BufDec

Decode a part of a buffer into a scalar

Description

```
BufDec( const buf[], offset, method )
```

This command will decode a portion of a buffer into a scalar variable.

Parameters

`buf`	Data buffer
`offset`	Byte offset into the buffer where to start the decoding
`method`	What type of value is encoded in the buffer

Available methods:

Method	Length	Type	Description
BT_INT8	8 bit	signed integer	Single octet value
BT_UINT8	8 bit	unsigned integer	Single octet value
BT_INT16	16 bit	signed integer	Big endian integer
BT_UINT16	16 bit	unsigned integer	Big endian integer
BT_INT32	32 bit	signed integer	Big endian integer
BT_UINT32	32 bit	unsigned integer	Big endian integer
BT_INT32LE	32 bit	signed integer	Little endian integer
BT_UINT32LE	32 bit	unsigned integer	Little endian integer
BT_INT32BLE	32 bit	signed integer	Big-Little endian integer
BT_UINT32BLE	32 bit	unsigned integer	Big-Little endian integer
BT_INT32LBE	32 bit	signed integer	Little-Big endian integer
BT_UINT32LBE	32 bit	unsigned integer	Little-Bit endian integer
BT_FLOAT	32 bit	float	IEEE 732 floating point
BT_FLOATLE	32 bit	float	IEEE 732 floating point little endian
BT_FLOAT64	64 bit	float64	IEEE 732 64-bit floating point (converts to 32 bit float)
BT_FLOAT64LE	64 bit	float64	IEEE 732 64-bit floating point little endian (converts to 32 bit float)

Also available for decoding bitmaps:

Bit offset : **BT_BITOFS0** through **BT_BITOFS15**
Bit mask : **BT_BITMASK1** through **BT_BITMASK8**

The BT_BITOFSx and BT_BITMASKx methods can be OR:ed together for bitmask decoding.

Example: To extract a 3 bit value from bit position 5,6 and 7, use `BT_BITOFS5 | BT_BITMASK3`

Return value

Returns the decoded value

Note that the returned value is always a 32 bit cell (signed 32 bit integer or 32 bit float).

Example usage

```
    new x;
    new s{10};

    if(ModbusRead(5, MB_HOLDING, 100, 2, s) == 4) {  // Request and receive
two registers (32 bits total)
        x = BufDec(s, 0, BT_INT32);                  // Decode the received
data as a 32 bit integer
        // ..further processing
    }
```