

Sollae Systems Co., Ltd.

# **PHPoC Internal Functions**

Revision V1.0

# Table of Contents

## PHPoC Functions

bin2int()	4
hexdump()	6
int2bin()	8
pid_bind()	10
pid_close()	11
pid_connect()	12
pid_ioctl()	13
pid_listen()	15
pid_lseek()	16
pid_open()	18
pid_read()	19
pid_recv()	21
pid_recvfrom()	23
pid_send()	25
pid_sendto()	27
pid_write()	29
_GET()	31
_POST()	32
_SERVER()	33

## PHP Functions

bin2hex()	34
count()	35
die()	36
explode()	37
exit()	38
flush()	39
header()	40
hex2bin()	41
inet_ntop()	42
inet_pton()	43
ltrim()	44
printf()	45

rtrim()	47
set_time_limit()	48
sleep()	49
sprintf()	50
str_replace()	52
strlen()	53
strpos()	54
strtolower()	55
strtoupper()	56
substr()	57
substr_replace()	59
system()	61
usleep()	62

## PHP Math Functions

abs()	63
acos()	64
asin()	65
atan()	66
atan2()	67
ceil()	68
cos()	69
deg2rad()	70
exp()	71
floor()	72
is_finite()	73
is_infinite()	74
is_nan()	75
log()	76
pow()	77
rad2deg()	78
rand()	79
round()	80
sin()	81
sqrt()	82
tan()	83

Revision History .....84

# bin2int()

---

int bin2int ( string *\$data*, int *\$offset*, int *\$len* [ , bool *\$swap* = false] )

---

## Description

bin2int() converts string to integer value, size of len from the offset (little endian data format) with optional swap function

## Parameters

*\$data*

string to convert

*\$offset*

offset bytes

*\$len*

string length from the \$offset to convert

*\$swap*

Swap operation performed when this field is true (default: false)

## Return values

Returns converted integer, PHP error on error

## Examples

```
<?php
$buf = "";
$buf = int2bin(0x11223344, 4);
$buf .= int2bin(0x55667788, 4);
hexdump($buf);
// OUTPUT: 0000 44 33 22 11 88 77 66 55          ID3". .wfU          |

$i = bin2int($buf, 0, 1);      // 1 byte from position 0
echo "i: $i\r\n";             // OUTPUT: i: 68
$i = bin2int($buf, 1, 2);      // 2 bytes from position 1
echo "i: $i\r\n";             // OUTPUT: i: 8755
$i = bin2int($buf, 1, 2, true); // 2 bytes from position 1 with swap
echo "i: $i\r\n";             // OUTPUT: i: 13090
$i = bin2int($buf, 0, 4);      // 4 bytes from position 0
echo "i: $i\r\n";             // OUTPUT: i: 287454020
$i = bin2int($buf, 0, 4, true); // 4 bytes from position 0 with swap
echo "i: $i\r\n";             // OUTPUT: i: 1144201745
$i = bin2int($buf, 0, 8);      // 8 bytes from position 0
echo "i: $i\r\n";             // OUTPUT: i: 6153737367135073092
```

```
$i = bin2int($buf, 0, 8, true); // 8 bytes from position 0 with swap
echo "i: $i\r\n";           // OUTPUT: i: 4914309077090657877
?>
```

```
<?php
// It receives data from the UART0 and transmits data to the UART0 after changing 0x20 data to 0x2e

$pid = pid_open("/mmap/uart0"); // opens UART0
// set the device to 115200bps, no parity, 8 databit, 1stop bit
pid_ioctl($pid, "set baud 115200");
pid_ioctl($pid, "set parity 0");
pid_ioctl($pid, "set data 8");
pid_ioctl($pid, "set stop 1");
pid_ioctl($pid, "set flowctrl 0");

$rbuf = ""; // declaring $rbuf
$wbuf = ""; // declaring $wbuf
while(1)
{
    $rlen = pid_read($pid, $rbuf, 16); // read upto 16bytes data from the UART0

    if($rlen > 0) // if there are received data
    {
        $wbuf = ""; // clear $wbuf
        for($i = 0; $i < $rlen; $i++)
        {
            $data = bin2int($rbuf, $i, 1); // getting 1 byte data from the $rbuf at position $i
            if($data == 0x20) $data = 0x2e; // changing $data to 0x2e if it is 0x20
            $wbuf .= int2bin($data, 1); // append $data to the $wbuf
        }
    }
    pid_write($pid, $wbuf, $rlen); // write the $rlen length of the $wbuf to the UART0
}
?>
```

## See also

[int2bin\(\)](#)

## Remarks

None

# hexdump()

```
int hexdump ( string $str [ ,int $start=0, int $len=128 ] )
```

## Description

hexdump() dumps a binary data stream into a human-friendly format to the standard output. It outputs maximum PHP\_STDOUT\_BUF/4 bytes.

## Parameters

*\$str*

string data to output

*\$start*

start position to output

If this value is minus it starts at the start'th character from the end of string.

*\$len*

string length to output

## Return values

Returns the output data length, PHP error on error

## Examples

```
<?php
$str = "Hello PHPoC";

hexdump($str);
// OUTPUT: 0000 48 65 6c 6c 6f 20 50 48 50 6f 43           |Hello PHPoC   |

hexdump($str, 1, 3);
// OUTPUT: 0001 65 6c 6c                                |ell           |

hexdump($str, -5, 3);
// OUTPUT: 0006 50 48 50                                |PHP           |

?>
```

Use the following code if the input data length is over the maximum length:

```
<?php
function hexdump1024($str)
{
    for($offset = 0; $offset < strlen($str); )
        $offset += hexdump($str, $offset);

    return $offset;
}

?>
```

**See also**

None

**Remarks**

None

# int2bin()

---

string int2bin ( int *\$val*, int *\$len* [ , bool *\$swap=false* ] )

---

## Description

int2bin() converts integer value to string (little endian data format) with optional swap function

## Parameters

*\$val*

integer value to convert

*\$len*

string length

*\$swap*

Swap operation performed when this field is true (default: false)

## Return values

Returns converted string, PHP error on error

## Examples

```
<?php
$buf = "";
$buf .= int2bin(0x55, 4);           // 0x55 0x00 0x00 0x00
$buf .= int2bin(0x55, 8);           // 0x55 0x00 0x00 0x00 0x00 0x00 0x00 0x00

$buf .= int2bin(0x1122334455667788, 8); // 0x88 0x77 0x66 0x55 0x44 0x33 0x22 0x11
$buf .= int2bin(0xaa88, 2, true); // 0xaa 0x88 (with swap)
$buf .= int2bin(7000, 2);           // 0x58 0x1b

hexdump($buf);
// OUTPUT
// 0000 55 00 00 00 55 00 00 00 00 00 00 00 88 77 66 55 |U...U.....wFU|
// 0010 44 33 22 11 aa 88 58 1b                               |D3"...X.      |
?>
```

```
<?php
// It receives data from the UART0 and transmits data to the UART0 after changing 0x20 data to 0x2e

$pid = pid_open("/mmap/uart0"); // opens UART0
// set the device to 115200bps, no parity, 8 databit, 1stop bit
pid_ioctl($pid, "set baud 115200");
pid_ioctl($pid, "set parity 0");
```

```

pid_ioctl($pid, "set data 8");
pid_ioctl($pid, "set stop 1");
pid_ioctl($pid, "set flowctrl 0");

$rbuf = ""; // declaring $rbuf
$wbuf = ""; // declaring $wbuf
while(1)
{
    $rlen = pid_read($pid, $rbuf, 16); // read upto 16bytes data from the UART0

    if($rlen > 0) // if there are received data
    {
        $wbuf = ""; // clear $wbuf
        for($i = 0; $i < $rlen; $i++)
        {
            $data = bin2int($rbuf, $i, 1); // getting 1 byte data from the $rbuf at position $i
            if($data == 0x20) $data = 0x2e; // changing $data to 0x2e if it is 0x20
            $wbuf .= int2bin($data, 1); // append $data to the $wbuf
        }
    }
    pid_write($pid, $wbuf, $rlen); // write the $rlen length of the $wbuf to the UART0
}
?>

```

## See also

[bin2int\(\)](#)

## Remarks

None

# pid\_bind()

---

int pid\_bind ( int *\$pid* [, string *\$addr* = "", int *\$port* = 0 ] )

---

## Description

pid\_bind() binds a name.

This bind function is necessary before using UDP reception.

## Parameters

*\$pid*

network device's PID to bind

*\$addr*

This field should be empty string ("")

*\$port*

port number to bind

## Return values

Returns 0 on success, PHP error on error.

## Examples

```
<?php
$buf = "Hello PHPoC!";
$peer_addr = "10.3.0.10";
$peer_port = 1470;

$pid = pid_open("/mmap/udp0");
pid_bind($pid, "", 1470);

$wlen = pid_sendto($pid, $buf, strlen($buf), 0, $peer_addr, $peer_port);
echo "udp sent ($wlen bytes)\r\n";
?>
```

## See also

pid\_open() / pid\_ioctl() / pid\_sendto() / pid\_recvfrom()

## Remarks

None

# pid\_close()

---

int pid\_close ( int *\$pid* )

---

## Description

pid\_close() closes the specified port/peripheral

## Parameters

*\$pid*

the valid device PID created with pid\_open()

## Return values

Returns 0 on success, PHP error on error

## Examples

```
<?php
$pid = pid_open("/mmap/uart0"); // open UART0

$buf = "Hello PHPoC!";
// set the device to 115200bps, no parity, 8 databit, 1stop bit
pid_ioctl($pid, "set baud 115200");
pid_ioctl($pid, "set parity 0");
pid_ioctl($pid, "set data 8");
pid_ioctl($pid, "set stop 1");
pid_ioctl($pid, "set flowctrl 0");

$wlen = pid_write($pid, $buf, 12); // write 12 bytes of the $buf to the $pid

pid_close($pid);
?>
```

## See also

pid\_open() / pid\_read() / pid\_write() / pid\_ioctl() / pid\_recv() / pid\_send()

## Remarks

Even though a pid\_close() function has been called, some devices can be accessed without pid\_open() function

# pid\_connect()

---

```
int pid_connect ( int $pid, string $addr, int $port )
```

---

## Description

pid\_connect() attempts to TCP connection to the specified IP address and port number  
This function is non-blocking so it returns immediately

## Parameters

*\$pid*

the valid network device PID created with pid\_open()

*\$addr*

destination IP address

*\$port*

destination port number

## Return values

Returns 0 on success, PHP error on error

## Examples

```
<?php
$pid = pid_open("/mmap/tcp0"); // open TCP0
$addr = "10.3.0.10"; // IP address to connect to
$port = 1470; // peer host's service port number
pid_connect($pid, $addr, $port); // trying to TCP connect to the specified host/port
do
{
    $state = pid_ioctl($pid, "get state");
}while(($state != TCP_CONNECTED) && ($state != TCP_CLOSED));
if($state == TCP_CONNECTED) echo "TCP connected\r\n";
else if($state == TCP_CLOSED) echo "TCP connection failed\r\n";

while(1);
?>
```

## See also

pid\_open() / pid\_close() / pid\_recv() / pid\_send() / pid\_ioctl() / pid\_listen()

## Remarks

None

# pid\_ioctl()

int/string pid\_ioctl ( int *\$pid*, string *\$req*, [ , string *\$arg*, ... ] )

---

## Description

pid\_ioctl() manipulates the device's parameters

## Parameters

*\$pid*

device's PID to manipulate

*\$req*

device's parameters (maximum length: PHP\_LLSTR\_BLK\_SIZE - 1)

*\$opt*

optional argument for the *\$req*

## Return values

Returns an integer or string value on success, PHP error occurs on fail

## Examples

```
<?php
$pid = pid_open("/mmap/uart0"); // open UART0
// set the device to 115200bps, no parity, 8 databit, 1stop bit
pid_ioctl($pid, "set baud 115200");
pid_ioctl($pid, "set parity 0");
pid_ioctl($pid, "set data 8");
pid_ioctl($pid, "set stop 1");
pid_ioctl($pid, "set flowctrl 0");

$rbuf = "";
while(1)
{
    $rxlen = pid_ioctl($pid, "get rxlen"); // get received data from the uart0
    if($rxlen >= 8)
    {
        $rlen = pid_read($pid, $rbuf, 8);
        if($rlen > 0 )
        {
            $wlen = pid_write($pid, $rbuf, $rlen);
            echo "$rlen bytes received and echoed\r\n";
        }
    }
}
```

```
}  
?>
```

## See also

`pid_open()` / `pid_close()` / `pid_read()` / `pid_write()` / `pid_recv()` / `pid_send()`

## Remarks

Refer to the device programming guide for more information for each devices.

This `pid_ioctl()` function supports another function format.

The words starting with '%' followed by a number in the string are replaced by parameters.

For example, the following two functions are exactly same.

```
pid_ioctl($pid, "set baud 115200", " kdfjkd");  
  
$baudrate_name = "baud";  
$baudrate = "115200";  
pid_ioctl($pid, "set %1 %2", $baudrate_name, $baudrate);
```

# pid\_listen()

```
int pid_listen ( int $pid, [ int $backlog = 1 ] )
```

---

## Description

pid\_listen() waits for and accepts a TCP connection from a foreign client.  
This function is non-blocking so it returns immediately.

## Parameters

*\$pid*

the valid TCP device PID created with pid\_open()

*\$backlog*

maximum number of incoming connections (default: 1)

Only 1 is valid for this field

## Return values

Returns 0 on success, PHP error on error

## Examples

```
<?php
$pid = pid_open("/mmap/tcp0"); // open TCP0
$port = 1470; // local port number to accept
pid_bind($pid, "", $port);
pid_listen($pid); // wait for a TCP connection from the foreign host
echo "listen...\r\n";
do
{
    $state = pid_ioctl($pid, "get state");
}while(($state != TCP_CONNECTED) && ($state != TCP_CLOSED));
if($state == TCP_CONNECTED) echo "TCP connected\r\n";
else if($state == TCP_CLOSED) echo "TCP connection failed\r\n";

while(1);
?>
```

## See also

pid\_open() / pid\_close() / pid\_recv() / pid\_send() / pid\_ioctl() / pid\_connect()

## Remarks

None

# pid\_lseek()

---

int pid\_lseek ( int *\$pid*, int *\$offset* [ , int *\$whence* = SEEK\_SET ] )

---

## Description

pid\_lseek() relocates read/write PID offset

## Parameters

*\$pid*

device's PID to seek

*\$offset*

the offset

To move to a position before the end-of-file, you need to pass a zero or negative value in offset and set whence to SEEK\_END

*\$whence*

SEEK\_SET – Set position equal to offset bytes

SEEK\_CUR – Set position to current location plus offset

SEEK\_END – Set position to end-of-file plus offset (the offset should be zero or negative when \$whence is SEEK\_END)

## Return values

Returns the offset from the beginning, PHP error on error

## Examples

```
<?php

$buf = "ABCDEFGHJKLMNOPabcdefghijklmnop";

$pid_um = pid_open("/mmap/um1"); // /mmap/um1 buffer size is 64

pid_lseek($pid_um, 0, SEEK_SET); // set the point to the beginning (position 0)
pid_write($pid_um, $buf, 32); // write $buf to the position 0 (position changed to 32 after this
operation)
pid_lseek($pid_um, 0, SEEK_SET); // set the point to the beginning (position 0)
pid_read($pid_um, $buf, 32); // read 32 bytes from the current position (position 0)
hexdump($buf); // outputs $buf
// OUTPUT
// 0000 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 |ABCDEFGHJKLMNOP|
// 0010 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 |abcdefghijklmnop|

pid_lseek($pid_um, 0, SEEK_SET); // set the point to the beginning (position 0)
pid_lseek($pid_um, 8, SEEK_CUR); // set the point to position 8
```

```

pid_read($pid_um, $buf, 16); // read 16 bytes from the current position (position 8)
hexdump($buf); // outputs $buf
// OUTPUT
// 0000 49 4a 4b 4c 4d 4e 4f 50 61 62 63 64 65 66 67 68 |IJKLMNOPabcdefgh|

pid_read($pid_um, $buf, 16); // read 16 bytes from the current position (position 24)
hexdump($buf);
// OUTPUT
// 0000 69 6a 6b 6c 6d 6e 6f 70 00 00 00 00 00 00 00 00 |ijklmnop.....|

pid_lseek($pid_um, -48, SEEK_END); // set the point to 48 from the end (position 16)
pid_read($pid_um, $buf, 16); // read 16 bytes from the current position (position 16)
hexdump($buf); // outputs $buf
// OUTPUT
// 0000 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 |abcdefghijklmnop|
?>

```

## See also

[pid\\_open\(\)](#) / [pid\\_close\(\)](#) / [pid\\_read\(\)](#) / [pid\\_write\(\)](#) / [pid\\_ioctl\(\)](#)

# pid\_open()

---

int pid\_open ( string *\$path* )

---

## Description

pid\_open() opens the specified file/port/peripheral

## Parameters

*\$path*

file/port/peripheral's path

## Return values

Returns file/port/peripheral's PID on success or PHP error on fail

## Examples

```
<?php
$pid = pid_open("/mmap/uart0"); // open UART0

// set the device to 115200bps, no parity, 8 databit, 1stop bit
pid_ioctl($pid, "set baud 115200");
pid_ioctl($pid, "set parity 0");
pid_ioctl($pid, "set data 8");
pid_ioctl($pid, "set stop 1");
pid_ioctl($pid, "set flowctrl 0");

$rbuf = "";
while(1)
{
    $rlen = pid_read($pid, $rbuf, 10);    // read upto 10 bytes from the $pid into $rbuf
    if($rlen > 0) $wlen = pid_write($pid, $rbuf, $rlen); // write $rlen bytes of the $rbuf to the $pid
}

pid_close($pid);
?>
```

## See also

pid\_close() / pid\_read() / pid\_write() / pid\_ioctl() / pid\_connect() / pid\_rcv() / pid\_send()

## Remarks

Refer to the product's manual for the paths for each product.

# pid\_read()

---

```
int pid_read ( int $pid, int/string &$buf [ , int $len ] )
```

---

## Description

pid\_read() attempts to read up to \$len bytes from the port or peripheral \$pid into the buffer \$buf

## Parameters

*\$pid*

device's PID to read

*\$buf*

destination buffer will be saved to

*\$len*

bytes to read

It reads upto size of the \$buf if this field is omitted (integer: 8bytes, string: MAX\_STRING\_LEN)

## Return values

On success, the number of bytes read is returned, otherwise PHP error

It is not an error if this number is smaller than the number of bytes requested. This may happen for example fewer bytes are actually available right now.

## Examples

```
<?php
$buf = "";
$pid = pid_open("/mmap/uart0"); // open UART0

// set the device to 115200bps, no parity, 8 databit, 1stop bit
pid_ioctl($pid, "set baud 115200");
pid_ioctl($pid, "set parity 0");
pid_ioctl($pid, "set data 8");
pid_ioctl($pid, "set stop 1");
pid_ioctl($pid, "set flowctrl 0");

while(1)
{
    $rlen = pid_read($pid, $buf, 10); // read maximum 10 bytes from the $pid into $buf
    if($rlen > 0) // if there is any received data
    {
        $wlen = pid_write($pid, $buf, $rlen); // write $rlen bytes of the $buf to the $pid
        echo "$rlen bytes received and echoed\r\n";
    }
}
```

```
pid_close($pid);  
?>
```

```
<?php  
$ch = 0;  
$pid = pid_open("/mmap/uart0"); // open UART0  
  
// set the device to 115200bps, no parity, 8 databit, 1stop bit  
pid_ioctl($pid, "set baud 115200");  
pid_ioctl($pid, "set parity 0");  
pid_ioctl($pid, "set data 8");  
pid_ioctl($pid, "set stop 1");  
pid_ioctl($pid, "set flowctrl 0");  
while(1)  
{  
    $rlen = pid_read($pid, $ch, 1); // read 1 byte from the $pid into $ch  
    if($rlen > 0) // if there is received data  
    {  
        printf("received data: 0x%02x(%c).\r\n", $ch, $ch); // print the data  
    }  
}  
pid_close($pid);  
?>
```

## See also

[pid\\_open\(\)](#) / [pid\\_close\(\)](#) / [pid\\_write\(\)](#) / [pid\\_ioctl\(\)](#)

## Remarks

None

# pid\_recv()

---

```
int pid_recv ( int $pid, int/string &$buf [ , int $len, int $flags = 0 ] )
```

---

## Description

pid\_recv() attempts to read up to \$len bytes from the network device \$pid into the buffer \$buf

## Parameters

*\$pid*

the valid TCP device PID created with pid\_open()

*\$buf*

destination buffer will be saved to

*\$len*

bytes to receive

It receives upto size of the \$buf if this field is omitted (integer: 8bytes, string: MAX\_STRING\_LEN)

*\$flags*

This field should be 0

## Return values

On success received the number of bytes is returned, otherwise PHP error

It is not an error if this number is smaller than the number of bytes requested. This may happen for example fewer bytes are actually available right now.

## Examples

```
<?php
$buf = "";

$pid = pid_open("/mmap/tcp0");
pid_connect($pid, "10.3.0.10", 1470);
do
{
    $state = pid_ioctl($pid, "get state"); // get the current TCP state
    if($state == TCP_CLOSED) // if TCP connection attempt fails, try to connect again
    {
        pid_connect($pid, "10.3.0.10", 1470);
        sleep(1);
    }
}while($state != TCP_CONNECTED);
echo "TCP connected\r\n";

while(1)
```

```
{
    $rlen = pid_recv($pid, $buf, 100); // read the received data into $buf upto 100 bytes
    if($rlen > 0)
    {
        echo "tcp received: $rlen bytes\r\n";
        $wlen = pid_send($pid, $buf, $rlen, 0);
        echo "tcp echo sent: $wlen bytes\r\n";
    }
}
?>
```

### **See also**

[pid\\_open\(\)](#) / [pid\\_close\(\)](#) / [pid\\_send\(\)](#) / [pid\\_ioctl\(\)](#)

### **Remarks**

None

# pid\_recvfrom()

---

int pid\_recvfrom ( int *\$pid*, int/string &*\$buf* [ , int *\$len*, int *\$flags* = 0, string &*\$addr*, int &*\$port* ] )

---

## Description

pid\_recvfrom() receives \$len bytes data in \$buf from the address \$addr on the \$port using UDP \$pid

## Parameters

*\$pid*

network device's PID

&*\$buf*

destination buffer will be saved to

*\$len*

maximum number of bytes to receive

It receives upto size of the \$buf if this field is omitted (integer: 8bytes, string: MAX\_STRING\_LEN)

*\$flags*

This field should be 0

&*\$addr*

peer host's IP address

&*\$port*

peer host's UDP port number

## Return values

Returns number of bytes received (it can be less than the \$len). PHP error on error

## Examples

```
<?php
define("MAX_BUF", 100);

$buf = "";
$peer_addr = "";
$peer_port = 0;

$pid = pid_open("/mmap/udp0");
pid_bind($pid, "", 1470);

while(1)
{
    $rlen = pid_recvfrom($pid, $buf, MAX_BUF, 0, $peer_addr, $peer_port);
```

```
if($rlen > 0)
{
    echo "udp received from $peer_addr:$peer_port ($rlen bytes)\r\n";
    $wlen = pid_sendto($pid, $buf, $rlen, 0, $peer_addr, $peer_port);
    echo "udp echo sent: $wlen bytes\r\n";

    // send data another host
    $wlen = pid_sendto($pid, $buf, $rlen, 0, "10.3.0.52", 2000);
    echo "udp sent to another host: $wlen bytes\r\n";
}
}
?>
```

## See also

[pid\\_open\(\)](#) / [pid\\_ioctl\(\)](#) / [pid\\_bind\(\)](#) / [pid\\_sendto\(\)](#)

## Remarks

None

# pid\_send()

---

```
int pid_send ( int $pid, int/string &$buf [ , int $len, int $flags = 0 ] )
```

---

## Description

pid\_send() sends \$len bytes from the \$buf to the network device \$pid

## Parameters

*\$pid*

the valid TCP device PID created with pid\_open()

*\$buf*

source buffer to send

*\$len*

bytes to send

It sends size of the \$buf if this field is omitted (integer: 8bytes, string: MAX\_STRING\_LEN)

*\$flags*

This field should be 0

## Return values

On success, the number of sent bytes is returned. -EPIPE is returned if the network session is closed. Otherwise PHP error

The return value may differ from the \$len because of the network status.

## Examples

```
<?php
$buf = "";

$pid = pid_open("/mmap/tcp0");
pid_connect($pid, "10.3.0.10", 1470);
do
{
    $state = pid_ioctl($pid, "get state"); // get the current TCP state
    if($state == TCP_CLOSED) // if TCP connection attempt fails, try to connect again
    {
        pid_connect($pid, "10.3.0.10", 1470);
        sleep(1);
    }
}while($state != TCP_CONNECTED);
echo "TCP connected\r\n";

while(1)
```

```
{
    $rlen = pid_recv($pid, $buf, 100); // read the received data into $buf upto 100 bytes
    if($rlen > 0)
    {
        echo "tcp received: $rlen bytes\r\n";
        $wlen = pid_send($pid, $buf, $rlen, 0);
        echo "tcp echo sent: $wlen bytes\r\n";
    }
}
?>
```

### **See also**

[pid\\_open\(\)](#) / [pid\\_close\(\)](#) / [pid\\_recv\(\)](#) / [pid\\_ioctl\(\)](#)

### **Remarks**

None

# pid\_sendto()

---

int pid\_sendto ( int *\$pid*, int/string *\$buf* [ , int *\$len*, int *\$flags*, string *\$addr*, int *\$port* ] )

---

## Description

pid\_sendto() transmits \$len bytes data from \$buf through UDP \$pid to the \$port at the address \$addr.

## Parameters

*\$pid*

the valid UDP device PID created with pid\_open()

*\$buf*

source buffer to send

*\$len*

bytes to send

It sends size of the \$buf if this field is omitted (integer: 8bytes, string: MAX\_STRING\_LEN)

*\$flags*

This field should be 0

*\$addr*

peer host's IP address

This field can be set by using pid\_ioctl() and it can be omitted in this case.

(Refer to the following Examples)

*\$port*

peer host's UDP port number

This field can be set by using pid\_ioctl() and it can be omitted in this case.

(Refer to the following Examples)

## Return values

Returns number of bytes sent, PHP error on error

## Examples

```
<?php
$buf = "Hello PHPoC!";
$peer_addr = "10.3.0.10";
$peer_port = 1470;

$pid = pid_open("/mmap/udp0");
pid_ioctl($pid, "set dstaddr $peer_addr");
pid_ioctl($pid, "set dstport $peer_port");
```

```
$wlen = pid_sendto($pid, $buf);  
echo "udp sent: $wlen bytes\r\n";  
?>
```

```
<?php  
$buf = "Hello PHPoC!";  
$peer_addr = "10.3.0.10";  
$peer_port = 1470;  
  
$pid = pid_open("/mmap/udp0");  
pid_bind($pid, "", 1470);  
  
$wlen = pid_sendto($pid, $buf, strlen($buf), 0, $peer_addr, $peer_port);  
echo "udp sent: $wlen bytes\r\n";  
?>
```

### **See also**

[pid\\_open\(\)](#) / [pid\\_ioctl\(\)](#) / [pid\\_bind\(\)](#) / [pid\\_recvfrom\(\)](#)

### **Remarks**

None

# pid\_write()

---

```
int pid_write ( int $pid, int/string &$buf [ , int $len ] )
```

---

## Description

pid\_write() writes \$len bytes from the \$buf to the port or peripheral \$pid

## Parameters

*\$pid*

device's PID to write

*\$wbuf*

source buffer to send

*\$len*

bytes to write

It writes size of the \$buf if this field is omitted (integer: 8bytes, string: MAX\_STRING\_LEN)

## Return values

On success, the number of bytes written is returned. Otherwise: PHP error

## Examples

```
<?php
$rbuf = "";
$pid = pid_open("/mmap/uart0"); // open UART0
// set the device to 115200bps, no parity, 8 databit, 1stop bit
pid_ioctl($pid, "set baud 115200");
pid_ioctl($pid, "set parity 0");
pid_ioctl($pid, "set data 8");
pid_ioctl($pid, "set stop 1");
pid_ioctl($pid, "set flowctrl 0");
while(1)
{
    $rlen = pid_read($pid, $rbuf, 10); // read maximum 10 bytes from the $pid into $rbuf
    if($rlen > 0) // if there is any received data
    {
        $wlen = pid_write($pid, $rbuf, $rlen); // write $rlen bytes of the $rbuf to the $pid
        echo "$rlen bytes received and $wlen bytes echoed\r\n";
    }
}
pid_close($pid);
?>
```

```

<?php
$pid = pid_open("/mmap/uart0"); // open UART0
// set the device to 115200bps, no parity, 8 databit, 1stop bit
pid_ioctl($pid, "set baud 115200");
pid_ioctl($pid, "set parity 0");
pid_ioctl($pid, "set data 8");
pid_ioctl($pid, "set stop 1");
pid_ioctl($pid, "set flowctrl 0");

$ch = '0';
$wlen = pid_write($pid, $ch, 1); // write the $ch to the UART0
echo "\$wlen = $wlen\r\n"; // sent length is 1

$str = "Hello";
$wlen = pid_write($pid, $str); // write the $str to the UART0
echo "\$wlen = $wlen\r\n"; // sent length is 5.

$ch = '0';
$wlen = pid_write($pid, $ch); // write the $ch to the UART0
echo "\$wlen = $wlen\r\n"; // sent length is 1.

$i = 1;
$wlen = pid_write($pid, $i); // write the $i to the UART0
echo "\$wlen = $wlen\r\n"; // sent length is 8 (value: 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00).

pid_close($pid);
?>

```

## See also

[pid\\_open\(\)](#) / [pid\\_close\(\)](#) / [pid\\_read\(\)](#) / [pid\\_ioctl\(\)](#)

## Remarks

None

# \_GET()

---

string `_GET ( string $str )`

---

## Description

`_GET()` finds the query string from the URL request initiated by HTTP GET method

Information sent from a form with the GET method is visible to everybody because it will be displayed in the browser's address window and has limits on the amount of information to send

## Parameters

*\$str*

key string which is the name of the form field

## Return values

If the value for the key string is found, returns the string value. It returns empty string if it is not found.

Otherwise PHP error.

## Examples

```
<?php
// test.php
$str_get = "";
$str_get = _GET("name");
if((bool)$str_get)
{
    echo "get data: $str_get";
    return;
}
?>

<html>
<form name = "submit_button" action = "test.php" method = "GET">
<input type = "text" name = "name">
<input type = "submit" value="Press GET" name="submit_button">
</form>
</html>
```

## See also

`_POST()`

## Remarks

This function is implemented for same operation to the predefined variable `$_GET` of the PHP Group's PHP.

# \_POST()

---

string `_POST` ( string *\$str* )

---

## Description

`_POST()` finds the query string from the HTTP POST method

## Parameters

*\$str*

key string which is the name of the form field

## Return values

If the value for the key string is found, returns the string value. It returns empty string if it is not found.  
Otherwise PHP error.

## Examples

```
<?php
// test.php
$str_post = "";

$str_post = _POST("name");
if((bool)$str_post)
{
    echo "posted data: $str_post";
    return;
}
?>

<html>
<form name = "submit_button" action = "test.php" method = "POST">
<input type = "text" name = "name">
<input type = "submit" value="Press POST" name="submit_button">
</form>
</html>
```

## See also

`_GET()`

## Remarks

This function is implemented for same operation to the predefined variable `$_POST` of the PHP Group's PHP.

# \_SERVER()

---

string `_SERVER ( string $str )`

---

## Description

`_SERVER()` returns information after searching the web browser's request

Can be used to prevent normal script's execution from being executed by web browsers

## Parameters

*\$str*

string to search

REQUEST\_METHOD: request method by the browser (GET/POST)

REMOTE\_ADDR: host's IP address what browser is running on

REMOTE\_PORT: host's port number of this connection

SCRIPT\_NAME: called script name

If *\$str* starts with HTTP\_, the string followed by HTTP\_ is searched at the HTTP header to return its value.

## Return values

If the value for the key string is found, returns the string value. It returns empty string if it is not found.

Otherwise PHP error.

## Examples

```
<?php
if((bool)_SERVER("REQUEST_METHOD"))
    return; /* avoid php execution via http */

// Normal code will be followed:
?>
```

```
<?php
$ret = _SERVER("REQUEST_METHOD");echo "REQUEST_METHOD = $ret<br>";
$ret = _SERVER("REMOTE_ADDR");echo "REMOTE_ADDR = $ret<br>";
$ret = _SERVER("HTTP_USER_AGENT");echo "USER_AGENT = $ret<br>";
?>
```

## See also

None

## Remarks

This function is implemented for same operation to the predefined variable `$_SERVER` of the PHP Group's PHP.

# bin2hex()

---

string bin2hex ( string *\$str* )

---

## Description

bin2hex() converts string into hexadecimal representation

## Parameters

*\$str*

string to convert

## Return values

Returns an ASCII string containing the hexadecimal representation of the given string

## Examples

```
<?php
$str = "ABC abc 012";
hexdump($str); // OUTPUT: 0000 41 42 43 20 61 62 63 20 30 31 32          |ABC abc 012  |
echo bin2hex($str); // OUTPUT: 4142432061626320303132
?>
```

```
<?php
$buf = "Hello";
$hex = bin2hex($buf);

echo $buf; // OUTPUT: Hello
echo "\r\n";
echo $hex; // OUTPUT: 48656C6C6F
?>
```

## See also

[hex2bin\(\)](#)

## Remarks

This function is identical to the PHP group's bin2hex() function.

# count()

---

int count ( bool/int/float/string/array *\$var* [, int *\$mode* = COUNT\_NORMAL] )

---

## Description

count() counts all elements in an array or a variable

## Parameters

*\$var*

an array or a variable to count elements

*\$mode*

COUNT\_NORMAL: normal operation

COUNT\_RECURSIVE: will recursively count the array. This is useful for counting all elements of a multidimensional array.

## Return values

Returns the number of elements in the array or variable. If the array is empty, returns 0. Otherwise returns 1.

## Examples

```
<?php
$a = array(1, 2, 3);

$b1 = array(1, 2, 3);
$b2 = array(4, 5, 6);
$b = array($b1, $b2);

$cnt1 = count($a);
$cnt2 = count($b, COUNT_RECURSIVE);

echo "cnt1: $cnt1, cnt2: $cnt2\r\n"; // OUTPUT: cnt1: 3, cnt2: 8
?>
```

## See also

## Remarks

This function is identical to the PHP group's count() function.

# die()

---

void die (int/string *\$status*)

---

## Description

die() outputs a message and terminates the current script

## Parameters

*\$status*

If *\$status* is a string, this function prints the status just before exiting.

If *\$status* is an integer, the value will not be printed.

## Return values

No return value returned.

## Examples

```
<?php
die("This script will be terminated.\r\n");
while(1);
?>
```

## See also

exit()

## Remarks

This function is identical to the PHP group's die() function except when the *\$status* is an integer.

# explode()

---

string(array of) explode (string *\$delimiter*, string *\$string* [ , int *\$limit* ] )

---

## Description

explode() returns an array of strings, each of which is a substring of *string* formed by splitting it on boundaries formed by the string *delimiter*

## Parameters

*\$delimiter*

the boundary string (maximum length: PHP\_LLSTR\_BLK\_SIZE – 1)

*\$string*

the input string

*\$limit*

The returned array will contain a maximum of *limit* elements with the last element containing the rest of string

If the *\$limit* is zero, then it is treated as 1

Negative is not supported

## Return values

Returns an array of strings created by splitting the string parameter on boundaries formed by the delimiter, PHP error on error

## Examples

```
<?php
$ret = array();
$str = "abc,def,ghi,jkl,mno,pqr,stu,vwx,yz";
$delimiter = ",";

$ret = explode($delimiter, $str, 3);
echo $ret[0],"\r\n"; // OUTPUT: abc
echo $ret[1],"\r\n"; // OUTPUT: def
echo $ret[2],"\r\n"; // OUTPUT: ghi,jkl,mno,pqr,stu,vwx,yz
?>
```

## See also

None

## Remarks

This function is identical to the PHP group's explode() function but it doesn't support negative *\$limit*.

# exit()

---

void exit (int/string *\$status*)

---

## Description

exit() outputs a message and terminates the current script

## Parameters

*\$status*

If *\$status* is a string, this function prints the status just before exiting.

If *\$status* is an integer, the value will not be printed.

## Return values

No return value returned.

## Examples

```
<?php
exit("This script will be terminated.\r\n");
while(1);
?>
```

## See also

die()

## Remarks

This function is identical to the PHP group's exit() function except when the *\$status* is an integer.

# flush()

---

void flush (void)

---

## Description

flush() flushes output buffer

This attempts to push current output all the way to the browser.

## Parameters

None

## Return values

No return value returned.

## Examples

```
<?php
echo "<html>\r\n";
echo "<head>\r\n";
echo "<title> This is a sample web page</title>\r\n";

echo "<body>\r\n";

for($cnt = 0; $cnt < 100; $cnt++)
{
    echo "This is a body[$cnt]:
0123456789ABCDEFGHIJKLMNQRSTUUVWXYZabcdefghijklmnopqrstuvwxyz<br>";
    if(!($cnt%100)) flush();
}

echo "</body>\r\n";

echo "</html>\r\n";
?>
```

## See also

None

## Remarks

This function is identical to the PHP group's flush() function.

# header()

---

void header(string *\$str*)

---

## Description

header() sends raw HTTP header or Status-Line in the response message. Each HTTP header field consists of a field-name followed by a colon(":") and a field-value. The Status-Line is the first line of a Response message consisting of the protocol version followed by a numeric status code and its associated textual phrase.

## Parameters

*\$str*

If there is no field-name in the message header it adds the field-name and the field value.

If there is field-name in the message it replace the field-value with a new field-value.(Some field-values can be mofied.)

## Return values

None

## Examples

```
<?php
// This script inform to the web browser that the page move to other page temporarily.
header("HTTP/1.1 302 FOUND");
header("Location: http://www.phpoc.com");

echo "Redirecting to http://www.phpoc.com";
?>
```

## See also

None

## Remarks

header() must be called before any actual output is sent,

This function is almost same but it doesn't support \$replace and \$http\_response\_code in the PHP.

Refer to RFC2616 for more information about the HTTP response header.

# hex2bin()

---

string hex2bin ( string *\$str* )

---

## Description

hex2bin() decodes hexadecimal encoded string

## Parameters

*\$str*

string to convert

## Return values

Returns the binary representation of the given string, PHP error on failure

## Examples

```
<?php
$hex = "48656C6C6F";
$bin = hex2bin($hex);

echo $bin; // OUTPUT: Hello
?>
```

## See also

[bin2hex\(\)](#)

## Remarks

This function is identical to the PHP group's hex2bin() function.

# inet\_ntop()

string inet\_ntop ( string *\$str* )

---

## Description

inet\_ntop() converts a string IP(IPv4) address to human readable IP address

## Parameters

*\$str*

string IP address

## Return values

Returns a human readable IP address string or FALSE on invalid IP address. Otherwise PHP error

## Examples

```
<?php
$ip1 = "192.168.0.100";
hexdump($ip1); // OUTPUT: 0000 31 39 32 2e 31 36 38 2e 30 2e 31 30 30          |192.168.0.100  |
$ip2 = inet_pton($ip1);
hexdump($ip2); // OUTPUT: 0000 c0 a8 00 64                                |...d          |
$ip1 = inet_ntop($ip2);
hexdump($ip1); // OUTPUT: 0000 31 39 32 2e 31 36 38 2e 30 2e 31 30 30          |192.168.0.100  |
?>
```

## See also

inet\_pton()

## Remarks

This function is identical to the PHP group's inet\_ntop().

# inet\_pton()

---

string inet\_pton ( string *\$str* )

---

## Description

inet\_pton() converts a human readable IP(IPv4) address to string

## Parameters

*\$str*

human readable IP address

## Return values

Returns a string IP address or FALSE when the \$str is syntactically invalid, otherwise: PHP error

## Examples

```
<?php
$ip1 = "192.168.0.100";
hexdump($ip1); // OUTPUT: 0000 31 39 32 2e 31 36 38 2e 30 2e 31 30 30          |192.168.0.100 |
$ip2 = inet_pton($ip1);
hexdump($ip2); // OUTPUT: 0000 c0 a8 00 64          |...d |
$ip1 = inet_ntop($ip2);
hexdump($ip1); // OUTPUT: 0000 31 39 32 2e 31 36 38 2e 30 2e 31 30 30          |192.168.0.100 |
?>
```

## See also

inet\_ntop()

## Remarks

This function is identical to the PHP group's inet\_pton().

# ltrim()

string ltrim (string *\$str* [, string *\$charlist* ])

---

## Description

ltrim() strip whitespace (or other characters) from the beginning of a string

## Parameters

*\$str*

the input string

*\$charlist*

You can also specify the characters you want to strip, by means of the charlist parameter. Simply list all characters that you want to be stripped. With .. you can specify a range of characters

default value is "\x20\x09\x0a\x0d\x00\x0b" if this parameter is omitted

## Return values

Returns the modified string, PHP error on error

## Examples

```
<?php
hexdump(ltrim("\x0b\x00\x0d\x0a\x09\x20ABCDEFGH\x0b\x00\x0d\x0a\x09\x20"));
// OUTPUT: 0000 41 42 43 44 45 46 47 0b 00 0d 0a 09 20          |ABCDEFGH.... |

echo ltrim(".0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ0123456789.", "0123.45678"), "\r\n";
// OUTPUT: 9ABCDEFGHIJKLMN0PQRSTUVWXYZ0123456789.

echo ltrim(".0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ0123456789.", "0..3....4..8"), "\r\n";
// OUTPUT: 9ABCDEFGHIJKLMN0PQRSTUVWXYZ0123456789.

?>
```

## See also

rtrim()

## Remarks

This function is identical to the PHP group's ltrim() function.

# printf()

int printf ( string *\$format*, bool/int/string *\$arg...* )

---

## Description

printf() outputs formatted data to its standard output.

The parameters (*\$args ...*) will be inserted at percent sign(%) in the *\$format* string.

If parameter's data type is different from the percent format, a juggling error occurs.

## Parameters

format:

%% - outputs a literal percent character

%b - outputs a binary number (argument: integer)

%c - outputs a character of ASCII value (argument: integer)

%d - outputs a decimal number (argument: integer)

%u - outputs an unsigned decimal number (argument: integer)

%o - outputs an octal number (argument: integer)

%e - outputs a scientific notation (argument: float)

%E - like %s but uses uppercase letter (argument: float)

%f - outputs a floating point number (argument: float)

%F - same to %s

%g - shorter of %e and %f (argument: float)

%G - shorter of %E and %f (argument: float)

%s - outputs a string (argument:string)

%x - outputs a hexadecimal number with lowercase letters (argument: string)

%X - outputs a hexadecimal number with uppercase letters (argument: string)

If a number is inserted between the percent sign(%) and the format letter, the number means that the digit of the data.

If the number starts with a zero(0), blank will be replaced with 0.

If '+' is inserted between the percent sign(%) and the format letter or number, it outputs sign (+/-) even though the number is positive.

## Return values

Returns output string length, PHP error on error

## Examples

```
<?php
$n = 43951789;
$u = -43951789;
$c = 65;          // decimal 65, hexadecimal 0x41, 'A'

printf("%%b = '%b'\r\n", $n);          // binary representation
printf("%%c = '%c'\r\n", $c);          // print the ascii character, same as chr() function
```

```

printf("%%d = '%d'\r\n", $n);          // standard integer representation
printf("%%+d = '%+d'\r\n", $n);       // standard integer representation with sign
printf("%%u = '%u'\r\n", $n);         // unsigned integer representation of a positive integer
printf("%%u = '%u'\r\n", $u);         // unsigned integer representation of a negative integer
printf("%%o = '%o'\r\n", $n);         // octal representation
printf("%%e = '%e'\r\n", (float)$n);  // scientific notation
printf("%%E = '%E'\r\n", (float)$n);  // scientific notation with a upper case E
printf("%%f = '%f'\r\n", (float)$n);  // floating point notation
printf("%%F = '%F'\r\n", (float)$n);  // floating point notation
printf("%%g = '%g'\r\n", (float)$n);  // shorter form
printf("%%G = '%G'\r\n", (float)$n);  // shorter form
printf("%%s = '%s'\r\n", (string)$n); // string representation
printf("%%x = '%x'\r\n", $n);         // hexadecimal representation (lower-case)
printf("%%X = '%X'\r\n", $n);         // hexadecimal representation (upper-case)

$msg = 'sollae';

printf("[%s]\r\n",      $msg); // standard string output
printf("[%10s]\r\n",   $msg); // right-justification with spaces
printf("[% -10s]\r\n", $msg); // left-justification with spaces
printf("[%010s]\r\n",  $msg); // zero-padding works on strings too
$ret = printf("[% '@10s]\r\n", $msg); // use the custom padding character '@'
echo "=>$ret bytes\r\n";

?>

```

## See also

[sprintf\(\)](#)

## Remarks

This function is identical to the PHP group's `printf()` function but PHPoC doesn't support floating-point related parameters.

# rtrim()

string rtrim (string *\$str* [, string *\$charlist* ])

---

## Description

rtrim() strips whitespace (or other characters) from the end of a string

## Parameters

*\$str*

the input string

*\$charlist*

You can also specify the characters you want to strip, by means of the charlist parameter. Simply list all characters that you want to be stripped. With .. you can specify a range of characters

Default value is "\x20\x09\x0a\x0d\x00\x0b" if this parameter is omitted

## Return values

Returns the modified string, PHP error on error

## Examples

```
<?php
hexdump(rtrim("\x0b\x00\x0d\x0a\x09\x20ABCDEFG\x0b\x00\x0d\x0a\x09\x20"));
//OUTPUT: 0000 0b 00 0d 0a 09 20 41 42 43 44 45 46 47          |..... ABCDEFG |

echo rtrim(".0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ9876543210.", "012.34567"), "\r\n";
//OUTPUT: .0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ98

echo rtrim(".0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ9876543210.", "0..2...3..7"), "\r\n";
//OUTPUT: .0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ98
?>
```

## See also

ltrim()

## Remarks

This function is identical to the PHP group's rtrim() function.

# set\_time\_limit()

---

void set\_time\_limit ( int *\$seconds* )

---

## Description

set\_time\_limit() limits maximum execution time of the PHP script.

The default limit time of the HTTP session is 4 seconds.

And the default limit time of the non-PHP script is infinite(0).

## Parameters

*\$seconds*

the maximum PHP execution time

If set to zero, no time limit is imposed.

## Return values

Returns none, PHP error on failure

## Examples

```
<?php
set_time_limit(5); // limits PHP execution time to 5 seconds
while(1)
{
    sleep(1);
    echo ".";
}
// After testing this code, you should run "set_time_limit(0);" to avoid terminating another codes.
?>
```

## See also

None

## Remarks

This function is identical to the PHP Groups's PHP.

Do not use it with infinite loop code.

# sleep()

---

int sleep ( int *\$seconds* )

---

## Description

sleep() delays the program execution for the specified number of seconds

## Parameters

*\$seconds*

sleeping time in seconds

If this parameter is negative the function is same to when this parameter is 0

## Return values

Returns 0, PHP error on error

## Examples

```
<?php
$pid_st0 = pid_open("/mmap/st0");
pid_ioctl($pid_st0, "set mode free");

$sleeping_time = 3;

echo "sleeping for $sleeping_time seconds...\r\n"; // OUTPUT: sleeping for 3 seconds...
pid_ioctl($pid_st0, "start");
sleep($sleeping_time);

$tick = pid_ioctl($pid_st0, "get count");
printf("slept time: %d milli seconds\r\n", $tick); // OUTPUT: slept time: 3000 milli seconds
?>
```

## See also

usleep()

## Remarks

This function is identical to the PHP group's sleep() function except error handling.

If the specified number of seconds is negative, the PHP group's sleep() function generates a E\_WARNING.

# sprintf()

string sprintf ( string *\$format*, bool/int/string *\$args...* )

---

## Description

sprintf() returns formatted string

The parameters (*\$args ...*) will be inserted at percent sign(%) in the *\$format* string.

If parameter's data type is different from the percent format, a juggling error occurs.

## Parameters

format:

%% - writes a literal percent character

%b - writes a binary number (argument: integer)

%c - writes a character of ASCII value (argument: integer)

%d - writes a decimal number (argument: integer)

%u - writes an unsigned decimal number (argument: integer)

%o - writes an octal number (argument: integer)

%e - writes a scientific notation (argument: float)

%E - like %s but uses uppercase letter (argument: float)

%f - writes a floating point number (argument: float)

%F - same to %s

%g - shorter of %e and %f (argument: float)

%G - shorter of %E and %f (argument: float)

%s - writes a string (argument:string)

%x - writes a hexadecimal number with lowercase letters (argument: string)

%X - writes a hexadecimal number with uppercase letters (argument: string)

If a number is inserted between the percent sign(%) and the format letter, the number means that the digit of the data.

If the number starts with a zero(0), blank will be replaced with 0.

If '+' is inserted between the percent sign(%) and the format letter or number, it outputs sign (+/-) even though the number is positive.

## Return values

Returns formatted string, PHP error on error

## Examples

```
<?php
$n = 43951789;
$u = -43951789;
$c = 65;          // decimal 65, hexadecimal 0x41, 'A'

echo sprintf("%%b = '%b'\r\n", $n);          // binary representation
echo sprintf("%%c = '%c'\r\n", $c);          // print the ascii character, same as chr() function
```

```

echo sprintf("%d = '%d'\r\n", $n); // standard integer representation
echo sprintf("%+d = '%+d'\r\n", $n); // standard integer representation with sign
echo sprintf("%u = '%u'\r\n", $n); // unsigned integer representation of a positive integer
echo sprintf("%u = '%u'\r\n", $n); // unsigned integer representation of a negative integer
echo sprintf("%o = '%o'\r\n", $n); // octal representation
echo sprintf("%e = '%e'\r\n", (float)$n); // scientific notation
echo sprintf("%E = '%E'\r\n", (float)$n); // scientific notation with a upper case E
echo sprintf("%f = '%f'\r\n", (float)$n); // floating point notation
echo sprintf("%F = '%F'\r\n", (float)$n); // floating point notation
echo sprintf("%g = '%g'\r\n", (float)$n); // shorter form
echo sprintf("%G = '%G'\r\n", (float)$n); // shorter form
echo sprintf("%s = '%s'\r\n", (string)$n); // string representation
echo sprintf("%x = '%x'\r\n", $n); // hexadecimal representation (lower-case)
echo sprintf("%X = '%X'\r\n", $n); // hexadecimal representation (upper-case)

$msg = 'sollae';

echo sprintf("[%s]\r\n", $msg); // standard string output
echo sprintf("[%10s]\r\n", $msg); // right-justification with spaces
echo sprintf("[% -10s]\r\n", $msg); // left-justification with spaces
echo sprintf("[%010s]\r\n", $msg); // zero-padding works on strings too
?>

```

## See also

[printf\(\)](#)

## Remarks

This function is identical to the PHP group's [sprintf\(\)](#) function.

# str\_replace()

string str\_replace ( string *\$search*, string *\$replace*, string *\$subject* [ , int *&\$count* ] )

## Description

str\_replace() returns string with all occurrences of \$search in \$subject replaced with the given \$replace value

## Parameters

*\$search*

The value being searched for (maximum length: PHP\_LLSTR\_BLK\_SIZE - 1)

*\$replace*

The replacement value that replaces found \$search values (maximum length: PHP\_LLSTR\_BLK\_SIZE - 1)

*\$subject*

The string searched and replaced on

*\$count*

If passed, this will be set to the number of replacement performed

## Return values

Returns a string with the replaced values, PHP error on error

## Examples

```
<?php
$search = "Mark";
$replace = "John";
$subject = "His name is Mark. Mark is handsome.";
$count = 0;
$ret = "";

$ret = str_replace($search, $replace, $subject, $count);
echo "count: $count, ret: $ret\r\n"; // OUTPUT: count: 2, ret: His name is John. John is handsome.
?>
```

## See also

strpos() / substr() / substr\_replace()

## Remarks

This function is identical to the PHP group's str\_replaced() function but doesn't support array.

# strlen()

---

int strlen ( string *\$str*)

---

## Description

strlen() calculates the length of the string *\$str*, excluding the terminating null bytes(0)

## Parameters

*\$str*

string to calculate

## Return values

Returns number of bytes of the *\$str*, PHP error on error

## Examples

```
<?php
$str = "Hello PHPoC!";
$ret = strlen($str);
printf("str = %s, ret = %d\r\n", $str, $ret); // OUTPUT: str = Hello PHPoC!, ret = 12
?>
```

## See also

None

## Remarks

This function is identical to the PHP group's strlen() function.

# strpos()

---

int strpos ( string *\$haystack*, string *\$needle* [ , int *\$offset*=0 ] )

---

## Description

strpos() finds the numeric position of the first occurrence of needle in the haystack string

## Parameters

*\$str*

string to search in

*\$needle*

string to search

*\$offset*

position to start to search

## Return values

Returns the position of where the needle exists relative to the beginning of the haystack string(independent of offset).

The string positions start at 0, and not 1.

Returns FALSE if needle was not found.

Otherwise PHP error

## Examples

```
<?php
$haystack = "Hello PHPoC World!";
$needle1 = "PHP";
$needle2 = "H";

$pos = strpos($haystack, $needle1);
echo "needle1: $needle1 at position $pos\r\n"; // OUTPUT: needle1: PHP at position 6
$pos = strpos($haystack, $needle2, 2);
echo "needle2: $needle2 at position $pos\r\n"; // OUTPUT: needle2: H at position 7
?>
```

## See also

[substr\(\)](#) / [str\\_replace\(\)](#) / [substr\\_replace\(\)](#)

## Remarks

This function is identical to the PHP group's strpos() function.

# strtolower()

---

string strtolower (string *\$str* )

---

## Description

strtolower() converts the string *\$str* to lower case

## Parameters

*\$str*

string to convert to lower case

## Return values

Returns lower case string of *\$str*, PHP error on error

## Examples

```
<?php
$str_org = "Hello PHPoC!";
$str_lower = strtolower($str_org);
printf("str_org = %s, str_lower = %s\r\n", $str_org, $str_lower);
// OUTPUT: str_org = Hello PHPoC!, str_lower = hello phpoc!
?>
```

## See also

strtoupper()

## Remarks

This function is identical to the PHP group's strtolower() function.

# strtoupper()

---

string strtoupper ( string *\$str* )

---

## Description

strtoupper() converts the string *\$str* to upper case

## Parameters

*\$str*

string to convert to upper case

## Return values

Returns upper case string of *\$str*, PHP error on error

## Examples

```
<?php
$str_org = "Hello PHPoC!";
$str_upper = strtoupper($str_org);
printf("str_org = %s, str_upper = %s\r\n", $str_org, $str_upper);
// OUTPUT: str_org = Hello PHPoC!, str_upper = HELLO PHPOC!
?>
```

## See also

strtolower()

## Remarks

This function is identical to the PHP group's strtoupper() function.

# substr()

---

string substr ( string *\$string*, int *\$start* [ , int *\$length* ] )

---

## Description

substr() returns the portion of string specified by the start and length parameters

## Parameters

*\$string*

the input string

*\$start*

If *\$start* is non-negative, the returned string will start at the start'th position in string

If *\$start* is negative, the returned string will start at the start'th character from the end of string

If *\$string* is less than or equal to *\$start* characters long, FALSE will be returned

*\$length*

If *\$length* is positive, the string returned will contain at most length characters beginning from start

If *\$length* is negative, the absolute value will be omitted from the end of string (after the start position has been calculated when a start is negative)

If *\$length* is given and is 0, an empty string will be returned

If *\$length* is omitted, the substring starting from *\$start* until the end of string will be returned

## Return values

Returns the extracted part of string, PHP error on error

## Examples

```
<?php
$str = "abcdef";
$ret = substr($str, 2); echo "$ret\r\n"; // OUTPUT: cdef
$ret = substr($str, -1); echo "$ret\r\n"; // OUTPUT: f
$ret = substr($str, -2); echo "$ret\r\n"; // OUTPUT: ef
$ret = substr($str, 2, 3); echo "$ret\r\n"; // OUTPUT: cde
$ret = substr($str, -3, 1); echo "$ret\r\n"; // OUTPUT: d
$ret = substr($str, 0, -1); echo "$ret\r\n"; // OUTPUT: abcde
$ret = substr($str, 2, -1); echo "$ret\r\n"; // OUTPUT: cde
$ret = substr($str, 4, -4); echo "$ret\r\n"; // OUTPUT: 0
$ret = substr($str, -3, -1); echo "$ret\r\n"; // OUTPUT: de

$ret = substr($str, 9);
if($ret === FALSE) echo "returned FALSE\r\n"; // OUTPUT: returned FALSE
?>
```

## See also

`strpos() / str_replace() / substr_replace()`

### **Remarks**

This function is identical to the PHP group's `substr()` function.

# substr\_replace()

---

string substr\_replace ( string *\$string*, string *\$replace*, int *\$start* [ , int *\$length* ] )

---

## Description

substr\_replace() replaces a copy of string delimited by the start and (optionally) length parameter with the string given in replacement

## Parameters

*\$string*

the input string

*\$replace*

the replacement string

*\$start*

If *\$start* is positive, the replacing will begin at the *\$start*'th offset.

If *\$start* is negative, the replacing will begin at the *\$start*'th character from the end of *\$string*.

*\$length*

If is positive, it represent the length of the portion of string which is to be replaced.

If it is negative, it represent the number of characters from the end of *\$string* at which to stop replacing.

If it is omitted, then it will default to strlen(*\$string*); i.e. end the replacing at the end of *\$string*. If *\$length* is zero then this function will have effect of inserting *\$replacement* into *\$string* at the given *\$start* offset.

## Return values

Returns the result string

## Examples

```
<?php
$str = "Hello PHPoC!";
$ret = substr_replace($str, "Hi", 0, 5); echo "$ret\r\n"; // OUTPUT: Hi PHPoC!
$ret = substr_replace($str, "Hi", -12, 5); echo "$ret\r\n"; // OUTPUT: Hi PHPoC!
$ret = substr_replace($str, "Hi", 0, -7); echo "$ret\r\n"; // OUTPUT: Hi PHPoC!
$ret = substr_replace($str, "Hi", 6); echo "$ret\r\n"; // OUTPUT: Hello Hi

$str = "";
$ret = substr_replace($str, "Hi", 6); echo "$ret\r\n"; // OUTPUT: Hi
?>
```

## See also

strpos() / str\_replace() / substr()

## Remarks

This function is identical to the PHP group's `substr()` function but it doesn't support array.

# system()

---

string system ( string *\$command* [ , string *\$arg*, ... ] )

---

## Description

system() executes an external function and display the output

## Parameters

*\$command*

system command (maximum length: PHP\_LLSTR\_BLK\_SIZE - 1)

*\$arg*

extended argument (maximum length: PHP\_LLSTR\_BLK\_SIZE - 1)

## Return values

Returns output string, PHP error on error

## Examples

```
<?php
$str = system("uname -svip");
echo "$str\r\n";
?>
```

## See also

None

## Remarks

Refer to the system() function other manual for detailed information.

This system() function supports another function format.

The words starting with '%' followed by a number in the string are replaced by parameters.

For example, the following two functions are exactly same.

```
system("php -d 500 main.php");

$delay = "500";
$filename = "main.php";
system("php -d %1 %2", $delay, $filename);
```

# usleep()

---

int usleep ( int *\$micro\_seconds* )

---

## Description

usleep() delays program execution for specified number of micro-seconds

## Parameters

*\$micro\_seconds*

sleeping time in micro-seconds (1 second = 1,000,000 micro-second)

If this parameter is negative the function is same to when this parameter is 0

The sleep time is inaccurate if this value is under 1,000 (1ms)

## Return values

Returns 0, PHP error on error

## Examples

```
<?php
$pid_st0 = pid_open("/mmap/st0");
pid_ioctl($pid_st0, "set mode free");

$sleeping_time = 1000000;

echo "sleeping for $sleeping_time micro-seconds...\r\n"; // OUTPUT: sleeping for 1000000 micro-seconds...

pid_ioctl($pid_st0, "start"); // timer start
usleep($sleeping_time);

$tick = pid_ioctl($pid_st0, "get count");
printf("slept time: %d ms\r\n", $tick); // OUTPUT: slept time: 1000 ms
?>
```

## See also

sleep()

## Remarks

This function is identical to the PHP group's usleep() function except error handling.

If the specified number of seconds is negative, the PHP group's usleep() function generates a E\_WARNING.

# abs()

int/float abs ( int/float *\$number* )

---

## Description

abs() returns the absolute value of \$number

## Parameters

*\$number*

The numeric value to process

## Return values

Returns absolute value of \$number. If the argument is float it returns float and if the argument is integer it returns integer

## Examples

```
<?php
$ret1 = abs(-10.11);
$ret2 = abs(10.11);
$ret3 = abs(-10);
$ret4 = abs(10);

echo "ret = $ret1\r\n"; // OUTPUT: ret = 10.11
echo "ret = $ret2\r\n"; // OUTPUT: ret = 10.11
echo "ret = $ret3\r\n"; // OUTPUT: ret = 10
echo "ret = $ret4\r\n"; // OUTPUT: ret = 10
?>
```

## See also

None

## Remarks

This function is identical to the PHP group's abs() function but it supports only integer and float type.

# acos()

---

float acos ( float *\$arg* )

---

## Description

acos() returns the arc cosine of \$arg in radians. acos() is the complementary function of the cos().

## Parameters

*\$arg*

The argument to process between -1 and 1

## Return values

Returns arc cosine of \$arg in radians

## Examples

```
<?php
$rad = 1.0;

$val1 = cos($rad);
$val2 = acos($val1);

echo "cos($rad) = $val1\r\n"; // OUTPUT: cos(1) = 0.54030230586814
echo "acos($val1) = $val2\r\n"; // OUTPUT: acos(0.54030230586814) = 1
?>
```

```
<?php
// 1.1 is invalid argument for the acos() so it returns NAN (Not A Number)
$val = acos(1.1);
echo "acos(1.1) = $val\r\n"; // OUTPUT: acos(1.1) = NAN
?>
```

## See also

cos()

## Remarks

This function is identical to the PHP group's acos() function.

# asin()

---

float asin ( float *\$arg* )

---

## Description

asin() returns the arc sine of \$arg in radians. asin() is the complementary function of the sin().

## Parameters

*\$arg*

The argument to process between -1 and 1

## Return values

Returns arc sine of \$arg in radians

## Examples

```
<?php
$rad = 1.0;

$val1 = sin($rad);
$val2 = asin($val1);

echo "sin($rad) = $val1\r\n"; // OUTPUT: sin(1) = 0.8414709848079
echo "asin($val1) = $val2\r\n"; // OUTPUT: asin(0.8414709848079) = 1
?>
```

```
<?php
// 1.1 is invalid argument for the asin() so it returns NAN (Not A Number)
$val = asin(1.1);
echo "asin(1.1) = $val\r\n"; // OUTPUT: asin(1.1) = NAN
?>
```

## See also

sin()

## Remarks

This function is identical to the PHP group's asin() function.

# atan()

float atan ( float *\$arg* )

---

## Description

atan() returns the arc tangent of *\$arg* in radians. atan() is the complementary function of the tan().

## Parameters

*\$arg*

The argument to process

## Return values

Returns arc tangent of *\$arg* in radians

## Examples

```
<?php
$y = 1.0;$x = 2.0;
$y3 = -1.0;$x3 = -2.0;

$rad = $y/$x;
$val1 = atan($rad);
$val2 = atan2($y, $x);
$val3 = atan2($y3, $x3);
$val4 = tan($val1);

echo "atan($rad) = $val1\r\n"; // OUTPUT: atan(0.5) = 0.46364760900081
echo "atan2($y, $x) = $val2\r\n"; // OUTPUT: atan2(1, 2) = 0.46364760900081
echo "atan2($y3, $x3) = $val3\r\n"; // OUTPUT: atan2(-1, -2) = -2.677945044589
echo "tan($val1) = $val4\r\n"; // OUTPUT: tan(0.46364760900081) = 0.5
?>
```

## See also

atan2() / tan()

## Remarks

This function is identical to the PHP group's atan() function.

# atan2()

float atan2 ( float \$y, float \$x )

---

## Description

This function calculates the arc tangent of the two variables \$x and \$y. It is similar to calculating the arc tangent of \$y/\$x, except that the signs of both arguments are used to determine the quadrant of the result.

## Parameters

\$y

Dividend parameter

\$x

Divisor parameter

## Return values

Returns arc tangent of \$y/\$x in radians

## Examples

```
<?php
$y = 1.0;$x = 2.0;
$y3 = -1.0;$x3 = -2.0;

$rad = $y/$x;
$val1 = atan($rad);
$val2 = atan2($y, $x);
$val3 = atan2($y3, $x3);
$val4 = tan($val1);

echo "atan($rad) = $val1\r\n"; // OUTPUT: atan(0.5) = 0.46364760900081
echo "atan2($y, $x) = $val2\r\n"; // OUTPUT: atan2(1, 2) = 0.46364760900081
echo "atan2($y3, $x3) = $val3\r\n"; // OUTPUT: atan2(-1, -2) = -2.677945044589
echo "tan($val1) = $val4\r\n"; // OUTPUT: tan(0.46364760900081) = 0.5
?>
```

## See also

[atan\(\)](#) / [tan\(\)](#)

## Remarks

This function is identical to the PHP group's atan2() function.

# ceil()

---

float ceil ( float *\$value* )

---

## Description

ceil() rounds fractions up

## Parameters

*\$value*

The numeric value to round

## Return values

Returns the next highest integer value by rounding up value if necessary

## Examples

```
<?php
$val = 3.14;

$ret = ceil($val);
echo "ret = $ret\r\n"; // OUTPUT: ret = 4
?>
```

## See also

floor() / round()

## Remarks

This function is identical to the PHP group's ceil() function.

# cos()

---

float cos ( float *\$arg* )

---

## Description

cos() returns the cosine of \$arg. The \$arg is in radians.

## Parameters

*\$arg*

An angle in radians

## Return values

Returns cosine of \$arg

## Examples

```
<?php
$rad = 1.0;

$val1 = cos($rad);
$val2 = acos($val1);

echo "cos($rad) = $val1\r\n"; // OUTPUT: cos(1) = 0.54030230586814
echo "acos($val1) = $val2\r\n"; // OUTPUT: acos(0.54030230586814) = 1
?>
```

## See also

acos()

## Remarks

This function is identical to the PHP group's cos() function.

# deg2rad()

---

float deg2rad ( float *\$number* )

---

## Description

deg2rad() converts the number in degrees to the radian equivalent

## Parameters

*\$number*

the angular value in degrees

## Return values

Returns the radian equivalent of \$number

## Examples

```
<?php
$deg = 45.0;

$rad = deg2rad($deg);
$deg = rad2deg($rad);

echo "degree = $deg\r\n"; // OUTPUT: degree = 45
echo "radian = $rad\r\n"; // OUTPUT: radian = 0.78539816339745
?>
```

## See also

rad2deg()

## Remarks

This function is identical to the PHP group's deg2rad() function.

# exp()

---

float exp ( float *\$arg* )

---

## Description

exp() returns e raised to the power of \$arg

('e' is the base of the natural system of logarithms, or approximately 2.718281828459)

## Parameters

*\$arg*

the argument to process

## Return values

Returns 'e' raised to the power of \$arg

## Examples

```
<?php
$f1 = exp(1);
$f2 = exp(2);

echo "f1 = $f1\r\n"; // OUTPUT: f1 = 2.718281828459
echo "f2 = $f2\r\n"; // OUTPUT: f2 = 7.3890560989307
?>
```

## See also

log()

## Remarks

This function is identical to the PHP group's exp() function.

# floor()

---

float floor ( float *\$value* )

---

## Description

floor() rounds fractions down

## Parameters

*\$value*

The numeric value to round

## Return values

Returns the next lowest integer value by rounding down value if necessary

## Examples

```
<?php
$val = 123.54;

$ret = floor($val);
echo "ret = $ret\r\n"; // OUTPUT: ret = 123
?>
```

## See also

ceil() / round()

## Remarks

This function is identical to the PHP group's floor() function.

# is\_finite()

---

bool is\_finite ( float *\$val* )

---

## Description

is\_finite() checks whether *\$val* is a legal finite on this platform

## Parameters

*\$val*

The value to check

## Return values

TRUE if *\$val* is a legal finite number within the allowed range for a PHPoC float on this platform, else FALSE

## See also

is\_infinite() / is\_nan()

## Remarks

This function is identical to the PHP group's is\_finite().

# is\_infinite()

---

bool is\_infinite ( float *\$val* )

---

## Description

is\_infinite() checks whether \$val is infinite on this platform

## Parameters

*\$val*

The value to check

## Return values

TRUE if \$val is infinite, like any value too big to fit into a float in this platform

## See also

is\_finite() / is\_nan()

## Remarks

This function is identical to the PHP group's is\_infinite().

# is\_nan()

---

bool is\_nan ( float *\$val* )

---

## Description

is\_nan() checks whether \$val is 'not a number', like the result of acos(1.01)

## Parameters

*\$val*

The value to check

## Return values

TRUE if \$val is 'not a number'; else FALSE

## See also

is\_finite() / is\_infinite()

## Remarks

This function is identical to the PHP group's is\_nan().

# log()

---

float log ( float *\$arg* [, float *\$base* = M\_E] )

---

## Description

If the optional *\$base* parameter is specified, log() returns  $\log_{\$base}\$arg$ , otherwise log() returns the natural logarithm of *\$arg*

## Parameters

*\$arg*

the value to calculate the logarithm for

*\$base*

the optional logarithmic base to use (default is 'e')

## Return values

Returns logarithm of *\$arg* to *\$base*, if given, or the natural logarithm

## Examples

```
<?php
$f1 = log(2); // logarithmic base: 'e'
$f2 = log(2, 10);

echo "f1 = $f1\r\n"; // OUTPUT: f1 = 0.69314718055995
echo "f2 = $f2\r\n"; // OUTPUT: f2 = 0.30102999566398
?>
```

## See also

exp()

## Remarks

This function is identical to the PHP group's log() function.

# pow()

---

float pow ( float *\$base*, float *\$exp* )

---

## Description

Returns *\$base* raised to the power of *\$exp*

## Parameters

*\$base*

the base to use

*\$exp*

the exponent

## Return values

Returns *\$base* raised to the power of *\$exp* ( $\$base^{\$exp}$ )

## Examples

```
<?php
$base = 2.0;
$exp = 3.0;
$ret = pow($base, $exp);

echo "ret = $ret\r\n"; // OUTPUT: ret = 8
?>
```

## See also

[exp\(\)](#)

## Remarks

This function is identical to the PHP group's [pow\(\)](#) function, but it supports only float.

# rad2deg()

---

float rad2deg ( float *\$number* )

---

## Description

rad2deg() converts \$numbers from radian to degrees

## Parameters

*\$number*

the radian value

## Return values

Returns the equivalent of \$number in degrees

## Examples

```
<?php
$deg = 45.0;

$rad = deg2rad($deg);
$deg = rad2deg($rad);

echo "degree = $deg\r\n"; // OUTPUT: degree = 45
echo "radian = $rad\r\n"; // OUTPUT: radian = 0.78539816339745
?>
```

## See also

deg2rad()

## Remarks

This function is identical to the PHP group's rad2deg() function.

# rand()

---

int rand ( [ int *\$min*, int *\$max* ] )

---

## Description

rand() generates a random number between *\$min* and *\$max*

## Parameters

*\$min*

minimum number from 0 to 9223372036854775807

*\$max*

maximum number from 0 to 9223372036854775807

## Return values

Returns a random number between *\$min* and *\$max*, PHP error on error

## Examples

```
<?php
echo (string)rand() . "\r\n";          // OUTPUT: random number (0 ~ 9223372036854775807)
echo (string)rand(10) . "\r\n";      // OUTPUT: random number (10 ~ 9223372036854775807)
echo (string)rand(5, 15) . "\r\n";   // OUTPUT: random number (5 ~ 15)
?>
```

## See also

None

## Remarks

This function is identical to the PHP group's rand() function except output value range and number of input parameter.

# round()

---

float round ( float *\$val* [, int *\$precision* = 0] )

---

## Description

round() rounds a float

## Parameters

*\$val*

The value to round

*\$precision*

The optional number of decimal digits to round to

## Return values

Returns the rounded value of *\$val* to specified precision (number of digits after the decimal point). The precision can be negative or zero.

## Examples

```
<?php
$val = 123.141592;

$ret = round($val, 2);
echo "ret = $ret\r\n"; // OUTPUT: ret = 123.14

$ret = round($val);
echo "ret = $ret\r\n"; // OUTPUT: ret = 123

$ret = round($val, 4);
echo "ret = $ret\r\n"; // OUTPUT: ret = 123.1416

$ret = round($val, -2);
echo "ret = $ret\r\n"; // OUTPUT: ret = 100
?>
```

## See also

ceil() / floor()

## Remarks

This function is identical to the PHP group's round() function but it doesn't support the third parameter.

# sin()

---

float sin ( float *\$arg* )

---

## Description

sin() returns the sine of \$arg. The \$arg is in radians.

## Parameters

*\$arg*

A value in radians

## Return values

Returns sine of \$arg.

## Examples

```
<?php
$rad = 1.0;

$val1 = sin($rad);
$val2 = asin($val1);

echo "sin($rad) = $val1\r\n"; // OUTPUT: sin(1) = 0.8414709848079
echo "asin($val1) = $val2\r\n"; // OUTPUT: asin(0.8414709848079) = 1
?>
```

## See also

asin()

## Remarks

This function is identical to the PHP group's sin() function.

# sqrt()

---

float sqrt ( float *\$arg* )

---

## Description

Returns the square root of *\$arg*

## Parameters

*\$arg*

the argument to process

## Return values

Returns the square root of *\$arg* or the special value NAN for negative numbers

## Examples

```
<?php
$arg1 = 2.0;
$ret1 = sqrt($arg1);
$arg2 = -2.0;
$ret2 = sqrt($arg2);

echo "ret1 = $ret1\r\n"; // OUTPUT: ret1 = 1.4142135623731
echo "ret2 = $ret2\r\n"; // OUTPUT: ret2 = NAN
?>
```

## See also

[pow\(\)](#)

## Remarks

This function is identical to the PHP group's [sqrt\(\)](#) function.

# tan()

---

float tan ( float *\$arg* )

---

## Description

atan() returns the tangent of \$arg. The \$arg is in radians.

## Parameters

*\$arg*

The argument to process in radians

## Return values

Returns tangent of \$arg

## Examples

```
<?php
$y = 1.0;$x = 2.0;
$y3 = -1.0;$x3 = -2.0;

$rad = $y/$x;
$val1 = atan($rad);
$val2 = atan2($y, $x);
$val3 = atan2($y3, $x3);
$val4 = tan($val1);

echo "atan($rad) = $val1\r\n"; // OUTPUT: atan(0.5) = 0.46364760900081
echo "atan2($y, $x) = $val2\r\n"; // OUTPUT: atan2(1, 2) = 0.46364760900081
echo "atan2($y3, $x3) = $val3\r\n"; // OUTPUT: atan2(-1, -2) = -2.677945044589
echo "tan($val1) = $val4\r\n"; // OUTPUT: tan(0.46364760900081) = 0.5
?>
```

## See also

atan() / atan2()

## Remarks

This function is identical to the PHP group's tan() function.

## Revision History

Date	Revision	Contents	Name
2014.Sep.25	1.0	Initial Release	Matthew