

PCIe TO M.2 HAT+

Overview

Introduction

PCIe to M.2 M key adapter board for Raspberry Pi 5, compatible with M.2 drives in 2230 / 2242 size, supports Gen2 and Gen3 modes, supports booting from SSD for PI5.

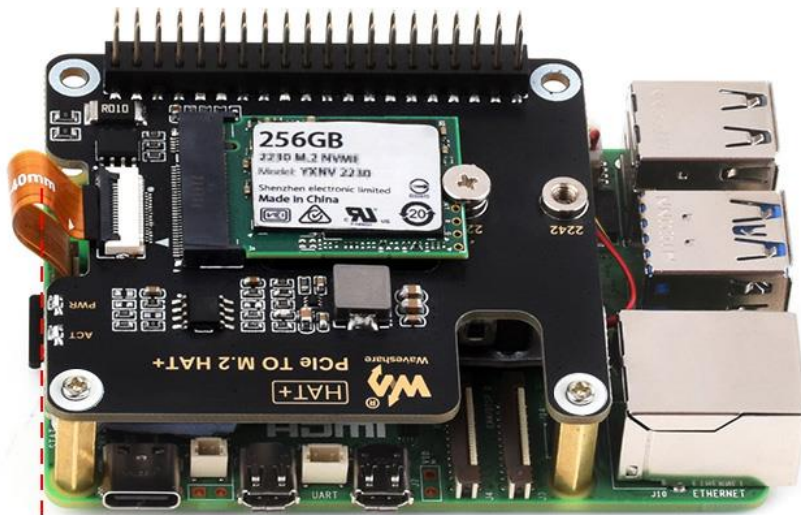
Features

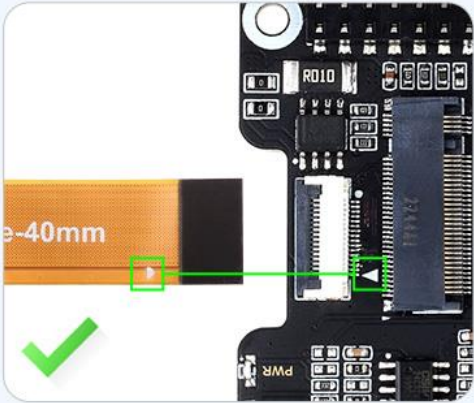
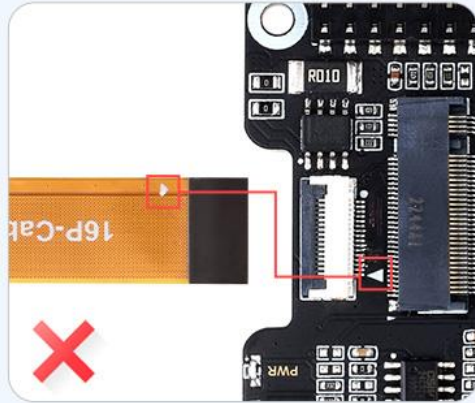
- Support NVMe protocol M.2 solid state drive, high-speed read/write, with high work efficiency.
- PCI-E×1 Gen2 or Gen3 mode.
- Only supports PI5B.
- Compatible with M.2 solid state drive of 2230/2242 sizes.
- Onboard working indicator lights, with PWR on continuously when powered, and ACT blinking during read/write, making the operating status clear at a glance.
- HAT+ design, with on-board EEPROM chip.
- Onboard power monitoring chip, for real-time monitoring of the working status of the solid-state drive.
- Reserved airflow vent for PI5 fans, increasing airflow, and reducing solid-state temperatures.

Hardware Description

Hardware Connection

Pay attention to the direction of the row of wires and connect as shown in the figure:



	
<p>Ensure the both triangles as shown above are in the same side when connecting the cable.</p>	<p>The HAT will not work if connection is reversed, and may cause malfunction or damage of the board</p>

Load

1: Enable PCIe Interface:

The PCIe interface is not enabled by default on PI5B. To enable it, add the following configuration in `/boot/firmware/config.txt`:

The PI5B does not have the PCIE interface enabled by default, add it in `/boot/firmware/config.txt`:

```
dtparam=pciex1
```

2: PCIE is gen2 by default, if you need to enable PCIE gen3, add it in `/boot/firmware/config.txt`:

```
dtparam=pciex1_gen=3
```

3: After modifying it and restarting PI5, you can recognize the device.

As shown below, the identified SM2263 is my SSD, and the other PI5 is the RPI chip.

```
pi@raspberrypi:~$ lspci
0000:00:00.0 PCI bridge: Broadcom Inc. and subsidiaries Device 2712 (rev 21)
0000:01:00.0 Non-Volatile memory controller: Silicon Motion, Inc. SM2263EN/SM2263XT SSD Controller (rev 03)
0001:00:00.0 PCI bridge: Broadcom Inc. and subsidiaries Device 2712 (rev 21)
0001:01:00.0 Ethernet controller: Device 1de4:0001
```

4: Note that skip this step if you have partitioned and formatted on other platforms (will delete all data from the SSD and proceed with caution).

`lsblk` for viewing the disk (If you want to see the details run `sudo fdisk -l`)

```
pi@raspberrypi:~$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
mmcblk0     179:0    0  29.7G  0 disk
├─mmcblk0p1 179:1    0   512M  0 part /boot/firmware
└─mmcblk0p2 179:2    0  29.2G  0 part /
nvme0n1     259:0    0 119.2G  0 disk
└─nvme0n1p1 259:1    0 119.2G  0 part
```

Partition

`sudo fdisk /dev/nvme0n1` The device number is the total device number, don't add p1, that's just one partition

How to use `fdisk` partition tool

n New Partition

q Exit without saving

p Print partition table

m Print selection menu

d Delete Partition

w Save to exit

t Modify ID number

Add the partition to execute n can be, the last w save exit

5: Format:

sudo mkfs. Then press the tab to see a variety of different suffixes, the different suffixes are the formats you need to format.

```
pi@raspberrypi:~$ sudo mkfs.  
mkfs.bfs  mkfs.cramfs  mkfs.exfat  mkfs.ext2  mkfs.ext3  mkfs.ext4  mkfs.fat  mkfs.minix  mkfs.msdos  mkfs.ntfs  mkfs.vfat
```

If I want to format to the ext4 file format, this would be done by executing the following command:

```
sudo mkfs.ext4 /dev/nvme0n1p1
```

Wait for a few moments, when done has appeared, it means that the formatting has been carried out.

```
pi@raspberrypi:~$ sudo mkfs.ext4 /dev/nvme0n1p1  
mke2fs 1.47.0 (5-Feb-2023)  
Discarding device blocks: done  
Creating filesystem with 31258368 4k blocks and 7815168 inodes  
Filesystem UUID: 1a84fb29-5460-475f-afb7-0a90271ef975  
Superblock backups stored on blocks:  
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,  
    4096000, 7962624, 11239424, 20480000, 23887872  
  
Allocating group tables: done  
Writing inode tables: done  
Creating journal (131072 blocks): done  
Writing superblocks and filesystem accounting information: done
```

6: Mount:

Create Mount Directory

```
sudo mkdir toshiba
```

Mount the device

```
sudo mount /dev/nvme0n1p1 ./toshiba
```

Checking disk status

```
df -h
```

Read/Write Test

Enter the directory where the disk is mounted:

```
cd toshiba
```

- Release memory:

```
sudo sh -c "sync && echo 3 > /proc/sys/vm/drop_caches"
```

- Copying the contents of the Raspberry Pi's memory to the hard drive (Write):

```
sudo dd if=/dev/zero of=./test_write count=2000 bs=1024k
```

```
pi@raspberrypi:~/toshiba $ sudo dd if=/dev/zero of=./test_write count=2000 bs=1024k
2000+0 records in
2000+0 records out
2097152000 bytes (2.1 GB, 2.0 GiB) copied, 3.78947 s, 553 MB/s
```

- Copying the contents of the hard drive's memory to the Raspberry Pi (Write):

```
sudo dd if=./test_write of=/dev/null count=2000 bs=1024k
```

```
pi@raspberrypi:~/toshiba $ dd if=./test_write of=/dev/null count=2000 bs=1024k
2000+0 records in
2000+0 records out
2097152000 bytes (2.1 GB, 2.0 GiB) copied, 3.53634 s, 593 MB/s
```

- Note: Different cards and environments have different test results. As the Raspberry Pi is more vulnerable to being affected, if you want to test the exact performance, you can use a PC to test.

NVMe SSD Booting

- 1: Start the Raspberry Pi using the SD card, mount, and test to ensure that the hardware is functioning properly.
- 2: Boot the Raspberry Pi using the SD card and modify the boot settings by changing the BOOT_ORDER configuration in the Raspberry Pi's bootloader.

```
sudo rpi-eeeprom-config --edit
```

```
Modify BOOT_ORDER=0xf41 as BOOT_ORDER=0xf416
```

Value	Mode	Description
0x0	SD CARD DETECT	Try SD then wait for card-detect to indicate that the card has changed - deprecated now that 0xf (RESTART) is available.
0x1	SD CARD	SD card (or eMMC on Compute Module 4).
0x2	NETWORK	Network boot - See Network boot server tutorial
0x3	RPIBOOT	RPIBOOT - See usbboot
0x4	USB-MSD	USB mass storage boot - See USB mass storage boot
0x5	BCM-USB-MSD	USB 2.0 boot from USB Type C socket (CM4: USB type A socket on CM4IO board). Not available on Raspberry Pi 5.
0x6	NVME	CM4 and Pi 5 only: boot from an NVMe SSD connected to the PCIe interface. See NVMe boot for more details.
0x7	HTTP	HTTP boot over ethernet. See HTTP boot for more details.
0xe	STOP	Stop and display error pattern. A power cycle is required to exit this state.
0xf	RESTART	Restart from the first boot-mode in the BOOT_ORDER field i.e. loop

For details, please refer to [BOOT_ORDER](#)

- 3: Reboot the Raspberry Pi and look at the serial port logs during boot up to see that:

```
USB-PD: src-cap PDO object1 0x0a0191f4
Current 5000 mA
Voltage 5000 mV
USB-PD: src-cap PDO object2 0x0002d12c
Current 3000 mA
Voltage 9000 mV
USB-PD: src-cap PDO object3 0x0003c0e1
Current 2250 mA
Voltage 12000 mV
USB-PD: src-cap PDO object4 0x0004b0b4
Current 1800 mA
Voltage 15000 mV
Trying partition: 0
type: 32 lba: 8192 'mkfs.fat' ' bootfs      ' clusters 261116 (4)
rsc 32 fat-sectors 2040 root dir cluster 2 sectors 0 entries 0
FAT32 clusters 261116
[sdcard] autoboot.txt not found
Trying partition: 0
type: 32 lba: 8192 'mkfs.fat' ' bootfs      ' clusters 261116 (4)
rsc 32 fat-sectors 2040 root dir cluster 2 sectors 0 entries 0
FAT32 clusters 261116
Read config.txt bytes      1695 hnd 0x2b0
SIG pieeprom.sig 483088fe21cfb6848e34db472960240da77cd243588a2684079ec3481f053868 1705300385
SELF-UPDATE timestamp current 1701752716 new 1705300385
Updating bootloader EEPROM
Reading EEPROM: 2097152 bytes 0x3c960000
1363ms
Writing EEPROM
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.....
.....
.....
.....+
5902ms

Verify BOOT EEPROM
Reading EEPROM: 2097152 bytes 0x3c960000
1363ms
BOOT-EEPROM: UPDATED
EEPROMs updated. Rebooting
RESET
```

This indicates that the modification was successful.

If you find that you can't modify the file several times, please connect to the network before modifying the file (wait for the network to time itself), or set the correct time before modifying the file.

4: Just program the system into NVME, then connect the board, remove the SD card, and re-power it up.

NVME Power Monitoring

Onboard the INA219 chip for detecting the voltage and current, easy to monitor the device status and monitor the input 5V voltage status (not 3.3V). The default I2C address is 0x40, addresses can be modified via back resistors to support stacking of different expansion boards.

Demo:

```
wget https://files.waveshare.com/wiki/PCIe-TO-M.2-HAT%2B/PCIe TO M.2 HAT%2B.zip
```

```
unzip -o PCIe_TO_M.2_HAT+.zip -d ./PCIe_TO_M.2_HAT+
```

```
cd PCIe_TO_M.2_HAT
```

```
sudo python INA219.py
```

```
pi@raspberrypi:~/PCIe_TO_M.2_HAT$ sudo python INA219.py
Load Voltage: 5.016 V
Current:      0.057 A
Power:       0.280 W
```

Resource

Datasheet

- [INA Datasheet](#)