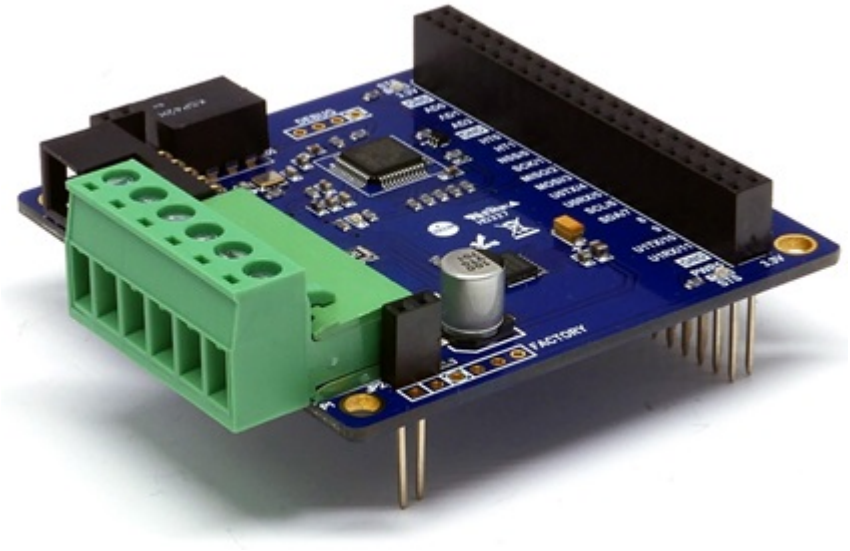
제품 소개



PES-2403

스텝모터 제어기 PES-2403은 PHPoC 보드형 제품 전용 스마트 확장보드입니다. 이 보드를 이용하면 여러 종류의 스텝모터를 손쉽게 컨트롤 할 수 있습니다.

PES-2403의 주요 특징

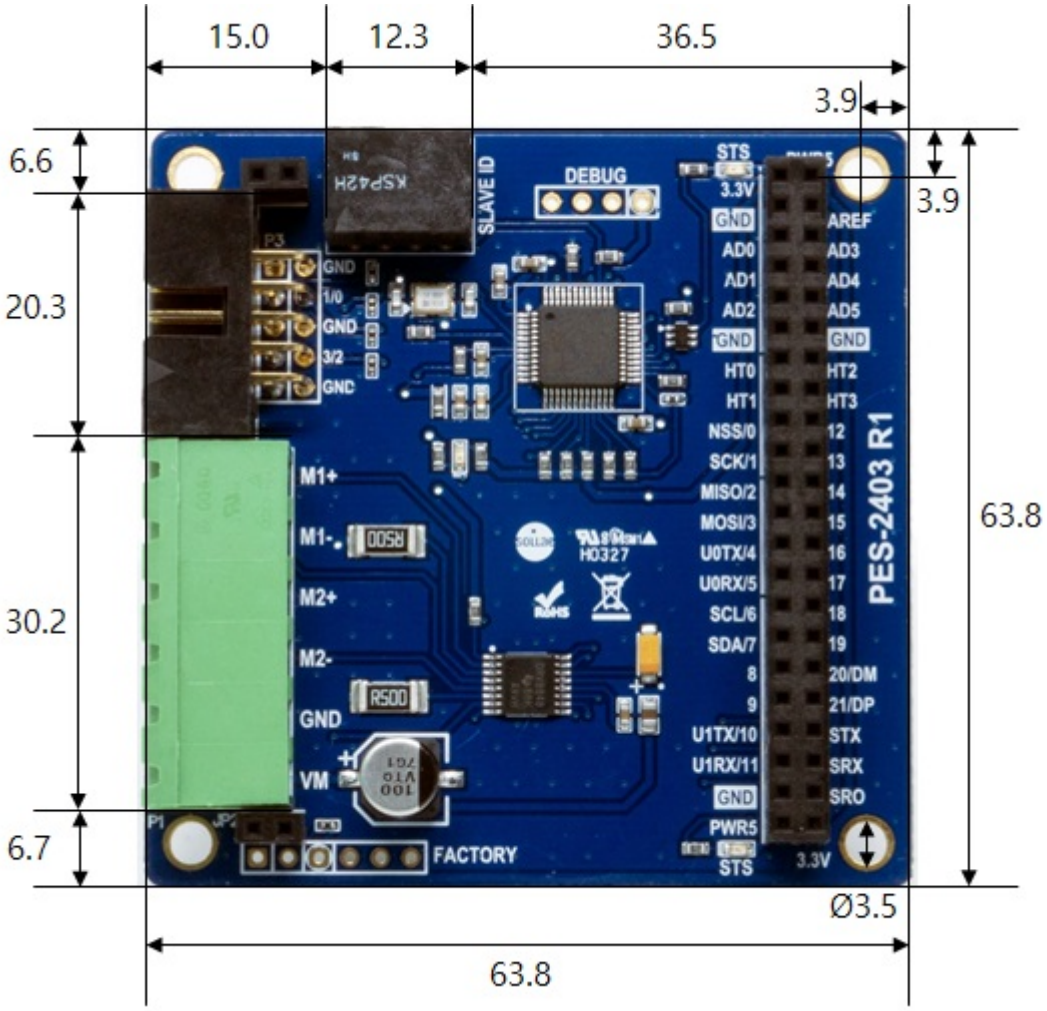
- 바이폴라 스텝모터 제어기
- 드라이빙 모드: full-step(2상 여자), half-step
- 모터 전압: 4 ~ 18V [DC]
- 모터 전류: 코일당 최대 1A

※ 주의: PES-2403을 사용하기 위해서는 펌웨어 버전 1.3.0이상의 PHPoC 보드가 필요합니다!

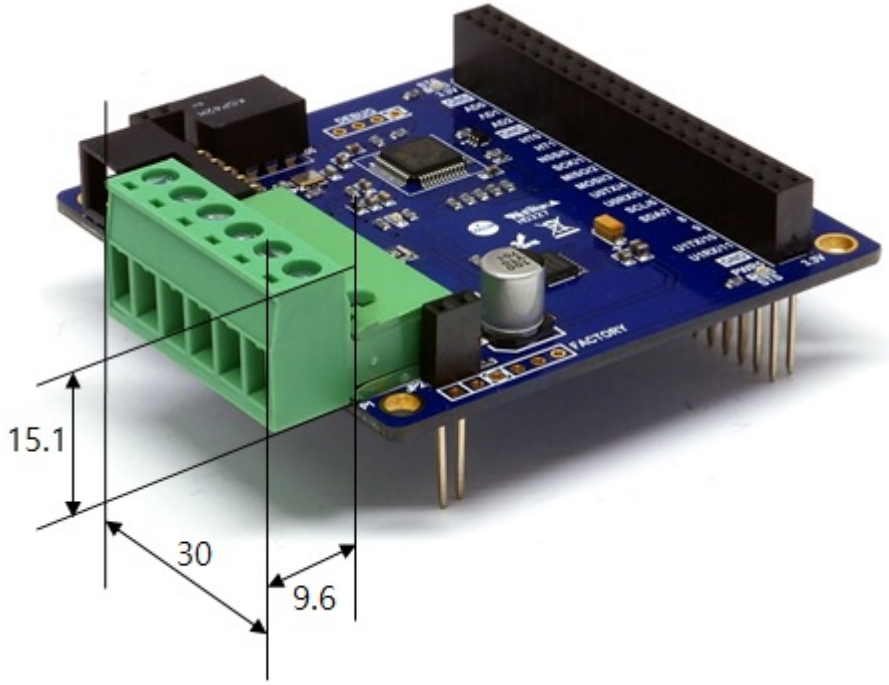
> 스마트 확장보드란?
>
> 스마트 확장보드는 일반 확장보드와는 달리 PHPoC보드의 디바이스 및 펌웨어와는 독립적인 자체 디바이스와 전용 펌웨어를 내장하고 있습니다. 이 보드는 PHPoC 보드와 전용 통신 포트를 이용해 마스터-슬레이브 방식으로 통신합니다. 하나의 PHPoC 보드에 여러 개의 스마트 확장보드를 연결할 수 있으며 각각의 스마트 확장보드에는 반드시 슬레이브 아이디를 설정해야 합니다.

치수

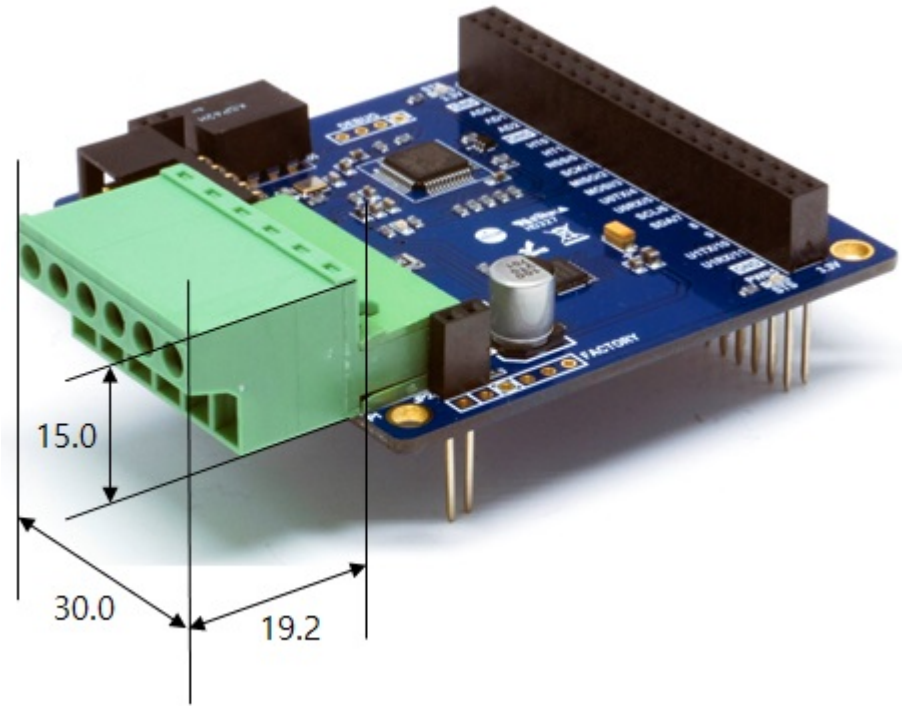
제품 본체


15.0
12.3
36.5
3.9
6.6
20.3
30.2
6.7
3.9
63.8
Ø3.5
63.8
PES-2403 R1
DEBUG
SLAVE ID
FACTORY
M1+
M1-
M2+
M2-
GND
VM
STS
3.3V
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
PWR5
STS
3.3V
AREF
AD3
AD4
AD5
GND
HT2
HT3
12
13
14
15
16
17
18
19
20/DM
21/DP
STX
SRX
SRO

터미널블록(T형)



터미널블록(S형)



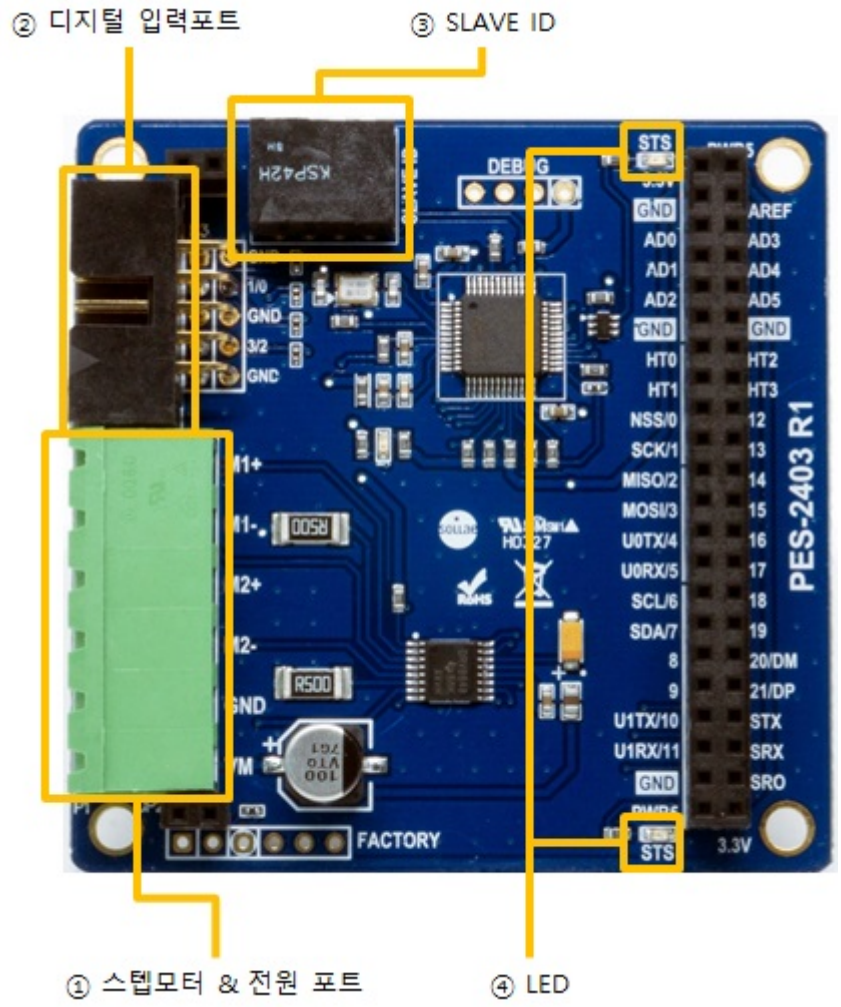
※ 치수(단위 : mm)는 제품 상태 및 재는 각도 등에 따라 약간의 오차가 있을 수 있습니다.

회로도

PES-2403의 회로도 입니다.

- PES-2403-R1-PO.pdf

레이아웃


② 디지털 입력포트
③ SLAVE ID
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
AREF
AD3
AD4
AD5
GND
HT2
HT3
12
13
14
15
16
17
18
19
20/DM
21/DP
STX
SRX
SRO
PES-2403 R1
FACTORY
STS
3.3V
① 스텝모터 & 전원 포트
④ LED

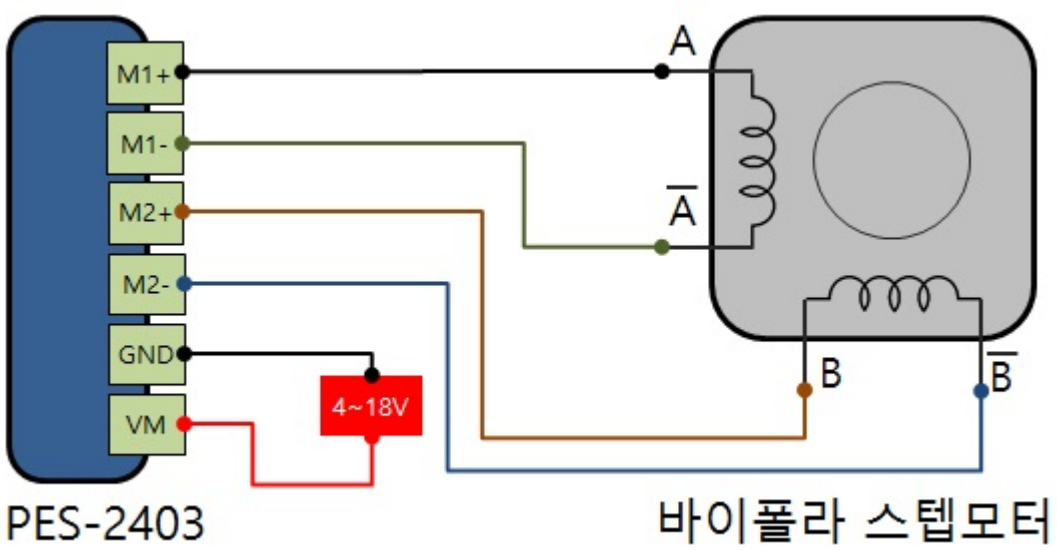
1. 스텝모터 & 전원포트

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

구분	설명
M1+, M1-, M2+, M2-	스텝모터 연결부
VM, GND	스텝모터 구동전원 연결부

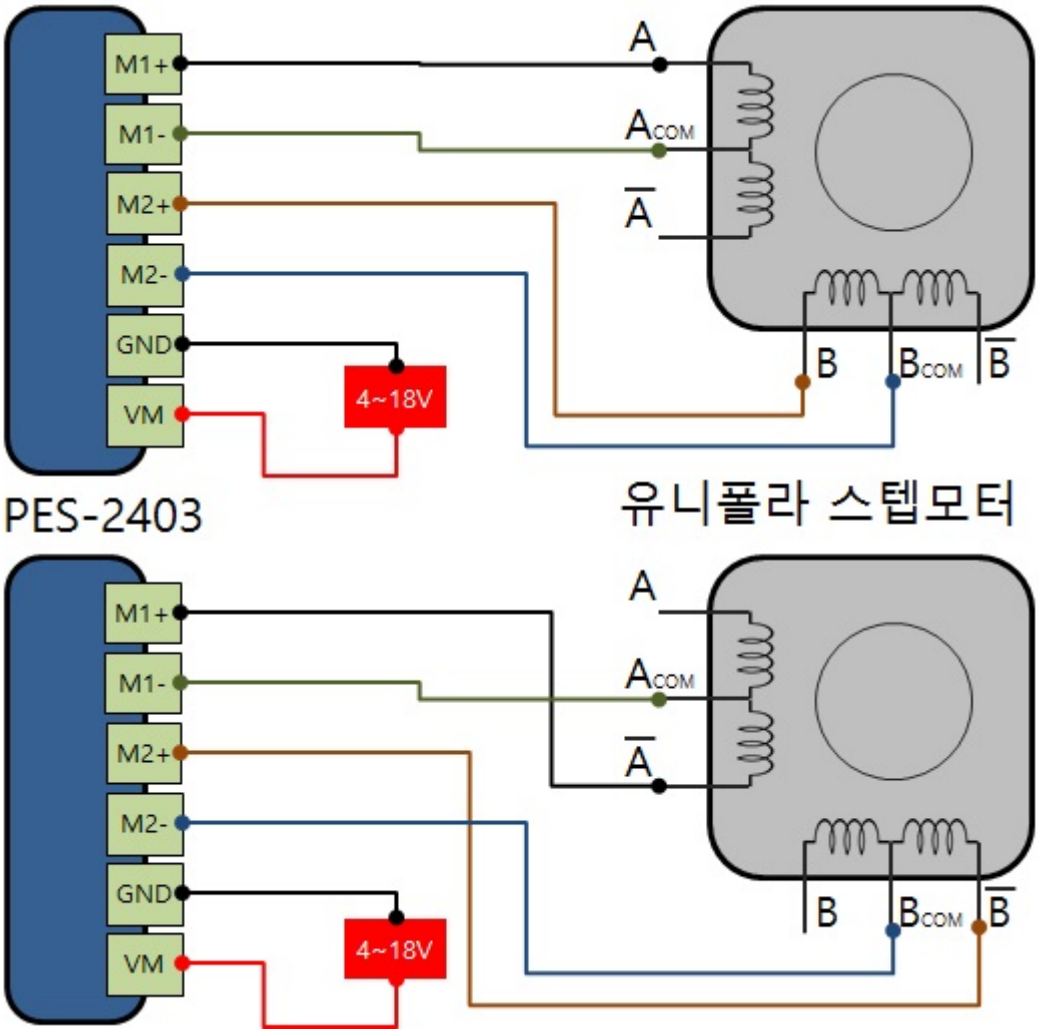
바이폴라 스텝모터 연결

PES-2403은 기본적으로 바이폴라(4선식) 스텝모터용 제어기입니다. 바이폴라 스텝모터의 연결 예는 다음과 같습니다.



유니폴라 스텝모터 연결

PES-2403은 유니폴라 스텝모터도 연결이 가능합니다. 유니폴라 스텝모터의 연결 예는 다음과 같습니다.

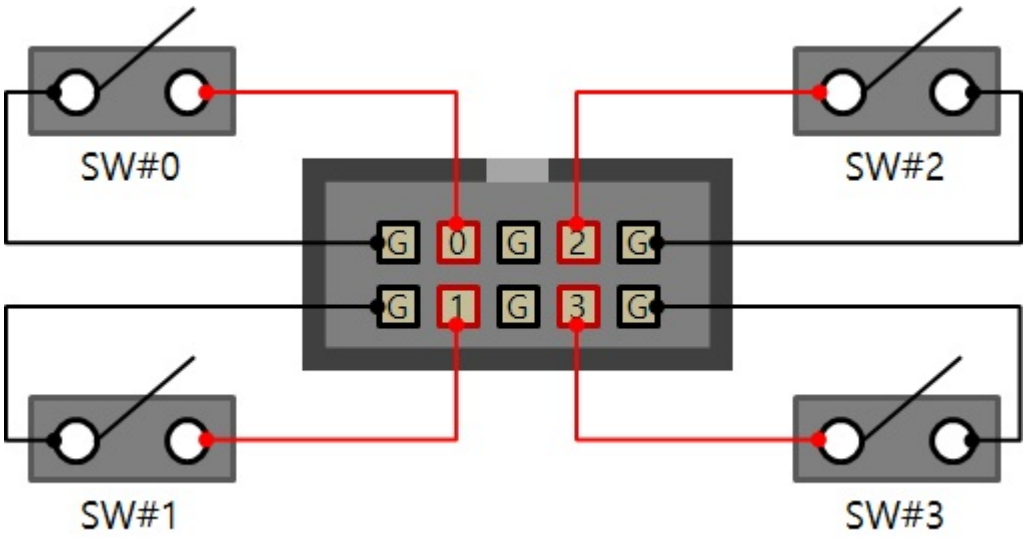


전원 연결

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

2. 디지털 입력포트

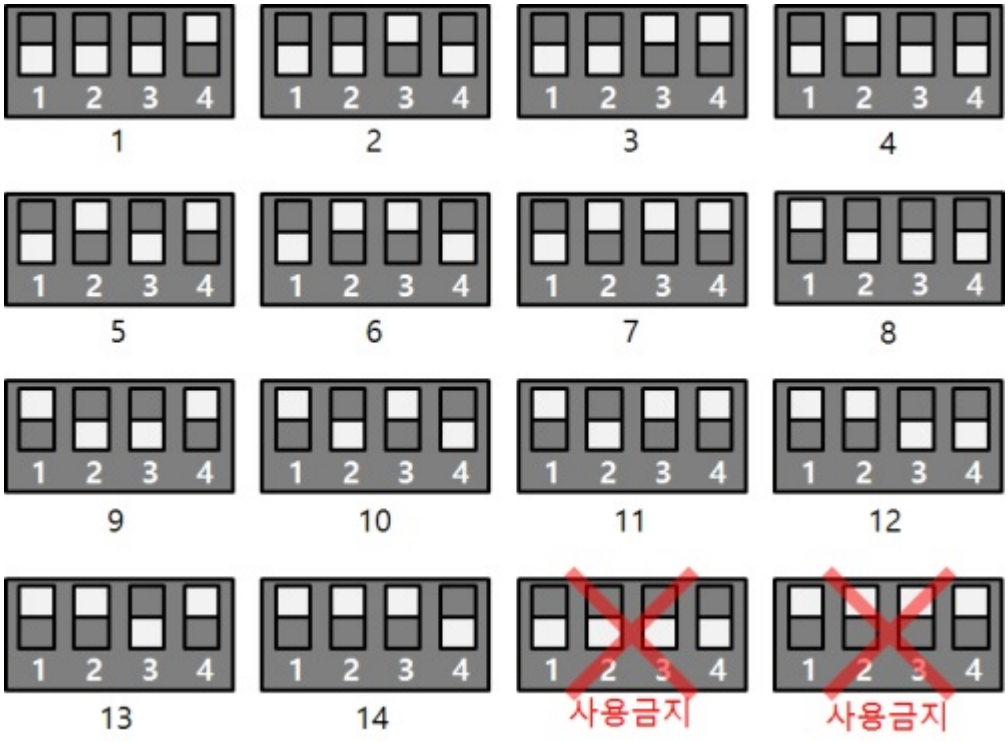
이 포트는 리미트스위치 연결 시 사용되는 포트이며 다른 용도로는 사용할 수 없습니다.
리미트스위치 연결 예는 다음과 같습니다.



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

3. 슬레이브 아이디 스위치

슬레이브 아이디는 마스터인 PHPoC 보드가 스마트 확장보드 각각을 구분하는데 사용됩니다. 하나의 PHPoC 보드에 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

PES-2403 보드에는 2개의 STS LED가 있습니다. JP1 상단의 STS LED는 3.3V에, 하단의 STS LED는 5V와 연결되어 있으며 동작은 다음과 같습니다.

- 아이디 설정 관련

상태	LED 동작
정상	1초마다 켜짐/꺼짐 반복
비정상	매우 빠르게 깜박임

- 모터 제어 관련

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

PES-2403을 사용하는 방법은 다음과 같습니다.

1. PHPoC 보드에 연결

PES-2403는 단독으로 사용할 수 없습니다. 반드시 PHPoC 보드에 연결하여 사용하시기 바랍니다.

2. 소프트웨어(IDE) 설치

PHPoC 디버거는 PHPoC 제품의 설정 및 개발에 사용되는 소프트웨어 입니다. PES-2403는 PHPoC 보드형 제품을 통해 제어할 수 있으므로 이 보드를 사용하기 위해서는 PC에 PHPoC 디버거를 반드시 설치해야 합니다.

- PHPoC 디버거 다운로드
- PHPoC 디버거 매뉴얼 페이지

3. SPC라이브러리 및 예제코드 활용

SPC라이브러리는 PES-2403을 비롯한 스마트 확장보드 라이브러리 입니다. 이 라이브러리를 사용하면 비교적 간단하게 PES-2403을 사용할 수 있습니다. 라이브러리와 함수에 대한 자세한 내용은 다음 문서를 참조하시기 바랍니다.

- SPC라이브러리 매뉴얼 페이지

명령어

스마트 확장보드를 사용하기 위해서는 spc_request_dev함수 또는 spc_request_sys함수가 필요합니다.

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

- $sid: 보드에 설정된 슬레이브 아이디 입니다.
- $cmd: 스텝모터를 사용하기 위한 명령어 문자열 입니다.

스마트 확장보드 공통 명령어

모든 스마트 확장보드가 공통으로 지원하는 명령어는 spc_request_sys함수를 사용하며, 명령어 목록은 다음과 같습니다.

명령어	인수	설명
get	did	디바이스 아이디 확인
get	uid	유니크 아이디 확인

PES-2403 명령어

스마트 확장보드별로 적용되는 명령어는 spc_request_dev함수를 사용하며, 명령어 목록은 다음과 같습니다.

명령어	인수1	인수2	인수3	인수4
set	mode	full	-	-
		half	-	-
	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

※ (): 인수 사용 필수, []: 인수 생략 가능

설정하기

스텝모터 제어에 관련된 각각의 파라미터들을 설정하기 위한 명령어는 set입니다.

드라이브 모드 설정

드라이브 모드를 설정하는 명령어는 mode입니다.

```
"set mode (drive)"
```

drive에 드라이브 모드를 입력해야 하며, PES-2403은 다음 2가지 드라이브 모드를 제공합니다.

drive	설명
full	Full-step(2상 여자)
half	Half-step

- 드라이브 모드 설정 예

```
spc_request_dev($sid, "set mode full");
spc_request_dev($sid, "set mode half");
```

제한전류 설정

제한전류를 설정하는 명령어는 vref입니다. 이 설정은 모터를 제어하기 위해 반드시 필요합니다.

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

state는 제한전류 설정이 필요한 다음 3가지 상태를 의미합니다.

state	설명
stop	정지 상태를 유지할 때의 제한전류
drive	동작할 때의 제한전류
lock	잠금 상태를 유지할 때의 제한전류

value는 제한전류의 양을 의미하며 각 상태별로 0 ~ 15까지 총 16단계로 설정할 수 있습니다.
이 값에 5를 설정하면 해당 상태에서의 전류를 5/15로 제한합니다.

- 제한전류 설정 예

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

공진 범위 설정

공진 범위를 설정하는 명령어는 rsnc입니다.

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

low_pps와 high_pps는 공진 범위 최소값과 최대값을 각각 의미합니다. 공진 범위를 설정하면 회전속도가 공진 범위 안에 해당할 때 자동으로 high_pps에 설정된 속도로 제어합니다.

- 공진 범위 설정 예

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

회전속도 설정

스텝모터의 회전속도를 설정하는 명령어는 speed입니다. 이 명령어는 모터를 제어하기 전에 회전속도만 따로 설정하고자 하는 경우에 사용합니다.

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

pps는 회전속도를 의미합니다. 회전속도의 단위는 pps(pulse per second)이며, PES-2403은 최대 20,000[pps]까지 지원합니다. 그러나 실제 최대속도는 스텝모터의 종류/전압/부하에 따라 달라질 수 있습니다.

- 회전속도 설정 예

```
spc_request_dev($sid, "set speed 400");
```

가속도 및 감속도 설정

스텝모터의 가속도 및 감속도를 설정하는 명령어는 accel입니다. 이 명령어는 모터를 제어하기 전에 가속도 및 감속도만 따로 설정하고자 하는 경우에 사용합니다.

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

accel은 가속도를, decel은 감속도를 의미합니다. 이 명령에서 가속도(accel)는 반드시 입력해야 하지만 감속도(decel)는 생략이 가능합니다. 감속도를 입력하지 않으면 가속도에 입력한 값이 감속도에 자동으로 설정됩니다.
가속도의 단위는 pps/s(pps per second)이며, PES-2403은 최대 200,000[pps/s]까지 지원합니다.

- 가속도 설정 예

```
spc_request_dev($sid, "set accel 1000"); // "set accel 1000 1000"과 같음
spc_request_dev($sid, "set accel 1000 0");
spc_request_dev($sid, "set accel 0 1000");
```

카운터 위치 설정

카운터 위치를 설정하는 명령어는 pos입니다. 카운터 위치 설정은 goto로 스텝모터를 제어할 때만 유효하고 move로 제어하는 경우 반영 되지 않습니다.

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

pos는 카운터의 위치를 의미합니다.
카운터 위치는 부호가 있는 32비트 정수 형태이고, 입력 가능한 범위는 -1000000000(10억) ~ +1000000000입니다.

- 카운터 위치 설정 예

```
spc_request_dev($sid, "set pos 400");
```

디지털 입력포트 설정

디지털 입력포트를 설정하기 위한 명령어는 eio set입니다.

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

p는 디지털 입력포트의 아이디를 의미하며 0, 1, 2 또는 3중 하나를 입력합니다.
mode는 디지털 입력포트의 입력모드를 의미합니다.

mode	설명
input	일반입력 모드
lock	제어잠금 모드

- 디지털 입력포트 설정 예: 일반입력 모드

```
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");
```

- 디지털 입력포트 설정 예: 제어잠금 모드

```
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");
```

상태 확인하기

스텝모터의 현재 상태를 확인하기 위한 명령어는 get입니다.

스텝모터 동작상태 확인

스텝모터의 동작상태를 확인하는 명령어는 state입니다.

```
"get state"
```

반환되는 동작상태의 종류는 다음과 같습니다.

반환 값	상태
0	정지됨
1	제어 잠김(lock)
그 외	동작 중

- 스텝모터 동작상태 확인 예

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
spc_request_dev($sid, "set mode full");
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";
?>
```

- 출력 예

```
state: 3
state: 2
```

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

카운터 위치 확인

현재 카운터 위치를 확인하는 명령어는 pos입니다.

```
"get pos"
```

- 카운터 위치 확인 예

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

spc_reset();
spc_sync_baud(460800);

$pos = -400;
$sid = 1;
spc_request_dev($sid, "set mode full");
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);
}
?>
```

- 출력 예

```
position: -315
position: -233
position: -151
position: -68
position: 14
position: 96
position: 178
```

```
position: 260
position: 338
position: 392
```

디지털 입력포트의 상태 확인

디지털 입력포트의 현재 상태를 확인하기 위한 명령어는 eio get입니다.

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

p는 입력포트의 아이디(0 ~ 3)를 의미합니다.

반환되는 입력포트 상태의 종류는 다음과 같습니다.

반환 값	상태
0	LOW
1	HIGH (기본 값)

- 디지털 입력포트의 상태 확인 예

```
<?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);
}
?>
```

- 출력 예

```
1111
0110
...생략...
```

goto로 제어하기

스텝모터를 모터의 현재 위치가 아닌 초기 위치를 기준으로 제어하고자 할 때 goto를 사용합니다.
이 방식은 모터가 동작 중인 상태에서도 사용이 가능합니다.

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

※ 주의: [sign]과 (pos) 사이에는 빈 공간(space)이 없습니다.

각 인수에 대한 설명은 다음과 같습니다.

인수	설명	필수/옵션
sign	회전 방향, "+"(정 회전) 또는 "-"(역 회전)	옵션(생략시 "+")
pos	목표 카운터 위치	필수
speed	회전 속도(단위: pps)	옵션
accel	가속도(단위: pps/s)	옵션
decel	감속도(단위: pps/s)	옵션

목표 카운터의 위치(pos)는 카운터의 현재위치가 아닌 초기 위치가 기준이 됩니다.

이 명령에서 가속도(accel)만 입력하고 감속도(decel)를 입력하지 않으면 가속도에 입력한 값이 감속도에 자동으로 설정됩니다.

- goto를 이용한 스텝모터 구동 예

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
spc_request_dev($sid, "set mode full");
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);
?>
```

※ 이 명령어는 스텝모터가 동작중일 때에도 사용이 가능합니다. 따라서 모터가 구동 중일 때 새로운 goto명령이 실행되면 그 즉시 해당 명령을 수행합니다.

move로 제어하기

스텝모터를 모터의 현재 위치를 기준으로 제어하고자 할 때 move를 사용합니다.
이 방식은 모터가 정지상태일 때만 사용이 가능합니다.

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

※ 주의: [sign]과 (pos) 사이에는 빈 공간(space)이 없습니다.

각 인수에 대한 설명은 다음과 같습니다.

인수	설명	필수/옵션
sign	회전 방향, "+"(정 회전) 또는 "-"(역 회전)	옵션(생략시 "+")
step	구동 시킬 스텝 수	필수
speed	회전 속도(단위: pps)	옵션
accel	가속도(단위: pps/s)	옵션
decel	감속도(단위: pps/s)	옵션

구동 시킬 스텝 수(step)는 모터의 현재 위치가 기준이 됩니다.

이 명령에서 가속도(accel)만 입력하고 감속도(decel)를 입력하지 않으면 가속도에 입력한 값이 감속도에 자동으로 설정됩니다.

- move를 이용한 스텝모터 구동 예

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
spc_request_dev($sid, "set mode full");
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);
?>
```

※ 이 명령어는 스텝모터가 정지상태일 때만 사용이 가능합니다. 따라서 위 예제에서처럼 항상 모터가

정지된 이후에 move명령이 실행되도록 프로그래밍 하시기 바랍니다.

정지시키기

동작 중인 스텝모터를 정지하는 명령어는 stop입니다.

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

decel은 감속도를 의미합니다. 단위는 pps/s이며 생략하면 현재 설정된 감속도 값을 사용합니다.

- 스텝모터 정지 예

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

spc_reset();
spc_sync_baud(460800);

$sid = 1;
spc_request_dev($sid, "set mode full");
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");

?>
```

초기위치 설정하기

PES-2403의 디지털 입력포트에 리미트스위치를 연결하면 초기위치 설정이 필요한 시스템에서 초기위치를 설정할 수 있습니다. 설정하는 과정은 다음과 같습니다.

1. 모터 구동

   시스템을 시작하면 스텝모터를 초기위치방향으로 구동시킵니다.

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

goto ~ sw명령을 이용하면 리미트스위치가 닫혔을 때 자동으로 스텝모터의 동작을 정지시킬 수 있습니다. 이 명령을 통해 모터가 정지되면 모터의 상태 값은 정지상태가 됩니다.

- goto ~ sw의 명령어 형식

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

※ 주의: [dir]과 sw(id) 사이에는 빈 공간(space)이 없습니다.

파라미터	설명	필수/옵션
dir	회전 방향, "+"(정 회전) 또는 "-"(역 회전)	옵션(생략시 "+")
id	디지털 입력포트 아이디, 0/1/2/3	필수
speed	회전 속도(단위: pps)	옵션
accel	가속도(단위: pps/s)	옵션
decel	감속도(단위: pps/s)	옵션

- 자동으로 정지시키기 예

```
<?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 full");

// 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";
?>
```

- 출력 결과

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

수동으로 정지시키기

리미트스위치가 닫혔을 때 stop명령을 이용하면 수동으로 스텝모터의 동작을 정지시킬 수 있습니다. 이때 리미트스위치의 입력 여부는 eio get을 이용합니다.

- 수동으로 정지시키기 예

```
<?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 full");

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");
?>
```

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

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

제어잠금 모드 설정

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

잠금의 해제

제어기 잠금 상태를 해제하는 명령어는 unlock입니다.

unlock명령을 수행하면 모터의 상태는 잠김상태에서 정지상태로 전환되고, 디지털 입력포트의 입력모드는 제어잠금 모드에서 일반입력 모드로 초기화 됩니다.

따라서 unlock수행 이후에는 정상적으로 모터 제어가 가능합니다.

- 제어잠금 및 해제 예

```
<?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 full");

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";
?>
```

- 출력 결과

```
step_state 1
step_state 0
```