EPC, CRC-16/EPC-C1G2, CRC-16/I-CODE
CRC-16/IBM-3740	Crc16Ibm3740	CRC-16/AUTOSAR, CRC-16/CCITT-FALSE
CRC-16/IBM-SDLC	Crc16IbmSdlc	CRC-16/ISO-HDLC, CRC-16/ISO-IEC-14443-3-B, CRC-16/X-25, CRC-B,
CRC-16/ISO-IEC-14443-3-A	Crc16IsoIec144433A	CRC-A
CRC-16/KERMIT	Crc16Kermit	CRC-16/CCITT, CRC-16/CCITT-TRUE, CRC-16/V-41-LSB, CRC-CCITT,
CRC-16/MAXIM-DOW	Crc16MaximDow	CRC-16/MAXIM
CRC-16/MODBUS	Crc16Modbus	MODBUS
CRC-16/PROFIBUS	Crc16Profibus	CRC-16/IEC-61158-2
CRC-16/SPI-FUJITSU	Crc16SpiFujitsu	CRC-16/AUG-CCITT
CRC-16/UMTS	Crc16Umts	CRC-16/BUYPASS, CRC-16/VERIFONE
CRC-16/XMODEM	Crc16Xmodem	CRC-16/ACORN, CRC-16/LTE, CRC-16/V-41-MSB, XMODEM, ZMODE
CRC-24/OPENPGP	Crc24Openpgp	CRC-24
CRC-32/AIXM	Crc32Aixm	CRC-32Q
CRC-32/BASE91-D	Crc32Base91D	CRC-32D
CRC-32/BZIP2	Crc32Bzip2	CRC-32/AAL5, CRC-32/DECT-B, B-CRC-32
CRC-32/CKSUM	Crc32Cksum	CKSUM, CRC-32/POSIX
CRC-32/ISCSI	Crc32Iscsi	CRC-32/BASE91-C, CRC-32/CASTAGNOLI, CRC-32/INTERLAKEN, CR
CRC-32/ISO-HDLC	Crc32IsoHdlc	CRC-32, CRC-32/ADCCP, CRC-32/V-42, CRC-32/XZ, PKZIP
CRC-32/JAMCRC	Crc32Jamcrc	JAMCRC
CRC-32/XFER	Crc32Xfer	XFER
CRC-64/ECMA-182	Crc64Ecma182	CRC-64
CRC-64/XZ	Crc64Xz	CRC-64/GO-ECMA

CHAPTER

FOUR

HOW-TO

4.1 How to quickly calculate a CRC/checksum

If only one data buffer needs to be processed a CRC/checksum can be generated quickly using the class method *calc()*:

```
from crccheck.crc import Crc32
crc = Crc32.calc(data)
```

If the result is needed as hexstring or bytes use *calchex()* or *calcbytes()*, respectively.

4.2 How to calculate over multiple data blocks

Create an instance and feed all data blocks to *process()*. Once done, use *final()* to get the final result:

```
from crccheck.crc import Crc32
crcinst = Crc32()
crcinst.process(data1)
crcinst.process(data2)
crcinst.process(data3)
crc = crcinst.final()
```

The intermediate value can be read using *value()* and, if required, set again using `init()`.

4.3 How to use a CRC not implemented by the package

The package implements all CRCs listed in the Catalogue of parametrised CRC algorithms.

The general class *crccheck.crc.Crc* can be used for any other CRCs. You need to provide the CRC parameters. These are described in detail in the publication A paninless guide to CRC error detection alogithms.

For advanced users is also possible to create an own subclass. See the source code for details.

4.4 How to calculate the CRC or checksum of a file

You need to provide an interable over all bytes in the file. For this mmap is recommended:

```
from mmap import ACCESS_READ, mmap
from crccheck.crc import Crc32

with open("somefile.ext", 'rb') as fh, mmap(fh.fileno(), 0, access=ACCESS_READ) as mm:
    crc = Crc32.calc((b[0] for b in mm))
```

CHAPTER

FIVE

CHANGELOG

5.1 v1.3.0 - 2022-10-13

- Updated unit tests to use correct asserts.
- Some code quality adjustments.
- Switched license from GPLv3 to MIT.

5.2 v1.2.0 - 2022-10-06

- Updated with newest CRC from catalogue.
- Changed project repository from Mercurial to Git.

5.3 v1.1.1 - 2022-10-06

- Switched unit tests from nose to unittest module.

5.4 v1.1 - 2021-11-25

- Fixed ignored byteorder in calchex().
- Updated documentation.

5.5 v1.0 - 2020-09-21

- Added further CRCs.
- Fixed missing storage of initial value for general `Crc` class. Before `reset()` did not work correctly.
- Updated tests to achieve 100% code coverage.

5.6 v0.6 - 2016-04-03

- Added compatibility with Python 2.7 and 3.3.

5.7 v0.5 - 2016-03-30

- Added general checksum classes Checksum and ChecksumXor.
- changed `process()` to return `self` so that calls can be chained.
- changed `init()` to return `self` so that calls can be chained.
- renamed `init()` to `reset()`.
- Updated documentation.

5.8 v0.4 - 2016-03-29

- Removed arguments startindex and endindex as they are not required.
- Optimized reflectbitorder().
- base: Added `byteorder` argument to `calchex()`.
- Removed outdated code.
- Added more unit tests.

5.9 v0.3 - 2016-03-28

- Renamed package to `crccheck` as old name was taken in PIP.
- Changed `bigendian=True/False` arguments to `byteorder='big'/'little'`.
- Added more docstring documentation.
- Removed outdated code from repository.

5.10 v0.2 - 2016-03-27

- Changes to support Python 3.
- Code reformatting.
- Some smaller fixes.
- Runtime optimisations.

5.11 v0.1 - 2015-09-23

- First version.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

crccheck, ??
crccheck.base, 3
crccheck.checksum, 6
crccheck.crc, 8

INDEX

C

calc() (*crccheck.base.CrccheckBase class method*), 5
calc() (*crccheck.crc.Crc method*), 11
calcbytes() (*crccheck.base.CrccheckBase class method*), 5
calcbytes() (*crccheck.crc.Crc method*), 11
calchex() (*crccheck.base.CrccheckBase class method*), 5
calchex() (*crccheck.crc.Crc method*), 11
Checksum (*class in crccheck.checksum*), 8
Checksum16 (*class in crccheck.checksum*), 7
Checksum32 (*class in crccheck.checksum*), 7
Checksum8 (*class in crccheck.checksum*), 7
ChecksumBase (*class in crccheck.checksum*), 6
ChecksumXor (*class in crccheck.checksum*), 8
ChecksumXor16 (*class in crccheck.checksum*), 7
ChecksumXor32 (*class in crccheck.checksum*), 7
ChecksumXor8 (*class in crccheck.checksum*), 8
ChecksumXorBase (*class in crccheck.checksum*), 7
Crc (*class in crccheck.crc*), 10
Crc10 (*in module crccheck.crc*), 16
Crc10Atm (*class in crccheck.crc*), 16
Crc10Cdma2000 (*class in crccheck.crc*), 16
Crc10Gsm (*class in crccheck.crc*), 16
Crc10I610 (*in module crccheck.crc*), 16
Crc11 (*in module crccheck.crc*), 16
Crc11Flexray (*class in crccheck.crc*), 16
Crc11Umts (*class in crccheck.crc*), 16
Crc123Gpp (*in module crccheck.crc*), 17
Crc12Cdma2000 (*class in crccheck.crc*), 17
Crc12Dect (*class in crccheck.crc*), 17
Crc12Gsm (*class in crccheck.crc*), 17
Crc12Umts (*class in crccheck.crc*), 17
Crc12X (*in module crccheck.crc*), 17
Crc13Bbc (*class in crccheck.crc*), 17
Crc14Darc (*class in crccheck.crc*), 17
Crc14Gsm (*class in crccheck.crc*), 17
Crc15 (*in module crccheck.crc*), 17
Crc15Can (*class in crccheck.crc*), 17
Crc15Mpt1327 (*class in crccheck.crc*), 17
Crc16 (*in module crccheck.crc*), 22
Crc16Acorn (*in module crccheck.crc*), 22
Crc16Arc (*class in crccheck.crc*), 18
Crc16AugCcitt (*in module crccheck.crc*), 21
Crc16Autosar (*in module crccheck.crc*), 19
Crc16Base (*class in crccheck.crc*), 12
Crc16Buypass (*in module crccheck.crc*), 22
Crc16Ccitt (*in module crccheck.crc*), 20
Crc16CcittFalse (*in module crccheck.crc*), 19
Crc16CcittTrue (*in module crccheck.crc*), 20
Crc16Cdma2000 (*class in crccheck.crc*), 18
Crc16Cms (*class in crccheck.crc*), 18
Crc16Darc (*in module crccheck.crc*), 19
Crc16Dds110 (*class in crccheck.crc*), 18
Crc16DectR (*class in crccheck.crc*), 18
Crc16DectX (*class in crccheck.crc*), 18
Crc16Dnp (*class in crccheck.crc*), 18
Crc16En13757 (*class in crccheck.crc*), 18
Crc16Epc (*in module crccheck.crc*), 19
Crc16EpcC1G2 (*in module crccheck.crc*), 19
Crc16Genibus (*class in crccheck.crc*), 19
Crc16Gsm (*class in crccheck.crc*), 19
Crc16Ibm3740 (*class in crccheck.crc*), 19
Crc16IbmSdlc (*class in crccheck.crc*), 19
Crc16ICode (*in module crccheck.crc*), 19
Crc16Iec611582 (*in module crccheck.crc*), 21
Crc16IsoHdlc (*in module crccheck.crc*), 19
Crc16IsoIec144433A (*class in crccheck.crc*), 20
Crc16IsoIec144433B (*in module crccheck.crc*), 19
Crc16Kermit (*class in crccheck.crc*), 20
Crc16Lha (*in module crccheck.crc*), 18
Crc16Lj1200 (*class in crccheck.crc*), 20
Crc16Lte (*in module crccheck.crc*), 22
Crc16M17 (*class in crccheck.crc*), 20
Crc16Maxim (*in module crccheck.crc*), 20
Crc16MaximDow (*class in crccheck.crc*), 20
Crc16Mcrf4Xx (*class in crccheck.crc*), 20
Crc16Mcrf4XX (*in module crccheck.crc*), 20
Crc16Modbus (*class in crccheck.crc*), 21
Crc16Nrsc5 (*class in crccheck.crc*), 21
Crc16OpensafetyA (*class in crccheck.crc*), 21
Crc16OpensafetyB (*class in crccheck.crc*), 21
Crc16Profibus (*class in crccheck.crc*), 21
Crc16R (*in module crccheck.crc*), 18

Crc16Riello (*class in crccheck.crc*), 21
Crc16SpiFujitsu (*class in crccheck.crc*), 21
Crc16T10Dif (*class in crccheck.crc*), 21
Crc16Teledisk (*class in crccheck.crc*), 21
Crc16Tms37157 (*class in crccheck.crc*), 21
Crc16Umts (*class in crccheck.crc*), 22
Crc16Usb (*class in crccheck.crc*), 22
Crc16V41Lsb (*in module crccheck.crc*), 20
Crc16V41Msb (*in module crccheck.crc*), 22
Crc16Verifone (*in module crccheck.crc*), 22
Crc16X (*in module crccheck.crc*), 18
Crc16X25 (*in module crccheck.crc*), 19
Crc16Xmodem (*class in crccheck.crc*), 22
Crc17CanFd (*class in crccheck.crc*), 22
Crc21CanFd (*class in crccheck.crc*), 22
Crc24 (*in module crccheck.crc*), 23
Crc24Ble (*class in crccheck.crc*), 22
Crc24FlexrayA (*class in crccheck.crc*), 22
Crc24FlexrayB (*class in crccheck.crc*), 23
Crc24Interlaken (*class in crccheck.crc*), 23
Crc24LteA (*class in crccheck.crc*), 23
Crc24LteB (*class in crccheck.crc*), 23
Crc24Openpgp (*class in crccheck.crc*), 23
Crc24OpenPgp (*in module crccheck.crc*), 23
Crc24Os9 (*class in crccheck.crc*), 23
Crc30Cdma (*class in crccheck.crc*), 23
Crc31Philips (*class in crccheck.crc*), 23
Crc32 (*in module crccheck.crc*), 25
Crc32Aal5 (*in module crccheck.crc*), 24
Crc32Adccp (*in module crccheck.crc*), 25
Crc32Aixm (*class in crccheck.crc*), 23
Crc32Autosar (*class in crccheck.crc*), 24
Crc32B (*in module crccheck.crc*), 24
Crc32Base (*class in crccheck.crc*), 12
Crc32Base91C (*in module crccheck.crc*), 24
Crc32Base91D (*class in crccheck.crc*), 24
Crc32Bzip2 (*class in crccheck.crc*), 24
Crc32C (*in module crccheck.crc*), 25
Crc32c (*in module crccheck.crc*), 25
Crc32Castagnoli (*in module crccheck.crc*), 25
Crc32CdRomEdc (*class in crccheck.crc*), 24
Crc32Cksum (*class in crccheck.crc*), 24
Crc32D (*in module crccheck.crc*), 24
Crc32d (*in module crccheck.crc*), 24
Crc32DectB (*in module crccheck.crc*), 24
Crc32Interlaken (*in module crccheck.crc*), 25
Crc32Iscsi (*class in crccheck.crc*), 24
Crc32IsoHdlc (*class in crccheck.crc*), 25
Crc32Jamcrc (*class in crccheck.crc*), 25
Crc32Mef (*class in crccheck.crc*), 25
Crc32Mpeg2 (*class in crccheck.crc*), 25
Crc32Posix (*in module crccheck.crc*), 24
Crc32Q (*in module crccheck.crc*), 23
Crc32q (*in module crccheck.crc*), 23
Crc32V42 (*in module crccheck.crc*), 25
Crc32Xfer (*class in crccheck.crc*), 25
Crc32Xz (*in module crccheck.crc*), 25
Crc3Gsm (*class in crccheck.crc*), 12
Crc3Rohc (*class in crccheck.crc*), 12
Crc40Gsm (*class in crccheck.crc*), 26
Crc4G704 (*class in crccheck.crc*), 12
Crc4Interlaken (*class in crccheck.crc*), 13
Crc4Itu (*in module crccheck.crc*), 13
Crc5Epc (*in module crccheck.crc*), 13
Crc5EpcC1G2 (*class in crccheck.crc*), 13
Crc5G704 (*class in crccheck.crc*), 13
Crc5Itu (*in module crccheck.crc*), 13
Crc5Usb (*class in crccheck.crc*), 13
Crc64 (*in module crccheck.crc*), 26
Crc64Ecma182 (*class in crccheck.crc*), 26
Crc64GoEcma (*in module crccheck.crc*), 26
Crc64GoIso (*class in crccheck.crc*), 26
Crc64Ms (*class in crccheck.crc*), 26
Crc64Redis (*class in crccheck.crc*), 26
Crc64We (*class in crccheck.crc*), 26
Crc64Xz (*class in crccheck.crc*), 26
Crc6Cdma2000A (*class in crccheck.crc*), 13
Crc6Cdma2000B (*class in crccheck.crc*), 13
Crc6Darc (*class in crccheck.crc*), 13
Crc6G704 (*class in crccheck.crc*), 13
Crc6Gsm (*class in crccheck.crc*), 14
Crc6Itu (*in module crccheck.crc*), 13
Crc7 (*in module crccheck.crc*), 14
Crc7Mmc (*class in crccheck.crc*), 14
Crc7Rohc (*class in crccheck.crc*), 14
Crc7Umts (*class in crccheck.crc*), 14
Crc8 (*in module crccheck.crc*), 16
Crc82Darc (*class in crccheck.crc*), 26
Crc8Aes (*in module crccheck.crc*), 16
Crc8Autosar (*class in crccheck.crc*), 14
Crc8Base (*class in crccheck.crc*), 12
Crc8Bluetooth (*class in crccheck.crc*), 14
Crc8Cdma2000 (*class in crccheck.crc*), 14
Crc8Darc (*class in crccheck.crc*), 14
Crc8DvbS2 (*class in crccheck.crc*), 14
Crc8Ebu (*in module crccheck.crc*), 16
Crc8GsmA (*class in crccheck.crc*), 14
Crc8GsmB (*class in crccheck.crc*), 14
Crc8Hitag (*class in crccheck.crc*), 14
Crc8I4321 (*class in crccheck.crc*), 15
Crc8ICode (*class in crccheck.crc*), 15
Crc8Itu (*in module crccheck.crc*), 15
Crc8Lte (*class in crccheck.crc*), 15
Crc8Maxim (*in module crccheck.crc*), 15
Crc8MaximDow (*class in crccheck.crc*), 15
Crc8MifareMad (*class in crccheck.crc*), 15
Crc8Nrsc5 (*class in crccheck.crc*), 15
Crc8Opensafety (*class in crccheck.crc*), 15

Crc8Rohc (*class in crccheck.crc*), 15
Crc8SaeJ1850 (*class in crccheck.crc*), 15
Crc8Smbus (*class in crccheck.crc*), 15
Crc8Tech3250 (*class in crccheck.crc*), 16
Crc8Wcdma (*class in crccheck.crc*), 16
CrcA (*in module crccheck.crc*), 20
CrcArc (*in module crccheck.crc*), 18
CrcB (*in module crccheck.crc*), 19
CrcBase (*class in crccheck.crc*), 9
Crcc16Mcrf4xx (*in module crccheck.crc*), 20
CrcCcitt (*in module crccheck.crc*), 20
crccheck
    module, 1, 27
crccheck.base
    module, 3
crccheck.checksum
    module, 6
crccheck.crc
    module, 8
CrccheckBase (*class in crccheck.base*), 4
CrccheckError, 4
CrcCksum (*in module crccheck.crc*), 24
CrcDow (*in module crccheck.crc*), 15
CrcIbm (*in module crccheck.crc*), 18
CrcJamcrc (*in module crccheck.crc*), 25
CrcKermit (*in module crccheck.crc*), 20
CrcModbus (*in module crccheck.crc*), 21
CrcPkzip (*in module crccheck.crc*), 25
CrcX25 (*in module crccheck.crc*), 19
CrcXfer (*in module crccheck.crc*), 26
CrcXmodem (*in module crccheck.crc*), 22
CrcZmodem (*in module crccheck.crc*), 22

F

final() (*crccheck.base.CrccheckBase method*), 4
final() (*crccheck.crc.CrcBase method*), 9
finalbytes() (*crccheck.base.CrccheckBase method*), 4
finalhex() (*crccheck.base.CrccheckBase method*), 4
find() (*in module crccheck.crc*), 9

I

identify() (*in module crccheck.crc*), 10

M

module
    crccheck, 1, 27
    crccheck.base, 3
    crccheck.checksum, 6
    crccheck.crc, 8

P

process() (*crccheck.base.CrccheckBase method*), 4
process() (*crccheck.checksum.ChecksumBase method*), 6
process() (*crccheck.checksum.ChecksumXorBase method*), 7
process() (*crccheck.crc.Crc16Base method*), 12
process() (*crccheck.crc.Crc32Base method*), 12
process() (*crccheck.crc.Crc8Base method*), 12
process() (*crccheck.crc.CrcBase method*), 9

R

reflectbitorder() (*in module crccheck.base*), 3
reset() (*crccheck.base.CrccheckBase method*), 4

S

selftest() (*crccheck.base.CrccheckBase class method*), 5
selftest() (*crccheck.checksum.ChecksumBase class method*), 7
selftest() (*crccheck.crc.Crc method*), 11

V

value() (*crccheck.base.CrccheckBase method*), 5