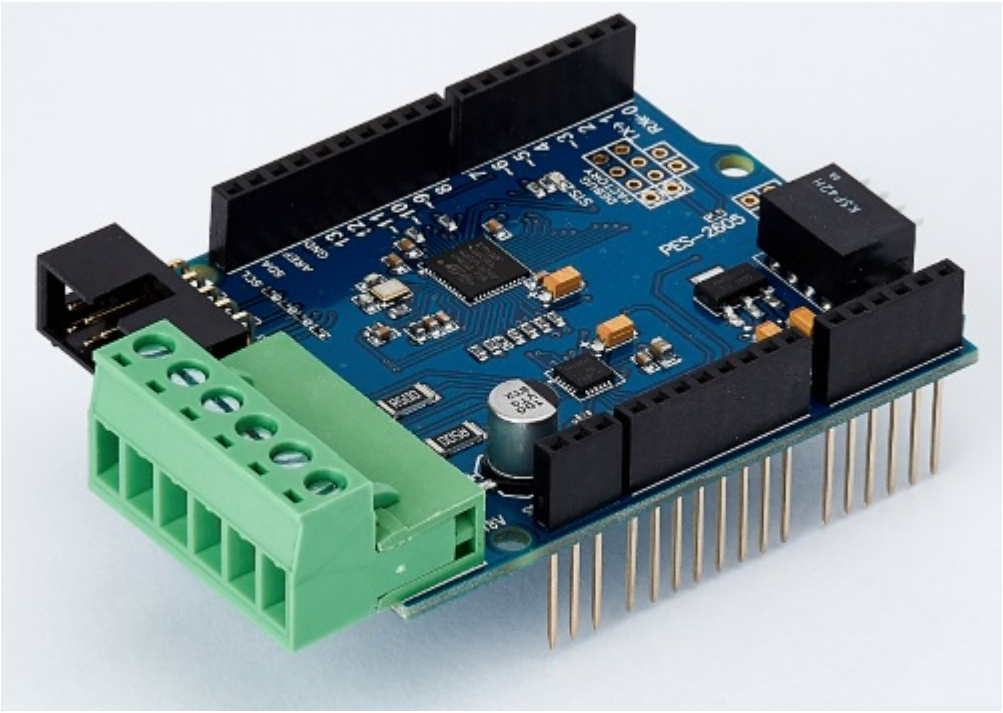
Introduction



PES-2605

PES-2605, stepper motor controller, is a smart expansion board for PHPoC shield for Arduino. You can accurately control various stepper motors with this board via the Arduino sketch.

Highlights of PES-2605

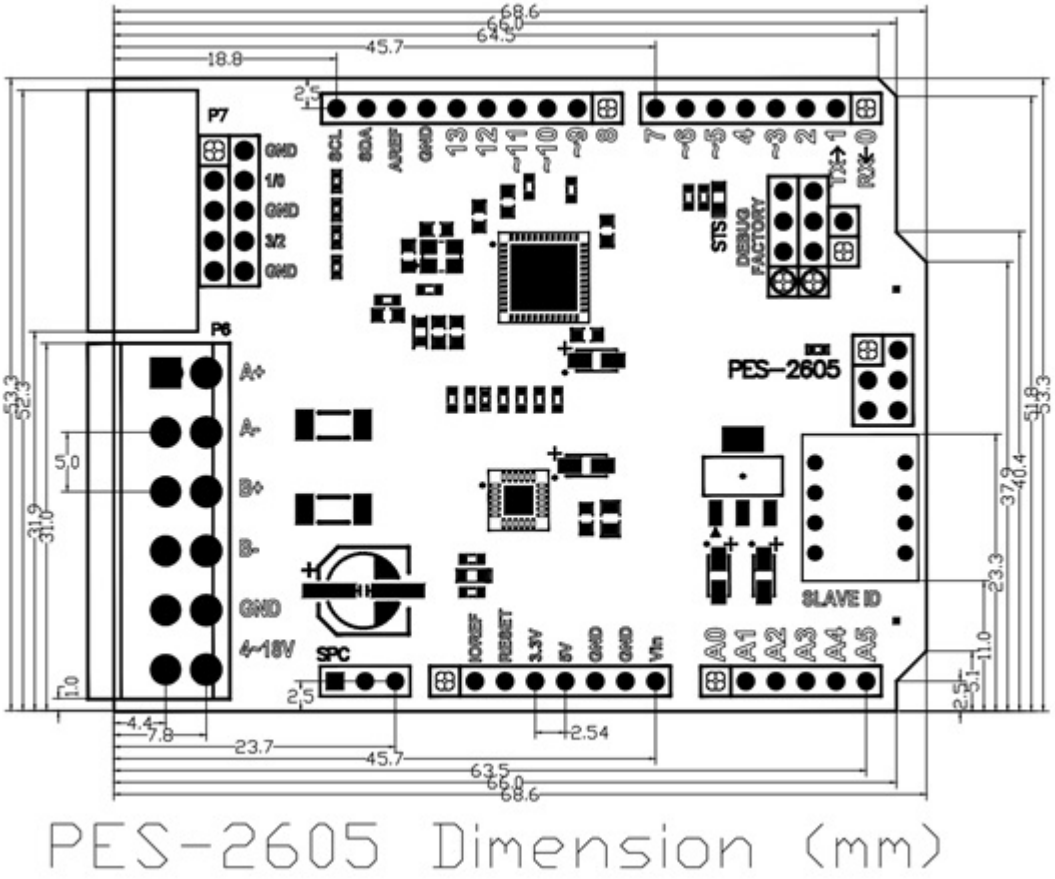
- bi-polar stepper motor controller
- driving modes: micro step(maximum division rate: 1/32)
- motor voltage: 4 ~ 18V [DC]
- motor current: maximum 1A on each coil

※ Caution : Both a PHPoC shield with R2 or later version and an Arduino board are required to use this board!

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

Dimension

Body



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

Terminal Block

This board uses two types of 6-pole terminal block. Refer to each datasheet for dimension.

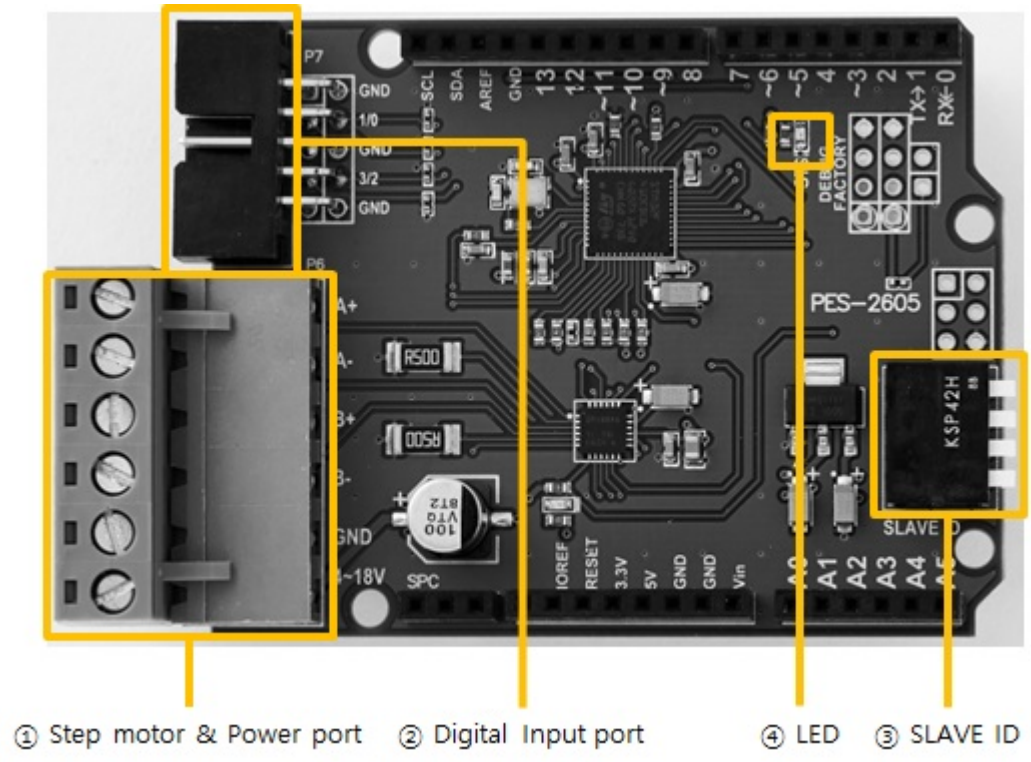
- Datasheet of T-type Terminal Block
- Datasheet of S-type Terminal Block

Schematic

This is the schematic of PES-2605.

- PES-2605-V12-PO.pdf

Layout



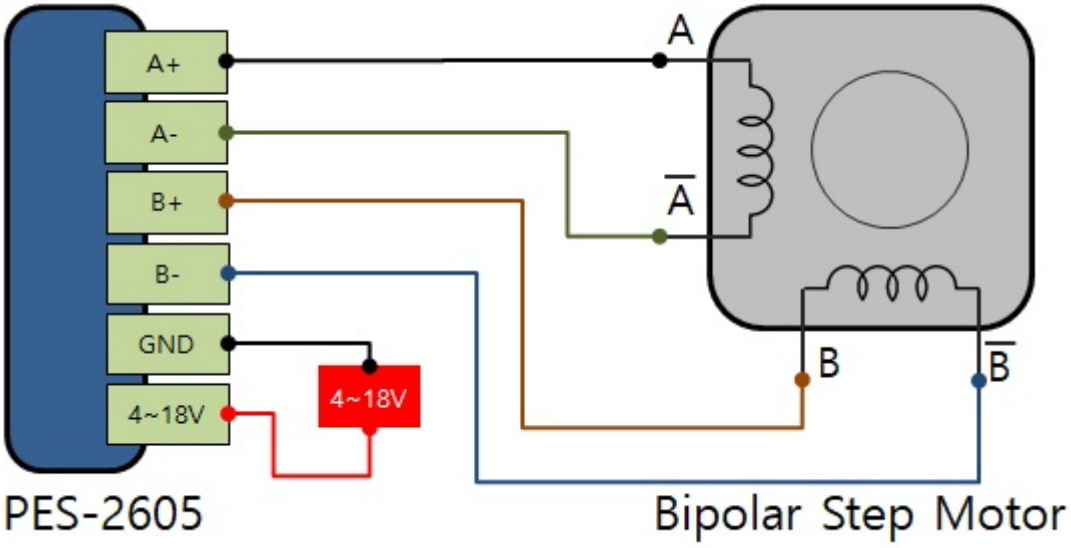
1. Stepper motor & Power port

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

label	description
A+, A-, B+, B-	connect to stepper motor
4~18V, GND	connect to power

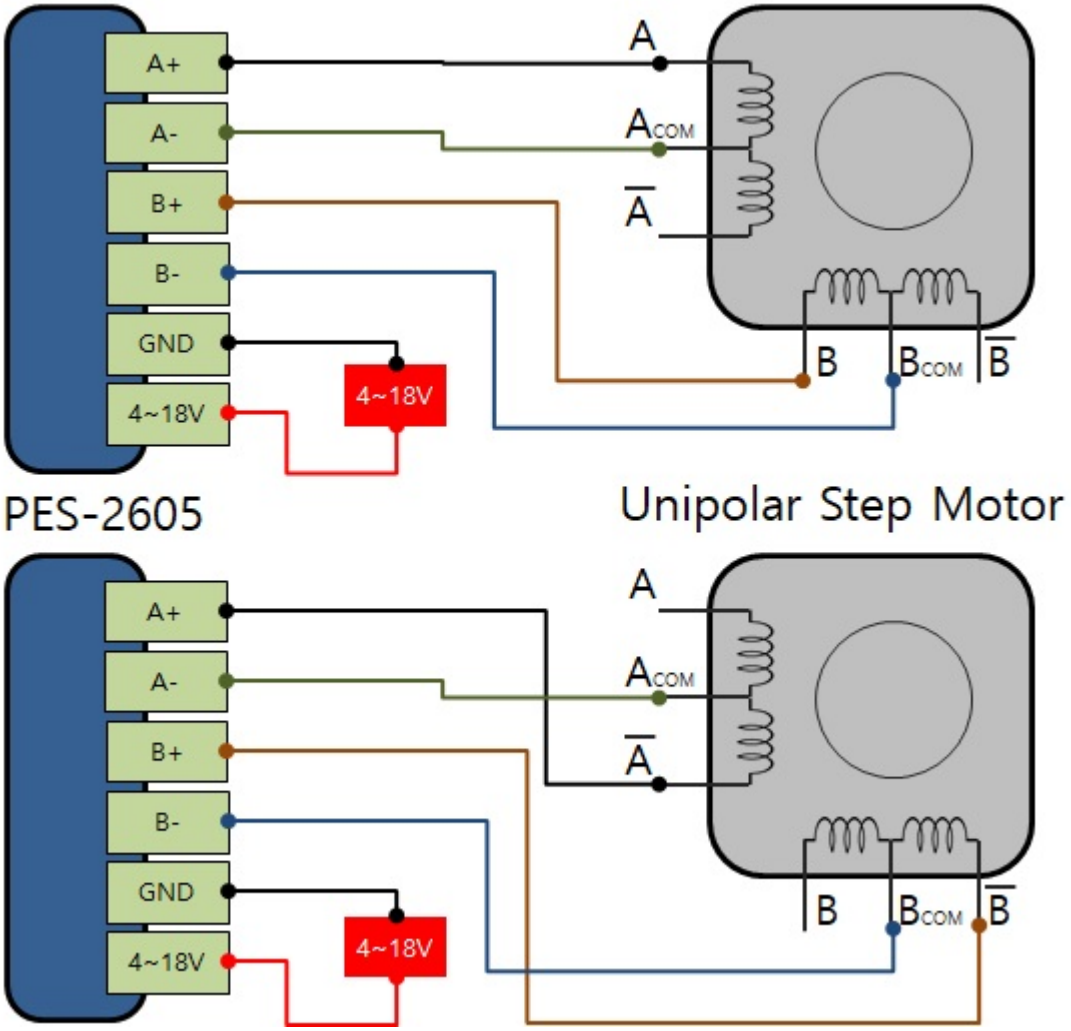
Connecting to a bipolar stepper motor

This board is a bipolar stepper motor controller. 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 this board refer to two examples below.



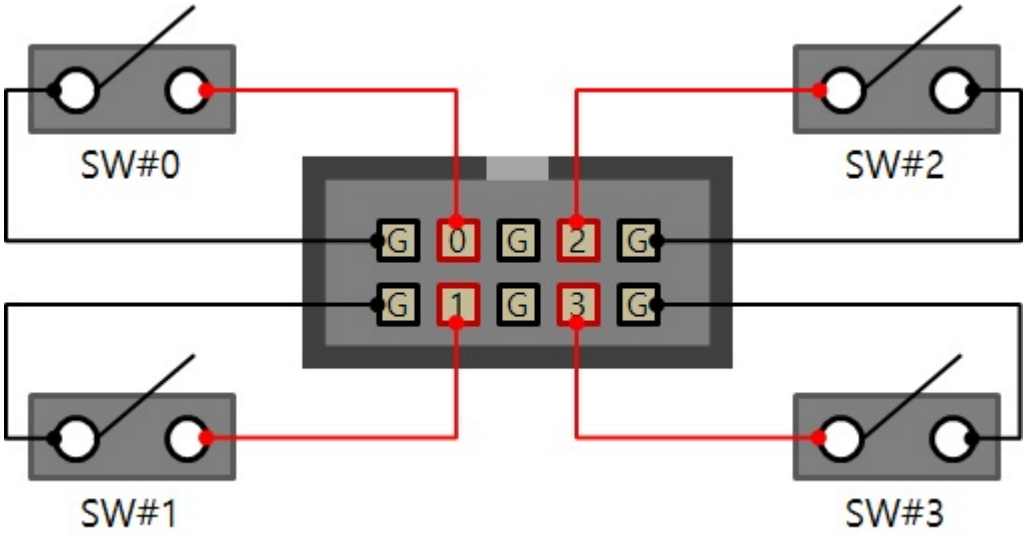
Supplying power

The GND pin and the 4~18V pin 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

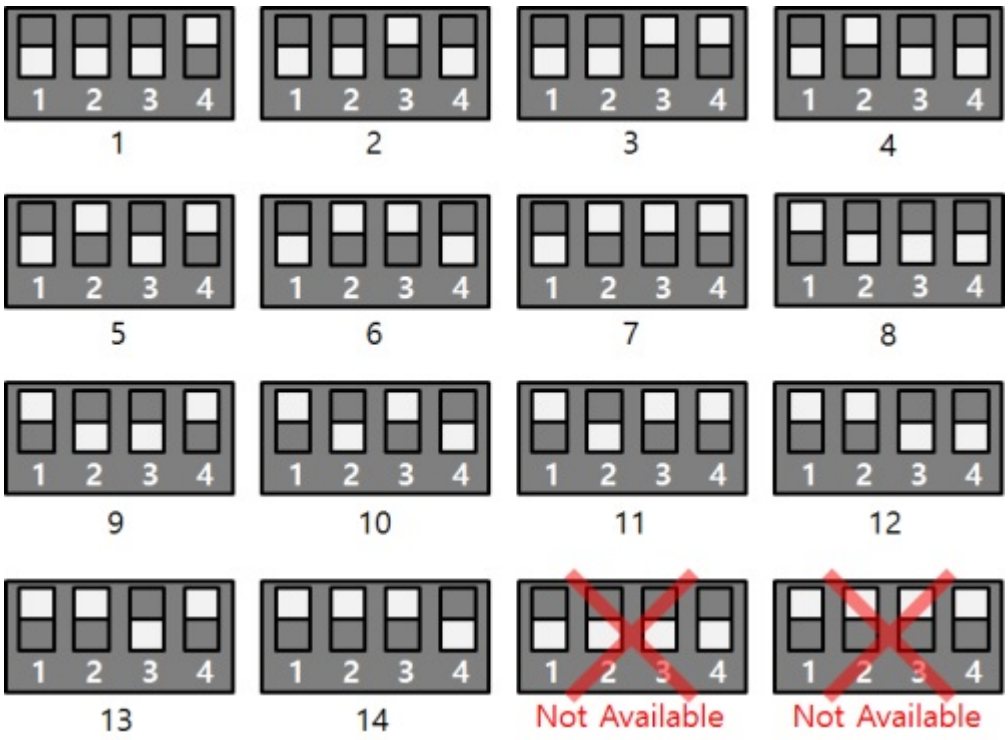
This port is used to connect limit switches and cannot be used for other purposes.
An example of connection with limit switches is as follows:



- 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 shield identifies each smart expansion board. So, each smart expansion board, which is connected to a PHPoC shield, 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

This board has STS LED which indicates the board's status.

State	Operation
ready	Repeat On/Off in every second
running	shortly blink 4 times every second under OFF state
locked	shortly blink every second under ON state
Fail to communicate with PHPoC shield	Off

How to Use

This board can be used by steps as follows.

1. Connect to a PHPoC Shield and an Arduino

It is not possible to use this board alone. Please be sure that connection to a PHPoC Shield and an Arduino.

2. Install Libraries for Arduino

Install PHPoC and PhpocExpansion library via library manager on Arduino IDE. Both libraries are required to use PHPoC shield and this board. Refer to the manual pages below for detail about the libraries.

- PHPoC shield library reference

3. Use Sample Codes

Use sample codes in libraries and examples in this manual.

Class and Functions

Class

To use this extension board, use the ExpansionStepper class of the PHPoC Expansion library.

Member Functions

Available member functions of the ExpansionStepper class are as follows:

Member Function	Description
int getPID(void)	get the product's ID
char *getName(void)	get the product's name
ExpansionStepper(int sid)	create an instance of a motor port
void reset(void)	stop a motor and initialize settings
void setMode(int mode)	set a division rate
void setVrefStop(int vref)	set a current limiting of stop state
void setVrefDrive(int vref)	set a current limiting of run state
void setVrefLock(int vref)	set a current limiting of lock state
void setResonance(int low, int high)	set a resonance range
void setSpeed(long speed)	set a speed
void setAccel(long accel)	set accelration and deceleration
void setPosition(long pos)	set a counter position
int getState(void)	get operation states
long getPosition(void)	get a counter position
void stepMove(long step)	drive a motor based on the initial position
void stepGoto(long pos)	drive a motor based on the current position
void stepGotoSW(int id, int dir)	drive a motor for initial positioning
void stop(long decel = -1)	stop a motor
void setEioMode(int id, int mode)	set digital input ports
int getEio(int id)	get a state of digital input ports

Settings

Setting a Division Rate

Set a division rate of micro step by using setMode() function.

```
step.setMode(mode)
```

- mode - the division rate of micro step

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

Current Limiting

Before starting the stepper motor, it is necessary to set the limit current to maintain each state for the following three states.

state	function for current limiting
stop	step.setVrefStop(vref)
drive	step.setVrefDrive(vref)
lock	step.setVrefLock(vref)

- vref - The amount of limiting current (0 ~ 15)

  ※ Note : There are 16 levels for limiting current from 0 to 15. If you set the vref to 5, the current is limited to 5 of 15.

Setting a Resonance Range

You can set a resonance range by using setResonance() function.

```
step.setResonance(low, high)
```

- low - the lowest value of the range in pps unit
- high - the highest value of the range in pps unit

  The unit of both values are pps(pulse per second). Once the resonance range is set, the board drives a motor with the speed which is set by highest value of the range if the rotation speed

falls within the resonance range.

Setting a Rotation Speed

You can set a rotation speed by using setSpeed() function.

```
step.setSpeed(speed)
```

- speed - the rotation speed in pps unit

  The unit of this value is pps(pulse per second) and the board provides 240,000[pps] as its maximum speed. However, the actual maximum speed depends on the type, voltage and loads of the stepper motor.

Setting Acceleration and Deceleration

You can set both acceleration and deceleration by using setAccel() function.

```
step.setAccel(accel)
step.setAccel(accel, decel)
```

- accel - acceleration in pps/s unit
- decel - deceleration in pps/s unit

  The unit of both is pps/s(pps per second) and the board provides 2,400,000[pps/s] as its maximum value. If the deceleration is omitted, it is automatically set to the same value of the acceleration.

Setting a Counter Position

You can set a counter position by using setPosition() function.

```
step.setPosition(pos)
```

- pos - the counter position

  This value is a signed 32-bit integer and can have a value between -1000000000(1 billion) and +1000000000. This setting is valid only when controlling stepper motor with stepGoto() and is not reflected when controlling with stepMove().

Setting Digital Input Ports

You can set digital input ports by using setEioMode() function.

```
step.setEioMode(id, mode)
```

- id - the id of digital input ports (0 ~ 3)
- mode - the type of the digital input port

mode	description
0	normal input
otherwise	control lock

Getting States

Getting Operation States

You can get the operation state by using getState() function.

```
state = step.getState()
```

This function returns the operation state of a stepper motor.
Returned values are as follows:

value	state
0	stopped
1	control locked
otherwise	running

Example

- source code for Arduino

```
#include <PhpocExpansion.h>
#include <Phpoc.h>

byte spcId = 1;

ExpansionStepper step(spcId);

int state;

void setup() {
    Serial.begin(9600);
    while(!Serial)
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();

    Serial.println(step.getPID());
    Serial.println(step.getName());

    step.setMode(4);
    step.setVrefStop(2);
    step.setVrefDrive(8);
    step.setResonance(120, 250);
    step.setSpeed(400);
    step.setAccel(800);

    step.stepGoto(400);

    while(state = step.getState()) {
```

```
        Serial.print("state: ");
        Serial.println(state);
        delay(200);
    }

    Serial.print("state: ");
    Serial.println(state);
}

void loop() {

}
```

- the output example

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

Getting a Counter Position

You can get a counter position by using getPosition() function.

```
pos = step.getPosition()
```

This function returns the current counter position of a stepper motor.

Example

- source code for Arduino

```
#include <PhpocExpansion.h>
#include <Phpoc.h>

byte spcId = 1;

ExpansionStepper step(spcId);

int pos = -400;

void setup() {
    Serial.begin(9600);
    while(!Serial)
```

```
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();

    Serial.println(step.getPID());
    Serial.println(step.getName());

    step.setMode(4);
    step.setVrefStop(2);
    step.setVrefDrive(8);
    step.setResonance(120, 250);
    step.setSpeed(400);
    step.setAccel(800);
    step.setPosition(pos);

    step.stepGoto(400);

    while(step.getState()) {
        pos = step.getPosition();
        Serial.print("position: ");
        Serial.println(pos);
        delay(200);
    }
}

void loop() {

}
```

- the output example

```
position: -400
position: -369
position: -302
position: -214
position: -126
position: -39
position: 48
position: 135
position: 223
position: 310
position: 375
position: 400
```

Getting States of Digital Input Ports

You can get states of digital input ports by using getEio() function.

```
state = step.getEio(id)
```

- id - the ID of input ports(0 ~ 3)

The returned states are as follows:

returned value	state
0	LOW
1	HIGH (default)

Example

- source code for Arduino

```
#include <PhpocExpansion.h>
#include <Phpoc.h>

byte spcId = 1;

ExpansionStepper step(spcId);

void setup() {
    Serial.begin(9600);
    while(!Serial)
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();

    Serial.println(step.getPID());
    Serial.println(step.getName());
}

void loop() {
    Serial.print(step.getEio(0));
    Serial.print(step.getEio(1));
    Serial.print(step.getEio(2));
    Serial.print(step.getEio(3));
    Serial.println();
    delay(1000);
}
```

- the output example

```
1111
1111
1110
0110
...
```

Controlling by stepGoto()

Controlling by stepGoto()

You can control a stepper motor relative to the initial position, not the current position by using stepGoto() function. This function can be executed even when the motor is running.

```
step.stepGoto(pos)
step.stepGoto(pos, speed)
step.stepGoto(pos, speed, accel)
```

- pos - the target counter position

  You can define the forward or reverse rotation by adding "+" or "-" in front of the pos value. (default : "+") The target counter position is relative to the initial position, not the current position.

- speed - rotation speed in pps unit

  ※ pps: pulse per second

- accel - acceleration in pps/s unit

  The deceleration is applied with the same value with the acceleration.

Example

- source code for Arduino

```
#include <PhpocExpansion.h>
#include <Phpoc.h>

byte spcId = 1;

ExpansionStepper step(spcId);

void setup() {
    Serial.begin(9600);
    while(!Serial)
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();

    Serial.println(step.getPID());
```

```
    Serial.println(step.getName());

    step.setMode(4);
    step.setVrefStop(2);
    step.setVrefDrive(8);
    step.setResonance(120, 250);
    step.setSpeed(400);
    step.setAccel(0, 0);
    step.setPosition(0);

    step.stepGoto(400);
    delay(2000);
    step.stepGoto(-400);
    delay(2000);
    step.stepGoto(400);
    delay(2000);

    while(step.getState()) {
        delay(1);
    }
}

void loop() {

}
```

Controlling by stepMove()

Controlling by stepMove()

You can control a stepper motor relative to the current position, not the initial position by using stepMove() function. This function can be executed only when the motor is NOT running.

```
step.stepMove(step)
step.stepMove(step, speed)
step.stepMove(step, speed, accel)
```

- step - the target counter position

  You can define the forward or reverse rotation by adding "+" or "-" in front of the pos value. (default : "+") The target counter position is relative to the current position, not the initial position.

- speed - rotation speed in pps unit

  ※ pps: pulse per second

- accel - acceleration in pps/s unit

  The deceleration is applied with the same value with the acceleration.

Example

- source code for Arduino

```
#include <PhpocExpansion.h>
#include <Phpoc.h>

byte spcId = 1;

ExpansionStepper step(spcId);

void setup() {
    Serial.begin(9600);
    while(!Serial)
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();

    Serial.println(step.getPID());
```

```
    Serial.println(step.getName());

    step.setMode(4);
    step.setVrefStop(2);
    step.setVrefDrive(8);
    step.setResonance(120, 250);
    step.setSpeed(400);
    step.setAccel(0, 0);
    step.setPosition(0);

    step.stepMove(400);
    delay(2000);
    step.stepMove(-400);
    delay(2000);
    step.stepMove(400);
    delay(2000);

    while(step.getState()) {
        delay(1);
    }
}

void loop() {

}
```

Stopping

Stopping

You can stop a motor by using stop() function.

```
step.stop()
step.stop(decel)
```

- decel - deceleration in pps/s unit

  ※ pps: pulse per second

Example

- source code for Arduino

```
#include <PhpocExpansion.h>
#include <Phpoc.h>

byte spcId = 1;

ExpansionStepper step(spcId);

int state;

void setup() {
    Serial.begin(9600);
    while(!Serial)
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();

    Serial.println(step.getPID());
    Serial.println(step.getName());

    step.setMode(4);
    step.setVrefStop(2);
    step.setVrefDrive(8);
    step.setSpeed(400);
    step.setAccel(1000);

    step.stepGoto(40000);
}

void loop() {
```

```
    state = step.getEio(0);

    if(state == 0)
        step.stop();

    delay(1);
}
```

Settings the Initial Position

By connecting a limit switch to the digital input port of this board, 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 Getting States and use it as the initial position.

Stop automatically

Stop automatically

You can automatically stop a stepper motor when a limit switch is closed by using stepGotoSW() function. The state of a motor will be stop if it stopped by this function.

```
step.stepGotoSW(id, dir)
step.stepGotoSW(id, dir, speed)
step.stepGotoSW(id, dir, speed, accel)
```

- id - an ID of a input port(0 ~ 3)
- dir - direction of rotation

dir	direction of rotation
0 or higher than 0	forward
otherwise	reverse

- speed - rotation speed in pps unit

  ※ pps: pulse per second

- accel - acceleration in pps/s unit

  The deceleration is applied with the same value with the acceleration.

Example

- source code for Arduino

```
#include <PhpocExpansion.h>
#include <Phpoc.h>

byte spcId = 1;

ExpansionStepper step(spcId);

void setup() {
    Serial.begin(9600);
    while(!Serial)
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();
```

```
    Serial.println(step.getPID());
    Serial.println(step.getName());

    step.setMode(4);
    step.setVrefDrive(8);
    step.setSpeed(400);
    step.setAccel(0);

    // rotate until digital input 0 is LOW
    Serial.print("find positive limit ...");
    step.stepGotoSW(0, 1);
    while(step.getState()) {
        delay(1);
    }
    Serial.println("done");

    delay(1000);

    // rotate until digital input 1 is LOW
    Serial.print("find negative limit ...");
    step.stepGotoSW(1, -1);
    while(step.getState()) {
        delay(1);
    }
    Serial.println("done");
}

void loop() {

}
```

- the output result

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

Stop manually

Stop manually

You can manually stop a stepper motor when a limit switch is closed by using stop() function. The states of limit switches can be monitored by getEio() function.

Example

- an example of stopping a motor manually

```
#include <PhpocExpansion.h>
#include <Phpoc.h>

byte spcId = 1;

ExpansionStepper step(spcId);

int state;

void setup() {
    Serial.begin(9600);
    while(!Serial)
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();

    Serial.println(step.getPID());
    Serial.println(step.getName());

    step.setMode(4);
    step.setVrefStop(2);
    step.setVrefDrive(8);
    step.setSpeed(400);
    step.setAccel(1000);

    step.stepGoto(40000);
}

void loop() {
    state = step.getEio(0);

    if(state == 0)
        step.stop();

    delay(1);
}
```

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 to control lock referring to Settings chapter.

Unlock

You can unlock a stepper motor by using unlock() function.

```
step.unlock()
```

When you execute the unlock function, the motor status changes from the lock state to the stop 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().

Example

- source code for Arduino

```
#include <PhpocExpansion.h>
#include <Phpoc.h>

byte spcId = 1;

ExpansionStepper step(spcId);

int state;

void setup() {
    Serial.begin(9600);
    while(!Serial)
        ;

    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET);
    Expansion.begin();
```

```
    Serial.println(step.getPID());
    Serial.println(step.getName());

    step.setMode(4);
    step.setVrefStop(2);
    step.setVrefDrive(8);
    step.setVrefLock(8);
    step.setSpeed(400);
    step.setAccel(4000);

    step.setEioMode(0, 1);
    step.setEioMode(1, 1);
    step.setEioMode(2, 1);
    step.setEioMode(3, 1);

    step.stepGoto(4000);

    while(step.getState() > 1) {
        delay(1);
    }

    // state: 0 - stop, 1 - locked
    Serial.print("step_state ");
    Serial.println(step.getState());

    step.unlock();

    // state: 0 - stop, 1 - locked
    Serial.print("step_state ");
    Serial.println(step.getState());
}

void loop() {

}
```

- the output result

```
step_state 1
step_state 0
```