-gbinary

GoFramegbinary[ ]byte//

Content Menu

-
-

```
import "github.com/gogf/gf/v2/encoding/gbinary"
```

https://pkg.go.dev/github.com/gogf/gf/v2/encoding/gbinary

```
func Encode(vs ...interface{}) ([]byte, error)
func EncodeInt(i int) []byte
func EncodeInt8(i int8) []byte
func EncodeInt16(i int16) []byte
func EncodeInt32(i int32) []byte
func EncodeInt64(i int64) []byte
func EncodeUint(i uint) []byte
func EncodeUint8(i uint8) []byte
func EncodeUint16(i uint16) []byte
func EncodeUint32(i uint32) []byte
func EncodeUint64(i uint64) []byte
func EncodeBool(b bool) []byte
func EncodeFloat32(f float32) []byte
func EncodeFloat64(f float64) []byte
func EncodeString(s string) []byte

func Decode(b []byte, vs ...interface{}) error
func DecodeToInt(b []byte) int
func DecodeToInt8(b []byte) int8
func DecodeToInt16(b []byte) int16
func DecodeToInt32(b []byte) int32
func DecodeToInt64(b []byte) int64
func DecodeToUint(b []byte) uint
func DecodeToUint8(b []byte) uint8
func DecodeToUint16(b []byte) uint16
func DecodeToUint32(b []byte) uint32
func DecodeToUint64(b []byte) uint64
func DecodeToBool(b []byte) bool
func DecodeToFloat32(b []byte) float32
func DecodeToFloat64(b []byte) float64
func DecodeToString(b []byte) string
```

```
func EncodeBits(bits []Bit, i int, l int) []Bit
func EncodeBitsWithUint(bits []Bit, ui uint, l int) []Bit
func EncodeBitsToBytes(bits []Bit) []byte
func DecodeBits(bits []Bit) uint
func DecodeBitsToUint(bits []Bit) uint
func DecodeBytesToBits(bs []byte) []Bit
```

Bit(01)

```
type Bit int8
```

github.com/gogf/gf/v2/blob/master/.example/encoding/gbinary/binary.go

```
package main

import (
        "fmt"
        "github.com/gogf/gf/v2/encoding/gbinary"
        "github.com/gogf/gf/v2/os/gctx"
        "github.com/gogf/gf/v2/os/glog"
)

func main() {
        // gbinary.Encoded
        if buffer := gbinary.Encode(18, 300, 1.01); buffer != nil {
                // glog.Error(err)
        } else {
                fmt.Println(buffer)
        }

        // gbinary.Decode
        // int8/16/32/64uint8/16/32/64float32/64
        // 1.01float64(64)
        if buffer := gbinary.Encode(18, 300, 1.01); buffer != nil {
                //glog.Error(err)
        } else {
                var i1 int8
                var i2 int16
                var f3 float64
                if err := gbinary.Decode(buffer, &i1, &i2, &f3); err !=
nil {
                        glog.Error(gctx.New(), err)
                } else {
                        fmt.Println(i1, i2, f3)
                }
        }

        // / int
        fmt.Println(gbinary.DecodeToInt(gbinary.EncodeInt(1)))
        fmt.Println(gbinary.DecodeToInt(gbinary.EncodeInt(300)))
        fmt.Println(gbinary.DecodeToInt(gbinary.EncodeInt(70000)))
        fmt.Println(gbinary.DecodeToInt(gbinary.EncodeInt(2000000000)))
        fmt.Println(gbinary.DecodeToInt(gbinary.EncodeInt(500000000000)))

        // / uint
        fmt.Println(gbinary.DecodeToUint(gbinary.EncodeUint(1)))
        fmt.Println(gbinary.DecodeToUint(gbinary.EncodeUint(300)))
        fmt.Println(gbinary.DecodeToUint(gbinary.EncodeUint(70000)))
        fmt.Println(gbinary.DecodeToUint(gbinary.EncodeUint(2000000000)))
        fmt.Println(gbinary.DecodeToUint(gbinary.EncodeUint(500000000000)))

        // / int8/16/32/64
        fmt.Println(gbinary.DecodeToInt8(gbinary.EncodeInt8(int8(100))))
        fmt.Println(gbinary.DecodeToInt16(gbinary.EncodeInt16(int16(100))))
        fmt.Println(gbinary.DecodeToInt32(gbinary.EncodeInt32(int32(100))))
        fmt.Println(gbinary.DecodeToInt64(gbinary.EncodeInt64(int64(100))))

        // / uint8/16/32/64
        fmt.Println(gbinary.DecodeToUint8(gbinary.EncodeUint8(uint8(100))))
        fmt.Println(gbinary.DecodeToUint16(gbinary.EncodeUint16(uint16
(100))))
        fmt.Println(gbinary.DecodeToUint32(gbinary.EncodeUint32(uint32
(100))))
        fmt.Println(gbinary.DecodeToUint64(gbinary.EncodeUint64(uint64
(100))))

        // / string
        fmt.Println(gbinary.DecodeToString(gbinary.EncodeString("I'm
string!")))
}
```

```
[18 44 1 41 92 143 194 245 40 240 63]
18 300 1.01
1
300
70000
2000000000
500000000000
1
300
70000
2000000000
500000000000
100
100
100
100
100
100
100
100
I'm string!
```

1.

   gbinary.Encode([ ]byte)gbinary.Encodeint1gbinary.Encode1byteint3002byte[ ]byte/int8/16/32/64

   gbinarygbinary.Encode*gbinary.EncodeInt/gbinary.EncodeUint[ ]byte1/2/4/8

2.

   ([ ]byte)gbinary.Decodeint8/16/32/64uint8/16/32/64float32/64int/uint

   gbinarygbinary.DecodeTo*gbinary.DecodeToInt/gbinary.DecodeToUint1-8

gbinaryBits

1.

   4(0:1: 2: 3:)2(2 bit)1byte(8 bit)4

   100(0)

   https://github.com/gogf/gf/v2/blob/master/.example/encoding/gbinary/bits1.go

```
package main


import (
    "fmt"
    "github.com/gogf/gf/v2/encoding/gbinary"
)


func main() {
    // 0:, 1:, 2: 3:
    count  := 100
    status := 1


    //
    bits := make([]gbinary.Bit, 0)
    for i := 0; i < count; i++ {
        bits = gbinary.EncodeBits(bits, int(status), 2)
    }
    buffer := gbinary.EncodeBitsToBytes(bits)
    fmt.Println("buffer length:", len(buffer))


    /* / */


    //
    alivecount := 0
    sensorbits := gbinary.DecodeBytesToBits(buffer)
    for i := 0; i < len(sensorbits); i += 2 {
        if gbinary.DecodeBits(sensorbits[i:i+2]) == 1 {
            alivecount++
        }
    }
    fmt.Println("alived sensor:", alivecount)
}
```

```
buffer length: 25
alived sensor: 100
```

10025byte100

2. **gkvdbMETA**

gkvdbGoFrameDRHKey-Value

```
[64(64bit,8byte) (8bit,1byte) (24bit,3byte) (40bit,5byte)]()
```

/

github.com/gogf/gf/v2/blob/master/.example/encoding/gbinary/bits2.go

```
package main


import (
    "fmt"
    "github.com/gogf/gf/v2/encoding/gbinary"
)


func main() {
    // Meta[64(64bit,8byte) (8bit,1byte) (24bit,3byte) (40bit,5byte)]
()
    hash   := 521369841259754125
    klen   := 12
    vlen   := 35535
    offset := 80000000


    //
    bits   := make([]gbinary.Bit, 0)
    bits    = gbinary.EncodeBits(bits, hash,   64)
    bits    = gbinary.EncodeBits(bits, klen,    8)
    bits    = gbinary.EncodeBits(bits, vlen,   24)
    bits    = gbinary.EncodeBits(bits, offset, 40)
    buffer := gbinary.EncodeBitsToBytes(bits)
    fmt.Println("meta length:", len(buffer))


        /* / */


    //
    metabits := gbinary.DecodeBytesToBits(buffer)
    fmt.Println("hash  :", gbinary.DecodeBits(metabits[0  : 64]))
    fmt.Println("klen  :", gbinary.DecodeBits(metabits[64 : 72]))
    fmt.Println("vlen  :", gbinary.DecodeBits(metabits[72 : 96]))
    fmt.Println("offset:", gbinary.DecodeBits(metabits[96 : 136]))
}
```

```
meta length: 17
hash  : 521369841259754125
klen  : 12
vlen  : 35535
offset: 80000000
```