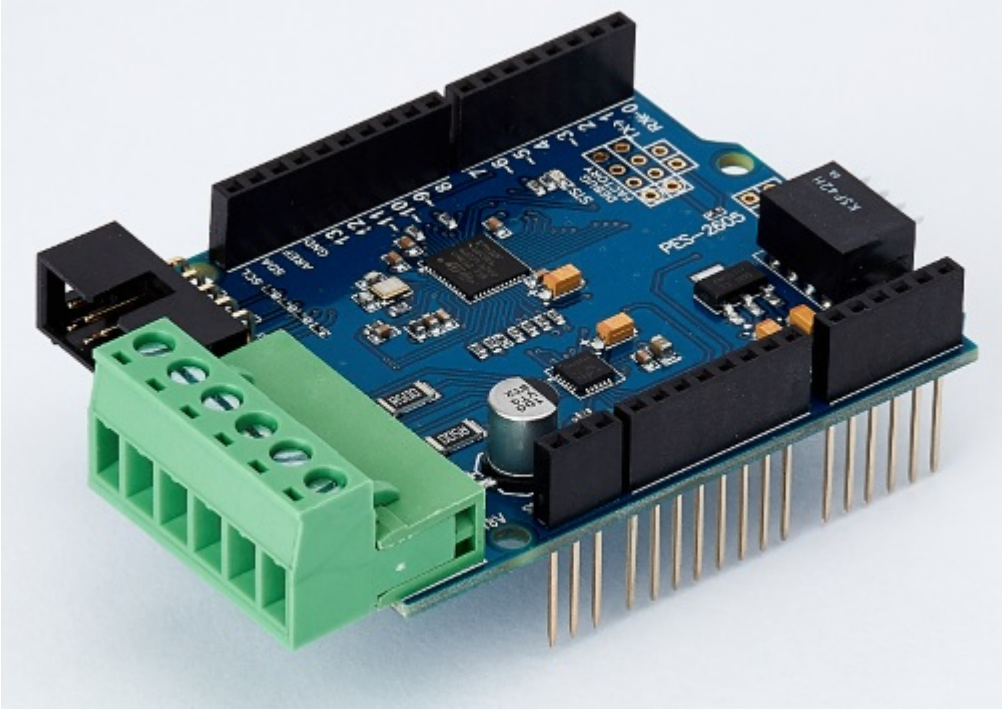
제품 소개



PES-2605

스텝모터 제어기 PES-2605는 아두이노용 PHPoC 쉴드 제품 전용 스마트 확장보드입니다. 이 보드를 이용하면 아두이노 스케치를 통해 원격으로 스텝모터를 정밀하게 컨트롤 할 수 있습니다.

PES-2605의 주요 특징

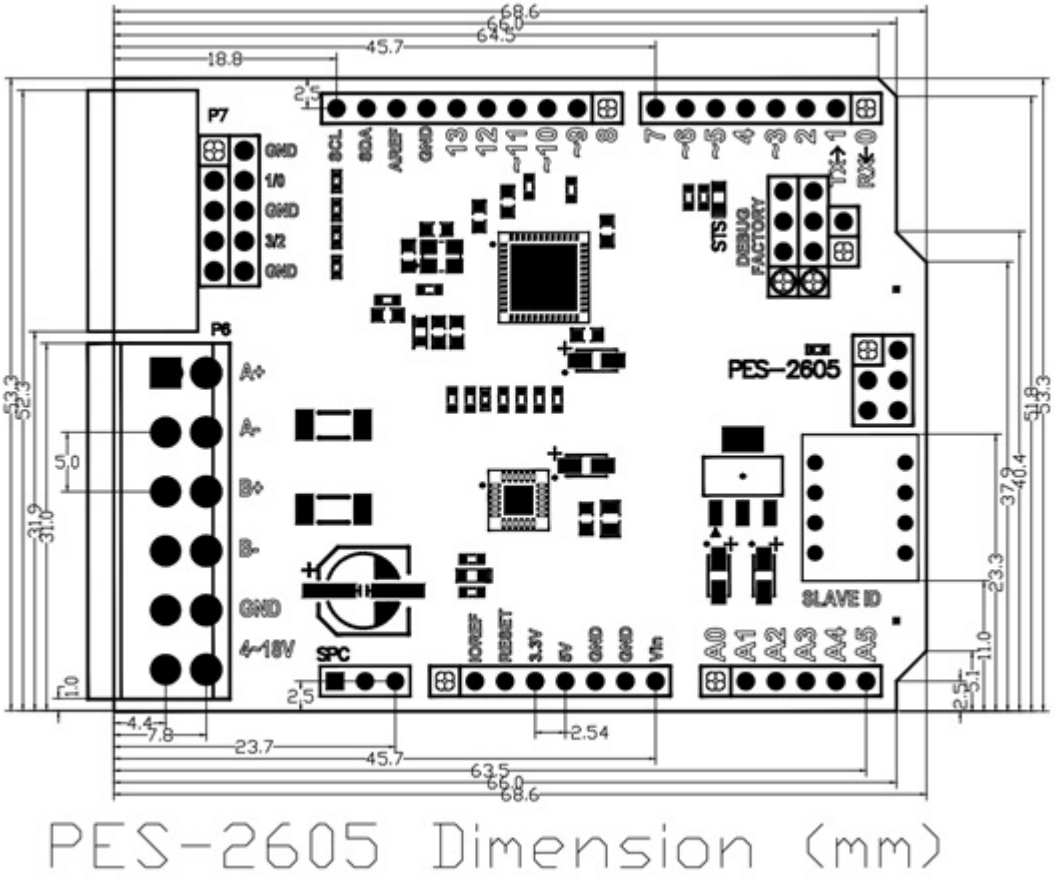
- 바이폴라 스텝모터 제어기
- 드라이빙 모드: 마이크로 스텝(최대 분주비 1/32)
- 모터 전압: 4 ~ 18V [DC]
- 모터 전류: 코일당 최대 1A

※ 주의 : 이 확장보드를 사용하기 위해서는 반드시 아두이노와 R2 이상 버전의 PHPoC 쉴드가 필요합니다!

> PHPoC 쉴드용 스마트 확장보드란?
>
> PHPoC 쉴드용 스마트 확장보드는 자체 디바이스와 전용 펌웨어를 내장하고 있습니다. 이 보드는 PHPoC 쉴드와 전용 통신 포트를 이용해 마스터-슬레이브 방식으로 통신합니다. 하나의 PHPoC 쉴드에 여러개의 스마트 확장보드를 연결할 수 있으며 각각의 스마트 확장보드에는 반드시 슬레이브 아이디를 설정해야 합니다.

치수

제품 본체



※ 치수(단위 : mm)는 제품 상태 및 재는 각도 등에 따라 약간의 오차가 있을 수 있습니다.

터미널블록

이 보드는 6폴 터미널블록을 사용합니다. 치수는 각 터미널블록의 데이터시트를 참조하시기 바랍니다.

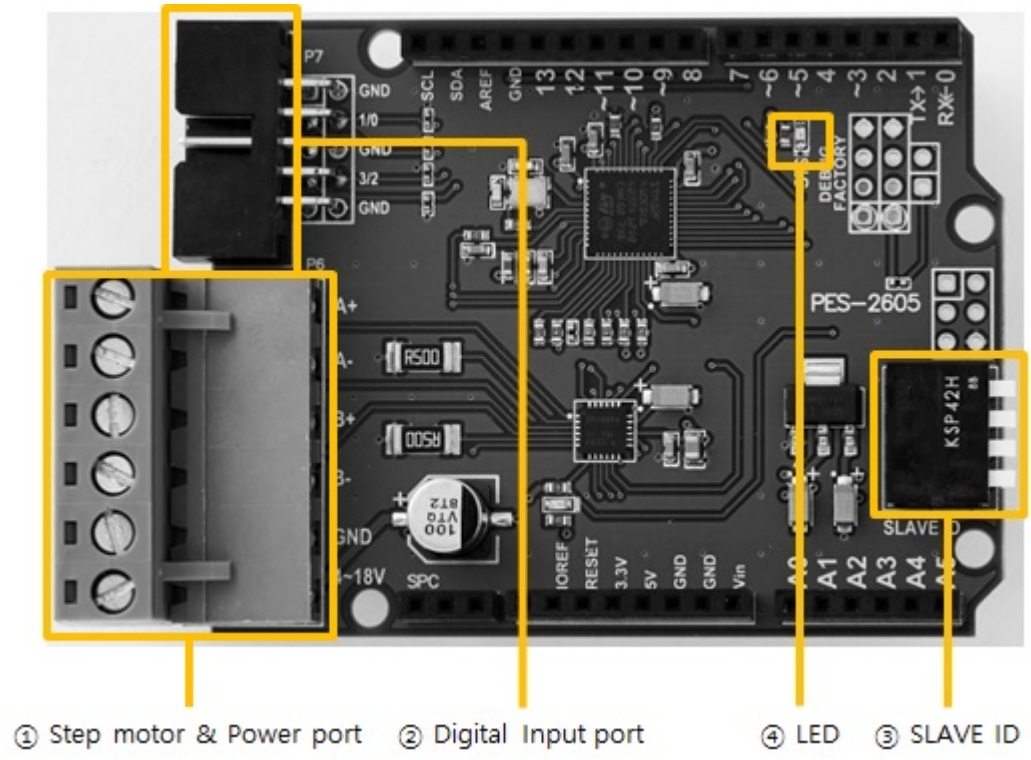
- T형 터미널블록 데이터시트
- S형 터미널블록 데이터시트

회로도

PES-2605의 회로도 입니다.

- PES-2605-V12-PO.pdf

레이아웃



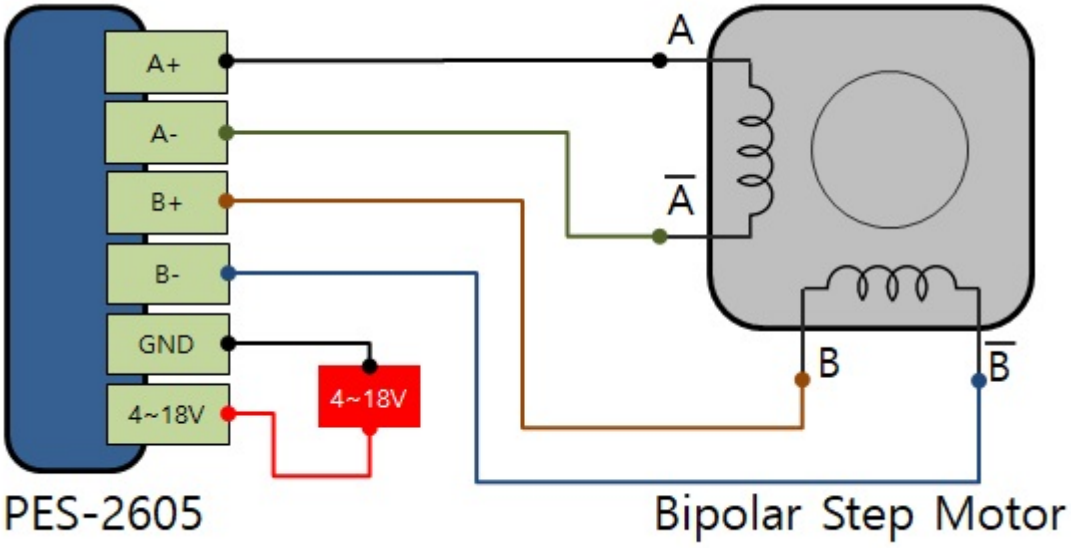
1. 스텝모터 & 전원포트

스텝모터 & 전원포트는 5mm간격의 1 by 6 터미널 블록으로 되어 있습니다.

구분	설명
A+, A-, B+, B-	스텝모터 연결부
4~18V, GND	스텝모터 구동전원 연결부

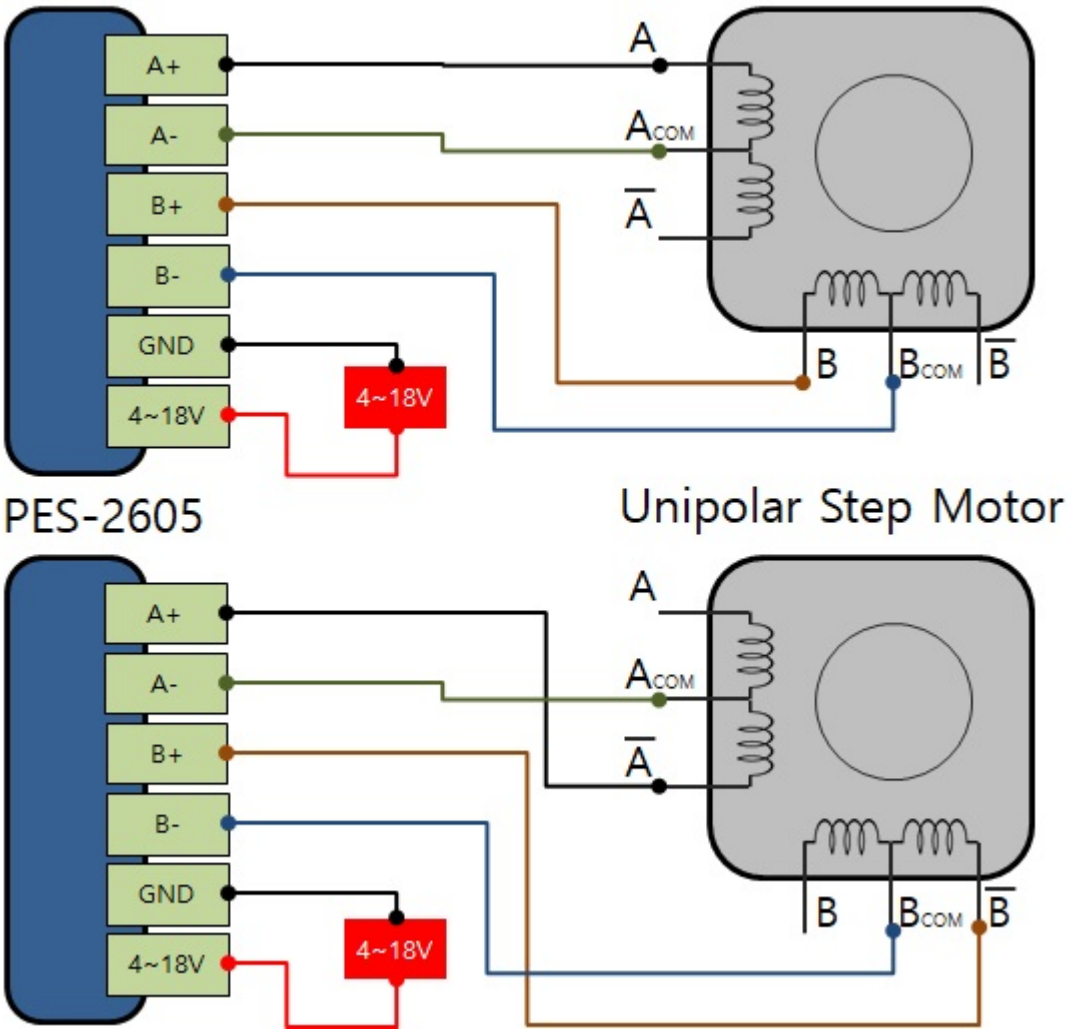
바이폴라 스텝모터 연결

이 보드는 기본적으로 바이폴라(4선식) 스텝모터용 제어기입니다. 바이폴라 스텝모터의 연결 예는 다음과 같습니다.



유니폴라 스텝모터 연결

이 보드는 유니폴라 스텝모터도 연결이 가능합니다. 유니폴라 스텝모터의 연결 예는 다음과 같습니다.

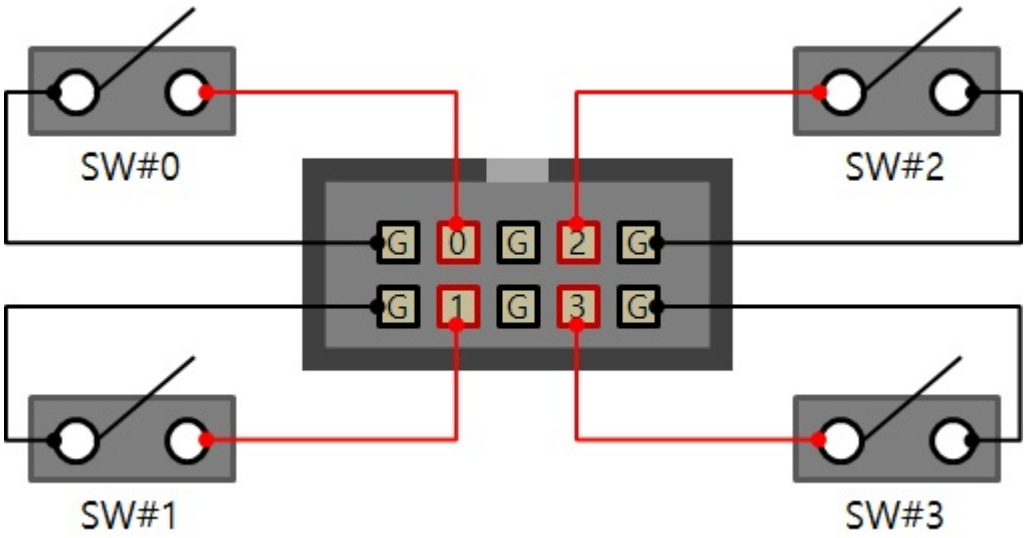


전원 연결

GND와 4~18V포트는 스텝모터 구동 전원(DC 4 ~ 18V)을 입력하는 포트입니다. 스텝모터 구동 시 반드시 이 포트를 통해 전원을 공급해야 합니다. DC극성을 꼭 확인하여 전원을 입력해주시기 바랍니다.

2. 디지털 입력포트

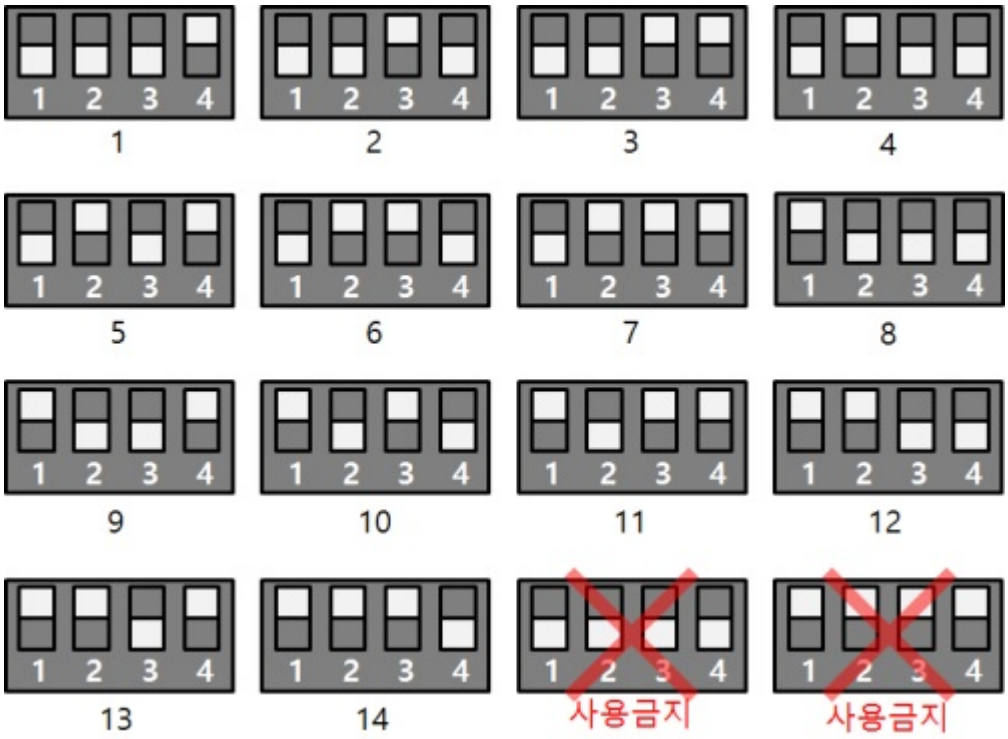
이 포트는 리미트스위치 연결 시 사용되는 포트이며 다른 용도로는 사용할 수 없습니다.
리미트스위치 연결 예는 다음과 같습니다.



- G는 그라운드(GND)이며 내부적으로 모두 연결되어 있습니다.
- 0 ~ 3은 디지털 입력포트이며 기본적으로 풀업(Pull-Up)되어 있습니다.

3. 슬레이브 아이디 스위치

슬레이브 아이디는 PHPoC 쉴드가 스마트 확장보드 각각을 구분하는데 사용됩니다. 따라서 PHPoC 쉴드에 연결되는 각 스마트 확장보드는 고유한 슬레이브 아이디를 사용해야 합니다. 슬레이브 아이디는 1부터 14까지 14개 중 하나로 설정할 수 있으며 다음과 같이 4개의 딥스위치를 조정하여 설정합니다.



4. LED

이 보드에는 상태를 나타내는 STS LED가 있습니다.

상태	LED 동작
대기	1초마다 켜짐/꺼짐 반복
제어 중	꺼진 상태에서 1초에 4번 짧게 켜짐
제어 잠김	켜진 상태에서 1초에 1번 짧게 꺼짐
제어 불가능	꺼짐

사용하기

이 보드를 사용하는 방법은 다음과 같습니다.

1. PHPoC 쉴드와 아두이노에 연결

이 보드는 단독으로 사용할 수 없습니다. 반드시 아두이노와 아두이노용 PHPoC 쉴드에 연결하여 사용하시기 바랍니다.

2. 아두이노용 라이브러리 설치

아두이노 IDE의 라이브러리 매니저를 통해 Phpoc 라이브러리와 PhpocExpansion 라이브러리를 설치합니다. 아두이노용 PHPoC 쉴드와 스마트 확장보드를 사용하려면 반드시 두 라이브러리를 모두 설치해야 합니다. 라이브러리에 대한 자세한 내용은 다음 문서를 참조하시기 바랍니다.

- PHPoC 쉴드 라이브러리 레퍼런스

3. 예제코드 활용

본 매뉴얼과 라이브러리에 포함된 예제코드를 활용하여 프로그래밍 하시기 바랍니다.

클래스 및 함수

클래스

이 확장보드를 사용하기 위해서는 아두이노 PHPoC 라이브러리의 ExpansionStepper 클래스를 사용합니다.

멤버 함수

ExpansionStepper 클래스의 사용 가능한 멤버함수는 다음과 같습니다.

멤버 함수	설명
int getPID(void)	제품 아이디 읽기
char *getName(void)	제품명 읽기
ExpansionStepper(int sid)	모터 포트의 인스턴스 생성
void reset(void)	모터 구동 정지 및 설정 초기화
void setMode(int mode)	마이크로 스텝 분주비 설정
void setVrefStop(int vref)	정지상태의 제한전류 설정
void setVrefDrive(int vref)	동작상태의 제한전류 설정
void setVrefLock(int vref)	제어 잠금상태의 제한전류 설정
void setResonance(int low, int high)	공진 범위 설정
void setSpeed(long speed)	회전 속도 설정
void setAccel(long accel)	가속도/감속도 설정
void setPosition(long pos)	카운터 값 초기화 또는 변경
int getState(void)	모터 상태 확인
long getPosition(void)	모터의 현재 카운터 값 확인
void stepMove(long step)	상대위치로 모터 구동
void stepGoto(long pos)	절대위치로 모터 구동
void stepGotoSW(int id, int dir)	초기위치 설정을 위한 모터 구동
void stop(long decel = -1)	모터 구동 정지
void setEioMode(int id, int mode)	디지털 입력포트 유형 설정
int getEio(int id)	디지털 입력포트 상태 확인

설정하기

마이크로 스텝 분주비 설정

setMode() 함수로 마이크로 스텝의 분주비를 설정하십시오.

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

- mode - 마이크로 스텝 분주비

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

제한전류 설정

스텝모터를 구동하기에 앞서 다음 세 가지 상태에 대하여 각각의 상태를 유지하기 위한 제한전류를 반드시 설정해야 합니다.

상태	제한 전류 설정 함수
정지 상태	step.setVrefStop(vref)
동작 상태	step.setVrefDrive(vref)
제어 잠금상태	step.setVrefLock(vref)

- vref - 각각의 상태를 유지하기 위한 제한전류의 양

  ※ 참고: 제한전류는 각 상태별로 0 ~ 15까지 총 16단계로 설정할 수 있습니다. 이 값에 5를 설정하면 해당 상태에서의 전류를 5/15로 제한합니다.

공진 범위 설정

setResonance() 함수를 이용하여 공진 범위를 설정할 수 있습니다.

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

- low - 공진 범위의 최소 값
- high - 공진 범위의 최대 값

  공진 범위의 설정 단위는 pps(pulse per second)입니다. 공진 범위를 설정하면 스텝모터의 회전속도가 공진 범위 안에 해당할 때 공진 범위의 최대 값으로 제어합니다.

회전속도 설정

setSpeed() 함수를 이용하여 스텝모터의 회전 속도를 설정할 수 있습니다.

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

- speed - 모터의 회전속도

  회전속도의 단위는 pps(pulse per second)이며, 이 보드는 최대 240,000[pps]까지 설정할 수 있습니다. 그러나 실제 최대속도는 스텝모터의 종류/전압/부하에 따라 달라질 수 있습니다.

가속도 및 감속도 설정

setAccel() 함수를 이용하여 스텝모터의 가속도 및 감속도를 설정할 수 있습니다.

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

- accel - 가속도
- decel - 감속도

  가속도와 감속도의 설정 단위는 pps/s(pps per second)이며, 이 보드는 최대 2,400,000[pps/s]까지 설정할 수 있습니다. 감속도를 입력하지 않으면 가속도에 입력한 값이 감속도에 자동으로 설정됩니다.

카운터 위치 설정

setPosition() 함수를 이용하여 카운터 위치를 초기화 또는 변경할 수 있습니다.

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

- pos - 카운터 위치

  카운터 위치의 설정 단위는 부호가 있는 32비트 정수 형태이고 입력 가능한 범위는 -1000000000(10억) ~ +1000000000입니다. 또한 카운터 위치 설정은 stepGoto()함수로 스텝모터를 제어할 때만 유효하고 stepMove()함수로 제어하는 경우에는 반영되지 않습니다.

디지털 입력포트 설정

setEioMode() 함수를 이용하여 디지털 입력포트를 설정할 수 있습니다.

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

- id - 디지털 입력포트의 아이디 (0 ~ 3)
- mode - 입력포트의 입력모드

mode	입력 모드
0	일반입력 모드
그 외	제어잠금 모드

상태 확인하기

스텝모터 동작상태 확인

getState() 함수로 스텝모터의 동작상태를 확인할 수 있습니다.

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

이 함수는 스텝모터의 동작 상태를 나타내는 코드를 반환합니다. 반환되는 동작상태의 종류는 다음과 같습니다.

반환 값	상태
0	정지됨
1	제어 잠김(lock)
그 외	동작 중

예제

- 아두이노 소스코드

```
#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() {

}
```

- 출력 예

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

카운터 위치 확인

getPosition() 함수를 이용하여 현재 카운터 위치를 확인할 수 있습니다.

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

이 함수는 스텝모터의 현재 카운터 위치를 반환합니다.

예제

- 아두이노 소스코드

```
#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() {

}
```

- 출력 예

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

디지털 입력포트의 상태 확인

getEio() 함수로 디지털 입력포트의 현재 상태를 확인할 수 있습니다.

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

- id - 입력포트의 아이디(0 ~ 3)

반환되는 입력포트 상태의 종류는 다음과 같습니다.

반환 값	상태
0	LOW
1	HIGH (기본 값)

예제

- 아두이노 소스코드

```
#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);
}
```

- 출력 예

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

stepGoto()로 제어하기

stepGoto()함수로 제어

스텝모터를 모터의 현재 위치가 아닌 초기 위치를 기준으로 제어하고자 할 때 stepGoto()함수를 사용합니다. 이 함수는 모터가 동작 중인 상태에서도 사용이 가능합니다. 모터가 동작 중일 때 새로운 setpGoto()함수가 실행되면 그 즉시 해당 함수를 수행합니다.

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

- pos - 목표 카운터 위치

  pos는 "+" 또는 "-" 부호를 앞에 붙여서 정회전 또는 역회전을 나타낼 수 있습니다. (생략시 "+")
  목표 카운터의 위치는 카운터의 현재위치가 아닌 초기 위치가 기준이 됩니다.

- speed - 회전 속도, pps 단위

  ※ pps: pulse per second

- accel - 가속도, pps/s 단위

  accel에 값을 설정하면 해당 값은 가속도뿐만 아니라 감속도에도 동일하게 적용됩니다.

예제

- 아두이노 소스코드

```
#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() {

}
```

stepMove()로 제어하기

stepMove()함수로 제어

스텝모터를 모터의 초기 위치가 아닌 현재 위치를 기준으로 제어하고자 할 때 stepMove()함수를 사용합니다. 이 함수는 모터가 정지상태일 때 사용이 가능합니다.

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

- step - 목표 카운터 위치

  step은 "+" 또는 "-" 부호를 앞에 붙여서 정회전 또는 역회전을 나타낼 수 있습니다. (생략시 "+")
  목표 카운터의 위치는 초기 위치가 아닌 카운터의 현재위치가 기준이 됩니다.

- speed - 회전 속도, pps 단위

  ※ pps: pulse per second

- accel - 가속도, pps/s 단위

  accel에 값을 설정하면 해당 값은 가속도뿐만 아니라 감속도에도 동일하게 적용됩니다.

예제

- 아두이노 소스코드

```
#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() {

}
```

정지시키기

정지시키기

stop() 함수를 이용하여 동작 중인 스텝모터를 정지시킬 수 있습니다.

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

- decel - 감속도, pps/s 단위

  ※ pps: pulse per second

예제

- 아두이노 소스코드

```
#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);
}
```

초기위치 설정하기

이 보드의 디지털 입력포트에 리미트스위치를 연결하면 초기위치 설정이 필요한 시스템에서 초기위치를 설정할 수 있습니다. 설정하는 과정은 다음과 같습니다.

1. 모터 구동

   시스템을 시작하면 스텝모터를 초기위치방향으로 구동 시킵니다.

2. 모터 정지

   모터의 움직임에 의해 리미트스위치가 닫히면 모터를 정지시킵니다. 모터를 정지시킬 때 자동 또는 수동으로 정지시킬 수 있습니다.

   - 자동으로 정지시키기
   - 수동으로 정지시키기

3. 초기위치 설정

   모터가 정지되면 상태 확인하기를 참고하여 카운터의 현재 위치를 확인하고 이를 초기위치로 사용합니다.

자동으로 정지시키기

자동으로 정지시키기

stepGotoSW() 함수를 이용해 리미트스위치가 닫혔을 때 자동으로 스텝모터의 동작을 정지시킬 수 있습니다. 이 명령을 통해 모터가 정지되면 모터의 상태 값은 정지상태가 됩니다.

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

- id - 입력포트의 아이디(0 ~ 3)
- dir - 모터의 회전 방향

dir	회전 방향
0 또는 0보다 큰 정수	정 방향
그 외	역 방향

- speed - 회전 속도, pps 단위

  ※ pps: pulse per second

- accel - 가속도, pps/s 단위

  accel에 값을 설정하면 해당 값은 가속도 뿐만 아니라 감속도에도 동일하게 적용됩니다.

예제

- 아두이노 소스코드

```
#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() {

}
```

- 출력 결과

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

수동으로 정지시키기

수동으로 정지시키기

리미트스위치가 닫혔을 때 stop() 함수를 이용하면 수동으로 스텝모터의 동작을 정지시킬 수 있습니다. 이 때 리미트스위치의 입력 여부는 getEio() 함수를 이용합니다.

예제

- 아두이노 소스코드

```
#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);
}
```

제어잠금 및 해제

제어잠금 기능은 일종의 물리적인 보호 기능입니다. 이 기능은 디지털 입력포트에 리미트스위치를 연결하고 스위치가 닫히면 추가적인 제어를 불가능하게 하는 것입니다. 따라서 스텝모터의 동작범위를 제한할 수 있습니다.

제어잠금 상태

모터의 동작이 리미트스위치에 의해 멈추면 모터의 상태는 제어잠금 상태가 되어 잠금을 해제하기 전까지 추가적인 제어가 불가능합니다.

제어잠금 모드 설정

설정하기의 "디지털 입력포트 설정"을 참고하여 디지털 입력모드를 제어잠금 모드로 설정합니다.

제어잠금 상태의 해제

제어잠금 상태를 해제하는 함수는 unlock()함수입니다.

```
step.unlock()
```

unlock()함수를 실행하면 모터의 상태는 제어잠금 상태에서 정지 상태로 전환되고, 디지털 입력포트의 입력모드는 제어잠금 모드에서 일반입력 모드로 초기화 됩니다.

따라서 unlock() 이후에는 정상적으로 모터 제어가 가능합니다.

예제

- 아두이노 소스코드

```
#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() {

}
```

- 출력 결과

```
step_state 1
step_state 0
```