Software Setup

1. Enable PCIe interface:

PCIE interface is enabled on the PI5B by default.
If the PCIE interface is not enabled, you add the following content in "/boot/firmware/config.txt":
dtparam=pciex1

2. The module only supports PCIE gen2 x1.

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 PCI bridge: ASMedia Technology Inc. ASM1182e 2-Port PCIe x1 Gen2 Packet Switch
0000:02:03.0 PCI bridge: ASMedia Technology Inc. ASM1182e 2-Port PCIe x1 Gen2 Packet Switch
0000:02:07.0 PCI bridge: ASMedia Technology Inc. ASM1182e 2-Port PCIe x1 Gen2 Packet Switch
0000:03:00.0 Non-Volatile memory controller: Sandisk Corp WD Blue SN580 NVMe SSD (DRAM-less) (rev 01)
0000:04:00.0 Non-Volatile memory controller: Sandisk Corp WD Blue SN580 NVMe SSD (DRAM-less) (rev 01)
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 do use fdisk

n New partition

q Exit without saving

p Print partition table

m Print selection menu

d Delete partition

w Save and exit

t Modify ID

Execute n to add the partition, at last execute w to save and 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, execute 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 the mounting directory:

sudo mkdir toshiba

Mount the device

sudo mount /dev/nvme0n1p1 ./toshiba

Check 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 memory to the hard driver (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 to the Raspberry Pi's memory (“/etc/fstab” for reading).

```
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 make 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.

Auto Mount

If the testing is sound and you don't need it as a system disk and only use an extended disk, you can set up an automatic mounting.

```
sudo nano /etc/fstab


#Add the following content at the end:

/dev/nvme0n1p1  /home/pi/toshiba  ext4  defaults  0  0

#/dev/nvme0n1p1 is the device name, /home/pi/toshiba is the the directory to be mounted, ext4 is the file system type, defaults are using the default mounting settings

#Make the changes take effect (make sure test without problems before rebooting, otherwise it will lead to failure to mount or boot)


sudo mount -a


#And then reboot

Check the device through “lsblk”
```

Booting from NVMe SSD

1: First, you can use an SD card to boot the Raspberry Pi, just test it to make sure the hardware can work properly.

2: Use the SD card to boot the Raspberry Pi and modify the config file, modify BOOT_ORDER:

```
sudo rpi-eeprom-config --edit
```

Add:

```
NVME_CONTROLLER=1
```

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 more details, you can refer to BOOT_ORDER

If you want to realize SD card boot priority, change to BOOT_ORDER=0xf461

Note: The board has onboard two or more M.2 ports, one of them is used as boot, it is recommended to connect the SSD used as boot to NVME1, the priority is NVME1 first.

3: Reboot the Raspberry Pi:

If you fail after trying several times, you can connect it to the network before modify again (wait for network time synchronization), or set the correct time before modifying the file.

4: Flash the system to NVME, and then connect to the board, remove the SD card, and power it on again.

NVMe Power Monitoring

The onboard INA219 chip can detect 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/upload/0/06/PCIE_HAT_INA219.zip
unzip -o PCIE_HAT_INA219.zip -d ./PCIE_HAT_INA219
cd PCIE_HAT_INA219
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
```