pcKM1024i 드라이버 및 GBus 제어기 매뉴얼

2025-4-28

다운로드 링크 :

테스트 소프트웨어 실행파일 (exe,dll)
테스트 소프트웨어 소스코드(VS2020 solution)
GData 3D 모델
pcKM014i 3D 모델

0. 기존 UData 와 달라진 점

- 엔코더 커넥터의 핀맵 달라짐

- 테스트 소프트웨어 UI
  -> 복수의 헤드를 구동할 때, 각 헤드마다 소프트웨어 인스턴스를 생성하여 사용
  -> 이미지 인쇄 관련 함수 추가

- 기가비트 이더넷으로 연결되며, 각 모듈은 IP address 와 channel 값 (port) 를 갖는다.

- writePixelData 관련
  기존

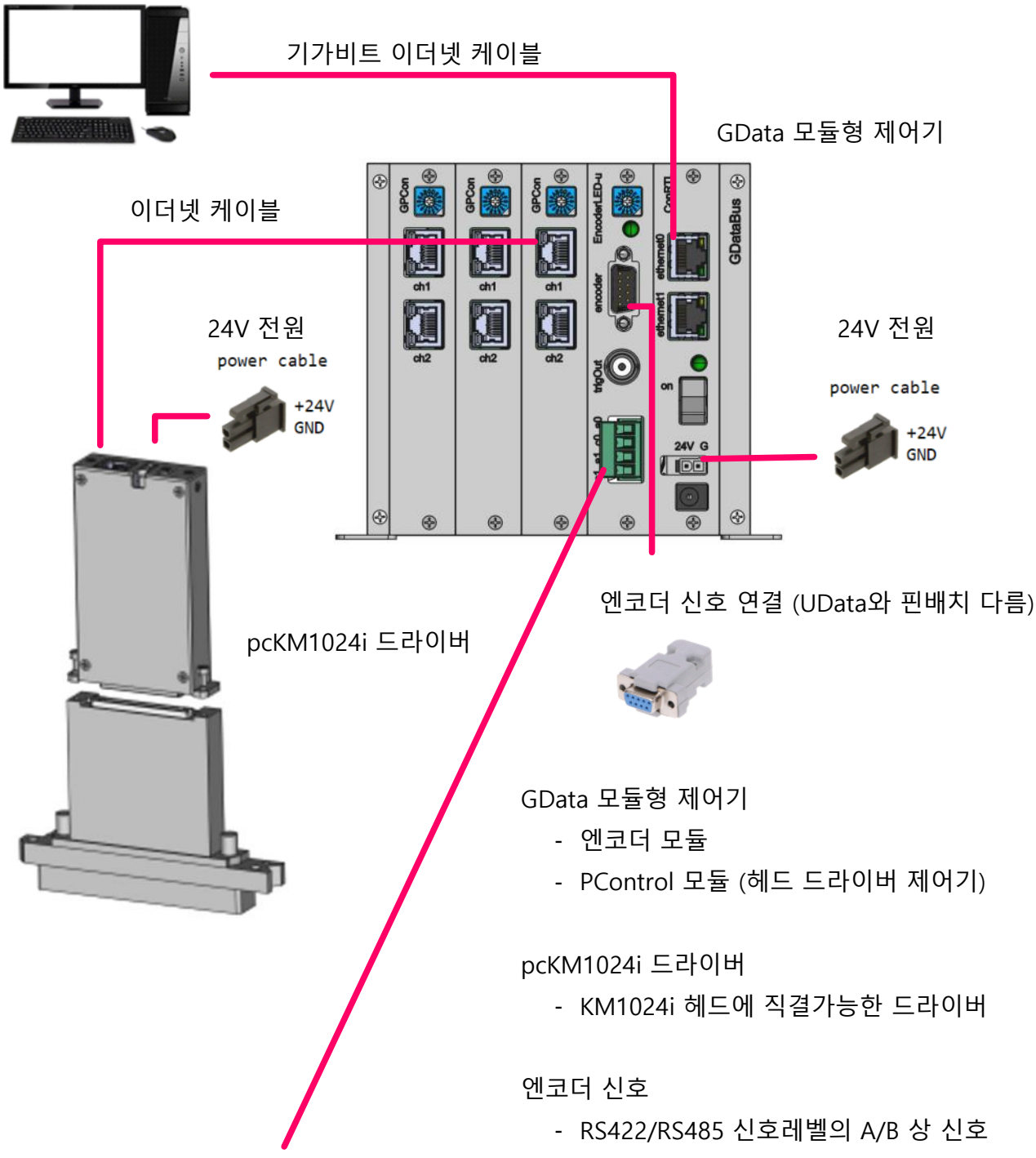
  beginWritePixel
  writePixel
  endWritePixel

  에서

  endWritePixel 이 없어지고 isWritePixelFinished 로 통신이 끝났는지 확인함.
  isWritePixelFinished 이후 대략 100 ms 이후에 인쇄 시작 가능.

- 헤드 온도 값을 인쇄중 (spitting/printing) 에도 읽을 수 있슴
  기존과 같이 다른 제어는 할 수 없슴. 온도값만 읽을 수 있슴.

1. 연결 개요



GData 모듈형 제어기
- 엔코더 모듈
- PControl 모듈 (헤드 드라이버 제어기)

pcKM1024i 드라이버
- KM1024i 헤드에 직결가능한 드라이버

엔코더 신호
- RS422/RS485 신호레벨의 A/B 상 신호

24V 전원
- DC 24V 전원

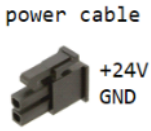
2. GData 연결

2-1. GData 이더넷 모듈

- 기가비트 이더넷 연결
  2 개의 포트가 있고, 내부 이더넷 스위치에 연결됨.
  어느 포트를 쓰던 상관없음.

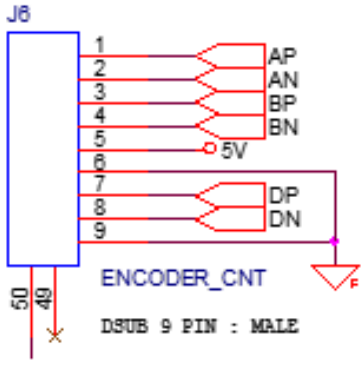
- 전원 LED
- 전원 스위치

- 24V 전원
  전원 케이블 커넥터 : Molex 0039301020 (5557-02R)
  대기전류 0.3 A, 최대전류 2 A



2-2. GData 엔코더 모듈

- IP 어드레스 설정
  192.168.0. (100 + (dip switch 값))
  기본설정값 192.168.0.115



- Trigger OUT 신호
  BNC 커넥터, 5V TTL 레벨

핀	설명
AP/AN	엔코더 출력신호 A 상 (A+/A-) , RS422/RS485 레벨 신호
BP/BN	엔코더 B 상
DP/DN	사용하지 않는다.
+5V	엔코더에 5V 전원 공급. 최대 400 mA 엔코더가 다른 전원을 공급받고 있다면 연결하지 않는다.
GND	엔코더 GND (0V) 핀으로 반드시 연결한다.

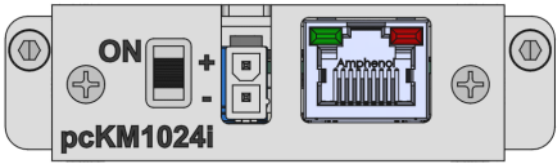
2-3. GData GPCon 모듈 (기가비트 PControl 모듈)

- IP 어드레스 설정
  192.168.0. (100 + (dip switch 값))
  기본설정값 192.168.0.100

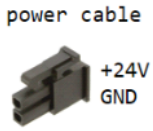
- 2 채널
  일반적인 8p8c 이더넷 케이블을 사용하며,
  CAT5 이상의 등급을 사용한다.

  드라이버 연결시 상부 LED 켜짐.
  인쇄시 하부 LED 켜짐.

3. pcKM1024i 드라이버 연결

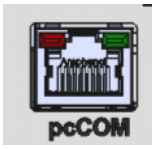


- 24V 전원
  전원 케이블 커넥터 : Molex 0039301020 (5557-02R)
  대기전류 0.03 A, 최대전류 2 A



- GData 와 연결

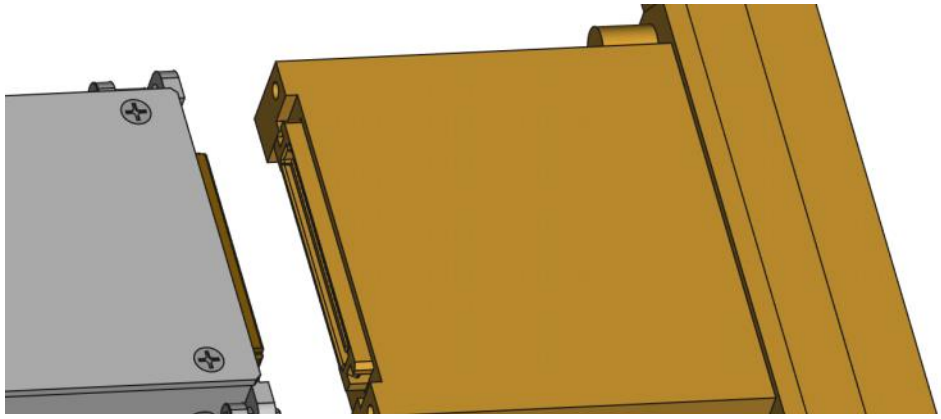
  일반적인 8p8c 이더넷 케이블을 사용하며, CAT5 이상의 등급을 사용한다.

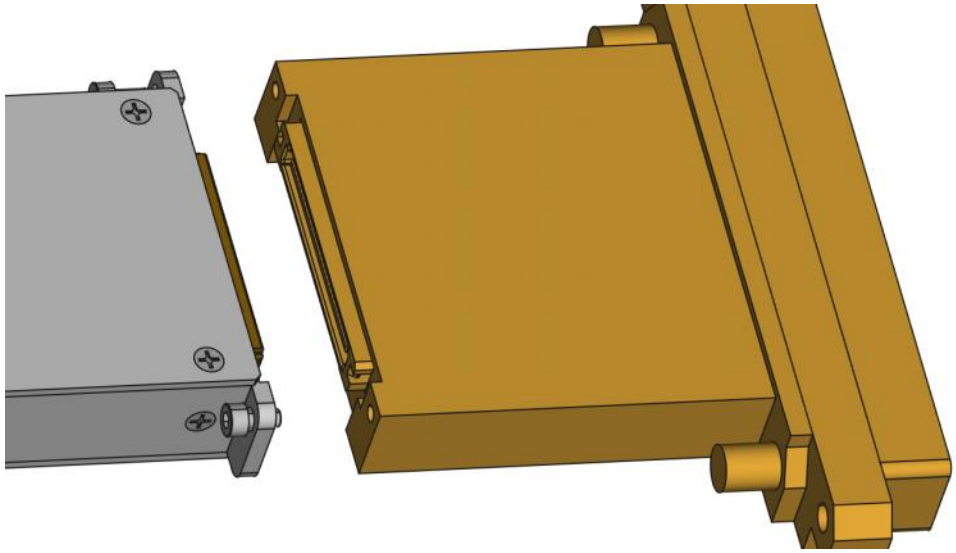


  GData 가 연결되면 (전원이 켜져 있는 경우) 녹색 등이 들어온다.
  spitting/printing 시 적색 등도 들어온다.

- 헤드와 연결

  두 커넥터의 1 번핀 위치를 맞추고, 직결한다.
  이후 M3 길이 6 mm 나사를 사용하여 고정한다.






pcKM1024i
250116-102
HS1000-002522
Mfg date 2019.09
2L: 11.6V 2R: 11.6V
1LA:6.1V 1LB:6.1V 1RA:6.0V 1RB:5.8V

3. 테스트 소프트웨어 (드라이버)

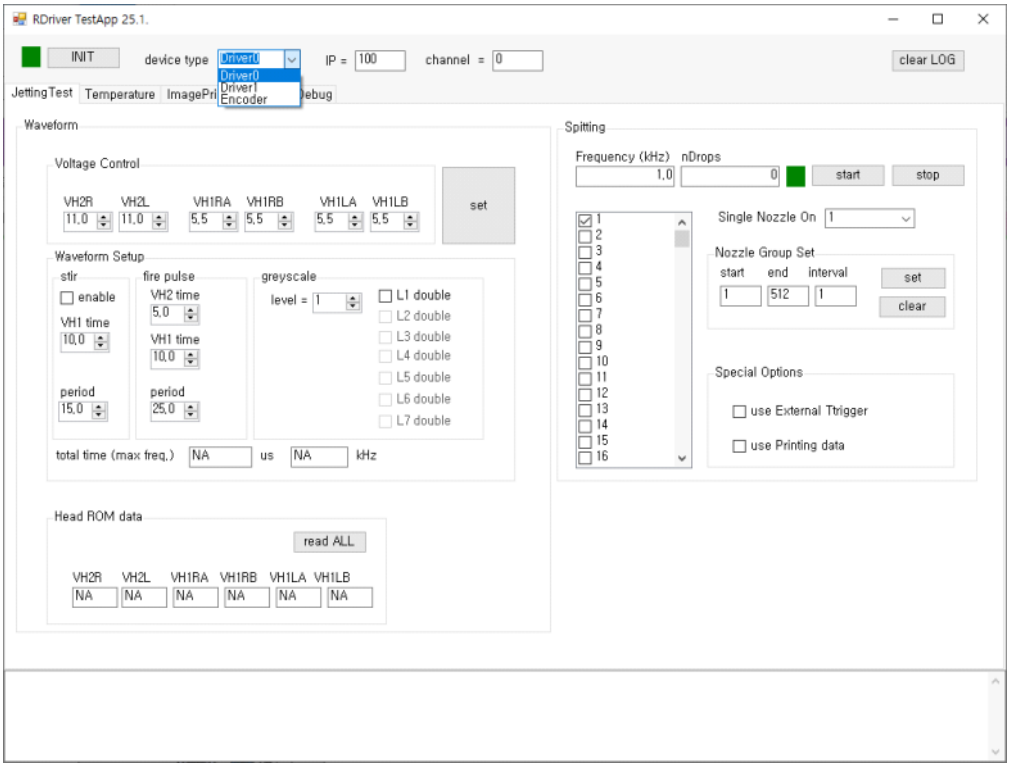
3-1. 소프트웨어 설치

빌드환경 : Windows 10 64 bit , WinForm Application

필요화일 :

testKM1024i.exe ：실행 파일

RDriver.dll ：라이브러리 파일 (C# class library)

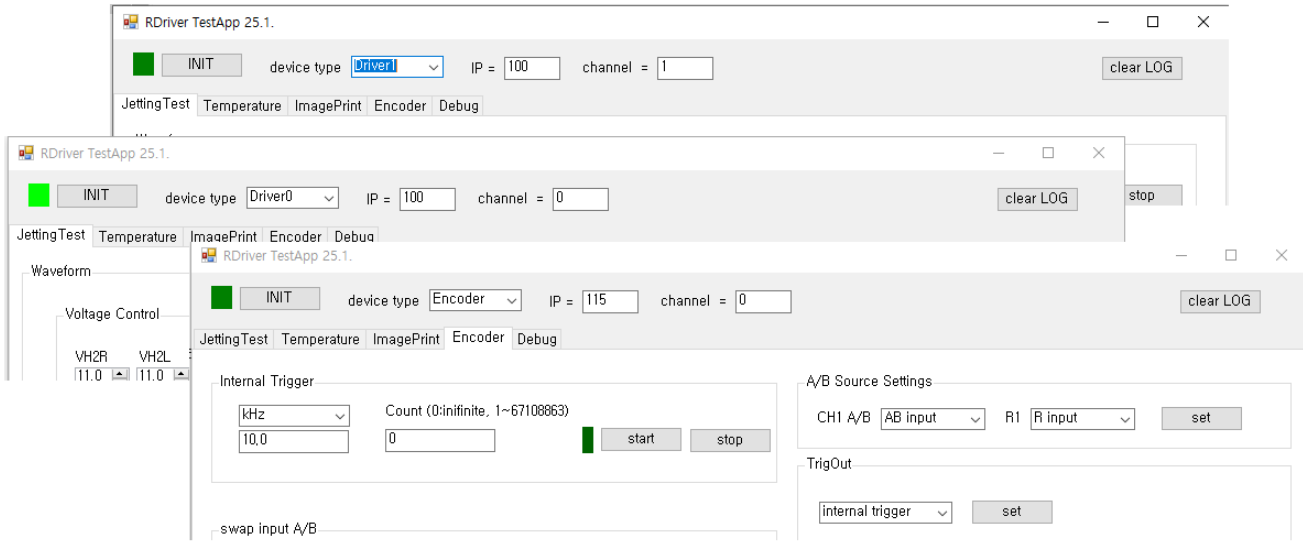


3-2. 초기화

- GData 내부 운영체제 로딩 시간
  전원을 켠 후 내부 리눅스 시스템이 부팅하기 위해서 26초 정도가 필요하다.

- 테스트 소프트웨어의 UI
  소프트웨어의 UI 는 1 개의 장치를 구동하도록 만들어 졌으므로, 3 개의 장치를 동시에 조작하기 위해서는 아래 그림처럼 소프트웨어를 3 개 실행시켜서 각각 장치에 할당한다.

  DeviceType 을 Encoder/Driver0/Driver1 중에 선택하면, IP 어드레스 및 채널 값이 자동으로 나타난다.
  이 값은 유저가 변경할 수 있다.
  INIT 버튼을 눌러 초기화 한다.



3-3. spitting 테스트

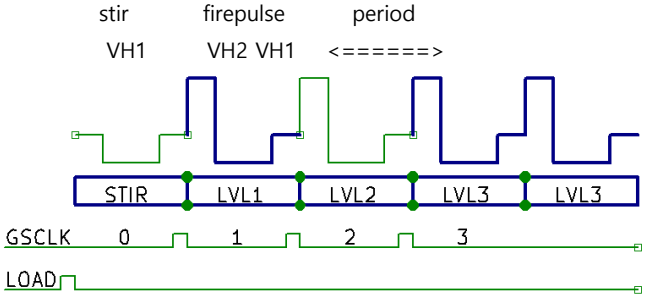
- Voltage Control

  VH2-L/R, VH1-L/R-A/B 로 6 채널이다.

  21.0 >= VH2 > VH1 >= 5.0 V

  0.1V 단위.

- Waveform Setup

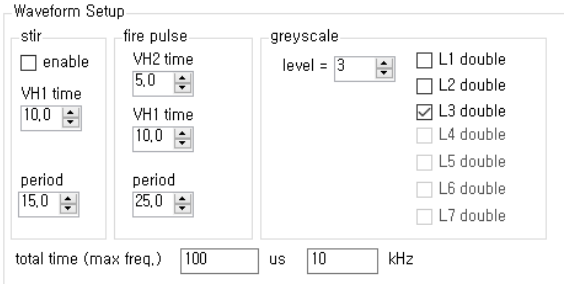


fire pulse : VH2, VH1 으로 이루어진다. period 는 multi-pulse 를 위한 설정값.
stir pulse : enable 시 사용가능하며 VH1 과 period 만을 넣는다.

0.1 us < VH1 < 25.5 us
0.1 us < VH2 < 25.5 us
(VH1+VH2) < period < (VH1+VH2+25.5 us)
0.1 us 단위

greyscale: level = 1 인 경우, SA[0..2]-L/R 에서 SA[0] 비트만 사용되며, 0 또는 1 이다.

설정예)



level=3 이므로 SA[0..1] 까지 사용되고, SA[2]=0 다.
SA=0 인 경우, off,
SA=1 인 경우, LVL1 파형 1개
SA=2 인 경우, LVL1 + LVL2 파형 2개
SA=3 인 경우, LVL1 + LVL2 + LVL3 x 2 (L3 double 체크) 로 파형 4 개
이때 파형간의 간격이 period 값이다.

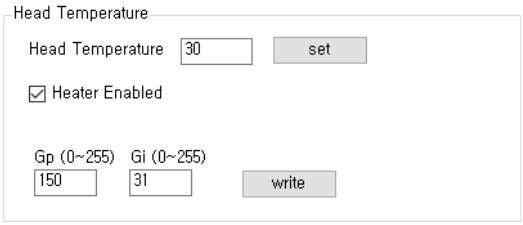
```
double  VH[] = { VH2R, VH2L, VH1RA, VH1RB, VH1LA, VH1LB };
int greysteps = 1;
int greybits = 1; //  2 : 2 <= greysteps <= 3,  3 : 4 <= greysteps <= 7
int firedouble = 0; // 1 @ L1 + 2 @ L2 + 4 @ L3 + 8 @ L4 + 16 @ L5 + 32 @ L6 + 64 @ L7
int greypattern[] = { greybits, greysteps, firedouble };
double stirtime[] = { 0, 0 }; // disabled or stirVH1time,  stirPaddingTime ( = period - VH1)
double firetime[] = { fireVH1time, fireVH2time, firePaddingTime }; // firePaddingTime = period -
(VH1+VH2)
setFGKM( VH, greypattern, stirtime, firetime);
```

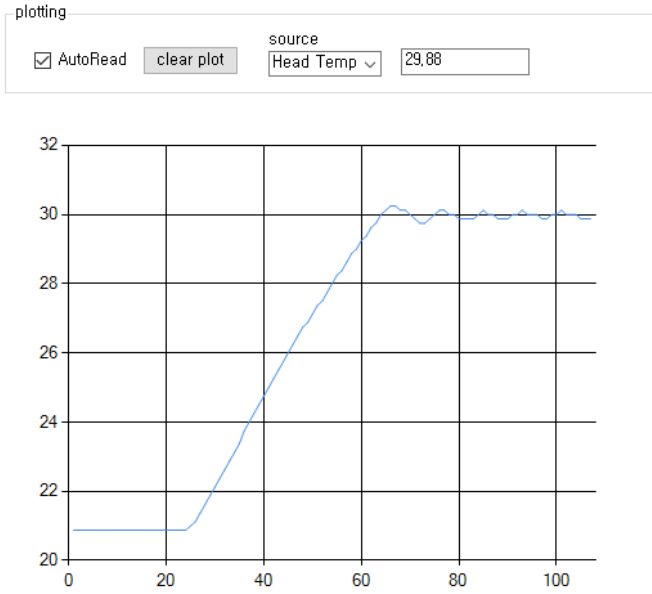
- Jetting Test 탭에서 분사할 노즐을 선택하고, 주파수 및 분사 수를 설정한다.
  이때 nDrops = 0 인 경우는 nDrops = ∞ 가 된다.
  nDrops = 1 ~ 32767 인 경우, nDrops 만큼 분사가 되면 자동으로 정지한다.
  "start" 버튼을 누르면 spitting 시작. "stop" 버튼을 누르면 중지.

- use External Trigger 가 체크되어 있는 경우 GData 내부 버스 상의 CLKE 에 의해 trigger 된다.
  Encode 모듈에서 발생시킬 수 있다.

- use Printing data 가 체크되어 있는 경우 "Image Print" 탭에서 설정한 인쇄 데이터 (이미지 파일)에 따라서 출력이 일어난다. printing 테스트에 대한 시뮬레이션 기능이다.

3-4. 헤드 온도 제어



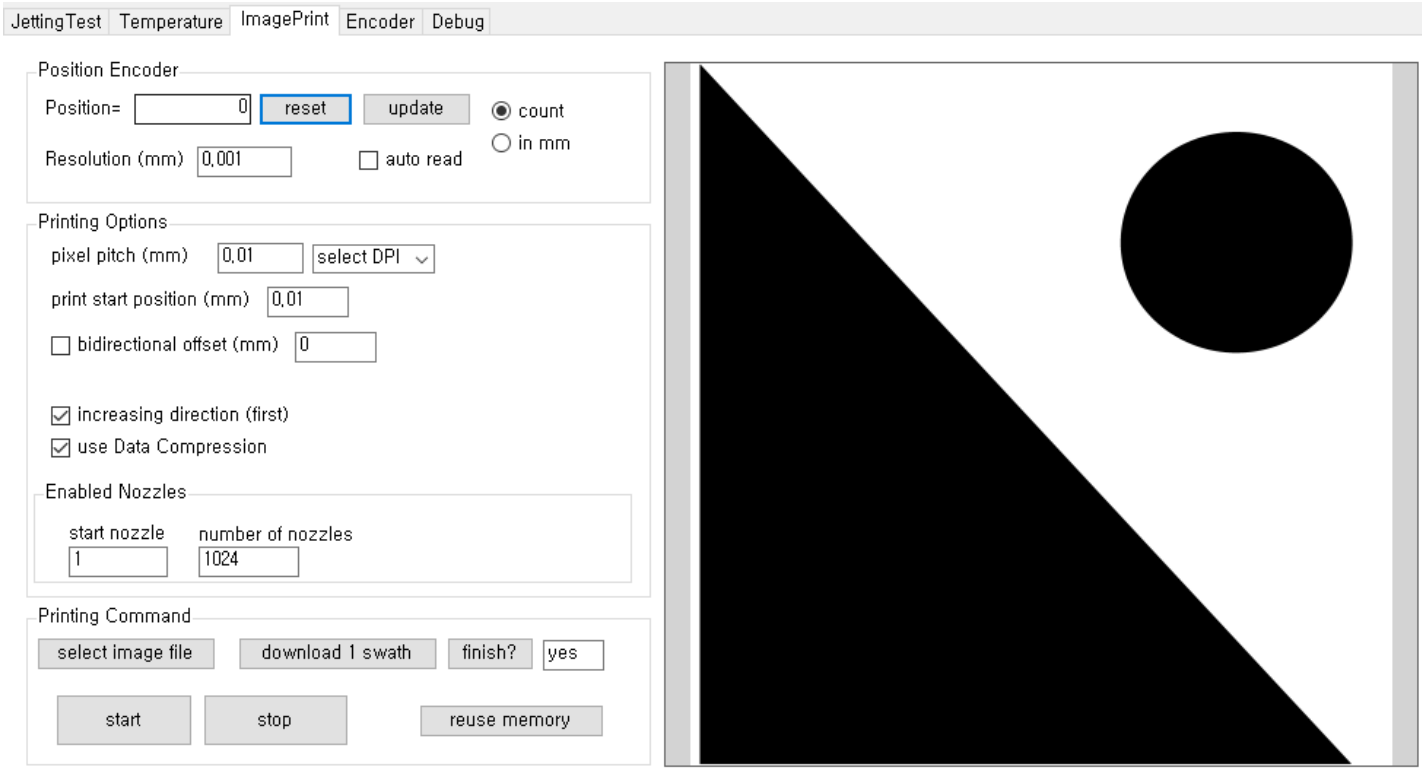


- 히터 제어기 게인은 PI 제어기로
  P = 150, I = 31 로 세팅할 수 있다.
  (내지는 P = 200, I = 0 로 가능하다.)

- AutoRead 를 체크하면 헤드 온도 값을 그래프로 표시할 수 있다.

- 목표 값을 Head Temperature 에 넣고, SET 을 눌른 후,
- Heater Enabled 체크 박스를 체크하여 온도 제어를 시작한다. 내지는 체크를 해제하여 온도 제어를 해제한다.
- 기존 모델과 달리 spitting/printing 중에도 온도 값을 읽을 수 있다.

3-5. 인쇄 테스트

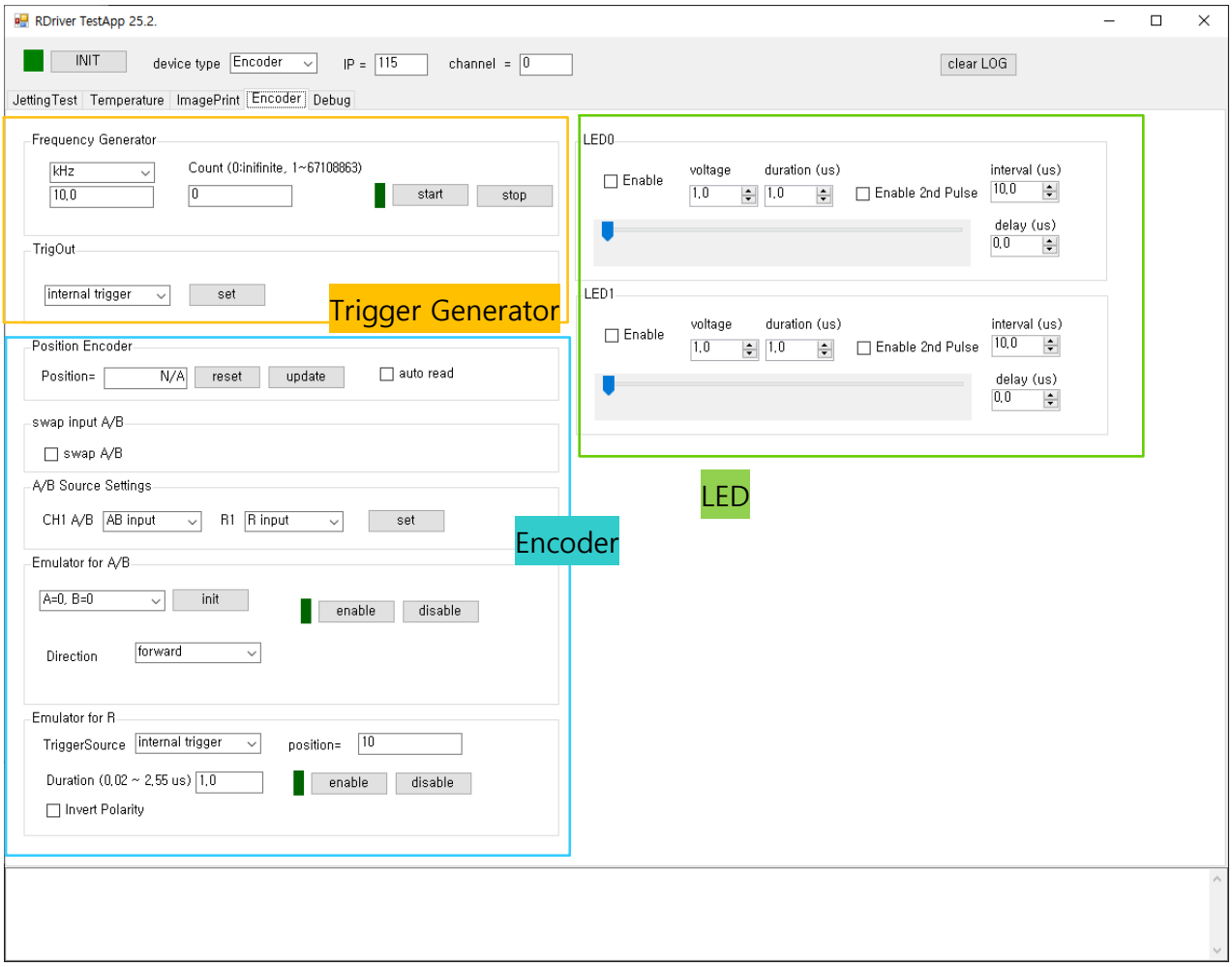


- 엔코더 값을 reset 하거나 update 할 수 있고, auto read 를 체크하면 자동으로 읽을 수 있다.
- printing options 는 아래의 이미지 인쇄를 하기 위한 파라미터들로,
  소프트웨어에서 writeImage 함수를 쓰지 않고, 직접 writePixel 함수를 사용하는 경우에는 이 옵션들은 모두

의미없다.

- 인쇄에 사용할 노즐을 선택한다.
  이것도 writeImage 함수가 아니고, 사용자가 직접 데이터를 쓰는 writePixel 함수를 쓰는 경우 의미없다.
- select image file 로 인쇄할 이미지를 불러온다. 인쇄에 사용할 노즐의 개수의 정수배의 높이를 가져야한다. 즉 1024 노즐을 쓰는 경우, 이미지의 세로방향 픽셀수는 1024 의 배수가 되어야 한다. (예제라서 간단하게 처리함)
- download 1 swath 를 눌러 첫번째 swath 데이터를 쓴다.
- start 버튼을 누른 후, 스테이지를 움직여 이미지 인쇄를 한다.
- 스테이지 움직임이 끝나고 stop 버튼을 눌러 인쇄를 마친다.
- 다음번 swath 데이터를 인쇄하기 위해 download 1 swath 부터 다시 시작한다.
- 현재 이미지 데이터를 처음부터 다시 활용하기 위해서 reuse memory 를 누르는 경우 이미지 데이터 올리기 없이 바로 download 1 swath 부터 시작한다.

4. 테스트 소프트웨어 (EncoderLED 모듈)



4-1. Tigger Generator

발생시킨 internal trigger 신호는 아래와 같이 사용된다.

- Bus 상에 모든 제어기로 전달되는 external trigger 신호로 사용
- LED strobo 의 기준 trigger 신호
- Encoder emulator 의 기준 신호
- BNC 커넥터로 출력되는 trigger out 신호

주파수 및 카운트 (무한 포함) 를 지정할 수 있다.

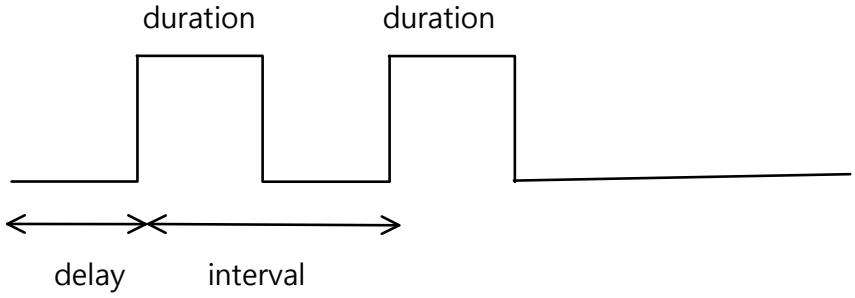
4-2. Encoder 처리기

- 기본적으로 외부에서 입력된 엔코더 신호를 사용하게 된다.
- 엔코더 신호 (외부/에뮬레이션)를 Bus 상의 모든 제어기로 전달한다.
- Position Encoder 에서 엔코더 카운터 값을 읽거나 리셋할 수 있다.
- A/B 를 소프트웨어적으로 swap 하여 진행방향을 반대로 설정할 수 있다.
- 엔코더 에뮬레이터를 내장하고 있어서 이 신호를 A/B 신호로 사용할 수 있다.

- Emulator 출력을 A/B 신호에 연결하기 위해, A/B source setting 에서 Emualtor 출력을 설정하고, Emualtor 를 enable 한다.
  이후 internal trigger 를 발생시키면 A/B 에뮬레이션 신호가 발생되고, Bus 상으로 공급된다.

4-3. LED 제어기

- internal trigger 신호에 동기한다.
- 2 channel 마다 voltage, delay, duration 값들을 조정할 수 있다.
- Enable 체크박스를 체크해서 enable 시킨다.
- Enable 2nd pulse 박스를 체크하고, interval 값을 조정하여 두 번째 strobo 를 사용할 수 있다.



소프트웨어에서는 interval 값을 쓰고 있지만, 라이브러리 상에서는
delay, delay2 로 쓰고 있다. delay2 = delay + interval 이다.

5. 소프트웨어 라이브러리 (드라이버)

1. 사용법

파일이름 : RDriver.dll (RDriver.dll 만 Reference 로 등록하면 됨)
형태 : C# class library
사용 예제는 VisualStudio WinForm Application 을 사용

using RDriver; // RCONTROL namespace

wKM1024i Driver = new wKM1024i();

2. 함수

초기화

bool init(int ip, int channel = 0)
ip: IP address 마지막 바이트 값. 192.168.0.100 인 경우 ip = 100
channel : GPCon 모듈은 2 개의 채널을 갖는다. 0, 1
예) Driver.init(100,0);

int getSerialNo()
드라이버의 시리얼 번호를 돌려준다.

파형 설정

bool setFGKM(double[] VH, int[] greypattern, double[] stirtime, double[] firetime)
파형을 설정한다.

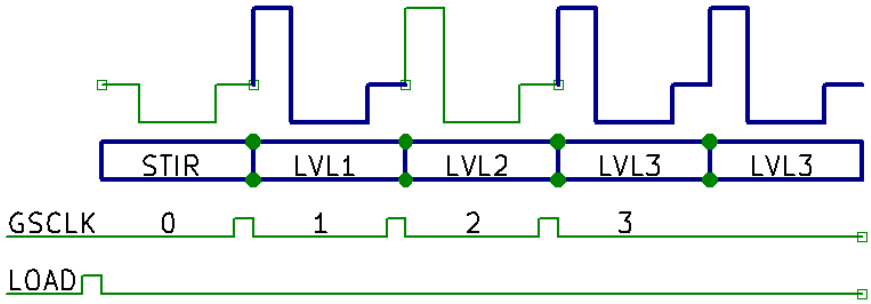
- Voltage Control

  VH2-L/R, VH1-L/R-A/B 로 6 채널이다.

  21.0 >= VH2 > VH1 >= 5.0 V
  0.1V 단위.

- Waveform Setup

  stir firepulse period
  VH1 VH2 VH1 <=======>

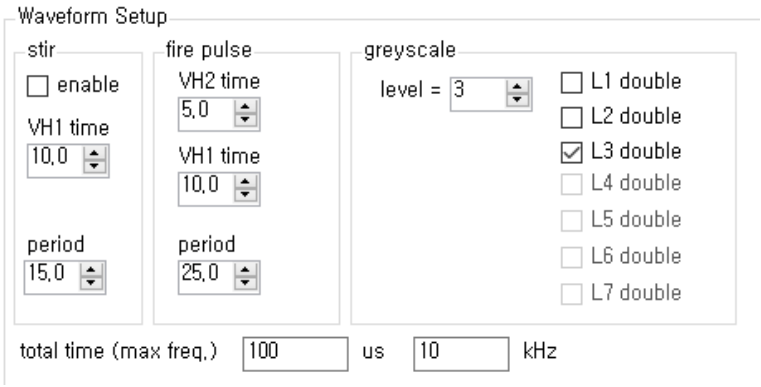


fire pulse : VH2, VH1 으로 이루어진다. period 는 multi-pulse 를 위한 설정값.
stir pulse : enable 시 사용가능하며 VH1 과 period 만을 넣는다.

0.1 us < VH1 < 25.5 us
0.1 us < VH2 < 25.5 us
(VH1+VH2) < period < (VH1+VH2+25.5 us)
0.1 us 단위

greyscale: level = 1 인 경우, SA[0..2]-L/R 에서 SA[0] 비트만 사용되며, 0 또는 1 이다.

설정예)



level=3 이므로 SA[0..1] 까지 사용되고, SA[2]=0 다.
SA=0 인 경우, off,
SA=1 인 경우, LVL1 파형 1개
SA=2 인 경우, LVL1 + LVL2 파형 2개
SA=3 인 경우, LVL1 + LVL2 + LVL3 x 2 (L3 double 체크) 로 파형 4 개
이때 파형간의 간격이 period 값이다.

```
double  VH[] = { VH2R, VH2L, VH1RA, VH1RB, VH1LA, VH1LB };
int greysteps = 1;
int greybits = 1; //  2 : 2 <= greysteps <= 3,  3 : 4 <= greysteps <= 7
int firedouble = 0; // 1 @ L1 + 2 @ L2 + 4 @ L3 + 8 @ L4 + 16 @ L5 + 32 @ L6 + 64 @ L7
int greypattern[] = { greybits, greysteps, firedouble };
double stirtime[] = { 0, 0 }; // disabled or stirVH1time,  stirPaddingTime ( = period - VH1)
```

```
double firetime[] = { fireVH1time, fireVH2time, firePaddingTime }; // firePaddingTime =
period - (VH1+VH2)
setFGKM( VH, greypattern, stirtime, firetime);
```

spitting 테스트

bool selectSpittingNozzle(int nz, bool onoff)
spitting 에 쓸 노즐을 선택한다. nz = 0 ~ 1023

bool enableSpitting(int oncode = 1, bool printdata = false)
oncode : 0 = off, 1 = on
printdata : false 인 경우 selectSpittingNozzle 에 로 노즐의 onoff 결정.
true 인 경우 writePixelData 로 쓴 패턴을 사용하여 spitting 함.
즉 printing 에뮬레이션

bool enableExternalTrigger(bool enable, bool invert = false)
GData 내부 trigger 신호인 ECLK 을 사용하여 spitting 함

bool startSpitting(double kHz, int nDrop = 0)
nDrop = 0 인 경우 stopSpitting 이 호출 될 때까지 무한히 spitting 함.
1 < nDrop < 32768 인 경우 nDrop 개수만큼 spitting 한 후 자동으로 멈춤

bool stopSpitting()
spitting 멈춤

bool isSpittingStopped()
nDrop > 0 인 경우 spitting 이 자동으로 멈추었는지 확인함.

인쇄 테스트

void enableCompression(bool onoff)
writePixelData 에서 출력하는 bit pattern 에서 반복되는 부분을 압축하여 표현할 것인지 설정.
호출하지 않는 경우 기본은 압축임.

bool beginWritePixel()
인쇄 데이터를 쓰기 전에 호출하여 메모리 포인터를 초기화한다.

bool writePixel(int nPixel, int[] PixelPosition, byte[] PixelData)
한 swath 데이터를 전송함.
PixelData.Length = nPixel * 1024

bool isWritePixelFinished()
writePixel 로 전송한 데이터가 이더넷을 통해 전송완료되었는지 확인함.

bool reuseMemory()
beginWritePixel 이후 writePixel 로 쓴 데이터들을 다시 처음부터 읽어서 인쇄에 사용함.

bool startPrinting()
인쇄 시작

bool stopPrinting()
인쇄 종료

헤드 온도 제어

double getHeadTemperature()
온도값 반환

bool setHeadTemperature(double temperature)
목표 온도값 설정

bool setHeadHeaterGain(int P = 150, int Ti = 0)
히터 제어기 게인 설정

bool enableHeadHeater()
bool disableHeadHeater()
히터 onoff

6. 소프트웨어 라이브러리 (EncoderLED)

6-1. 초기화

bool init(int ip)
ip 는 IP addresss 의 마지막 바이트 값으로 192.168.0.100 일 경우 100 이다.

6-2. internal trigger 발생기

bool enableInternalTrigger(double period_us, int count)
모듈 내부 펄스 발생기로 이 출력을 바로 GData 버스 내의 ECLK 로 사용하기도 하고,
이 신호를 에뮬레이션 클럭으로 사용하기도 한다.
period = 0.01 ~ 1,342,180 μs (100kHz ~ 1 Hz)
count = 0 인 경우 무한
1 ~ 67,108,863 인 경우, 자동 멈춤

bool disableInternalTrigger()
모듈 내부 펄스 발생기 멈춤

bool isDoneInternalTrigger()
모듈 내부 펄스 발생기가 멈추었는지 확인

6-3. Trigger Out 신호 출력

bool setTriggerOut(int source)
trigger out 단자에 신호 할당
0 : internal trigger 내부 펄스 발생기 출력
1: position trigger 엔코더 A/B 신호 4분주 신호
2: R 신호
3: A 신호

6-4. 엔코더

bool swapAB(bool swap)
A/B 채널을 서로 바꾸어서 엔코더 진행방향을 바꿀 수 있다.

bool setSourceAB(int source) // 0 = external, 1 = emulation, 2 = none

0: 엔코더 모듈의 출력 A/B 를 외부 엔코더 신호로 하는 경우
1: 엔코더 내부의 에뮬레이션 신호를 출력하는 경우

bool setSourceR(int source)
0: 엔코더 모듈의 출력 R 를 외부 엔코더 신호로 하는 경우
1: 엔코더 내부의 에뮬레이션 신호를 출력하는 경우

bool resetPosition()
엔코더 내부 카운터 리셋 (디버깅용)

int getPosition()
엔코더 내부 카운터 (디버깅용)

bool resetABEmulation(int initialstate)
내부 A/B 신호 발생기 초기화
initialstate = 초기의 A/B 상태값 0 ~ 3

bool enableABEmulation(int triggersource, int direction)
내부 A/B 신호 발생기 클럭 및 방향 설정
triggersouce = 0: 내부 펄스 발생기
1: 외부 trigger in 신호
direction = 0: forward
1: backward

bool disableABEmulation()
내부 A/B 신호 발생기 정지

bool enableREmulation(int triggersource, double duration, bool invert, int triggerposition)
bool disableREmulation()
내부 R 신호 발생기

6-5. LED

2채널을 구분하는 인자 : channel = 0, 1

bool setLEDVoltage(int channel, double volt)

각 채널별로 LED 의 전압을 설정한다. volt = 2 ~ 14.5V
LED strobo 는 turn on 시간이 매우 짧으므로, LED 전압값은 규격값보다 높은 값을 사용할 수 있

다. 즉, 3.5V 에서 켜지는 LED 의 경우 3.5V 보다 높은 전압을 설정하여 순간적인 전류가 더 강하게 흐르게 할 수 있다.
LED 는 열에 의해 파괴되므로, 순간적으로 전류가 정격보다 많이 흘러도, 그 주기에 비해서 turn on 시간이 짧다면 파괴되지 않는다.

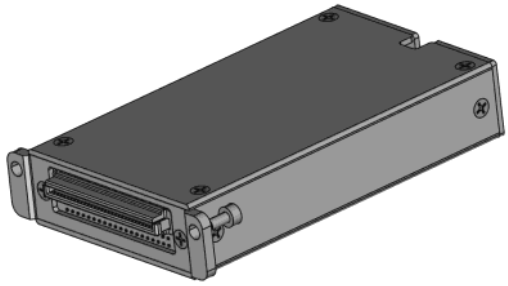
bool setLEDTime(int channel,double delay,double duration,double delay2 = 0)
delay2 = delay + interval (발광시간 간격)

bool enableLED(int channel,bool second = false)
LED 를 켤 수 있다.
이때 두 번째 발광 (delay2 에 의해 타이밍이 주어지는) 을 사용하려면 second = true 로 주어야한다.

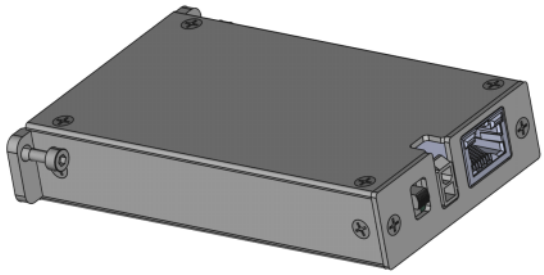
bool disableLED(int channel)
LED 를 끈다.

7. GData 사양

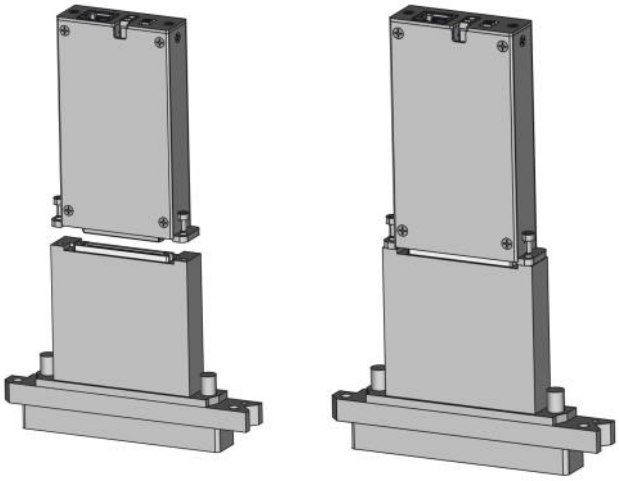
8. pcKM1024i 사양



케이블 없이 헤드에 직결하는 커넥터



전원 커넥터 및 스위치는 RD256M 과 동일



결합은 헤드에 직결 후, 나사 체결

전체적인 크기도 기존보다 더 작아짐.