Introduction

PES-2405

PES-2405, micro stepper motor controller, is a smart expansion board for PHPoC boards. With this board, you can accurately control various stepper motors.

Highlights of PES-2405

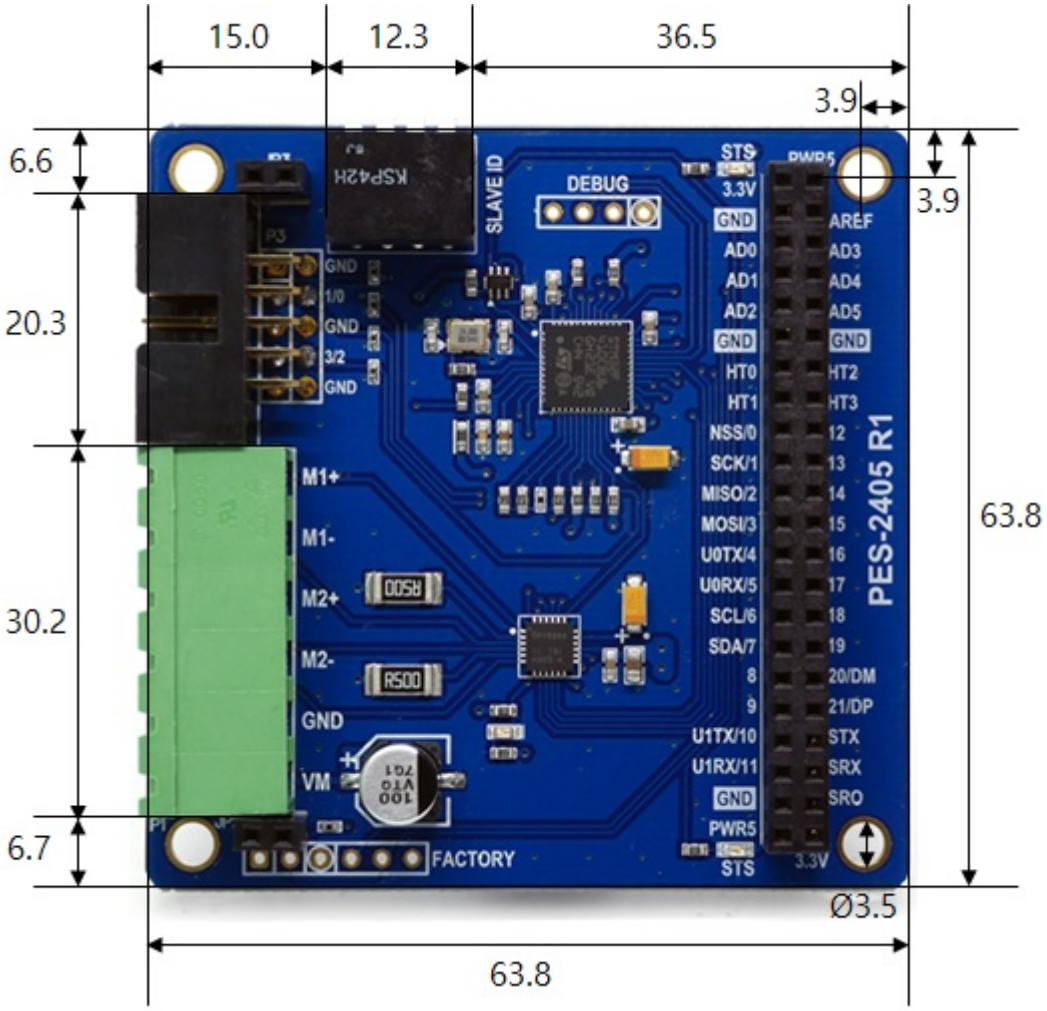
- bi-polar stepper motor controller
- input voltage: 4 ~ 18V [DC]
- drive: micro step(maximum division rate: 1/32)

※ Caution: PES-2405 requires a PHPoC board which has a firmware 1.3.0 or higher version.

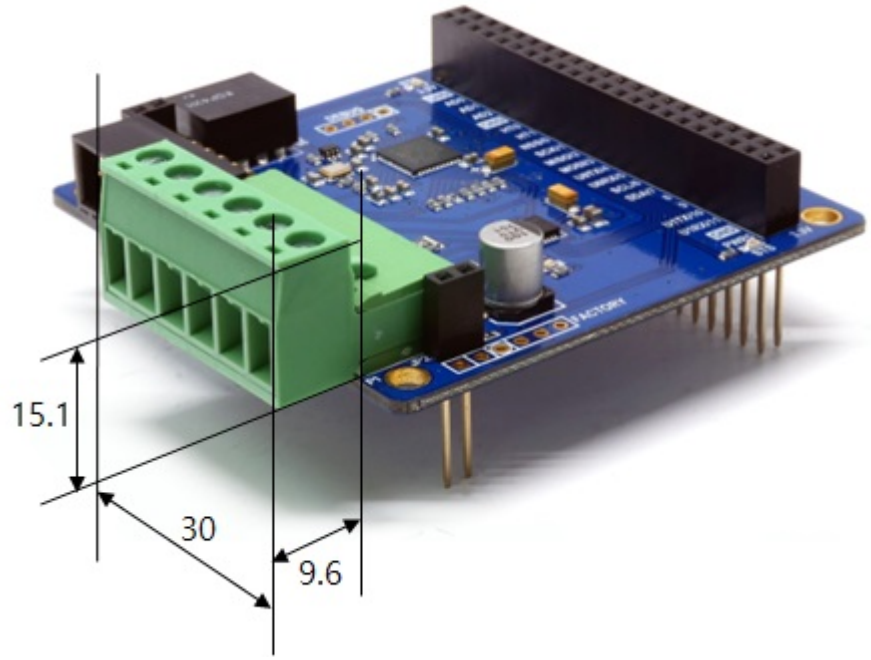
> What is the Smart Expansion Board?
>
> A smart expansion board has own devices and firmware unlike the other expansion boards. This board communicate in a master-slave protocol through the designated port. Two or more smart expansion boards can be connected to one PHPoC board and each of them required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

Dimension

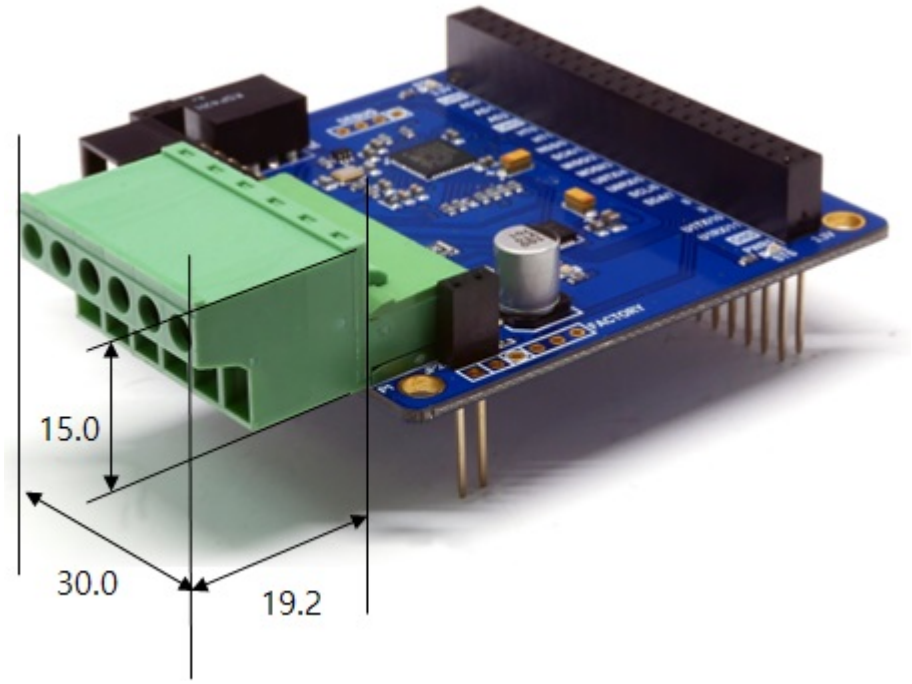
Body



with Terminal Block (T type)

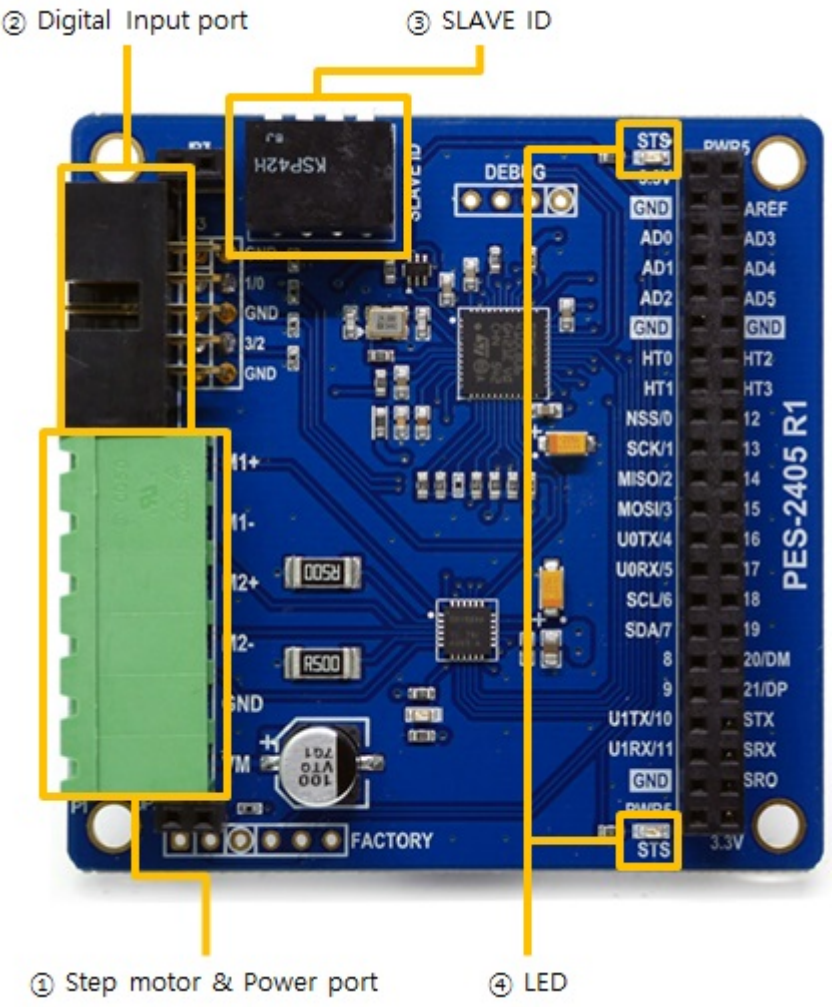


with Terminal Block (S type)



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

Layout


② Digital Input port
③ SLAVE ID
① Step motor & Power port
④ LED
STS
GND
AD0
AD1
AD2
HT0
HT1
NSS/0
SCK/1
MISO/2
MOSI/3
U0TX/4
U0RX/5
SCL/6
SDA/7
U1TX/10
U1RX/11
AREF
AD3
AD4
AD5
HT2
HT3
12
13
14
15
16
17
18
19
20/DM
21/DP
STX
SRX
SRO
3.3V
FACTORY
PES-2405 R1

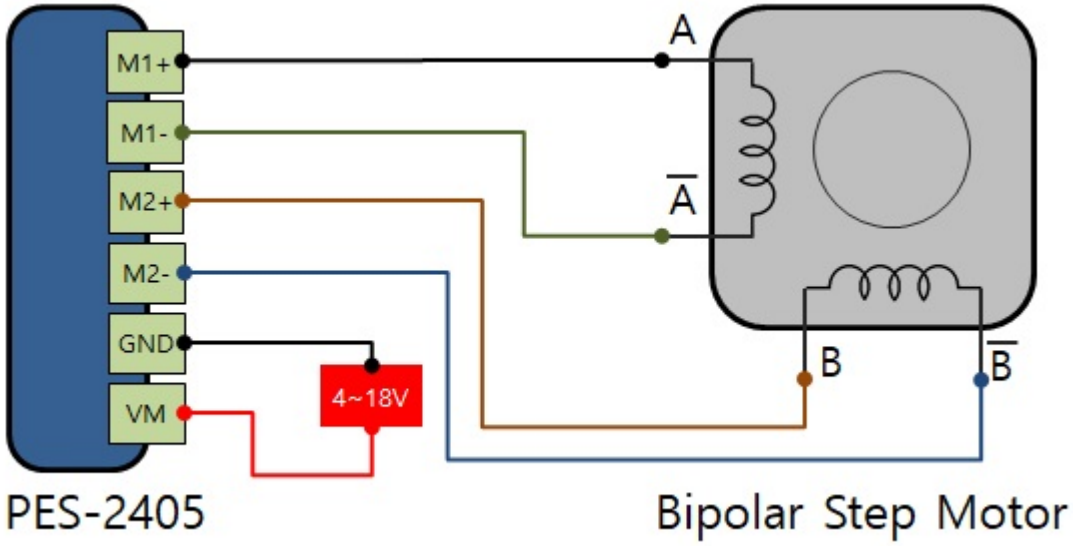
1. Stepper motor & Power port

Stepper motor & Power port is a terminal block. It has six terminals and a 5mm pitch.

label	description
M1+, M1-, M2+, M2-	connect to stepper motor
VM, GND	connect to power

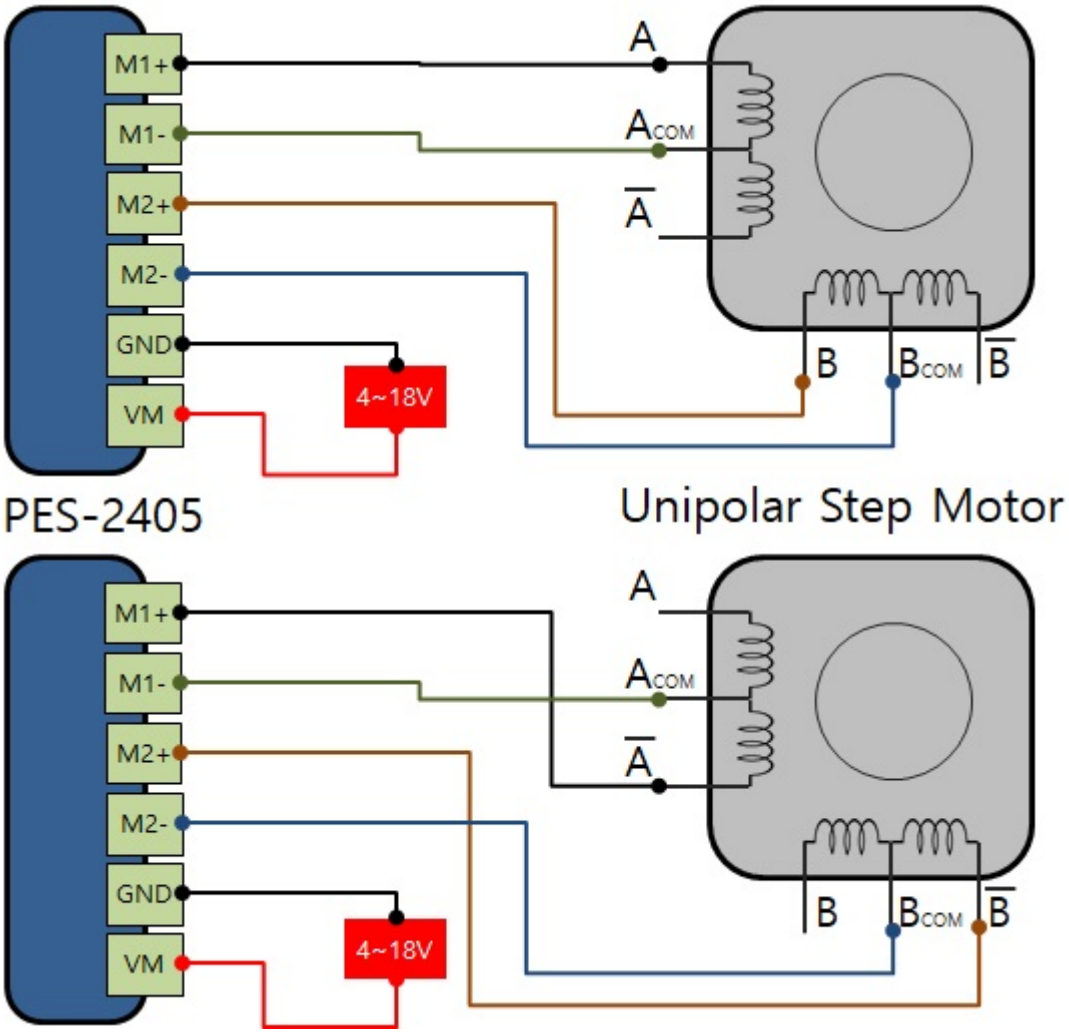
Connecting to a bipolar stepper motor

PES-2405 is a controller for bipolar stepper motors. An example of connection with a bipolar stepper motor is as follows:



Connecting to a unipolar stepper motor

If you want to connect a unipolar stepper motor to PES-2405 refer to two examples below.

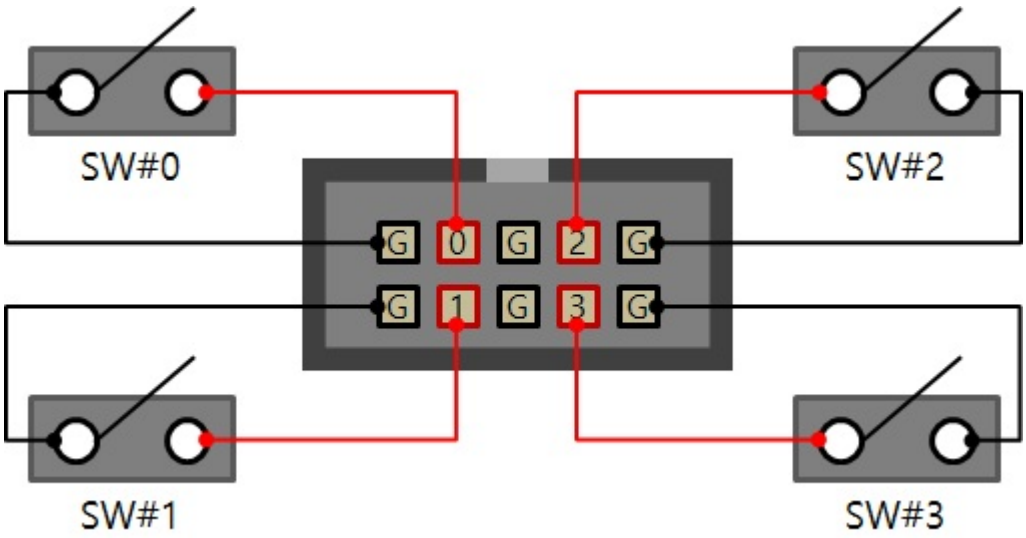


Supplying power

GND and VM are input port of supplying power(DC 4 ~ 18V) to run motors. Supplying power through this port is positively necessary. Check the DC polarity when you input the power.

2. Digital input port

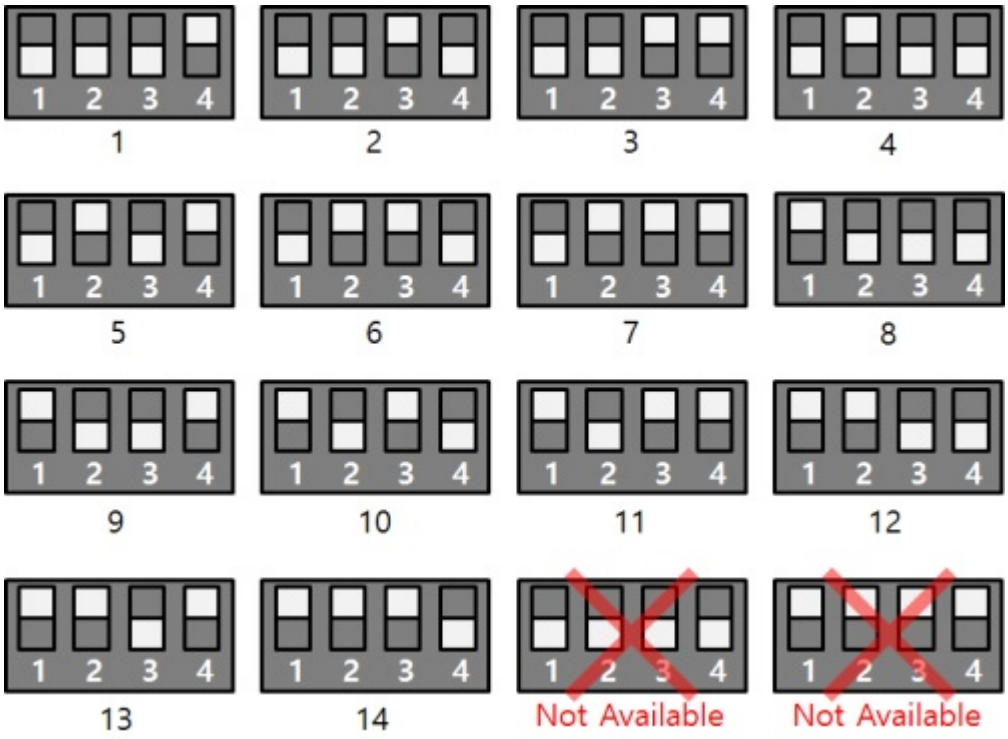
This port is used to connect limit switches and cannot be used for other purposes.
An example of connection with limit switches is as follows:



- G means ground and all ground pins are connected each other
- 0, 1, 2 and 3 are the index number of input ports and all of them are pulled up

3. 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:



4. LED

PES-2405 has 2 status LEDs. 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 operations of both LEDs are the same and they are as follows:

- about ID setting

state	operation
normal	repeat On/Off in every second
abnormal	blinks very quickly

- about running motor

state	operation
ready	repeat On/Off in every second
running	shortly turned on 4 times in every second
locked	shortly turned off 4 times in every second
out of control	off

How to Use

The steps for using PES-2405 are as follows.

1. Connect to a PHPoC board

PES-2405 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-2405 must be controlled by PHPoC.

- Download PHPoC Debugger
- PHPoC Manual Page

3. Use SPC Library and Sample Codes

The SPC library is for smart expansion boards such as PES-2405. 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_dev or spc_request_sys function when setting or using a smart expansion board.

```
spc_request_dev($sid, $cmd)
spc_request_sys($sid, $cmd)
```

- $sid: slave ID
- $cmd: a command string for setting and controlling stepper motors

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-2405 Commands

Dedicated commands for each smart expansion board can be used with spc_request_dev.
A list of dedicated commands for PES-2405 is as follows:

cmd	arg1	arg2	arg3	arg4
set	mode	(1, 2, 4, 8, 16 or 32)	-	-
	vref	stop	(0~15)	-
		drive	(0~15)	-
		lock	(0~15)	-
	rsnc	(low_pps)	(high_pps)	
	speed	(pps)	-	-
	accel	(accel)	[decel]	-
	pos	(-1000000000 ~ +1000000000)	-	-
get	state	-	-	-
	pos	-	-	-
move	step	[speed]	[accel]	[decel]
goto	pos	[spped]	[accel]	[decel]
	sw(0~3)	[speed]	[accel]	[decel]
stop	[decel]	-	-	-
eio	get	(0~3)	-	-
	set	(0~3)	mode	input
				lock

※ (): mandatory, []: optional

Settings

A command set is for setting parameters which is related with controlling stepper motors.

Setting a division rate

A command mode is for setting the division rate of microstepping.

```
"set mode (division)"
```

You can input one of divisions to drive argument.

drive	description
1	Full-step
2	Half-step
4	1/4-step
8	1/8-step
16	1/16-step
32	1/32-step

- examples of setting drive modes

```
spc_request_dev($sid, "set mode 1");
spc_request_dev($sid, "set mode 2");
spc_request_dev($sid, "set mode 4");
spc_request_dev($sid, "set mode 8");
spc_request_dev($sid, "set mode 16");
spc_request_dev($sid, "set mode 32");
```

Current limiting

A command vref is for current limiting. This is necessary to control motors.

```
"set vref (state) (value)"
```

The argument state means one of three states which requires to set limiting current.

state	description
stop	the limiting current to keep the stop state
drive	the limiting current to run motors
lock	the limiting current to keep the lock state

The argument value means the amount of current and there are 16 levels from 0 to 15. If you set this value to 5, PES-2405 limits the current of the specific state to 5 of 15.

- examples of setting limiting current

```
spc_request_dev($sid, "set vref stop 2");
spc_request_dev($sid, "set vref drive 8");
spc_request_dev($sid, "set vref lock 0");
```

Setting a resonance range

A command rsnc is for setting a resonance range.

```
"set rsnc (low_pps) (high_pps)"
```

The argument low_pps and high_pps mean the minimum and maximum values of the resonance range, respectively. When the resonance range is set, PES-2405 controls at the speed set in high_pps when the rotation speed falls within the resonance range.

- an example of setting a resonance range

```
spc_request_dev($sid, "set rsnc 120 250");
```

Setting a speed

A command speed is for setting a speed. This command is available when you want to set a speed in advance of controlling your motor.

```
"set speed (pps)"
```

The argument pps means a speed. The unit of this value is pps(pulse per second) and PES-2405 provides 20,000[pps] as its maximum speed. However, the actual maximum speed depends on the type, voltage and loads of the stepper motor.

When you set the speed, if you put k after setting value, it is possible to perform scaling up to 1000 times. For example, 1000 and 1k mean the same value.

- an example of setting a speed

```
spc_request_dev($sid, "set speed 400");
spc_request_dev($sid, "set speed 4k"); // the same result with "set speed 4000"
```

Setting acceleration and deceleration

A command accel is for setting acceleration and deceleration. This command is available when you want to set acceleration and deceleration in advance of controlling your motor.

```
"set accel (accel) [decel]"
```

The argument accel and decel mean the acceleration and deceleration respectively. The acceleration is a mandatory but the deceleration is an optional. If the deceleration is omitted, it is automatically set to the same value of the acceleration. The unit of both is pps/s(pps per second) and PES-2405 provides 200,000[pps/s] as its maximum value.

When you set these values, if you put k after setting values, it is possible to perform scaling up to 1000 times. For example, 1000 and 1k mean the same value.

- examples of setting acceleration and deceleration

```
spc_request_dev($sid, "set accel 1000"); // the same result with "set accel 1000 1000"
spc_request_dev($sid, "set accel 1000 0");
spc_request_dev($sid, "set accel 0 1k"); // the same result with "set accel 0 1000"
spc_request_dev($sid, "set accel 1k 2k");
```

Setting a counter position

A command pos is for setting a counter position. This command is valid only when controlling stepper motor with goto and is not reflected when controlling with move.

```
"set pos (pos)"
```

The argument pos means a counter position. This value is a signed 32-bit integer and can have a value between -1000000000(1 billion) and +1000000000.

When you set the counter position, if you put k after setting value, it is possible to perform scaling up to 1000 times. For example, 1000 and 1k mean the same value.

- an example of setting a counter position

```
spc_request_dev($sid, "set pos 400");
spc_request_dev($sid, "set pos 80k"); // the same result with "set pos 80000"
```

Setting digital input ports

A command eio set is for setting digital input ports.

```
"eio set (p) mode (mode)"
```

The argument p means id of digital input ports and it can be set to 0, 1, 2 or 3. The argument mode means a type of the digital input port.

mode	description
input	normal input
lock	control lock

- examples of setting digital input ports: normal input

```
spc_request_dev($sid, "eio set 0 mode input");
spc_request_dev($sid, "eio set 1 mode input");
spc_request_dev($sid, "eio set 2 mode input");
spc_request_dev($sid, "eio set 3 mode input");
```

- examples of setting digital input ports: control lock

```
spc_request_dev($sid, "eio set 0 mode lock");
spc_request_dev($sid, "eio set 1 mode lock");
spc_request_dev($sid, "eio set 2 mode lock");
```

```
spc_request_dev($sid, "eio set 3 mode lock");
```

Getting States

A command get is for getting states of a stepper motor.

Getting operation states

A command state is for getting an operation state of a stepper motor.

```
"get state"
```

Return values of this command are as follows:

value	state
0	stopped
1	control locked
otherwise	running

- an example of getting states of a stepper motor

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
spc_request_dev($sid, "set mode 4");
spc_request_dev($sid, "set vref stop 2");
spc_request_dev($sid, "set vref drive 8");
spc_request_dev($sid, "set speed 400");
spc_request_dev($sid, "set accel 800");
spc_request_dev($sid, "set rsnc 120 250");

$state = 0;
spc_request_dev($sid, "move 400");
while($state = (int)spc_request_dev($sid, "get state"))
{
  echo "state: $state₩r₩n";
  usleep(200000);
}
echo "state: $state₩r₩n";
?>
```

- the output example

```
state: 2
state: 2
```

```
state: 2
state: 2
state: 2
state: 2
state: 2
state: 0
```

Getting a counter position

A command pos is for getting a current counter position of a stepper motor.

```
"get pos"
```

- an example of getting a counter position

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

spc_reset();
spc_sync_baud(460800);

$pos = -400;
$sid = 1;
spc_request_dev($sid, "set mode 4");
spc_request_dev($sid, "set vref stop 2");
spc_request_dev($sid, "set vref drive 8");
spc_request_dev($sid, "set speed 400");
spc_request_dev($sid, "set accel 800");
spc_request_dev($sid, "set rsnc 120 250");
spc_request_dev($sid, "set pos $pos");

spc_request_dev($sid, "goto 400");
while((int)spc_request_dev($sid, "get state"))
{
    $pos = (int)spc_request_dev($sid, "get pos");
  echo "position: $pos₩r₩n";
  usleep(200000);
}
?>
```

- the output example

```
position: -400
position: -376
position: -317
position: -235
position: -153
position: -70
position: 12
```

```
position: 94
position: 175
position: 258
position: 336
position: 391
```

Getting a state of digital input ports

A command eio get is for getting a state of digital input ports.

```
"eio get (p) input"
```

The argument p means an ID(0 ~ 3) of digital input ports. Return values of this command are as follows:

value	state
0	LOW
1	HIGH (default)

- an example of getting states of digital input ports.

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
while(1)
{
    echo spc_request_dev($sid, "eio get 0 input");
    echo spc_request_dev($sid, "eio get 1 input");
    echo spc_request_dev($sid, "eio get 2 input");
    echo spc_request_dev($sid, "eio get 3 input");

    echo "₩r₩n";
    sleep(1);
}
?>
```

- the output example

```
1111
0110
...omitted...
```

Controlling by goto

A command goto is for controlling a stepper motor based on the initial position, not the current position of the motor.
This method can be used even when the motor is running.

```
"goto [sign](pos) [speed] [accel] [decel]"
```

※ Caution: there is no space between [sign] and (pos)

Descriptions of the arguments are as follows:

argument	description	mandatory/optional
sign	direction, "+"(forward) or "-"(reverse)	optional(default: "+")
pos	target counter position	mandatory
speed	speed(unit: pps)	optional
accel	acceleration(unit: pps/s)	optional
decel	deceleration(unit: pps/s)	optional

The argument pos is based on the initial position, not the current position of the counter.

If the decel(deceleration) is omitted, it is automatically set to the same value of the acceleration.

- an example of using goto

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
spc_request_dev($sid, "set mode 4");
spc_request_dev($sid, "set vref stop 2");
spc_request_dev($sid, "set vref drive 8");
spc_request_dev($sid, "set rsnc 120 250");
spc_request_dev($sid, "set pos -400");

spc_request_dev($sid, "goto +400 200 0");
sleep(1);
spc_request_dev($sid, "goto -400 200 0");
sleep(1);
spc_request_dev($sid, "goto +400 200 0");
while((int)spc_request_dev($sid, "get state"))
    usleep(1);
?>
```

※ This command can be used even when stepper motor is in operation. Therefore, if a new goto command is executed while the motor is running, it is executed immediately.

Controlling by move

A command move is for controlling a stepper motor based on the current position of the motor. This method is only available when the motor is stopped.

```
"move [sign](step) [speed] [accel] [decel]"
```

※ Caution: there is no space between [sign] and (step)

Descriptions of the arguments are as follows:

argument	description	mandatory/optional
sign	direction, "+"(forward) or "-"(reverse)	optional(default: "+")
step	the number of steps to move	mandatory
speed	speed(unit: pps)	optional
accel	acceleration(unit: pps/s)	optional
decel	deceleration(unit: pps/s)	optional

The argument step is based on the current position of the motor.

If the decel(deceleration) is omitted, it is automatically set to the same value of the acceleration.

- an example of using move

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
spc_request_dev($sid, "set mode 4");
spc_request_dev($sid, "set vref stop 2");
spc_request_dev($sid, "set vref drive 8");
spc_request_dev($sid, "set rsnc 120 250");

spc_request_dev($sid, "move 800 400 800 0");
while((int)spc_request_dev($sid, "get state"))
  usleep(1);

sleep(1);

spc_request_dev($sid, "move -800 400 0 800");
while((int)spc_request_dev($sid, "get state"))
  usleep(1);
?>
```

※ This command can be used only when the stepper motor is stopped. So, as in the example above, always program the move command to run after the motor is stopped.

Stopping

A command stop is for stopping a stepper motor which is in operation.

```
"stop [decel]"
```

The argument decel means deceleration.
The unit is pps/s.
If the decel is omitted, the current deceleration is used.

- an example of stopping a motor

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
spc_request_dev($sid, "set mode 4");
spc_request_dev($sid, "set vref stop 2");
spc_request_dev($sid, "set vref drive 8");
spc_request_dev($sid, "set rsnc 120 250");

// without stop command
spc_request_dev($sid, "move +800 200 200 200");
while((int)spc_request_dev($sid, "get state"))
    usleep(1);

sleep(1);

// with stop command
spc_request_dev($sid, "move +800 200 200 200");
sleep(1);
spc_request_dev($sid, "stop 200");

?>
```

Setting the Initial Position

By connecting a limit switch to the digital input port of the PES-2405, you can set the initial position in a system that requires initial positioning. The procedure for setting is as follows.

1. Driving the motor

   Drive the stepper motor to the direction of initial position when the system is started.

2. Stopping the motor

   Stop the stepper motor once the limit switch is closed. You can stop the motor automatically or manually.

   - stop automatically
   - stop manually

3. Setting the Initial Position

   When the motor stops, check the current position of the counter referring to Check Status and use it as the initial position.

Stop automatically

A command goto ~ sw can be used to automatically stop the stepper motor operation when the limit switch is closed.

If the motor stops by this command, the status value of the motor becomes the stopped status.

- a format of the goto ~ sw command

```
"goto [dir]sw(id) [speed] [accel] [decel]"
```

※ Caution: there is no space between [dir] and sw(id)

argument	description	mandatory/optional
dir	direction, "+"(forward) or "-"(reverse)	optional(default: "+")
id	ID of a digital input port, 0/1/2/3	mandatory
speed	speed(unit: pps)	optional
accel	acceleration(unit: pps/s)	optional
decel	deceleration(unit: pps/s)	optional

- an example of stopping a motor automatically

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
spc_request_dev($sid, "set vref stop 2");
spc_request_dev($sid, "set vref drive 8");
spc_request_dev($sid, "set mode 4");

// rotate until digital input 0 is LOW
echo "find positive limit ...";
spc_request_dev($sid, "goto +sw0 400 0");
while((int)spc_request_dev($sid, "get state"))
    usleep(1);
echo "done₩r₩n";

sleep(1);

// rotate until digital input 1 is LOW
echo "find negative limit ...";
spc_request_dev($sid, "goto -sw1 400 0");
while((int)spc_request_dev($sid, "get state"))
    usleep(1);
echo "done₩r₩n";
```

```
?>
```

- the output example

```
find positive limit ...done
find negative limit ...done
```

Stop manually

A command stop can be used to manually stop the stepper motor operation when the limit switch is closed.

At this time, a command eio get is used to detect the input of the limit switch.

- an example of stopping a motor manually

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
$state = 0;

spc_request_dev($sid, "eio set 0 mode input");

spc_request_dev($sid, "set vref stop 2");
spc_request_dev($sid, "set vref drive 8");
spc_request_dev($sid, "set mode 4");

spc_request_dev($sid, "goto 40000 400 1000");

while(1)
{
    $state = (int)spc_request_dev($sid, "eio get 0 input");
    if($state == 0)
        break;
    usleep(1);
}

spc_request_dev($sid, "stop");
?>
```

Lock and unlock

The control lock is a kind of physical protection function. This function is to connect a limit switch to the digital input port and disable further control if the switch is closed. Therefore, the operating range of the stepper motor can be limited.

Control lock

If the operation of the motor is stopped by the limit switch, the motor status is locked and no further control is possible until the lock is released.

Setting the control lock

Set the input mode of the digital input port referring to Settings chapter.

Unlock

A command unlock is for releasing the state of control lock.

When you execute the unlock command, the motor status changes from the locked state to the stopped state and the input mode of the digital input port is reset to the normal input mode from the control lock mode.

That means you can control the motor normally after executing unlock.

- an example of control lock

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
spc_request_dev($sid, "set vref stop 2");
spc_request_dev($sid, "set vref drive 8");
spc_request_dev($sid, "set vref lock 8");
spc_request_dev($sid, "set mode 4");

spc_request_dev($sid, "eio set 0 mode lock");
spc_request_dev($sid, "eio set 1 mode lock");
spc_request_dev($sid, "eio set 2 mode lock");
spc_request_dev($sid, "eio set 3 mode lock");

spc_request_dev($sid, "move 4000 400 4000");

while((int)spc_request_dev($sid, "get state") > 1)
    usleep(1);

// state: 0 - stop, 1 - locked
```

```
echo "step_state ", spc_request_dev($sid, "get state"), "₩r₩n";

spc_request_dev($sid, "unlock");

// state: 0 - stop, 1 - locked
echo "step_state ", spc_request_dev($sid, "get state"), "₩r₩n";
?>
```

- the output

```
step_state 1
step_state 0
```