

Overview

Internal functions are functions that can be used without requiring any external library files. They are part of the PHPoC core. They can be divided into two main categories: PHP-based function and PHPoC function. Most of PHP-based functions are identical to standard PHP function. PHPoC function are additional functions which interact directly with hardware and peripherals (e.g. I/O, UART, I2C, SPI...). This document describes the meaning, syntax, parameter and return value of each internal function.

- `_GET()`
- `_POST()`
- `_SERVER()`
- `abs()`
- `acos()`
- `asin()`
- `atan()`
- `atan2()`
- `bin2int()`
- `ceil()`
- `cos()`
- `count()`
- `date()`
- `deg2rad()`
- `die()`
- `echo()`
- `error_log()`
- `exit()`
- `exp()`
- `explode()`
- `floor()`
- `flush()`
- `hash()`
- `hash_final()`
- `hash_hmac()`
- `hash_init()`
- `hash_pbkdf2()`
- `hash_update()`
- `header()`
- `hex2bin()`
- `hexdump()`
- `inet_ntop()`
- `inet_pton()`
- `int2bin()`
- `is_array()`
- `is_bool()`
- `is_finite()`
- `is_float()`
- `is_infinite()`
- `is_int()`

- `is_nan()`
- `is_string()`
- `log()`
- `ltrim()`
- `mktime()`
- `pid_bind()`
- `pid_close()`
- `pid_connect()`
- `pid_ioctl()`
- `pid_listen()`
- `pid_lseek()`
- `pid_open()`
- `pid_read()`
- `pid_recv()`
- `pid_recvfrom()`
- `pid_send()`
- `pid_sendto()`
- `pid_write()`
- `pow()`
- `printf()`
- `rad2deg()`
- `rand()`
- `round()`
- `rtrim()`
- `set_time_limit()`
- `sin()`
- `sleep()`
- `sprintf()`
- `sqrt()`
- `str_repeat()`
- `str_replace()`
- `strlen()`
- `strpos()`
- `strtolower()`
- `strtoupper()`
- `substr()`
- `substr_replace()`
- `system()`
- `tan()`
- `time()`
- `trim()`
- `usleep()`

bin2int()

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

Description

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

※ available F/W version : all

Parameters

- \$data - the string to be converted
- \$offset - the position of offset byte
- \$len - the string's length from the \$offset to be converted
- \$swap - Swap operation performed when this field is true (default: false)

Return values

Returns the converted integer, PHP error on error

Example

```
<?php
$buf = "";
$buf = int2bin(0x11223344, 4);
$buf .= int2bin(0x55667788, 4);
hexdump($buf);
// OUTPUT: 0000 44 33 22 11 88 77 66 55 |D3"..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++)
        {
            // getting 1 byte data from the $rbuf at position $i
            $data = bin2int($rbuf, $i, 1);
            // changing $data to 0x2e if it is 0x20
            if($data == 0x20) $data = 0x2e;
            // append $data to the $wbuf
            $wbuf .= int2bin($data, 1);
        }
        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=256 ] )
```

Description

hexdump() dumps a binary data stream into a human-friendly format to the standard output.

※ available F/W version : all

Parameters

- \$str - the string to be output
- \$start - the starting position to be output. If this value is negative it starts at the \$start'th character from the end of string.
- \$len - the string's length to be output and the maximum value is 256

Return values

Returns the length of output data, PHP error on error

Example

```
<?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 an integer to a string (little endian data format) with optional swap function

※ available F/W version : all

Parameters

- \$val - the integer value to be converted
- \$len - string length
- \$swap - Swap operation performed when this field is true (default: false)

Return values

Returns the 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.    |
?>
```

Use the following code if the length of input data exceeds the maximum length:

```
<?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++)
        {
            // getting 1 byte data from the $rbuf at position $i
            $data = bin2int($rbuf, $i, 1);
            // changing $data to 0x2e if it is 0x20
            if($data == 0x20) $data = 0x2e;
            // append $data to the $wbuf
            $wbuf .= int2bin($data, 1);
        }
    }
    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.

※ available F/W version : all

Parameters

- \$pid - the network device's PID to bind
- \$addr - This field should be empty string ("")
- \$port - the port number to bind

Return values

Returns 0 on success, PHP error on error.

Example

```
<?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

※ available F/W version : all

Parameters

- \$pid - the valid device PID created with [pid_open\(\)](#)

Return values

Returns 0 on success, PHP error on error

Example

```
<?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 establish a TCP connection to the specified IP address and port number. This function is non-blocking so it returns immediately.

※ available F/W version : all

Parameters

- \$pid - the valid device PID created with [pid_open\(\)](#)
- \$addr - the destination IP address
- \$port - the destination port number

Return values

Returns 0 on success, PHP error on error

Example

```
<?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_read\(\)](#) / [pid_write\(\)](#) / [pid_ioctl\(\)](#) / [pid_recv\(\)](#) / [pid_send\(\)](#) / [pid_listen\(\)](#)

Remarks

None

pid_ioctl()

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

Description

pid_ioctl() manipulates the device's parameters

※ available F/W version : all

Parameters

- \$pid - th device's PID to be manipulated
- \$req - the device's parameters (maximum length: PHP_LLSTR_BLK_SIZE - 1)
- \$opt - the optional argument for the \$req

Return values

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

Example

```
<?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.

The `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 the same.

```
pid_ioctl($pid, "set baud 115200");

$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.

※ available F/W version : all

Parameters

- \$pid - the valid TCP device PID created with [pid_open\(\)](#)
- \$backlog - the maximum number of the incoming connections (default: 1). Only 1 is valid for this field

Return values

Returns 0 on success, PHP error on error

Example

```
<?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...WrWn";
do
{
    $state = pid_ioctl($pid, "get state");
}while(($state != TCP_CONNECTED) && ($state != TCP_CLOSED));
if($state == TCP_CONNECTED) echo "TCP connectedWrWn";
else if($state == TCP_CLOSED) echo "TCP connection failedWrWn";

while(1);
?>
```

See also

[pid_open\(\)](#) / [pid_close\(\)](#) / [pid_ioctl\(\)](#) / [pid_read\(\)](#) / [pid_write\(\)](#) / [pid_recv\(\)](#) / [pid_send\(\)](#) / [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

※ available F/W version : all

Parameters

- \$pid - the 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

Example

```
<?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_ioctl\(\)](#) / [pid_read\(\)](#) / [pid_write\(\)](#) /

Remarks

None

pid_open()

```
int pid_open ( string $path [ , int $flags ] )
```

Description

pid_open() opens the specified file/port/peripheral

※ available F/W version : all

Parameters

- \$path - file/port/peripheral's path
- \$flags
 - O_NODIE - PHPoC will not stop running script even a situation of PHP error

Return values

- without O_NODIE flag
 - Returns the file/port/peripheral's PID on success, PHP error on error.
- with O_NODIE flag
 - Returns the file/port/peripheral's PID on success, -ENOENT when there's no device in the entry and -EBUSY when the device is busy.

Example

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

pid_close($pid);
```

```
?>
```

See also

[pid_close\(\)](#) / [pid_ioctl\(\)](#) / [pid_read\(\)](#) / [pid_write\(\)](#) / [pid_recv\(\)](#) / [pid_send\(\)](#) / [pid_connect\(\)](#)

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.

※ available F/W version : all

Parameters

- \$pid - the device's PID to read
- \$buf - the destination buffer will be saved to
- \$len - the number of bytes to read
It reads up to the size of the \$buf if this field is omitted (integer: 8 bytes, 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 in some cases, for example fewer bytes are actually available right now.

Example

```
<?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).\WrWn", $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 incoming bytes from the internal buffer of the network device \$pid into the buffer \$buf

※ available F/W version : all

Parameters

- \$pid - the valid TCP device PID created with [pid_open\(\)](#)
- \$buf - the destination buffer will be saved to
- \$len - the number of bytes to read
It reads up to the size of the \$buf if this field is omitted (integer: 8 bytes, string: MAX_STRING_LEN)
- flags - This field should be 0

Return values

On success the number of bytes received is returned, otherwise PHP error. It is not an error if this number is smaller than the number of bytes requested. This happen in the case the length of available data in device's buffer is smaller than the requested length

Example

```
<?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 connectedWrWn";

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 of the incoming data from UDP socket's buffer

※ available F/W version : all

Parameters

- \$pid - the network device's PID
- &\$buf - the destination buffer will be saved to
- \$len - the maximum number of bytes to receive
It receives up to the size of the \$buf if this field is omitted (integer: 8 bytes, string: MAX_STRING_LEN)
- \$flags - This field should be 0
- &\$addr - the peer host's IP address
- &\$port - the peer host's UDP port number

Return values

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

Example

```
<?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_close\(\)](#) / [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

※ available F/W version : all

Parameters

- \$pid - the valid TCP device PID created with pid_open()
- \$buf - the source buffer to send
- \$len - the number of bytes to send
It sends the size of the \$buf bytes if this field is omitted (integer: 8 bytes, 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.

Example

```
<?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 of data from \$buf through UDP \$pid to the \$port at the address \$addr.

※ available F/W version : all

Parameters

- \$pid - the valid UDP device PID created with pid_open()
- \$buf - the source buffer to send
- \$len - the number of bytes to send
It sends the size of the \$buf bytes if this field is omitted (integer: 8 bytes, string: MAX_STRING_LEN)
- \$flags - This field should be 0
- \$addr - the 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 - the 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 the 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 bytesWrWn";
?>
```

```
<?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 bytesWrWn";  
?>
```

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

※ available F/W version : all

Parameters

- \$pid - the device's PID to write
- \$wbuf - the source buffer to send
- \$len - the number of bytes to write
It writes size of the \$buf bytes if this field is omitted (integer: 8 bytes, 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 $ch to the UART0
echo "W$wlen = $wlenWrWn"; // sent length is 1

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

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

$i = 1;
$wlen = pid_write($pid, $i); // write the $i to the UART0
echo "W$wlen = $wlenWrWn"; // 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

※ available F/W version : all

Parameters

- \$str - a key string which is the name of the form field

Return values

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

Example

```
<?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

※ available F/W version : all

Parameters

- \$str - a key string which is the name of the form field

Return values

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

Example

```
<?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.

※ available F/W version : all

Parameters

- \$str - the string to search
REQUEST_METHOD: a request method by the browser (GET/POST)
REMOTE_ADDR: the host's IP address what browser is running on
REMOTE_PORT: the host's port number of this connection
SCRIPT_NAME: the called script name
REQUEST_URI: the requested URI 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 key string is found, returns a value string corresponding to the key string. It returns an 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 a string into a hexadecimal representation

※ available F/W version : all

Parameters

- \$str - the string to be converted

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 "WrWn";
echo $hex; // OUTPUT: 486556C6C6F
?>
```

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

※ available F/W version : all

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.

Example

```
<?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: $cnt2WrWn"; // OUTPUT: cnt1: 3, cnt2: 8
?>
```

See also

None

Remarks

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

date()

string date (string \$format [, int \$timestamp = time()])

Description

date() returns a string formatted according to the given integer timestamp or the current time from the RTC if no timestamp is given. In other words, \$timestamp is optional and defaults to the value of time().

※ available F/W version : all

Parameters

- \$format
 - Y - A full numeric representation of a year, 4 digits (example: 2015)
 - y - A two digit representation of a year (example: 15)
 - M - A short texture representation of a month, three letters (example: Mar)
 - m - Numeric representation of a month with leading zeros (example: 03)
 - n - Numeric representation of a month without leading zeros (example: 3)
 - d - Day of the month, 2 digits with leading zeros (01 to 31)
 - j - Day of the month without leading zeros (1 to 31)
 - D - A textual representation of a day , three letters (example: Mon)
 - g - 12-hour format of an hour without leading zeros (1 to 12)
 - G - 24-hour format of an hour without leading zeros (0 to 23)
 - h - 12-hour format of an hour with leading zeros (01 to 12)
 - H - 24-hour format of an hour with leading zeros (00 to 23)
 - i - Minutes with leading zeros (00 to 59)
 - s - Seconds with leading zeros (00 to 59)
 - a - Lowercase Ante meridiem and Post meridiem (am or pm)
 - A - Uppercase Ante meridiem and Post meridiem (AM or PM)
- \$timestamp - the optional integer Unix timestamp. The default value is the current time from the RTC if this parameter is omitted.

Return values

Returns a formatted string of current system time.

Example

```
<?php
$str = date("Y:M-d-TH:i:s");
echo "$str\r\n"; // OUTPUT: 2015:Mar-16-T16:27:35
while(1);
?>
```

See also

[mktime\(\)](#) / [time\(\)](#)

Remarks

This function is identical to the PHP group's `date()` function. However, it supports only 16 types of arguments.

die()

void die (int/string \$status)

Description

die() outputs a message and terminates the current script

※ available F/W version : all

Parameters

- \$status
If \$status is a string, this function prints the status just before exiting.
If \$status is an integer, the value is not printed.

Return values

No value is returned.

Example

```
<?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.

echo

```
void echo " string $arg1" [ , "string $..." ]
```

Description

echo outputs one or more strings to web pages in response to HTTP requests, otherwise it outputs these to PHPoC's console for debugging.

echo is not a PHPoC function but language construct so it doesn't require parentheses.

※ available F/W version : all

Parameters

- \$arg1 - string to output

Return values

No value is returned.

Example

```
<?php
//this is a web page
echo "<html>";
echo "<head>";
echo "<title>Sample Web Page</title>";
echo "</head>";

echo "<body>";
echo "This is the first line<br />";
echo "This is the second line<br />";
echo "</body>";
echo "</html>";
?>
```

```
// OUTPUT TO A WEB BROWSER
This is the first line
This is the second line
```

```
<?php
$var1 = 1;
$var2 = 2;
$sum = $var1 + $var2;
```

```
$mul = $var1 * $var2;

echo "result:WrWn";
echo "sum: $var1 + $var2 = $sumWrWn";
echo "multiplication: ", "$var1 ", "* ", "$var2 ", "= ", "$mul", "WrWn";

// OUTPUT
// result:
// sum: 1 + 2 = 3
// multiplication: 1 * 2 = 2

?>
```

See also

[printf\(\)](#)

Remarks

This function is identical to the PHP group's `echo` but it doesn't support PHP group's shortcut syntax.

error_log()

bool error_log (string \$message [, int \$message_type = 0 , string \$destination])

Description

error_log() sends a user's message to PHPoC's console or a log file(/mmap/log2). It is usually used for debugging.

※ available F/W version : all

Parameters

- \$message - the message that should be logged
- \$message_type
 - 0 - outputs to PHPoC's console
 - 3 - outputs to a log file (/mmap/log2)
- \$destination - only "/mmap/log2" available when \$message_type is 3.

Return values

Returns TRUE

Example

```
<?php
$ret1 = error_log("debug message to be output to PHPoC's console\r\n");
sleep(1);
$ret2 = error_log("debug message to be output to PHPoC's /mmap/log2\r\n", 3, "/mmap/log2");
?>
```

See also

None

Remarks

This function is for debugging out, especially for programming web pages.

explode()

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

Description

explode() sends a user's message 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 PHPoC's console or a log file(/mmap/log2). It is usually used for debugging.

※ available F/W version : all

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

Example

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

$ret = explode($delimiter, $str, 3);
echo $ret[0],"rWn"; // OUTPUT: abc
echo $ret[1],"rWn"; // OUTPUT: def
echo $ret[2],"rWn"; // 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

※ available F/W version : all

Parameters

- \$status
If \$status is a string, this function prints the status just before exiting.
If \$status is an integer, the value is not printed.

Return values

No value is returned.

Example

```
<?php
exit("This script will be terminated.WrWn");
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 the output buffer
This attempts to push current output all the way to the browser.

※ available F/W version : all

Parameters

None

Return values

No value is returned.

Example

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

echo "<body>WrWn";

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

echo "</body>WrWn";

echo "</html>WrWn";
?>
```

See also

None

Remarks

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

hash()

```
string hash(string $algo, string $data [ , bool $raw_output = false ] )
```

Description

hash() generates a hash value. (message digest)

※ available F/W version : all

Parameters

- \$algo - the name of the selected hashing algorithm (i.e. "md5", "sha1" or "sha256")
- \$data - the message to be hashed
- \$raw_output
When it is set to TRUE: the function outputs raw binary data. When it is set to FALSE: the function outputs lowercase hexits.

Return values

Returns a string containing the calculated message digest as lowercase hexits unless \$raw_output is set to TRUE, in that case, the raw binary representation of the message digest is returned.

Example

```
<?php
$data = "abcdefghijklmn";

$hash_value1 = hash("md5", $data, true);
$hash_value2 = hash("md5", $data, false);
hexdump($hash_value1);
// OUTPUT: 0000 08 45 a5 97 2c d9 ad 4a 46 ba d6 6f 12 53 58 1f |.E...JF..o.SX.|
hexdump($hash_value2);
// OUTPUT:
// 0000 30 38 34 35 61 35 39 37 32 63 64 39 61 64 34 61 |0845a5972cd9ad4a|
// 0010 34 36 62 61 64 36 36 66 31 32 35 33 35 38 31 66 |46bad66f1253581f|
?>
```

See also

[hash_init\(\)](#) / [hash_update\(\)](#) / [hash_final\(\)](#) / [hash_hmac\(\)](#)

Remarks

hash() is identical to PHP group's hash function. However, it supports only "md5", "sha1" and "sha256" algorithms.

※ "sha256" algorithm is only available on the firmware version 1.3.1 and later versions.

hash_final()

```
string hash_final(string $context [ , bool $raw_output = false ] )
```

Description

hash_final() finalizes an incremental hash and return resulting digest

※ available F/W version : all

Parameters

- \$context - the hashing context returned by hash_init()
- \$raw_output
When being set to TRUE, the function outputs raw binary data. When being set to FALSE, the function outputs lowercase hexits.

Return values

Returns a string containing the calculated message digest as lowercase hexits unless \$raw_output is set to true, in which case, the raw binary representation of the message digest is returned.

Example

```
<?php
$data1 = "abcdefg";
$data2 = "hijklmn";

$context = hash_init("md5");
hash_update($context, $data1);
hash_update($context, $data2);
$hash_value = hash_final($context, true);

hexdump($hash_value);
// OUTPUT: 0000 08 45 a5 97 2c d9 ad 4a 46 ba d6 6f 12 53 58 1f |.E...JF..o.SX.|
?>
```

See also

[hash\(\)](#) / [hash_init\(\)](#) / [hash_update\(\)](#) / [hash_hmac\(\)](#)

Remarks

hash_final () is identical to PHP group's hash_final function.

hash_hmac()

```
string hash_hmac(string $algo, string $data, string $key [ , bool $raw_output = false ] )
```

Description

hash_hmac() generates a keyed hash value using the HMAC method

※ available F/W version : all

Parameters

- \$algo - the name of the selected hashing algorithm (i.e. "md5", "sha1" or "sha256")
- \$data - the message to be hashed
- \$key - the shared secret key used for generating the HMAC variant of the message digest
- \$raw_output - When being set to TRUE, the function outputs raw binary data. When being set to FALSE, the function outputs lowercase hexits

Return values

Returns a string containing the calculated message digest.

Example

```
<?php
$data = "abcdefghijklmn";
$key = "0123456789";

$hash_value = hash_hmac("md5", $data, $key, true);

hexdump($hash_value);
// OUTPUT: 0000 4d 2b 59 a5 12 ab 3c 0c 78 98 94 5a 13 97 86 50 |M+Y...<.x..Z...P|
?>
```

See also

[hash\(\)](#) / [hash_init\(\)](#) / [hash_update\(\)](#) / [hash_final\(\)](#)

Remarks

hash_hmac() is identical to PHP group's hash_hmac function. However, it supports only "md5", "sha1" and "sha256" algorithm.

※ "sha256" algorithm is only available on the firmware version 1.3.1 and later versions.

hash_init()

string hash_init(string \$algo)

Description

hash_init() initializes an incremental hashing context

※ available F/W version : all

Parameters

- \$algo - the name of the selected hashing algorithm (i.e. "md5", "sha1" or "sha256")

Return values

Returns a hasing context for use with hash_update(), hash_final().

Example

```
<?php
$data1 = "abcdefg";
$data2 = "hijklmn";

$context = hash_init("md5");
hash_update($context, $data1);
hash_update($context, $data2);
$hash_value = hash_final($context, true);

hexdump($hash_value);
// OUTPUT: 0000 08 45 a5 97 2c d9 ad 4a 46 ba d6 6f 12 53 58 1f |.E...JF..o.SX.|
?>
```

See also

[hash\(\)](#) / [hash_update\(\)](#) / [hash_final\(\)](#) / [hash_hmac\(\)](#)

Remarks

hash_init() is identical to PHP group's hash_init function. However, it supports only "md5", "sha1" and "sha256" algorithms and it doesn't support the second and the third parameters.

※ "sha256" algorithm is only available on the firmware version 1.3.1 and later versions.

hash_pbkdf2()

```
bool hash_pbkdf2(string $algo, string $password, string $salt, int $iterations [ , int $length = 0 [ ,
bool $raw_output = false ] ] )
```

Description

hash_pbkdf2() generates a PBKDF2 key derivation of a supplied password

※ available F/W version : all

Parameters

- \$algo - the name of the selected hashing algorithm (i.e. "md5", "sha1" or "sha256")
- \$password - the password to use for the derivation
- \$salt - the salt to use for the derivation
- \$iterations - the number of internal iterations to perform for the derivation
- \$length - the length of the output string
If \$raw_output is TRUE, this corresponds to the byte-length of the derived key. If \$raw_output is FALSE, this corresponds to twice the byte-length of the derived key (as every bytes of the key is returned as two hexits)
- \$raw_output - when being set to TRUE, the function outputs raw binary data. when being set to FALSE, the function outputs lowercase hexits.

Return values

Returns a string containing the derived key as lowercase hexits unless \$raw_output is set to TRUE, in which case the raw binary representation of the derived key is returned.

Example

```
<?php
$passphrase = "0123456789";
$ssid = "ssid_test";

$psk = hash_pbkdf2("sha1", $passphrase, $ssid, 4096, 32, true);

hexdump($psk);
// OUTPUT
// 0000 43 e7 47 9c 66 51 60 dd 35 a8 f9 a5 86 2e a9 de |C.G.fQ`.5.....|
// 0010 47 c3 9e 64 b5 1a 75 36 61 aa 32 3d 5f e2 cc 38 |G..d..u6a.2=_..8|
?>
```

See also

[hash\(\)](#) / [hash_init\(\)](#) / [hash_update\(\)](#) / [hash_final\(\)](#) / [hash_hmac\(\)](#)

Remarks

hash_pbkdf2() is identical to PHP group's hash_pbkdf2 function, but it supports only "md5", "sha1" and "sha256" algorithms. This function is very used to calculate a PSK for wireless LAN security.

※ "sha256" algorithm is only available on the firmware version 1.3.1 and later versions.

hash_update()

bool hash_update(string \$context , string \$data)

Description

The hash_update() pumps data into an active hashing context

※ available F/W version : all

Parameters

- \$context - the hashing context returned by hash_init()
- \$data - the message to be included in the hash digest

Return values

Returns TRUE.

Example

```
<?php
$data1 = "abcdefg";
$data2 = "hijklmn";

$context = hash_init("md5");
hash_update($context, $data1);
hash_update($context, $data2);
$hash_value = hash_final($context, true);

hexdump($hash_value);
// OUTPUT: 0000 08 45 a5 97 2c d9 ad 4a 46 ba d6 6f 12 53 58 1f |.E...JF..o.SX.|
?>
```

See also

[hash\(\)](#) / [hash_init\(\)](#) / [hash_final\(\)](#) / [hash_hmac\(\)](#)

Remarks

hash_update() is identical to PHP group's hash_update function.

header()

void header(string \$str)

Description

header() sends a 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.

※ available F/W version : all

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 modified.)

Return values

None

Example

```
<?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 the 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 a hexadecimal encoded string

※ available F/W version : all

Parameters

- \$str - the string to be converted

Return values

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

Example

```
<?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 a human-readable IP address

※ available F/W version : all

Parameters

- \$str - the string IP address to be converted

Return values

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

Example

```
<?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 a string

※ available F/W version : all

Parameters

- \$str - the human readable IP address to be converted

Return values

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

Example

```
<?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().

is_array()

bool is_array (mixed \$var)

Description

is_array() checks whether the variable is an array or not

※ available F/W version : all

Parameters

- \$var - the variable to be checked

Return values

Returns TRUE if \$var is an array, otherwise FALSE.

Example

```
<?php
$val1 = array(1, 2, 3);
$val2 = 1;

$ret1 = is_array($val1);
$ret2 = is_array($val2);

if($ret1) echo "ret1 is an array\n"; // OUTPUT: ret1 is an array
else echo "ret1 is NOT an array\n";

if($ret2) echo "ret2 is an array\n";
else echo "ret2 is NOT an array\n"; // OUTPUT: ret2 is NOT an array
?>
```

See also

[is_bool\(\)](#) / [is_float\(\)](#) / [is_int\(\)](#) / [is_string\(\)](#)

Remarks

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

is_bool()

bool is_bool (mixed \$var)

Description

is_bool() checks whether a variable is a boolean or not.

※ available F/W version : all

Parameters

- \$var - the variable to be checked

Return values

Returns TRUE if \$var is a boolean, otherwise FALSE.

Example

```
<?php
$val1 = TRUE;
$val2 = 1;

$ret1 = is_bool($val1);
$ret2 = is_bool($val2);

if($ret1) echo "ret1 is a bool\r\n"; // OUTPUT: ret1 is a bool
else echo "ret1 is NOT a bool\r\n";

if($ret2) echo "ret2 is a bool\r\n";
else echo "ret2 is NOT a bool\r\n"; // OUTPUT: ret2 is NOT a bool
?>
```

See also

[is_array\(\)](#) / [is_float\(\)](#) / [is_int\(\)](#) / [is_string\(\)](#)

Remarks

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

is_float()

bool is_float (mixed \$var)

Description

is_float() checks whether a variable is a float or not.

※ available F/W version : all

Parameters

- \$var - the variable to be checked.

Return values

Returns TRUE if \$var is a float, otherwise FALSE.

Example

```
<?php
$val1 = 3.14;
$val2 = 1;

$ret1 = is_float($val1);
$ret2 = is_float($val2);

if($ret1) echo "ret1 is a float\n"; // OUTPUT: ret1 is a float
else echo "ret1 is NOT a float\n";

if($ret2) echo "ret2 is a float\n";
else echo "ret2 is NOT a float\n"; // OUTPUT: ret2 is NOT a float
?>
```

See also

[is_array\(\)](#) / [is_bool\(\)](#) / [is_int\(\)](#) / [is_string\(\)](#)

Remarks

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

is_int()

bool is_int (mixed \$var)

Description

is_int() checks whether a variable is an integer or not.

※ available F/W version : all

Parameters

- \$var - the variable to be checked

Return values

Returns TRUE if \$var is an integer, otherwise FALSE.

Example

```
<?php
$val1 = 100;
$val2 = 100.0;

$ret1 = is_int($val1);
$ret2 = is_int($val2);

if($ret1) echo "ret1 is an intWrWn"; // OUTPUT: ret1 is an int
else echo "ret1 is NOT an intWrWn";

if($ret2) echo "ret2 is an intWrWn";
else echo "ret2 is NOT an intWrWn"; // OUTPUT: ret2 is NOT an int
?>
```

See also

[is_array\(\)](#) / [is_bool\(\)](#) / [is_float\(\)](#) / [is_string\(\)](#)

Remarks

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

is_string()

bool is_string (mixed \$var)

Description

is_string() checks whether a variable is a string or not.

※ available F/W version : all

Parameters

- \$var - the variable to be evaluated

Return values

Returns TRUE if \$var is a string, otherwise FALSE.

Example

```
<?php
$val1 = "100";
$val2 = 100;

$ret1 = is_string($val1);
$ret2 = is_string($val2);

if($ret1) echo "ret1 is a string\r\n"; // OUTPUT: ret1 is a string
else echo "ret1 is NOT a string\r\n";

if($ret2) echo "ret2 is a string\r\n";
else echo "ret2 is NOT a string\r\n"; // OUTPUT: ret2 is NOT a string
?>
```

See also

[is_array\(\)](#) / [is_bool\(\)](#) / [is_float\(\)](#) / [is_int\(\)](#)

Remarks

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

ltrim()

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

Description

ltrim() strips a white space (or other characters) from the beginning of a string

※ available F/W version : all

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

Example

```
<?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

[trim\(\)](#) / [rtrim\(\)](#)

Remarks

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

mktime()

```
int mktime ([ int $hour = date("H") [, int $minute = date("i") [, int $second = date("s") [, int $month = date("n") [, int $day = date("j") [, int $year = date("Y") ]]]]] ] )
```

Description

mktime() returns the Unix timestamp corresponding to the given arguments.

※ available F/W version : all

Parameters

- \$hour - specifies the hour, the result of date("H") if this parameter is omitted
- \$minute - specifies the minute, the result of date("i") if this parameter is omitted
- \$second - specifies the second, the result of date("s") if this parameter is omitted
- month - specifies the month, the result of date("n") if this parameter is omitted
- day - specifies the day, the result of date("j") if this parameter is omitted
- year - specifies the year, the result of date("Y") if this parameter is omitted

Return values

Returns the Unix timestamp corresponding to the given arguments.

Example

```
<?php
$time = mktime(20,30,0,6,4,2002);
$str = date("Y:M-d-TH:i:s", $time); // OUTPUT: date = 2002:Jun-04-T20:30:00
echo "date = $str\n";
?>
```

See also

[date\(\)](#) / [time\(\)](#)

Remarks

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

printf()

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

Description

printf() outputs the 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.

※ available F/W version : all

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 the output string's length, PHP error on error

Example

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

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

```

printf("%d = %dWrWn", $n); // standard integer representation
printf("%+d = %+dWrWn", $n); // standard integer representation with sign
printf("%u = %uWrWn", $n); // unsigned integer representation of a positive integer
printf("%u = %uWrWn", $u); // unsigned integer representation of a negative integer
printf("%o = %oWrWn", $n); // octal representation
printf("%e = %eWrWn", (float)$n); // scientific notation
printf("%E = %EWrWn", (float)$n); // scientific notation with a upper case E
printf("%f = %fWrWn", (float)$n); // floating point notation
printf("%F = %FWrWn", (float)$n); // floating point notation
printf("%g = %gWrWn", (float)$n); // shorter form
printf("%G = %GWrWn", (float)$n); // shorter form
printf("%s = %sWrWn", (string)$n); // string representation
printf("%x = %xWrWn", $n); // hexadecimal representation (lower-case)
printf("%X = %XWrWn", $n); // hexadecimal representation (upper-case)

$msg = 'sollae';

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

```

See also

[sprintf\(\)](#)

Remarks

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

rtrim()

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

Description

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

※ available F/W version : all

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

Example

```
<?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"), "WrWn";
//OUTPUT: .0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ98

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

See also

[trim\(\)](#) / [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 the 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).

※ available F/W version : all

Parameters

- \$seconds
the maximum PHP execution time. If set to zero, no time limit is imposed

Return values

Returns none, PHP error on failure

Example

```
<?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

※ available F/W version : all

Parameters

- \$seconds - the 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

Example

```
<?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 the 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.

※ available F/W version : all

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 the formatted string, PHP error on error

Example

```

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

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

```

```

echo sprintf("%d = %dWrWn", $n); // standard integer representation
echo sprintf("%+d = %+dWrWn", $n); // standard integer representation with sign
echo sprintf("%u = %uWrWn", $n); // unsigned integer representation of a positive integer
echo sprintf("%u = %uWrWn", $u); // unsigned integer representation of a negative integer
echo sprintf("%o = %oWrWn", $n); // octal representation
echo sprintf("%e = %eWrWn", (float)$n); // scientific notation
echo sprintf("%E = %EWrWn", (float)$n); // scientific notation with a upper case E
echo sprintf("%f = %fWrWn", (float)$n); // floating point notation
echo sprintf("%F = %FWrWn", (float)$n); // floating point notation
echo sprintf("%g = %gWrWn", (float)$n); // shorter form
echo sprintf("%G = %GWrWn", (float)$n); // shorter form
echo sprintf("%s = %sWrWn", (string)$n); // string representation
echo sprintf("%x = %xWrWn", $n); // hexadecimal representation (lower-case)
echo sprintf("%X = %XWrWn", $n); // hexadecimal representation (upper-case)

$msg = 'sollae';

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

```

See also

[printf\(\)](#)

Remarks

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

str_repeat()

string str_repeat (string \$input, int \$multiplier)

Description

str_repeat() is used to repeat a string, a specified number of times

※ available F/W version : all

Parameters

- \$input - the string to be repeated.
- \$multiplier - the number of time the \$input string should be repeated.

Return values

Returns the repeated string

Example

```
<?php
$input = "PHPoC!";
$multiplier = 3;

$ret = str_repeat($input, $multiplier);

echo "result: $ret\r\n"; // OUTPUT: result: PHPoC!PHPoC!PHPoC!
?>
```

See also

[str_replace\(\)](#) / [strpos\(\)](#) / [substr\(\)](#) / [substr_replace\(\)](#)

Remarks

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

str_replace()

string str_replace (string \$search, string \$replace, string \$subject [, int &\$count])

Description

str_replace() replaces some characters with some other characters in a string.

※ available F/W version : all

Parameters

- \$search - the value to search for (maximum length: PHP_LLSTR_BLK_SIZE - 1)
- \$replace - the value to replace the value in \$search (maximum length: PHP_LLSTR_BLK_SIZE - 1)
- \$subject - the string to be searched
- \$count - A optional variable that counts the number of replacements

Return values

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

Example

```
<?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\n"; // OUTPUT: count: 2, ret: His name is John. John is handsome.
?>
```

See also

[str_repeat\(\)](#) / [strpos\(\)](#) / [substr\(\)](#) / [substr_replace\(\)](#)

Remarks

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

strlen()

int strlen (string \$str)

Description

strlen() returns the length of a string, excluding the terminating null bytes(0)

※ available F/W version : all

Parameters

- \$str - the string to calculate

Return values

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

Example

```
<?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

※ available F/W version : all

Parameters

- \$str - the string to search in
- \$needle - the string to search for
- \$offset - the 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

Example

```
<?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

[str_repeat\(\)](#) / [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 all alphabetic characters of a string to lowercase

※ available F/W version : all

Parameters

- \$str - the string to convert to lower case

Return values

Returns the lowercased string, PHP error on error

Example

```
<?php
$str_org = "Hello PHPoC!";
$str_lower = strtolower($str_org);
printf("str_org = %s, str_lower = %s\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 all alphabetic characters of a string to uppercase.

※ available F/W version : all

Parameters

- \$str - the string to convert to upper case

Return values

Returns the uppercased string, PHP error on error

Example

```
<?php
$str_org = "Hello PHPoC!";
$str_upper = strtoupper($str_org);
printf("str_org = %s, str_upper = %sWrWn", $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

※ available F/W version : all

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 on success, FALSE on failure.

Example

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

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

See also

[str_repeat\(\)](#) / [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

※ available F/W version : all

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

Example

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

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

See also

[str_repeat\(\)](#) / [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 command

※ available F/W version : all

Parameters

- \$command - the system command (maximum length: PHP_LLSTR_BLK_SIZE - 1)
- \$arg - the extended argument (maximum length: PHP_LLSTR_BLK_SIZE - 1)

Return values

Returns the output string, PHP error on error

Example

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

See also

None

Remarks

1. Refer to the system() function in other manual for detailed information.
2. 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 the same.

```
system("php -d 5 main.php");

$delay = "5";
$filename = "main.php";
```

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

3. The PHP group's system() function outputs the result to the standard output and returns the last line. But PHPoC's system() returns the result.

time()

int time (void)

Description

time() returns current Unix timestamp.

※ available F/W version : all

Parameters

None

Return values

Returns current Unix timestamp from the RTC.

Example

```
<?php
$str = date("Y:M-d-TH:i:s", time());
echo "$str\r\n";
while(1);
?>
```

See also

[date\(\)](#) / [mktime\(\)](#)

Remarks

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

trim()

string trim (string \$str [, string \$charlist])

Description

trim() strips a white space (or other characters) from the beginning and end of a string

※ available F/W version : 1.3.1 ~

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

Example

```
<?php
hexdump(trim("\x0b\x00\x0d\x0a\x09\x20ABCDEFGH\x0b\x00\x0d\x0a\x09\x20"));
// OUTPUT: 0000 41 42 43 44 45 46 47          |ABCDEFGH      |

echo trim(".0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ012345678.", "0123.45678"), "\r\n";
// OUTPUT: 9ABCDEFGHIJKLMN0PQRSTUVWXYZ

echo trim(".0123456789ABCDEFGHIJKLMN0PQRSTUVWXYZ012345678.", "0..3...4..8"), "\r\n";
// OUTPUT: 9ABCDEFGHIJKLMN0PQRSTUVWXYZ
?>
```

See also

[ltrim\(\)](#) / [rtrim\(\)](#)

Remarks

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

usleep()

int usleep (int \$micro_seconds)

Description

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

※ available F/W version : all

Parameters

- \$micro_seconds - the sleeping time in micro-seconds (1 second = 1,000,000 micro-second)
If this parameter is negative the function is the 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

Example

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

$sleeping_time = 1000000;

echo "sleeping for $sleeping_time micro-seconds...WrWn"; // 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 msWrWn", $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 a given number

※ available F/W version : all

Parameters

- \$number - the numeric value to process

Return values

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

Example

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

echo "ret = $ret1WrWn"; // OUTPUT: ret = 10.11
echo "ret = $ret2WrWn"; // OUTPUT: ret = 10.11
echo "ret = $ret3WrWn"; // OUTPUT: ret = 10
echo "ret = $ret4WrWn"; // OUTPUT: ret = 10
?>
```

See also

None

Remarks

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

acos()

float acos (float \$arg)

Description

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

※ available F/W version : all

Parameters

- \$arg - the argument to process between -1 and 1

Return values

Returns the arc cosine of \$arg in radians

Example

```
<?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().

※ available F/W version : all

Parameters

- \$arg - the argument to process between -1 and 1

Return values

Returns the arc sine of \$arg in radians

Example

```
<?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().

※ available F/W version : all

Parameters

- \$arg - the argument to process

Return values

Returns the arc tangent of \$arg in radians

Example

```
<?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) = $val1WrWn"; // OUTPUT: atan(0.5) = 0.46364760900081
echo "atan2($y, $x) = $val2WrWn"; // OUTPUT: atan2(1, 2) = 0.46364760900081
echo "atan2($y3, $x3) = $val3WrWn"; // OUTPUT: atan2(-1, -2) = -2.677945044589
echo "tan($val1) = $val4WrWn"; // OUTPUT: tan(0.46364760900081) = 0.5
?>
```

See also

[tan\(\)](#) / [atan2\(\)](#)

Remarks

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

atan2()

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

Description

atan2() 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.

※ available F/W version : all

Parameters

- \$y - Dividend parameter
- \$x - Divisor parameter

Return values

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

Example

```
<?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) = $val1WrWn"; // OUTPUT: atan(0.5) = 0.46364760900081
echo "atan2($y, $x) = $val2WrWn"; // OUTPUT: atan2(1, 2) = 0.46364760900081
echo "atan2($y3, $x3) = $val3WrWn"; // OUTPUT: atan2(-1, -2) = -2.677945044589
echo "tan($val1) = $val4WrWn"; // OUTPUT: tan(0.46364760900081) = 0.5
?>
```

See also

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

Remarks

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

ceil()

float ceil (float \$value)

Description

ceil() rounds fractions up

※ available F/W version : all

Parameters

- \$value - the numeric value to round

Return values

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

Example

```
<?php
$val = 3.14;

$ret = ceil($val);
echo "ret = $ret\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.

※ available F/W version : all

Parameters

- \$arg - an angle in radians

Return values

Returns the cosine of \$arg

Example

```
<?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

※ available F/W version : all

Parameters

- \$number - the angular value in degrees

Return values

Returns the radian equivalent of \$number

Example

```
<?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)

※ available F/W version : all

Parameters

- \$arg - the argument to process

Return values

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

Example

```
<?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

※ available F/W version : all

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\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

※ available F/W version : all

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, FALSE otherwise

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

※ available F/W version : all

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, FALSE otherwise

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)

※ available F/W version : all

Parameters

- \$val - the value to check

Return values

TRUE if \$val is 'not a number', FALSE otherwise

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 the logarithm of \$arg to base \$base, otherwise log() returns the natural logarithm of \$arg

※ available F/W version : all

Parameters

- \$arg - the value to calculate the logarithm for
- \$base - the optional logarithmic base to use (default is 'e')

Return values

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

Example

```
<?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

pow() returns \$base raised to the power of \$exp

※ available F/W version : all

Parameters

- \$base - the base to use
- \$exp - the exponent

Return values

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

Example

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

echo "ret = $ret\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 \$number from radian to degrees

※ available F/W version : all

Parameters

- \$number - the radian value

Return values

Returns the equivalent of \$number in degrees

Example

```
<?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

※ available F/W version : all

Parameters

- \$min - the minimum number from 0 to 9223372036854775807
- \$max - the maximum number from 0 to 9223372036854775807

Return values

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

Example

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

※ available F/W version : all

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\n"; // OUTPUT: ret = 123.14
$ret = round($val);
echo "ret = $ret\n"; // OUTPUT: ret = 123
$ret = round($val, 4);
echo "ret = $ret\n"; // OUTPUT: ret = 123.1416
$ret = round($val, -2);
echo "ret = $ret\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.

※ available F/W version : all

Parameters

- \$arg - a value in radians

Return values

Returns the sine of \$arg.

Example

```
<?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

sqrt() returns the square root of \$arg

※ available F/W version : all

Parameters

- \$arg - the argument to process

Return values

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

Example

```
<?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

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

※ available F/W version : all

Parameters

- \$arg - the argument to process in radians

Return values

Returns the tangent of \$arg

Example

```
<?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) = $val1WrWn"; // OUTPUT: atan(0.5) = 0.46364760900081
echo "atan2($y, $x) = $val2WrWn"; // OUTPUT: atan2(1, 2) = 0.46364760900081
echo "atan2($y3, $x3) = $val3WrWn"; // OUTPUT: atan2(-1, -2) = -2.677945044589
echo "tan($val1) = $val4WrWn"; // OUTPUT: tan(0.46364760900081) = 0.5
?>
```

See also

[atan\(\)](#) / [atan2\(\)](#)

Remarks

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