PHPoC system() Function Manual

Version v1.0

Sollae Systems Co., Ltd.

http://www.sollae.co.kr

Table of Contents

1 system() Function

1.1 What is system() function

The system() is an internal function for system related commands and extended commands which could be added or removed.

Please refer to the Appendix for commands list of each device.

1.2 Command Format

The format of system() function is as follows:

```
string system(string $command_string[, string arg1, string arg2, ...]);
```

The system function returns a string after operation.

1.2.1 Format 1: only a command string without parameter

The followings are examples which have a command string without any parameter.

```
<?php
system("php main.php");  // Run main.php
?>
```

```
<?php
system("php -d 3 main.php");  // Run main.php (restart delay: 3 seconds)
?>
```

```
<?php
// Run main.php (task ID: 0, CPU time: 500us, restart delay: 3 seconds)
system("php -i 0 -t 500 -d 3 main.php");
?>
```

1.2.2 Format 2: command with parameter(s)

The words starting with '%' followed by a number in the command string are replaced by parameters. This format is useful when a command includes space or control character.

The followings are examples which have a command string with parameters.

```
<?php
$script = "main.php";
system("php %1",$script);  // Run main.php
```

```
?>
```

```
<?php
$delay = "3";
$script = "main.php";
system("php -d %1 %2", $delay, $script);  // Run main.php (restart delay: 3 seconds)
?>
```

```
<?php
$php_id = "0";
$cpu_time = "500";
$delay = "3";
$script = "main.php";
// Run main.php (task ID: 0, CPU time: 500us, restart delay: 3 seconds)
system("php  -i %1 -t %2 -d %3 %4", $php_id, $cpu_time, $delay, $script);
?>
```

2 Control / Information Commands

2.1 uname command

This command returns firmware version, PHPoC version, processor information, and hardware information. It requires a parameter string starting with '-' and supports multiple parameters with a parameter string. Example: system("uname –sv");

Parameter	Description
s	PHPoC engine name
v	PHPoC engine version
i	Hardware platform information
p	Processor information

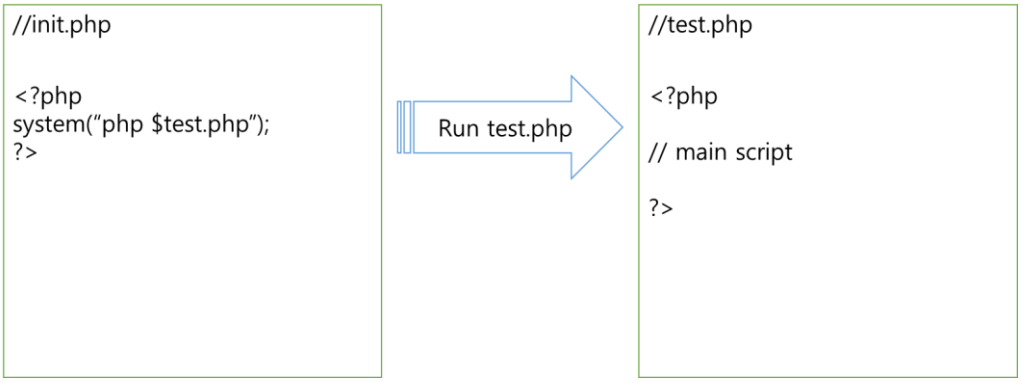
The following is an example of uname command.

```
<?php
// when the device is PBH-101
echo system("uname -s"), "\r\n";  // OUTPUT: PHPoC
echo system("uname -v"), "\r\n";  // OUTPUT: 1.0.0
echo system("uname -i"), "\r\n";  // OUTPUT: PBH-101
echo system("uname -p"), "\r\n";  // STM32F207 Cortex-M3 120MHz
echo system("uname –svip"), "\r\n";
// OUTPUT: PHPoC 1.0.0 STM32F207 Cortex-M3 120MHz PBH-101
?>
```

2.2 php command

When the system boots up, the init.php is run at first.

User can program user defined function in the init.php but user can run other file with a "php" system command. In addition, user can adjust CPU running time per a loop with this command.



This command returns the script name to be run.

The following is the php command format.

- string system("php [-i $tid –t $cpu_time –d $restart_delay $script_name]");

Parameter	Description
$tid	$tid is a PHPoC task ID. The default value is 0 if it is omitted. Please refer to the Appendix for supported task ID of each device.
$cpu_time	$cpu_time is CPU running time per a loop of the PHPoC engine. The default value is 500us if it is omitted (configuration range: 10 ~ 10000) If this value is larger, CPU occupation rate will be higher. But there might be network data losses when network load is high if this parameter is high.
$restart_delay	The script will be restarted automatically after this parameter time when the PHP script has been terminated. If this value is 0 the script will not be restarted. If it is omitted the default value is 5. The unit for this parameter is second.
$script_name	If the $script_name is used for a parameter, this script will be restarted if the script is terminated. If it is omitted, CPU occupation time will be changed immediately with the designated in the -i parameter value.

The following example is an example which the script is restarted if the current script is terminated.

```
system("php test.php");  // run test.php after finishing this script
```

The following example is a script which changes the CPU occupation time of the HTTP task per a loop to 50us.

```
system("php -i 7 -t 50");  // set CPU time of the HTTP task(3) to 50u seconds
```

2.3 reboot command

User can restart PHPoC script or the system with a "reboot" command.

The format of the reboot command is followed:

- string system(“reboot %1 [ms]”, $target);

The operation of this command is different depending on the $target as follows:

$target	Description
php	restarts the PHPoC engine.
sys	restarts the device (system).

The [ms] is delay time and its unit is millisecond. If the [ms] is omitted the default value is 100.

This command returns the delay time.

```
<?php
echo "Restart PHPoC in 1 second...\r\n";
system("reboot php 1000");

while(1);
?>
```

3 Commands for flash memory

3.1 Introduction

3.1.1 The environmental variable area

There is some information which should be kept even though system power is off. This information is stored in the flash memory(non-volatile memory).

There are a system data area for the system and a user data area for user data in the flash memory.

Data area	Description	Usage
system data area /mmap/envs	the area which values for system are saved into	network related data other system data
user data area /mmap/envu	the area for user data	user data

There is only user data area information in this document. Please refer to other document for the system data area.

3.2 nvm command

To save user data to the flash memory, save the data with a "nvm write" command after getting a key with a "nvm wkey" command.

- system("nvm wkey %1", $target);

  After generating a key for saving to flash memory, returns the key.

Parameter	Description
$target	area to save into (envs: system data area, envu: user data area)

- system("nvm write envs/envu wkey env");

  Saving the data to the flash area with the key which was generated with "nvm wkey".

Parameter	Description
envs/envu	envs – system data area / envu – user data area
wkey	the key which was returned from the "nvm wkey" command
env	data to be saved

When user saved data to the flash memory user cannot save data in the same area within 2 seconds. And the number of saving operation is limited because of hardware limitation.

The following is an example which saves "abcdefghij" to the user data area.

```
<?php
$str = "abcdefghij";

echo "setup /mmap/envu (user non-volatile meory)\r\n";
$wkey = system("nvm wkey envu");
echo "write \$str to /mmap/envu\r\n";
system("nvm write envu $wkey %1", $str);  // write $str  to /mmap/envu (flash)

echo "open /mmap/envu and read it\r\n";
$pid_envu = pid_open("/mmap/envu");  // open /mmap/envu
$buf = "";
pid_read($pid_envu, $buf, 10);  // read /mmap/envu
echo "/mmap/envu : $buf\r\n";

while(1);
?>
```

4 Crypto commands

4.1 Encryption/Decryption

4.1.1 rc4 command

The RC4 is a stream cipher made by Ron Rivest, which is used in TLS and WEP.

Because RC4 is an symmetric crypto algorithm both encryption and decryption command are same.

The following is the methods of encryption and decryption in PHPoC.

- system("rc4 init %1", $rc4_key);

  This command is for initializing PHPoC's RC4 crypto engine. So this command should be executed before the encryption/decryption. This command returns a context which is used for encryption and decryption operation.

Parameter	Description
$rc4_key	pre-defined key for the RC4

- system("rc4 crypt %1 %2", $rc4, $rc4_text);

  The RC4 encryption or decryption is performed with this command.
  It returns the data which was encrypted or decrypted.

Parameter	Description
$rc4	the context when the crypto engine was initialized
$rc4_text	plain text to be encrypted or cipher text to be decrypted

- system("rc4 skip %1", $rc4);

  This command skips encryption or decryption operation. The skipping operation should be performed to improve the weakness of the RC4.

Parameter	Description
$rc4	the context when the crypto engine was initialized

The following is an example of the RC4 encryption and decryption.

```
// encryption
$rc4 = system("rc4 init %1", $rc4_key);  // initialize
$out = system("rc4 crypt %1 %2", $rc4, $rc4_pt);  // encryption
```

```
// decryption test
$rc4 = system("rc4 init %1", $rc4_key);  // initialize
$out = system("rc4 crypt %1 %2", $rc4, $rc4_ct);  // decryption
```

4.1.2 des command

The DES(Data Encryption Standard) is a symmetric key algorithm. And the triple DES gives stronger data security because it performs DES operation 3 times.

There are ECB and CBC methods in the DES.

The following is ECB data encryption/decryption algorithm.

- system("des init ecb/ede3_ecb enc/dec %1", $ecb_key);

  This command is for initializing PHPoC's DES crypto engine. So this command should be executed before the encryption/decryption operation. This command returns a context which is used for encryption and decryption.

Parameter	Description
ecb/ede3_ecb	DES ECB / 3DES ECB
enc/dec	encryption-enc / decryption-dec
$ecb_key	64 bits-key to be used during the ECB encryption and decryption

- system("des crypt %1 %2", $des, $text");

  The DES encryption or decryption is performed with this command.
  It returns the data which was encrypted or decrypted.

Parameter	Description
$des	the context when the crypto engine was initialized
$text	plain text to be encrypted or cipher text to be decrypted

The following is an example of the ECB-based DES encryption and decryption.

```
// encryption
$des = system("des init ecb enc %1", $ecb_key);  // initialize
$out = system("des crypt %1 %2", $des, $ecb_pt); // encryption

// decryption
$des = system("des init ecb dec %1", $ecb_key);  // initialize
$out = system("des crypt %1 %2", $des, $ecb_ct); // decryption
```

The following is an example of the ECB-based triple DES encryption and decryption.

```
// encryption
$des = system("des init ede3_ecb enc %1", $ecb_key);
$out = system("des crypt %1 %2", $des, $ecb_pt);

// decryption
$des = system("des init ede3_ecb dec %1", $ecb_key);
$out = system("des crypt %1 %2", $des, $ecb_ct);
```

The following is CBC data encryption/decryption algorithm.

- system("des init cbc/ede3_cbc enc/dec %1 %2", $cbc_key, $iv);

  This command is for initializing PHPoC's DES crypto engine. So this command should be executed before the encryption/decryption operation. This command returns a context which is used for encryption and decryption.

Parameter	Description
cbc/ede3_cbc	DES CBC / 3DES CBC
enc/dec	encryption-enc / decryption-dec
$cbc_key	64 bits-key to be used during the ECB encryption and decryption
$iv	64 bits initialization vector

- system("des crypt %1 %2", $des, $text");

  The DES encryption or decryption is performed with this command.
  It returns the data which was encrypted or decrypted.

Parameter	Description
$des	the context when the crypto engine was initialized
$text	plain text to be encrypted or cipher text to be decrypted

The following is an example of the CBC-based DES encryption and decryption.

```
// encryption
$des = system("des init cbc enc %1 %2", $cbc_key, $cbc_iv);  // initialize
$out = system("des crypt %1 %2", $des, $cbc_pt);             // encryption

// decryption
$des = system("des init cbc dec %1 %2", $cbc_key, $cbc_iv);  // initialize
$out = system("des crypt %1 %2", $des, $cbc_ct);             // decryption
```

4.1.3 aes command

The AES(Advanced Encryption Standard) is a widely-used crypto specification made by the NIST. There are ECB and CBC methods in the AES.

The following is ECB data encryption/decryption algorithm.

- system("aes init ecb enc/dec %1", $ecb_key);

  This command is for initializing PHPoC's AES crypto engine. So this command should be executed before the encryption/decryption. This command returns a context which is used for encryption and decryption.

Parameter	Description
enc/dec	encryption-enc / decryption-dec
$ecb_key	the 128/192/256 bits key to be used encryption and decryption

- system("aes crypt %1 %2", $aes, $text");

  The AES encryption or decryption is performed with this command according to initialization command.

  It returns the data which was encrypted or decrypted.

Parameter	Description
$aes	the context when the crypto engine was initialized
$text	plain text to be encrypted or cipher text to be decrypted

The following is CBC data encryption/decryption algorithm.

- system("aes init cbc enc/dec %1 %2", $cbc_key, $iv);

  This command is for initializing PHPoC's AES crypto engine. So this command should be executed before the encryption/decryption. This command returns a context which is used for encryption and decryption.

Parameter	Description
enc/dec	encryption-enc / decryption-dec
$cbc_key	the 128/192/256 bits key to be used in encryption and decryption
$iv	128 bits initialization vector

- system("aes crypt %1 %2", $aes, $text");

  The AES encryption or decryption is performed with this command according to initialization command.

  It returns the data which was encrypted or decrypted.

Parameter	Description
$aes	the context when the crypto engine was initialized
$text	plain text to be encrypted or cipher text to be decrypted

The following is an example of the CBC-based AES encryption and decryption.

```
// encryption
$aes = system("aes init cbc enc %1 %2", $tv_cbc_key, $tv_cbc_iv);
$out = system("aes crypt %1 %2", $aes, $tv_cbc_pt16);

// decryption
$aes = system("aes init cbc dec %1 %2", $tv_cbc_key, $tv_cbc_iv);
$out = system("aes crypt %1 %2", $aes, $tv_cbc_ct16);
```

4.1.4 seed command

The SEED is a widely-used crypto specification in South Korea made by the KISA. There are ECB and CBC methods in the SEED.

The following is ECB data encryption/decryption algorithm.

- system("seed init ecb enc/dec %1", $type, $ecb_key);

  This command is for initializing PHPoC's SEED crypto engine. So this command should be executed before the encryption/decryption operation. This command returns a context which is used for encryption and decryption.

Parameter	Description
enc/dec	encryption-enc / decryption-dec
$ecb_key	The 128/256 bits key to used used in encryption and decryption

- system("seed crypt %1 %2", $seed, $text");

  The SEED encryption or decryption is performed with this command according to the initialization command.

  It returns the data which was encrypted or decrypted.

Parameter	Description
$seed	the context when the crypto engine was initialized
$text	plain text to be encrypted or cipher text to be decrypted

The following is CBC data encryption/decryption algorithm.

- system("seed init cbc enc/dec %1 %2", $ecb_key, $iv);

  This command is for initializing PHPoC's SEED crypto engine. So this command should be executed before the encryption/decryption. This command returns a context which is used for encryption and decryption.

Parameter	Description
enc/dec	encryption-enc / decryption-dec
$cbc_key	The 128/256 bits key to be used in encryption and decryption
$iv	128 bits initialization vector

- system("seed crypt %1 %2", $seed, $text");

  The SEED encryption or decryption is performed with this command according to the initialization command.

  It returns the data which was encrypted or decrypted.

Parameter	Description
$seed	the context when the crypto engine was initialized
$text	plain text to be encrypted or cipher text to be decrypted

The following is an example of the CBC-based SEED encryption and decryption.

```
// encryption
$seed = system("seed init cbc enc %1 %2", $tv_cbc_key, $tv_cbc_iv);
$out = system("seed crypt %1 %2", $seed, $tv_cbc_pt32);

// decryption
$seed = system("seed init cbc dec %1 %2", $tv_cbc_key, $tv_cbc_iv);
$out = system("seed crypt %1 %2", $seed, $tv_cbc_ct32);
```

4.1.5 base64 command

The BASE64 is an encryption and decryption algorithm to convert binary data to ASCII, and vice versa. The BASE64 is used in email and XML.

There are lots of alternations according to the usage. The PHPoC supports 3 types – standard type, URL type, and MIME type.

The following is BASE64 encryption/decryption algorithm.

- system("base64 enc/dec %1 [std/mime/url]", $msg);

Parameter	Description
enc/dec	encryption-enc / decryption-dec
$msg	plain text to be encrypted or cipher text to be decrypted
std/mime/url	std – standard, mime – MIME, url – URL The default is standard if it is omitted.

The following is an example of BASE64.

```
$enc_out = system("base64 enc %1", $msg0);
$dec_out = system("base64 dec %1", $enc_out);
```

4.2 Hash

4.2.1 md5 command

The MD5 is a 16 bytes hash algorithm to test data integrity.

The MD5 calculation procedure consists of 3 steps – init->update->final. The operation of each step is followed:

init	initializing MD5 engine
update	calculating MD5 (can be split with multiple operations, refer to the example)
final	finalizing result

- system("md5 init");

  This command initialize MD5 engine. A context which will be used in MD5 calculation is returned. This command should be performed prior to operating MD5.

- system("md5 update %1 %2", $md5, $msg);

  This command calculate MD5 hash value. This command can be used in multiple.

Parameter	Description
$md5	the context when the MD5 engine was initialized
$msg	data to be computed

- system("md5 final %1", $md5);

  This command calculate final hash value. It returns 16 bytes hash value.

Parameter	Description
$md5	the context when the MD5 engine was initialized

The following is an example of MD5.

```
<?php
$md5 = system("md5 init");  // initialize
system("md5 update %1 %2", $md5, "0123456789");
$out = system("md5 final %1", $md5);

$md5 = system("md5 init");
system("md5 update %1 %2", $md5, "012");
system("md5 update %1 %2", $md5, "3456789");
$out = system("md5 final %1", $md5);
?>
```

4.2.2 sha1 command

The SHA1(Secure Hash Algorithm) is 20bytes hash algorithm which was designed by the NSA and published by the NIST.

The SHA1 calculation procedure consists of 3 steps – init->update->final. The operation of each step is followed:

init	initializing SHA1 engine
update	calculating SHA1 (can be split with multiple operations, refer to the example)
final	finalizing result

- system("sha1 init");

  This command initialize SHA1 engine. A context which will be used in SHA1 calculation is returned.

  This command should be performed prior to operating SHA1.

- system("sha1 update %1 %2", $sha1, $msg);

  This command calculate SHA1 hash value. This command can be used in multiple.

Parameter	Description

$sha1	the context when the SHA1 engine was initialized
$msg	data to be computed

- system("md5 final %1", $sha1);

  This command calculate final hash value. It returns 20 bytes hash value.

Parameter	Description
$sha1	the context when the SHA1 engine was initialized

The following is examples of SHA1.

```
$sha1 = system("sha1 init");
system("sha1 update %1 %2", $sha1, $msg);
system("sha1 final %1", $sha1);
```

```
$sha1 = system("sha1 init");
system("sha1 update %1 %2", $sha1, "ab");
system("sha1 update %1 %2", $sha1, "c");
system("sha1 final %1", $sha1);
```

4.2.3 pbkdf2 command

The pbkdf2 command calculates PSK(Pre-Shared Key) which is used wireless LAN. The PSK is computed by using SSID and passphrase as following example.

```
<?php
$psk = "";
$pskpp = "0123456789";
$ssid = "csw_dev";
/* valid psk output
 * 30 d2 43 f2 bc fa 69 d0 c1 10 2b 0e ef 49 b3 af
 * 1d 0a 74 07 6e 12 3f dd 9e 76 69 be 3e 87 6b 5c
 */

for($id = 1; $id <= 2; $id++)
{
  $fu_next = system("pbkdf2 hmac_sha1 init %1 %2 $id", $pskpp, $ssid);

   for($i = 0; $i < 64; $i++)
   {
     if(!$i)
       $fu_next = system("pbkdf2 hmac_sha1 update %1 %2 63", $fu_next, $pskpp);
```

```
      else
    $fu_next = system("pbkdf2 hmac_sha1 update %1 %2 64", $fu_next, $pskpp);
  }

  $psk .= system("pbkdf2 hmac_sha1 final %1", $fu_next);
}
$psk = substr($psk, 0, 32);

hexdump($psk);
?>
```

4.2.4 crc command

The crc command computes 8/16/32 bits CRC and its format is followed:

- system("crc bits %1 [init div msb/lsb]", $msg);

  It returns CRC value after calculating with the parameters.

Parameter	Description
bits	8 – 8bit CRC 16 – 16bit CRC 32 – 32bit CRC
$msg	original message to be computed
init	CRC initial value If it is omitted the default value is: 8 bits - ff, 16 bits -1d0f, 32 bits-ffffffff
div	The divisor(polynomial) to be used for CRC calculation If it is omitted the default value is: 8 bits – e0, 16 bits -1021, 32 bits- edb88320
msb/lsb	the CRC calculation order msb: calculated from the MSB to LSB lsb: calculated from the LSB to MSB If it is omitted the default value is: 8 bits - lsb, 16 bits - msb, 32bits - lsb

The following is an example code for each CRC types.

```
<?php

$string = "123456789";
```

```
  printf("CRC-16-ANSI : %04x\r\n", (int)system("crc 16 %1 0000 a001 lsb", $string));

  printf("CRC-16-Modbus : %04x\r\n", (int)system("crc 16 %1 ffff a001 lsb", $string));

  printf("CRC-CCITT FFFF: %04x\r\n", (int)system("crc 16 %1 ffff 1021 msb", $string));

  printf("CRC-CCITT 1D0F: %04x\r\n", (int)system("crc 16 %1 1d0f 1021 msb", $string));

  printf("CRC-CCITT XModem : %04x\r\n", (int)system("crc 16 %1 0000 1021 msb",
$string));

  $crc16_out = (int)system("crc 16 123456789 %1 8408 lsb", $string);
  $crc16_out = bin2int(int2bin($crc16_out, 2, true), 0, 2);
  printf("CRC-CCITT Kermit : %04x\r\n", $crc16_out);

  $crc16_out = (int)system("crc 16 123456789 ffff 8408 lsb");
  $crc16_out = $crc16_out ^ 0xffff;
  printf("CRC-CCITT PPP    : %04x\r\n", $crc16_out);

  $crc16_out = ~(int)system("crc 16 %1 0000 a6bc lsb"m $string);
  $crc16_out = bin2int(int2bin($crc16_out, 2, true), 0, 2);
  printf("CRC-16-DNP       : %04x\r\n", $crc16_out);

  ?>
```

5 Appendix

5.1 Supported system commands depending on firmware

system command	P10	P20
uname	O	O
php	O	O
reboot	O	O
crc	O	O
rc4	X	O
des	X	O
aes	X	O
seed	X	O
base64	O	O
md5	X	O
sha1	X	O
pbkdf2	X	O

5.2 Supported task ID depending on firmware

task ID	P10	P20
0	default task	default task
1~6	Not available	Not available
7	web server task	web server task

6 Technical Support

Please use the following link for technical support.

- E-mail: support@eztcp.com
- Homepage: http://www.phpoc.com/forum/

7 History

Date	Version	Description	Writer
2014.Sep.23	0.1	○ Initial release	Matthew