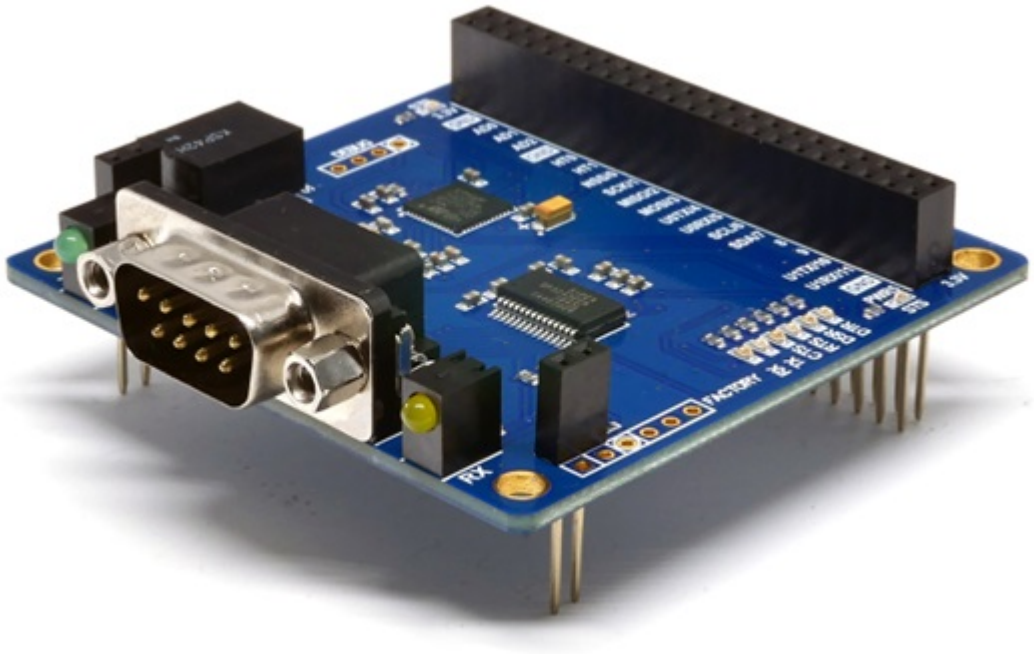
Introduction



PES-2406

PES-2406, smart RS-232 board, is a smart expansion board for PHPoC boards. With this board, you can easily implement the RS-232 communication function on the PHPoC board.

Highlights of PES-2406

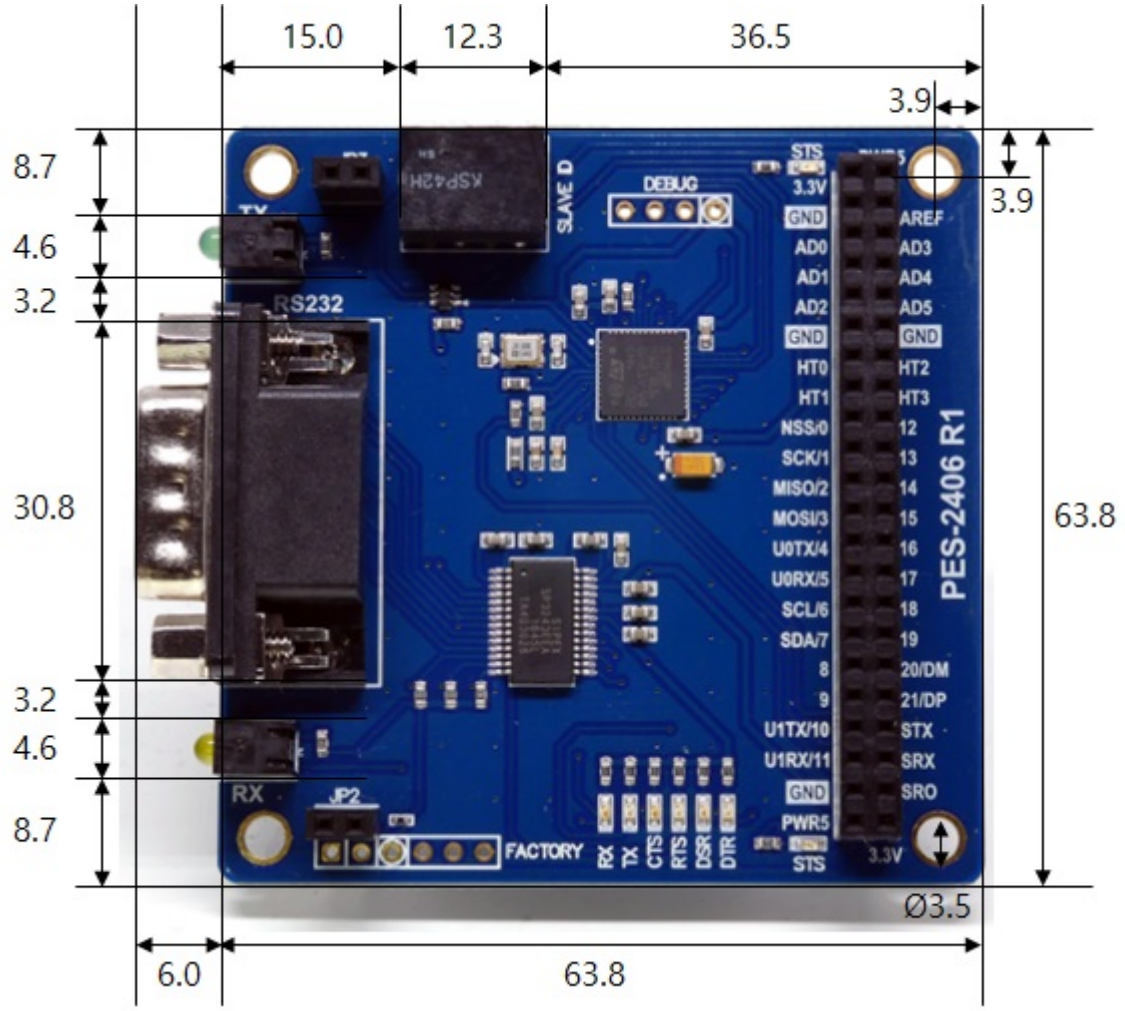
- 1 X RS-232 port: 1200bps ~ 115200bps
- Supports H/W and S/W flow control
- Provides inter frame delimiter function
- Provides inter frame gap function
- Current comsumption: approximately 25[mA]

※ Caution: PES-2406 requires a PHPoC board which has a firmware 1.3.0 or higher version.

> What is the Smart Expansion Board?
>
> A smart expansion board has own devices and firmware unlike the other expansion boards. This board communicate in a master-slave protocol through the designated port. Two or more smart expansion boards can be connected to one PHPoC board and each of them required to be setting a slave id.

Dimension

Body



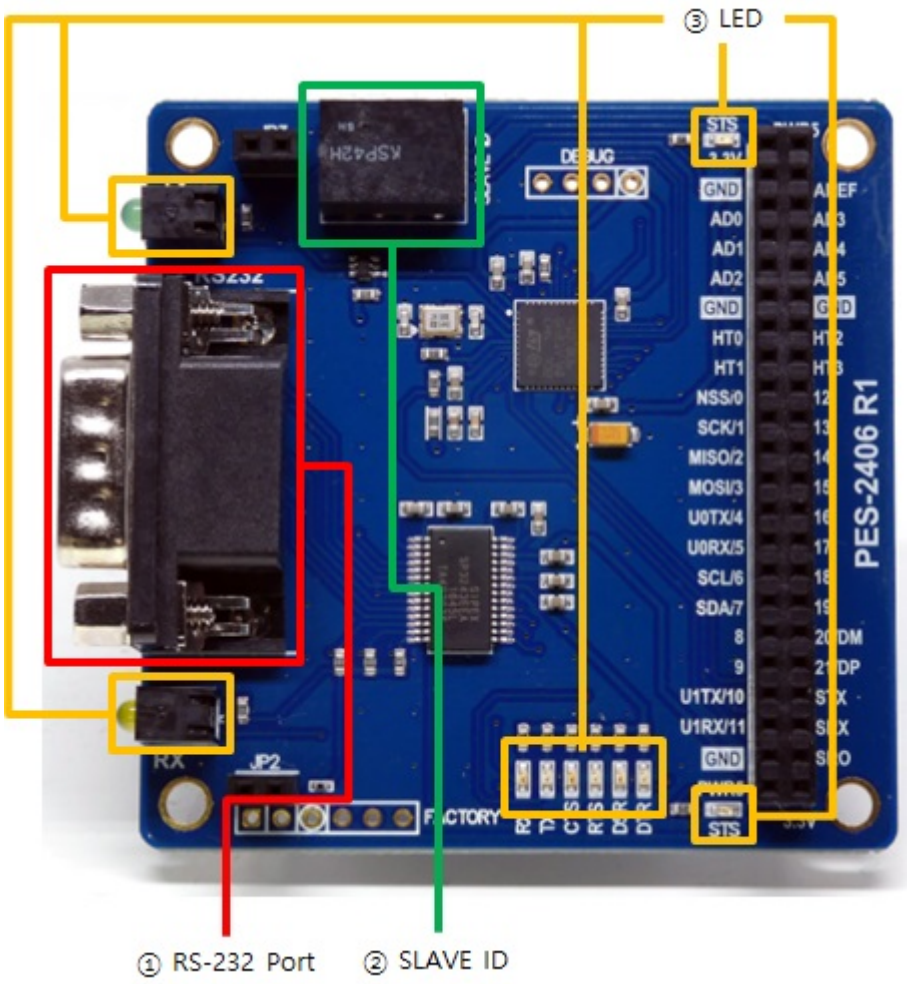
※ Dimensions(unit : mm) may vary according to a method of measurement.

Schematic

This is the schematic of PES-2406.

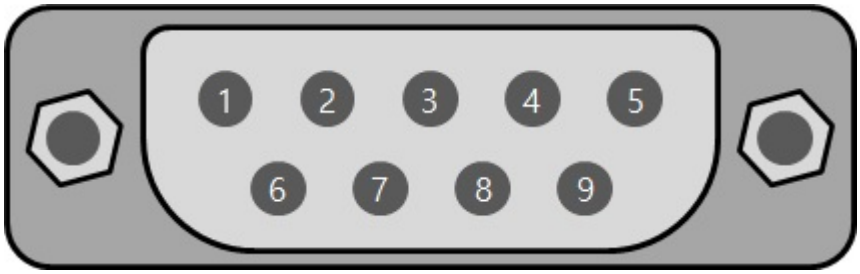
- PES-2406-R1-PO.pdf

Layout


③ LED
STS
DEBUG
GND
AD0
AD1
AD2
GND
HT0
HT1
NSS/0
SCK/1
MISO/2
MOSI/3
U0TX/4
U0RX/5
SCL/6
SDA/7
8
9
U1TX/10
U1RX/11
GND
PES-2406 R1
RX
JP2
FACTORY
STS
① RS-232 Port
② SLAVE ID

1. RS-232 Port

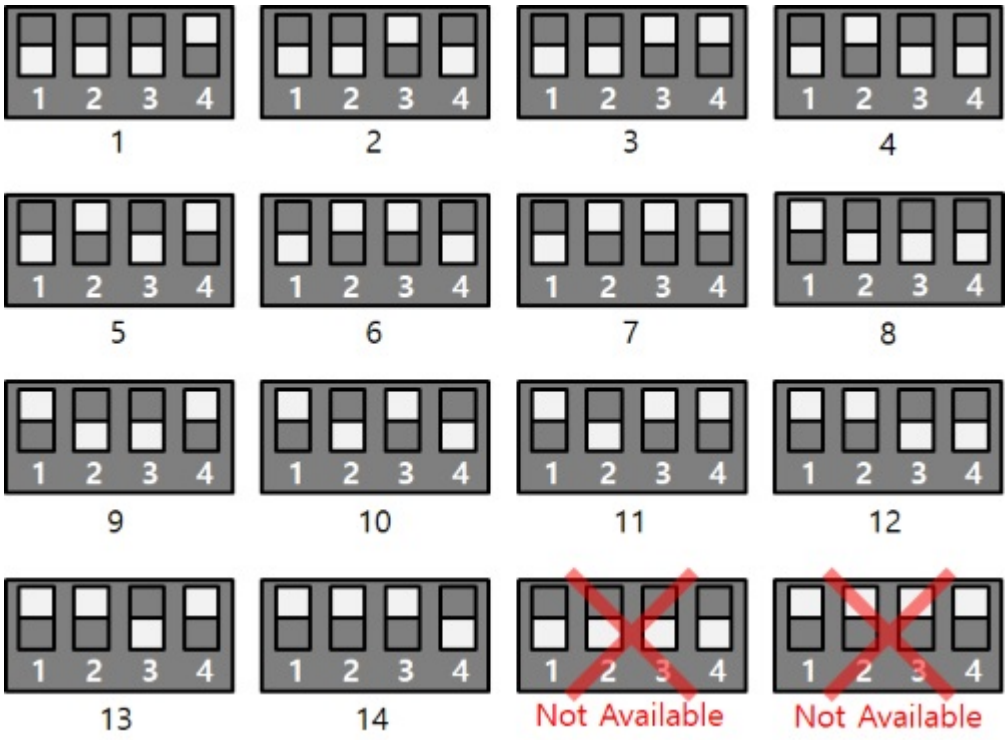
The RS-232 port of the PES-2406 is a D-SUB 9-pin male connector. The pin map is as follows.



Number	Name	Description	Level	I/O	Wiring
1	DCD	Data Carrier Detect	RS-232	In	Optional
2	RXD	Receive Data	RS-232	In	Required
3	TXD	Transmit Data	RS-232	Out	Required
4	DTR	Data Terminal Ready	RS-232	Out	Optional
5	GND	Ground	Ground	-	Required
6	DSR	Data Set Ready	RS-232	In	Optional
7	RTS	Request To Send	RS-232	Out	Optional
8	CTS	Clear To Send	RS-232	In	Optional
9	RI	Ring Indicator	RS-232	In	Optional

2. SLAVE ID Switch

A slave ID is used when PHPoC board identifies each smart expansion board. So, each smart expansion board, which is connected to a PHPoC board, should have a unique slave ID. The slave ID can be set one of the numbers from 1 to 14 by 4 DIP switches as follows:



3. LED

PES-2406 has 10 LEDs. Two of these are STS LEDs that indicate the status of the board. The one, on the top of the board, is connected to 3.3V and the other one, on the bottom of the board is connected to 5V. The rest eight of them are LEDs indicating various states of serial communication. The operation and meanings of each LED are as follows:

Name	Quantity	Type	Color	Operation
STS	2	SMD	Red	ID setting is normal > Repeated on / off every 1 second ID setting is incorrect > Fast flashing
TX	2	DIP, SMD	Green	Blinks when sending data to the serial port
RX	2	DIP, SMD	Yellow	Blinks when receiving data to the serial port
CTS	1	SMD	Yellow	On when CTS signal is ON
RTS	1	SMD	Green	On when RTS signal is ON
DSR	1	SMD	Yellow	On when DSR signal is ON
DTR	1	SMD	Green	On when DTR signal is ON

How to Use

The steps for using PES-2406 are as follows.

1. Connect to a PHPoC board

PES-2406 cannot be used alone. Be sure to connect to PHPoC board.

2. Install Software (IDE)

PHPoC Debugger is a software which is used for configuring PHPoC products and developing PHPoC script. It is required to install this software on your PC because PES-2406 must be controlled by PHPoC.

- PHPoC Debugger Download Page
- PHPoC Debugger Manual Page

3. Use SPC Library and Sample Codes

The SPC library is for smart expansion boards such as PES-2406. This library makes it easy for you to use smart expansion boards. Refer to the manual page of SPC library for more information.

- SPC Library Manual Page

Commands

You can use spc_request, spc_request_dev or spc_request_sys function when setting or using a smart expansion board.

```
spc_request($sid, 6, $wbuf)
spc_request($sid, 7, $rbuf)
spc_request_dev($sid, $cmd)
spc_request_sys($sid, $cmd)
```

- $sid: slave ID
- $wbuf: send buffer
- $rbuf: receive buffer
- $cmd: command string

Common Commands of Smart Expansion Boards

Common commands can be used with spc_request_sys function and a list of the commands is as follows:

Command	Option	Description
get	did	get a device ID
get	uid	get a unique ID

PES-2406 Commands

Dedicated commands for each smart expansion board can be used with spc_request and spc_request_dev. The spc_request function is used for sending and receiving data, and the spc_request_dev function is used for setting and checking status.

The list of PES-2406 commands is as follows:

cmd	arg1	arg2	arg3
set	uart	(parameters)	-
	modem	(signal)	-
		rts	(rts signal)
		dtr	(dtr signal)
	count	(counter)	(value)
	ifg	(bits)	-
	ifd	(del)	-
		(start_del)	(end_del)
		-	-
	txdelay	(bits)	-
	break	(time)	-

get	uart	-	-
	modem	(signal)	-
	count	(counter)	-
	rxlen	[del]	-
	txfree	-	-
	rxbuf	-	-
	txbuf	-	-
	ifg	-	-
	ifd	-	-
	txdelay	-	-

※ (): mandatory, []: optional

Settings

The command to set PES-2406 is set.

Communication Parameter

The command to set RS-232 communication parameters is uart.

```
"set uart (parameter)"
```

Specify a string of the following form in parameter.

```
"(baudrate)[parity[data bit[stop bit[flow control]]]]"
```

※ (): mandatory, []: optional

Paramater	Values	Description	Default Value
baudrate	1200 ~ 115200	baudrate(bps)	115200
parity	N, E, O, M or S	parity bit (N: None, E: Even, O: Odd, M: Mark, S: Space)	N
data bit	8 or 7	data bit	8
stop bit	1 or 2	stop bit	1
flow control	N, H or S	flow control (N: None, H: RTS/CTS, S: Xon/Xoff)	N

- an example of setting communication parameters

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200");
echo spc_request_dev($sid, "get uart"), "₩r₩n";   // output: 115200N81N

spc_request_dev($sid, "set uart 115200N81");
echo spc_request_dev($sid, "get uart"), "₩r₩n";   // output: 115200N81N

spc_request_dev($sid, "set uart 9600E72H");
echo spc_request_dev($sid, "get uart");           // output: 9600E72H

?>
```

※ Note: The flow control(H and S) and the setting of inter frame gap ("set ifg") cannot be used at the same time.

Modem Line Signal

The command to set modem line signals is modem. The modem line signals that can be controlled by this command are RTS and DTR, and both signals can be set simultaneously or individually.

Simultaneous setting

```
"set modem (signal)"
```

Enter a two digit binary number in signal.
The first digit indicates the value of the RTS signal, and the second digit indicates the value of the DTR signal. A value of 0 indicates an active state, and a value of 1 indicates an inactive state.

Value	RTS state	DTR state
00	active	active
01	active	inactive
10	inactive	active
11	inactive	inactive

- an example of simultaneous setting

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

spc_request_dev($sid, "set modem 11");                // RTS & DTR: active
echo spc_request_dev($sid, "get modem"), "₩r₩n";      // output(e.g.): 111111
sleep(1);

spc_request_dev($sid, "set modem 00");                // RTS & DTR: inactive
echo spc_request_dev($sid, "get modem"), "₩r₩n";      // output(e.g.): 110101
sleep(1);
?>
```

Individual setting

```
"set modem rts (rts signal)"
"set modem dtr (dtr signal)"
```

Input 1 digit of binary in signal.
A value of 0 indicates an active state, and a value of 1 indicates an inactive state.

- an example of individual setting

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

spc_request_dev($sid, "set modem rts 1");              // RTS: active
echo spc_request_dev($sid, "get modem rts"), "₩r₩n";  // output(e.g.): 1
sleep(1);

spc_request_dev($sid, "set modem dtr 1");              // DTR: active
echo spc_request_dev($sid, "get modem dtr");           // output(e.g.): 1
?>
```

※ Note: The setting modem line signal ("set modem") and hardware flow control (RTS / CTS) can not be used at the same time.

Counter Values

The command to set internal counter values is count.

> "set count (counter) (value)"

Specify a counter name and value to counter and value.

Counter	Description
rx	received byte
tx	transmitted byte
rf	received frame
tf	transmitted byte
pe	perity error
fe	framing error
oe	overrun error
be	break error
rbo	receive buffer overflow
tbo	transmit buffer overflow
rfo	received frame pointer overflow
tfo	transmitted frame pointer overflow

- an example of setting counter values

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

spc_request_dev($sid, "set count rx 0");
spc_request_dev($sid, "set count tx 0");
spc_request_dev($sid, "set count rf 0");
spc_request_dev($sid, "set count tf 0");
spc_request_dev($sid, "set count pe 0");
spc_request_dev($sid, "set count fe 0");
spc_request_dev($sid, "set count oe 0");
spc_request_dev($sid, "set count be 0");
spc_request_dev($sid, "set count rbo 0");
spc_request_dev($sid, "set count tbo 0");
spc_request_dev($sid, "set count rfo 0");
spc_request_dev($sid, "set count tfo 0");
?>
```

Inter Frame Gap

The command to set inter frame gap is ifg.

```
"set ifg (bits)"
```

Specify a interval to bits.

The inter frame gap is a way to distinguish frames using time. The interval can be set from 0 to 30000 in bit unit. For example, if you set the frame interval to 10 at 9600 bps, the actual setup time will be about 0.001 seconds (= 10/9600). Even if the unit is set to the same value, the set time differs if the communication speed is different.

In the case of transmitting data

The transmitted frames are transmitted at the set frame interval.

- an example of transmit frames with inter frame gap

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");
// set frame to frame interval : 100 milliseconds
spc_request_dev($sid, "set ifg 11520");

spc_request($sid, 7, "This is the first frame.₩r₩n");
spc_request($sid, 7, "This is the second frame.₩r₩nIt will be transmitted 100 milliseconds later right after the first packet has been transmitted. ₩r₩n");
?>
```

In the case of receiving data

If there is no new data until the set time while receiving the data, the data received up to that point is recognized as one frame. At this time, you can check the frame length with "get rxlen" command and receive data in frame units.

- an example of receive frames with inter frame gap

```
<?php
include "/lib/sd_spc.php";
```

```
$rbuf = "";
$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");
// set frame to frame interval : 100 milliseconds
spc_request_dev($sid, "set ifg 11520");

while(1)
{
    $rlen = (int)spc_request_dev($sid, "get rxlen");
    if($rlen > 0)
    {
        $rbuf = spc_request($sid, 6, "$rlen");
        echo "frame length = $rlen₩r₩n";
        hexdump($rbuf);
    }
}
?>
```

※ Note: The setting of inter frame gap ("set ifg") and the flow control(H and S) cannot be used at the same time.
※ Note: The setting of inter frame gap ("set ifg") and the setting of inter frame delimiter ("set ifd") cannot be used at the same time.

Inter Frame Delimiter

The command to set inter frame delimiter is ifd. While receiving data, PES-2406 distinguishes frames with the designated delimiter by this command. At this time, you can check the frame length with "get rxlen" command and receive data in frame units.

Setting inter frame delimiter

- Separating frames from the end

  Set the delimiter as follows to separate frames from the end.

  ```
  "set ifd (del)"
  ```

  If you set a delimiter in the del, data up to the delimiter is considered as one frame. The delimiter must be set in hexadecimal string form and can be set from 2 bytes to 64 bytes.

- Separating frames from both ends

  Set two delimiters as follows to separate frames from both ends.

  ```
  "set ifd (start_del) (end_del)"
  ```

  If you set two frame delimiters, the data from the first delimiter to the second delimiter becomes one frame. Both delimiters and the data between delimiters are valid but the rest of the data is ignored. The total length of two frame delimiters can not exceed 64 bytes.

Unsetting inter frame delimiter

How to unset inter frame delimiter is as follows:

```
"set ifd"
```

If you do not specify anything after the ifd command as above, the frame delimiter is unset.

- an example of setting and unsetting delimiter

```
<?php
include "lib/sd_spc.php";

$sid = 14;
```

```
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200");

spc_request_dev($sid, "set ifd 1b01");
echo spc_request_dev($sid, "get ifd"), "₩r₩n";  // output: 1b01

spc_request_dev($sid, "set ifd 1b02 1b03");
echo spc_request_dev($sid, "get ifd"), "₩r₩n";  // output: 1b02 1b03

spc_request_dev($sid, "set ifd");
echo spc_request_dev($sid, "get ifd"), "₩r₩n";  // output:
?>
```

- an example of sending and receiving data with delimiter

  This example uses 0x0d as a delimiter to receive data on a frame-by-frame basis and send the data back.

```
<?php
include "/lib/sd_spc.php";

$rwbuf = "";
$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");
spc_request_dev($sid, "set ifd 0d");

while(1)
{
    $txfree = (int)spc_request_dev($sid, "get txfree");
    $rlen = (int)spc_request_dev($sid, "get rxlen");
    if($rlen > 0)
    {
        if($rlen <= $txfree)
        {
            // receive data
            $rwbuf = spc_request($sid, 6, "$rlen");

            // send data
            spc_request($sid, 7, $rwbuf);

            // print data
            echo $rwbuf;
        }
    }
    usleep(1000);
}
```

```
?>
```

※ Note: The setting of inter frame delimiter ("set ifd") and the setting of inter frame gap ("set ifg") cannot be used at the same time.

Serial Transmission Delay

The command to set serial transmission delay is txdelay. This setting is to add a delay time between each byte when the PES-2406 transmits data to the serial port.

```
"set txdelay (bits)"
```

Specify delay time to bits. The setting unit is bit and the value can be set from 0 to 30000.

- an example of serial transmission delay

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

spc_request_dev($sid, "set txdelay 10");
echo spc_request_dev($sid, "get txdelay"), "₩r₩n"; // output: 10

spc_request_dev($sid, "set txdelay 0");
echo spc_request_dev($sid, "get txdelay");          // output: 0

?>
```

Sending Break Signal

The command to transmit the break signal is break.

```
"set break (time)"
```

Specify break time to time.

The break time can be set in bits or microseconds. When setting in bit units, you only need to enter the value, but when setting in microseconds, you must put "us" after the value. The available setting value is from 10 bits to 60 seconds.

※ Note: The actual break time may be longer than the specified time when system is busy.

- an example of sending break signal

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set break 10");          // sending break: 10bits duration

sleep(1);

spc_request_dev($sid, "set break 1000000us");  // sending break: 1 second
?>
```

Getting Status

The command to get the status of PES-2406 is get.

- Communication parameter(get uart)
- Modem line signal(get modem)
- Counter values(get count)
- Inter frame gap(get ifg)
- Inter frame delimiter(get ifd)
- Transmission delay(get txdelay)
- Received data length(get rxlen)
- Size of receive buffer(get rxbuf)
- Free space of send buffer(get txfree)
- Size of transmit buffer(get txbuf)

Communication Parameter

The command to get communication parameters is uart.

```
"get uart"
```

The response type by this command is a string and is the same as the setting format of "set uart".

```
"(baudrate)(parity)(data bit)(stop bit)(flow control)"
```

Parameter	Values	Description
baudrate	1200 ~ 115200	baudrate(bps)
parity	N, E, O, M or S	parity bit (N: None, E: Even, O: Odd, M: Mark, S: Space)
data bit	8 or 7	data bit
stop bit	1 or 2	stop bit
flow control	N, H or S	flow control (N: None, H: RTS/CTS, S: Xon/Xoff)

- an example of getting communication parameters

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200");
echo spc_request_dev($sid, "get uart"), "₩r₩n";    // output: 115200N81N

spc_request_dev($sid, "set uart 115200N81");
echo spc_request_dev($sid, "get uart"), "₩r₩n";    // output: 115200N81N

spc_request_dev($sid, "set uart 9600E72H");
echo spc_request_dev($sid, "get uart");            // output: 9600E72H

?>
```

Modem Line Signal

The command to check the modem line signal is modem.

There are 6 signals, which can be checked at once or individually.

Simultaneous checking

```
"get modem"
```

In this case, the return value is a string of 6 digits in binary form, with the following meanings for each digit:

```
(RI)(CTS)(RTS)(DSR)(DTR)(CD)
```

A value of 0 indicates an active state, and a value of 1 indicates an inactive state.

- an example of simultaneous checking

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

spc_request_dev($sid, "set modem 11");                  // RTS & DTR: active
echo spc_request_dev($sid, "get modem"), "₩r₩n";        // output(e.g.): 111111
sleep(1);

spc_request_dev($sid, "set modem 00");                  // RTS & DTR: inactive
echo spc_request_dev($sid, "get modem"), "₩r₩n";        // output(e.g.): 110101
sleep(1);
?>
```

Individual checking

```
"get modem (signal)"
```

In this case, specify the name of the signal to check on signal.

Signal	Description
ri	Ring Indicator
cts	Clear To Send
rts	Request To Send
dsr	Data Set Ready
dtr	Data Terminal Ready
cd	Carrier Detect

- an example of individual checking

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

spc_request_dev($sid, "set modem rts 1");              // RTS: active
echo spc_request_dev($sid, "get modem rts"), "₩r₩n";  // output(e.g.): 1
sleep(1);

spc_request_dev($sid, "set modem dtr 1");              // DTR: active
echo spc_request_dev($sid, "get modem dtr");           // output(e.g.): 1
?>
```

Counter Values

The command to get internal counter values is count.

> "get count (counter)"

Specify a name of counter to counter.

Counter	Description
rx	received byte
tx	transmitted byte
rf	received frame
tf	transmitted byte
pe	perity error
fe	framing error
oe	overrun error
be	break error
rbo	receive buffer overflow
tbo	transmit buffer overflow
rfo	received frame pointer overflow
tfo	transmitted frame pointer overflow

- an example of getting counter values

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

echo spc_request_dev($sid, "get count rx"), "₩r₩n";
echo spc_request_dev($sid, "get count tx"), "₩r₩n";
echo spc_request_dev($sid, "get count rf"), "₩r₩n";
echo spc_request_dev($sid, "get count tf"), "₩r₩n";
echo spc_request_dev($sid, "get count pe"), "₩r₩n";
echo spc_request_dev($sid, "get count fe"), "₩r₩n";
echo spc_request_dev($sid, "get count oe"), "₩r₩n";
echo spc_request_dev($sid, "get count be"), "₩r₩n";
echo spc_request_dev($sid, "get count rbo"), "₩r₩n";
echo spc_request_dev($sid, "get count tbo"), "₩r₩n";
echo spc_request_dev($sid, "get count rfo"), "₩r₩n";
echo spc_request_dev($sid, "get count tfo");
?>
```

Inter Frame Gap

The command to get inter frame gap is ifg.

```
"get ifg"
```

The return value is in bit unit.

- an example of getting inter frame gap

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

spc_request_dev($sid, "set ifg 50");
echo spc_request_dev($sid, "get ifg"), "₩r₩n";

spc_request_dev($sid, "set ifg 0");
echo spc_request_dev($sid, "get ifg");
?>
```

Inter Frame Delimiter

The command to get inter frame delimiter is ifd.

```
"get ifd"
```

The return value is a hexadecimal string with the following form.

```
[start_del [end_del]]
```

- an example of getting inter frame delimiter

```
<?php
include "lib/sd_spc.php";

$sid = 14;

spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200");

spc_request_dev($sid, "set ifd 1b01");
echo spc_request_dev($sid, "get ifd"), "₩r₩n";  // output: 1b01

spc_request_dev($sid, "set ifd 1b02 1b03");
echo spc_request_dev($sid, "get ifd"), "₩r₩n";  // output: 1b02 1b03

spc_request_dev($sid, "set ifd");
echo spc_request_dev($sid, "get ifd"), "₩r₩n";  // output:
?>
```

Serial Transmission Delay

The command to get serial transmission delay is txdelay.

```
"get txdelay"
```

The return value is in bit unit.

- an example of getting serial transmission delay

```
<?php
include "/lib/sd_spc.php";

$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

spc_request_dev($sid, "set txdelay 10");
echo spc_request_dev($sid, "get txdelay"), "₩r₩n"; // output: 10

spc_request_dev($sid, "set txdelay 0");
echo spc_request_dev($sid, "get txdelay");          // output: 0

?>
```

Received Data Size

The command to get the size of received data is rxlen. In the case of setting inter frame gap("set ifg") or inter frame delimiter ("set ifd"), the return value is the length of received frame. If there are multiple frames received, the length of the first received frame is returned.

```
"get rxlen [del]"
```

The return value is a string in integer form.

If you specify a delimiter in del, it returns the length to the delimiter.

※ Refer to the "Separating frames from the end" in Inter Frame Delimiter page about how to set the delimiter.

- an example of getting received data size

  This example receives data from the serial port and send it back to the same port.

```
<?php
include "/lib/sd_spc.php";

$rwbuf = "";
$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

while(1)
{
    $txfree = (int)spc_request_dev($sid, "get txfree");
    $rlen = (int)spc_request_dev($sid, "get rxlen");
    if($rlen > 0)
    {
        if($rlen <= $txfree)
        {
            // receive data
            $rwbuf = spc_request($sid, 6, "$rlen");

            // send data
            spc_request($sid, 7, $rwbuf);

            // print data
            echo $rwbuf;
        }
    }
    usleep(1000);
```

```
}
?>
```

- an example of getting received data size with the delimiter

  This example receives data in frame unit with a delimiter (0x0d) from the serial port and send it back to the same port.

```
<?php
include "/lib/sd_spc.php";

$rwbuf = "";
$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

while(1)
{
    $txfree = (int)spc_request_dev($sid, "get txfree");
    $rlen = (int)spc_request_dev($sid, "get rxlen 0d");
    if($rlen > 0)
    {
        if($rlen <= $txfree)
        {
            // receive data
            $rwbuf = spc_request($sid, 6, "$rlen");

            // send data
            spc_request($sid, 7, $rwbuf);

            // print data
            echo $rwbuf;
        }
    }
    usleep(1000);
}
?>
```

Size of Receive Buffer

The command to get the size of receive buffer is rxbuf.

```
"get rxbuf"
```

The response value is a string in integer form.

- an example of getting the size of receive buffer

```
<?php
include "/lib/sd_spc.php";

$rwbuf = "";
$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

$rxbuf_len = (int)spc_request_dev($sid, "get rxbuf");
echo $rxbuf_len;     // output(e.g.): 12288
?>
```

Free Space in Send Buffer

The command to get the free space in send buffer is txfree.

```
"get txfree"
```

The response value is a string in integer form.

- an example of getting the free space in send buffer

  This example receives data from the serial port and send it back to the same port.

```
<?php
include "/lib/sd_spc.php";

$rwbuf = "";
$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

while(1)
{
    $txfree = (int)spc_request_dev($sid, "get txfree");
    $rlen = (int)spc_request_dev($sid, "get rxlen");
    if($rlen > 0)
    {
        if($rlen <= $txfree)
        {
            // receive data
            $rwbuf = spc_request($sid, 6, "$rlen");

            // send data
            spc_request($sid, 7, $rwbuf);

            // print data
            echo $rwbuf;
        }
    }
    usleep(1000);
}
?>
```

Size of Send Buffer

The command to get the size of send buffer is txbuf.

```
"get txbuf"
```

The response value is a string in integer form.

- an example of getting the size of send buffer

```
<?php
include "/lib/sd_spc.php";

$rwbuf = "";
$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

$txbuf_len = (int)spc_request_dev($sid, "get txbuf");
echo $txbuf_len;     // output(e.g.): 12288
?>
```

Receiving Data

To receive data, use the spc_request function. You must input 6 for the second argument.

```
$rbuf = spc_request($sid, 6, $rlen)
```

- $rbuf: receive buffer
- $sid: slave ID
- $rlen: byte count to receive

an example of receiving data

This example receives data from the serial port and send it back to the same port.

```
<?php
include "/lib/sd_spc.php";

$rwbuf = "";
$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

while(1)
{
    $txfree = (int)spc_request_dev($sid, "get txfree");
    $rlen = (int)spc_request_dev($sid, "get rxlen");
    if($rlen > 0)
    {
        if($rlen <= $txfree)
        {
            // receive data
            $rwbuf = spc_request($sid, 6, "$rlen");

            // send data
            spc_request($sid, 7, $rwbuf);

            // print data
            echo $rwbuf;
        }
    }
    usleep(1000);
}
?>
```

Sending Data

To send data, use the spc_request function. You must input 7 for the second argument.

```
spc_request($sid, 7, $wbuf)
```

- $sid: slave ID
- $wbuf: data to send

an example of sending data

This example receives data from the serial port and send it back to the same port.

```
<?php
include "/lib/sd_spc.php";

$rwbuf = "";
$sid = 14;
spc_reset();
spc_sync_baud(115200);

spc_request_dev($sid, "set uart 115200N81");

while(1)
{
    $txfree = (int)spc_request_dev($sid, "get txfree");
    $rlen = (int)spc_request_dev($sid, "get rxlen");
    if($rlen > 0)
    {
        if($rlen <= $txfree)
        {
            // receive data
            $rwbuf = spc_request($sid, 6, "$rlen");

            // send data
            spc_request($sid, 7, $rwbuf);

            // print data
            echo $rwbuf;
        }
    }
    usleep(1000);
}
?>
```