入门

要求

• 1 x 带数据传输功能的USB Type-C®电缆（用于将PC连接到主板的数据端口）
• 1 x 12~19V电源 *
• 1 x 显示器，带HDMI电缆或USB Type-C®（DP）电缆
• 1 x 键盘和鼠标套装
* 电源单独购买。

软件准备

获取Tinker Edge R ROM

查看华硕Tinker Edge R官方网站以获取最新镜像。
https://tinker-board.asus.com/download-list.html，从菜单中选择Tinker Edge R。

获取Tinker Edge R Flash工具（Windows/Linux命令行）

在ROM镜像目录中找到命令行flash工具。

[Windows]安装Rockchip驱动程序

在ROM镜像目录中找到DriverAssitant压缩包，解压缩后执行DriverInstall.exe安装驱动程序。

烧录的Tinker Edge R (2021/8/1 之后发布的镜像)

***从板载 eMMC 启动**

注意：从板载 eMMC 启动仅适用于带有 eMMC 的型号。

要求：

• 1 x 具有数据传输功能的 Type-C® 数据线

• 1 x 电源

• 1 x 显示器

• 1 x 键盘和鼠标套装

步骤：

i. 确保 Recovery 标头上没有跳线。
ii. 使用 Type-C® 数据线将 Tinker Edge R 连接至 PC。
iii. 将电源适配器连接到 Tinker Edge R。
iv. 从 Tinker Board 网站 （https://tinker-board.asus.com/download-list.html?product=tinker-edge-r） 下载 TinkerOS 映像，然后使用第三方 ISO 软件（如 balenaEtcher）将其刻录到 Tinker Board 中。
v. 成功刻录 TinkerOS 映像后，请断开 Tinker Board 上的所有电缆。
vi. 将电源、键盘、鼠标和显示器连接到 Tinker Board 以启动。

自定义设置

更改键盘布局

键盘布局设置为英语（美国）作为默认设置。请参阅下面的过程更改语言。

```
sudo dpkg-reconfigure keyboard-configuration
sudo reboot
```

更改时区/日期/时间

使用操作系统中的Timedatectl内置工具更改时间。

- **时区**

  打印时区列表。

  ```
  timedatectl list-timezones
  ```

  确定哪个时区与您所在的位置相符后，以sudo用户身份运行以下命令：

  ```
  sudo timedatectl set-timezone your_time_zone
  ```

  例如，要将系统的时区更改为`Europe/Ljubljana`，您可以运行：

  ```
  sudo timedatectl set-timezone Europe/Ljubljana
  ```

- **日期/时间**

  启用NTP

  ```
  sudo timedatectl set-ntp yes
  ```

  禁用NTP

  ```
  sudo timedatectl set-ntp no
  ```

  设置时间（需要禁用NTP）

  ```
  sudo timedatectl set-time “2020-05-12”
  sudo timedatectl set-time “18:10:40”
  sudo timedatectl set-time “2020-05-12 18:10:40”
  ```

- **验证**

  通过发出timedatectl命令打印以验证更改：

  ```
  timedatectl
  ```

范例：

```
Local time: Mon 2019-03-11 22:51:27 CET
Universal time: Mon 2019-03-11 21:51:27 UTC
RTC time: Mon 2019-03-11 21:51:26
Time zone: Europe/Ljubljana (CET, +0100)
Network time on: yes
NTP synchronized: yes
RTC in local TZ: no
```

检查屏幕分辨率

- 方法1：从UI界面
  - 使用监视器设置直接更改分辨率。
- 方法2：终端（命令行）-xrandr

```
#列出所有可用的输出分辨率
$xrandr
```

您也可以使用xrandr设置不同的分辨率（必须存在于上面的列表中）

```
$ xrandr --output HDMI-1 --mode 1920 x1080
```

为未列出的解决方案

```
$cvt 1024 768 60
#1024x768 59.92 Hz（CVT 0.79M3）hsync：47.82 kHz; pth：63.50 MHz
Modeline "1024x768_60.00"63.50 1024 1072 1176 1328 768 771 775 798 -hsync +vsync


$xrandr --newmode"1024x768"63.50 1024 1072 1176 1328 768 771 775 798 -hsync +vsync
$ xrandr --addmode HDMI 1 1024x768
$ xrandr --output HDMI1 --mode 1024 x768
```

在xrandr的wiki上查看详细信息

https://xorg-team.pages.debian.net/xorg/howto/use-xrandr.html

检查音频输出接口

- **输出设备：**

输出设备	描述
rockchiprk809	音频插孔
rkhdmidpsound	HDMI/DP音频

检查Internet连接

- **以太网**

  1. 将以太网电缆连接到电路板。
  2. 使用以下命令检查详细的连接信息。

  ```
  Ifconfig eth0
  ```

- **Wi-Fi**

  通过在设备terminal运行以下命令来选择Wi-Fi网络

  ```
  nmtui
  ```

  然后选择`激活连接`并从`Wi-Fi（wlan 0）`下的列表中选择网络。

  或者，使用以下命令连接到已知的网络名称：

  ```
  nmcli dev wifi connect <NETWORK_NAME> password <PASSWORD> ifname wlan0
  ```

  使用以下命令验证您的连接：

  ```
  nmcli connection show
  ```

  您应该会在输出中看到所选的网络，举例来说：

名称	UUID	类型	装置
MyNetworkName	61f5d6b2-5f52-4256-83ae-7f148546575a	802-11-无线	wlan0

GPIO

下表显示了接头引脚，包括每个端口的sysfs路径，这通常是使用外围库时所需的名称您还可以通过在命令行中键入pinout来查看接头引脚。

注意事项：

I. 32、33、37号I/O引脚均为+3.0V电平，内部有61 K欧姆下拉电阻，3 mA驱动电流容量。

II. 除32、33、37号引脚外，其余引脚均为+3.3V电平，内置5 K ~ 10 K Ohm上拉电阻，驱动电流容量为50 mA。

sysfs路径	引脚功能	销		引脚功能	sysfs路径
	+3.3V电源	**1**	**2**	+5V电源	
/dev/i2c-6 /sys/class/gpio/gpio73	I2C 6 （SDA） GPIO2_B1	**3**	**4**	+5V电源	
/dev/i2c-6 /sys/class/gpio/gpio74	I2C 6 （SCL） GPIO2_B 2	**5**	**6**	Gnd	
/sys/class/gpio/gpio89	测试 （CLKOUT1） GPIO2_D1	**7**	**8**	N0（TX） GPIO2_C1	/设备/ttyS 0 /sys/class/gpio/gpio81
	Gnd	**9**	**10**	接口0 （RX） GPIO2_C0	/dev/ttyS0 /sys/class/gpio/gpio80
/dev/ttyS0 /sys/class/gpio/gpio83	网络0 （RTSN） GPIO2_C3	**11**	**12**	I2 S0 （SCLK） GPIO3_D0	/sys/class/gpio/gpio120
/dev/spidev5 /sys/class/gpio/gpio85	SPI 5 （TXD） GPIO2_C5	**13**	**14**	Gnd	
/dev/spidev5 /sys/class/gpio/gpio84	SPI 5 （RXD） GPIO2_C4	**15**	**16**	SPI 5 （CLK） GPIO 2_C6	/dev/spidev5 /sys/class/gpio/gpio86
	+3.3V电源	**17**	**18**	SPI 5 （CSN 0） GPIO2_C7	/dev/spidev5.0 /sys/class/gpio/gpio87

/dev/spidev1 /dev/ttyS4 /sys/class/gpio/gpio40	SPI 1（TXD） UART 4 （TX） GPIO1_B0	**19**	**20**	gnd	
/dev/spidev32766 /dev/ttyS4 /sys/class/gpio/gpio39	SPI 1（RXD） USB 4 （RX） GPIO 1_A7	**21**	**22**	I2S0 （SDI1SDO3） GPIO3_D4	/sys/class/gpio/gpio124
/dev/spidev1 /sys/class/gpio/gpio41	SPI 1 （SLK） GPIO1_B 1	**23**	**24**	SPI 1 （CSN 0） GPIO 1_B2	/dev/spidev1.0 /sys/class/gpio/gpio42
	gnd	**25**	**26**	PWM3A GPIO0_A 6	/sys/class/pwm/pwm chip3/pwm 0 /sys/class/gpio/gpio6
/dev/i2c-7 /sys/class/gpio/gpio71	I2C7 （SDA） GPIO2_A 7	**27**	**28**	I2C7 （SCL） GPIO2_B 0	/dev/i2c-7 /sys/class/gpio/gpio72
/sys/class/gpio/gpio126	IS2S0 （SDI3SDO1）	**29**	**30**	gnd	

sysfs路径	引脚功能	销		引脚功能	sysfs路径
	GPIO3_D6				
/sys/class/gpio/gpio125	IS2S0 （SDI2SDO2） GPIO3_D5	**31**	**32**	PWM0G PIO4_C2	/sys/class/pwm/pwm chip0/pwm 0 /sys/class/gpio/gpio146
/sys/class/pwm/pwm chip1/pwm 0 /sys/class/gpio/gpio150	PWM 1GPIO4_ C6	**33**	**34**	gnd	
/sys/class/gpio/gpio121	I2S0 （LRCK） GPIO3_D1	**35**	**36**	UART 0 （CTSN） GPIO2_C2	/设备/ttyS 0 /sys/class/gpio/gpio82
/sys/class/gpio/gpio149	SPDIF （TX） GPIO4_C 5	**37**	**38**	I2S0 （SDI0） GPIO 3_D3	/sys/class/gpio/gpio123
	gnd	**39**	**40**	I2S0 （SDO0） GPIO3_D7	/sys/class/gpio/gpio127

警告：处理GPIO引脚时要小心，避免静电放电或接触导电材料（金属）。如果不能正确处理Tinker Edge R，可能会导致短路、触电、严重伤害、死亡、火灾或损坏您的电路板和其他财产。

使用其他库

要访问Tinker Edge T上的头引脚，您可以使用标准的Linux sysfs接口。但是如果你想要一个Python API，我们建议你使用python-periphery库，它构建在sysfs接口之上。

您可以按如下方式在开发板上安装库

```
sudo apt-get update
sudo apt-get install python3-pip

sudo pip3 install python-periphery
```

注意事项：

- 要访问开发板上的外围硬件资源，您需要使用sudo权限运行代码
- 需要Python 3版本的编译器 -

通过配置库，您可以选择带有引脚号的GPIO或PWM引脚。其他接口（如I2C和UART引脚）必须使用引脚的器件路径指定请参见以下示例。

```
#硬件接口配置#

##注意：uart4和spi1是相同的引脚。当两者都打开时，设置后者。#注意：fiq_debugger和uart0使用相同的引脚。当两者都打开时，首先设置fiq_debugger。##

intf:
fiq_debugger=on
#intf: uart0=off
#intf: uart4=off
#intf: i2c6=off
#intf: i2c7=off
#intf: i2s0=off
#intf: spi1=off
#intf: spi5=off
#intf: spim0 =off
#intf: spim1 =off
#intf: spim3a =off
```

- 您可以编辑/boot/boot. txt来切换40针功能，目前可切换的功能如下：

注意事项：

##注意：uart4和spi1是相同的引脚。当两者都打开时设置后者##

##注意：fiq_debugger和uart0使用相同的引脚。当两者都打开时，首先设置fiq_debugger##

GPIO

下面的代码显示了如何使用配置实例化每个GPIO引脚：

注：如果/boot/config.txt 没有预设，所有 40 引脚都可以使用 GPIO

```
gpio3 = GPIO（3, “in”）
gpio5 = GPIO（5, “in”）
gpio7 = GPIO（7, “in”）
gpio120 = GPIO（120, “in”）
gpio124 = GPIO（124, “in”）
```

有关更多示例，请参见https://python-periphery.readthedocs.io/en/latest/gpio.html。

PWM

编辑/boot/pwm. txt以启用pwm功能。

```
intf: pwm0=on
intf: pwm1=on
intf: pwm3a=on
```

下面的代码显示了如何使用编译器实例化每个PWM引脚：

```
# PWM0 = pwmchip0, pwm0
pwm0 = PWM(0, 0)
# PWM1 = pwmchip1, pwm0
pwm1 = PWM(1, 0)
# PWM3 = pwmchip3, pwm0
pwm3 = PWM(2, 0)
```

有关用法示例，请参见https://python-periphery.readthedocs.io/en/latest/pwm.html。

I2c

编辑/boot/boot. txt以启用i2c功能。

```
intf: i2c6=on
intf: i2c7=on
```

下面的代码显示了如何使用配置实例化每个I2C端口：

```
i2c2 = I2C（“/dev/i2c-6”）
i2c3 = I2C（“/dev/i2c-7”）
```

有关用法示例，请参见https://python-periphery.readthedocs.io/en/latest/i2c.html。

SPI

编辑config.txt以启用SPI功能：

```
intf: spi1=on
intf: spi5=on
```

下面的代码显示了如何使用配置实例化每个SPI端口：

```
#SPI 1, SS 0, Mode 0, 10MHz
spi1_0 = SPI（"/dev/spidev1.0", 0, 10000000）
#SPI 5, SS 0, Mode 0, 10MHz
spi1_1 = SPI（"/dev/spidev5.0", 0, 10000000）
```

有关用法示例，请参见https://python-periphery.readthedocs.io/en/latest/spi.html。

UART

##注意：fiq_debugger和uart0使用相同的引脚。当两者都打开时，首先设置fiq_debugger

编辑/boot/boot. txt以启用启动功能：

```
intf: fiq_debugger=off
intf: uart0=on
intf: uart4=on
```

下面的代码显示了如何使用配置实例化每个端口

```
# UART0, 115200波特
uart 1 = Serial（"/dev/ttyS 0", 115200）
#UART 4, 9600波特
uart3 = Serial（"/dev/ttyS4", 9600）
```

有关用法示例，请参见https://python-periphery.readthedocs.io/en/latest/serial.html。

示例代码

blink.py

```
from periphery import GPIO
import time
LED_Pin = 73 #Physical Pin-3 is GPIO 73
# Open GPIO /sys/class/gpio/gpio73 with output direction
LED_GPIO = GPIO(73, "out")
while True:
	try: #Blink the LED
		LED_GPIO.write(True)
		# Send HIGH to switch on LED
		print("LED ON!")
		time.sleep(0.5)
		LED_GPIO.write(False)
		# Send LOW to switch off LED
		print("LED OFF!")
		time.sleep(0.5)
	except KeyboardInterrupt:
		# Turn LED off before stopping
		LED_GPIO.write(False)
		break
	except IOError:
		print ("Error")
LED_GPIO.close()
```

示例（运行）

```
sudo python3 blink.py
```

如何查看当前硬件信息

当前CPU频率

读取当前实时CPU频率：

双核Cortex-A72（高达1.8GHz）

```
sudo cat/sys/devices/system/cpu/cpu4/cpufreq/scaling_cur_freq
```

四核Cortex-A53（高达1.4GHz）

```
sudo cat/sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```

当前GPU频率

读取当前GPU频率

```
sudo cat/sys/class/devfreq/ff9a0000.gpu/cur_freq
```

当前CPU GPU温度

实时监控SoC温度

```
Watch -n 1 sudo cat/sys/class/thermal/thermal_zone0/temp
```

高级脚本

将下面的文本保存为hwinfo_moniter. sh

```
#!/bin/bash
soc_temp=$(sudo cat /sys/class/thermal/thermal_zone0/temp | awk '{printf "%.2f", $0 / 1000}')
cpu_freq=$(sudo cat /sys/devices/system/cpu/cpufreq/policy0/cpuinfo_cur_freq | awk '{printf "%.2f", $0 / 1000000}')
gpu_freq=$(sudo cat /sys/class/devfreq/ff9a0000.gpu/cur_freq | awk '{printf "%.2f", $0 / 1000000}')
echo "SoC Temp=> $soc_temp degree C"
echo "CPU Freq=> $cpu_freq GHz"
echo "GPU Freq=> $gpu_freq MHz"
```

范例:

```
$ sudo chmod +x hwinfo_moniter.sh
$ ./hwinfo_moniter.sh
SoC => 55.00°C
```