Getting Started

Requirement

• 1 x USB Type-C® cable with data transfer function (to connect your PC to the board's data port)
• 1 x 12~19V power supply*
• 1 x Monitor with HDMI cable or USB Type-C® (DP) cable
• 1 x Keyboard and Mouse set
* The power supply is purchased separately.

Software preparation

Get Tinker Edge R ROM Image

Check ASUS Tinker Edge R official website to get latest image.
https://tinker-board.asus.com/download-list.html, select Tinker Edge R from dropdown menu.

Get Edge R Flash tool (Windows GUI version)

Check ASUS Tinker Edge R official website to get newest version.
https://tinker-board.asus.com/download-list.html, select Tinker Edge R from dropdown menu.

Get Edge R Flash tool (Windows/Linux Command line)

Find the command line flash tool in ROM image directory.

[Windows] Install Rockchip Driver

Find the DriverAssitant zip package in ROM image directory, unzip it and execute DriverInstall.exe to install driver.

Flashing the Tinker Edge R

Initiating MASKROM mode

I. Connect the USB Type-C® cable to the USB Type-C® ports on the Tinker Edge R and your host computer.
II. Before you begin the flashing procedure, please ensure of the following:
• The board is completely powered off, and the power cord and cables connecting the board to your computer are all disconnected.
• In order to set boot mode to **MASKROM** mode, use metal object or a jumper cap to short recovery header.
III. Power on the Tinker Edge R, board should automatically be booted into MASKROM mode.
IV. Remember to remove the jumper cap upon power on.

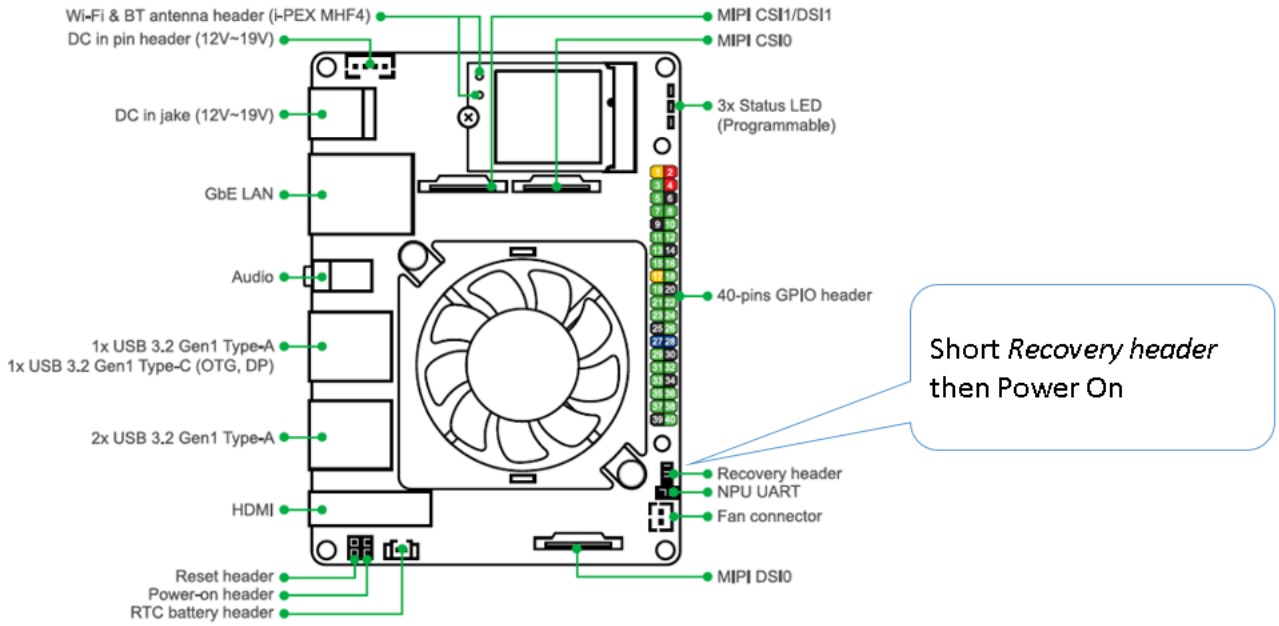

Figure1: Recovery header pin position

(Notes: Remember to remove jumper upon power on)

Executing the flash tool

I. Download the OS image from the Tinker Edge R website, then unzip the image file.
II. Run the GUI flash tool (Windows OS) or command line (Linux) to start up the flash process. The flash process should take a few minutes.
III. Once flash completed, Tinker board will be automatically rebooted.

A. Windows Flash tool (GUI)

- Check Device Manager to ensure “Rockusb device” is detected. if problem encountered:
  - Try to reconnect cable directly to PC’s USB port without hub

- Short Recovery header then power on again
- Try to re-install Rockchip driver.

- Unzip GUI flash tool package, run **Tinker-Flash-Tool.exe** (Run as **Administrator** option)
- Follow the instruction, select OS image file, pick the target Tinker board and execute the flash.

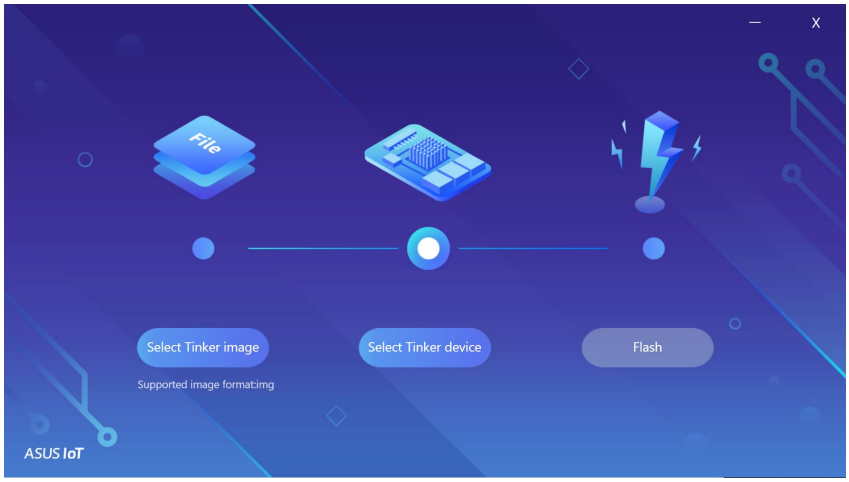

Figure2: Main user interface of ASUS GUI flash tool

B. Flash tool-command Line (Windows, Linux)

- Make sure Recovery header is no longer being shorted after power on.
- Run the flash script **flash.cmd** for Windows or **flash.sh** for Linux to start the flash process.
- Refer to README.txt for more information.

Customize Settings

Change Keyboard Layout

The keyboard layout is set to English (US) as default setting. Refer to the process below to change the language.

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

Change Time Zone/Date/Time

Use Timedatectl built-in tool in OS to change Time.

- **Time Zone**

  Print Time Zone list.

  ```
  timedatectl list-timezones
  ```

  Once you identify which time zone is accurate to your location, run the following command as sudo user:

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

  For example, to change the system's timezone to `Europe/Ljubljana` you would run:

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

- **Date/Time**

  Enable NTP

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

  Disable NTP

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

  Set Time (need disable 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"
  ```

- **Verify**

  Print to verify the change by issuing the `timedatectl` command:

  ```
  timedatectl
  ```

  Example:

  ```
  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
  ```

Check Screen's Resolution

- Method 1: from UI interface
  - Using the Monitor settings to change the resolution directly.
- Method 2: Terminal (Command line) – xrandr

```
# list all the available output resolution
$ xrandr
```

You can also use xrandr to set different resolution (must be present in the above list) on some output:

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

Adding for unlisted resolution

```
$ cvt 1024 768 60
# 1024x768 59.92 Hz (CVT 0.79M3) hsync: 47.82 kHz; pclk: 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 HDMI1 1024x768
$ xrandr --output HDMI1 --mode 1024x768
```

Check details on the wiki of the xrandr: https://xorg-team.pages.debian.net/xorg/howto/use-xrandr.html

Check Audio's Output Interface

- **Output Devices:**

Output Device	Description
rockchiprk809	Audio Jack
rkhdmidpsound	HDMI/DP Audio

Check Internet connection

- **Ethernet**
  1. Connect an Ethernet cable to the board.
  2. Use the following command to check detailed connection information.

     ```
     Ifconfig eth0
     ```

- **Wi-Fi**

  Select a Wi-Fi network by running the following command in the device shell:

  ```
  nmtui
  ```

  Then select `Activate a connection` and select a network from the list under `Wi-Fi (wlan0)`.

  Alternatively, use the following command to connect to a known network name:

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

  Verify your connection with this command:

  ```
  nmcli connection show
  ```

  You should see your selected network listed in the output. For example:

  ```
  NAME                UUID                                  TYPE             DEVICE
  MyNetworkName       61f5d6b2-5f52-4256-83ae-7f148546575a  802-11-wireless  wlan0
  ```

GPIO

Following table shows the header pinout, including the sysfs paths for each port, which is often the name required when using the periphery library. You can also see the header pinout from the command line by typing pinout.

Note:

I. No. 32, 33, 37 I/O pins are +3.0V level, it has 61K ohm internal pull-down resistor, 3mA drive current capacity.

II. In addition to no. 32, 33, 37 pins, all the others are +3.3V level, 5K~10K Ohm internal pull-up resistors, 50mA drive current capacity.

sysfs path	Pin function	Pin		Pin function	sysfs path
	+3.3V Power	1	2	+5V Power	
/dev/i2c-6 /sys/class/gpio/gpio73	I2C 6 (SDA) GPIO2_B1	3	4	+5V Power	
/dev/i2c-6 /sys/class/gpio/gpio74	I2C 6 (SCL) GPIO2_B2	5	6	Ground	
/sys/class/gpio/gpio89	TEST(CLKOUT1) GPIO2_D1	7	8	UART 0 (TX) GPIO2_C1	/dev/ttyS0 /sys/class/gpio/gpio81
	Ground	9	10	UART 0 (RX) GPIO2_C0	/dev/ttyS0 /sys/class/gpio/gpio80
/dev/ttyS0 /sys/class/gpio/gpio83	UART 0 (RTSN) GPIO2_C3	11	12	I2S0(SCLK) GPIO3_D0	/sys/class/gpio/gpio120
/dev/spidev5 /sys/class/gpio/gpio85	SPI 5 (TXD) GPIO2_C5	13	14	Ground	
/dev/spidev5 /sys/class/gpio/gpio84	SPI 5 (RXD) GPIO2_C4	15	16	SPI 5 (CLK) GPIO2_C6	/dev/spidev5 /sys/class/gpio/gpio86
	+3.3V Power	17	18	SPI 5 (CSN0) 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	Ground	
/dev/spidev32766 /dev/ttyS4 /sys/class/gpio/gpio39	SPI 1 (RXD) UART 4 (RX) GPIO1_A7	21	22	I2S0(SDI1SDO3) GPIO3_D4	/sys/class/gpio/gpio124
/dev/spidev1 /sys/class/gpio/gpio41	SPI 1 (SLK) GPIO1_B1	23	24	SPI 1 (CSN0) GPIO1_B2	/dev/spidev1.0 /sys/class/gpio/gpio42
	Ground	25	26	PWM3A GPIO0_A6	/sys/class/pwm/pwmchip3/pwm0 /sys/class/gpio/gpio6
/dev/i2c-7 /sys/class/gpio/gpio71	I2C7 (SDA) GPIO2_A7	27	28	I2C7(SCL) GPIO2_B0	/dev/i2c-7 /sys/class/gpio/gpio72
/sys/class/gpio/gpio126	IS2S0(SDI3SDO1)	29	30	Ground	

sysfs path	Pin function	Pin		Pin function	sysfs path
	GPIO3_D6				
/sys/class/gpio/gpio125	IS2S0(SDI2SDO2) GPIO3_D5	**31**	**32**	PWM0 GPIO4_C2	/sys/class/pwm/pwmchip0/pwm0 /sys/class/gpio/gpio146
/sys/class/pwm/pwmchip1/pwm0 /sys/class/gpio/gpio150	PWM 1 GPIO4_C6	**33**	**34**	Ground	
/sys/class/gpio/gpio121	I2S0(LRCK) GPIO3_D1	**35**	**36**	UART0 (CTSN) GPIO2_C2	/dev/ttyS0 /sys/class/gpio/gpio82
/sys/class/gpio/gpio149	SPDIF (TX) GPIO4_C5	**37**	**38**	I2S0(SDI0) GPIO3_D3	/sys/class/gpio/gpio123
	Ground	**39**	**40**	I2S0(SDO0) GPIO3_D7	/sys/class/gpio/gpio127

Warning: Use caution when handling the GPIO pins to avoid electrostatic discharge or contact with conductive materials (metals). Failure to properly handle the Tinker Edge R can result in a short circuit, electric shock, serious injury, death, fire, or damage to your board and other property.

Using the Periphery library

To access the header pins on the Tinker Edge T, you can use standard Linux sysfs interfaces. But if you'd like a Python API, we recommend you use the python-periphery library, which is built atop the sysfs interfaces.

You can install the library on your Dev Board as follows:

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

sudo pip3 install python-periphery
```

Note:

- To access peripheral hardware resources on the Dev Board, you need to run your code with sudo privileges.
- Python 3 version of Periphery is required.

The Periphery library allows you to select a GPIO or PWM pin with a pin number. Other interfaces, such as I2C and UART pins must be specified using the pin's device path. See the following examples.

- You can edit /boot/config.txt to switch 40-pin functions, current switchable functions are listed below:

```
##### Hardware Interface Config #####

## Note: uart4 and spi1 are the same pins. Set the latter one while both on. ##
## Note: fiq_debugger and uart0 use the same pin. Set fiq_debugger first while
both on. ##

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:pwm0=off
#intf:pwm1=off
#intf:pwm3a=off
```

Caution:

Note: uart4 and spi1 are the same pins. Set the latter one while both on.

Note: fiq_debugger and uart0 use the same pin. Set fiq_debugger first while both on.

GPIO

The following code shows how to instantiate each of the GPIO pins with Periphery:

Notes: All 40-pin can be GPIO usage if /boot/config.txt is not preset

```
gpio3 = GPIO(3, "in")
gpio5 = GPIO(5, "in")
gpio7 = GPIO(7, "in")
gpio120 = GPIO(120, "in")
gpio124 = GPIO(124, "in")
gpio126 = GPIO(126, "in")
```

For more examples, see the https://python-periphery.readthedocs.io/en/latest/gpio.html.

PWM

Edit /boot/config.txt to enable pwm function.

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

The following code shows how to instantiate each of the PWM pins with Periphery:

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

For usage examples, see the https://python-periphery.readthedocs.io/en/latest/pwm.html.

I2C

Edit /boot/config.txt to enable i2c function.

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

The following code shows how to instantiate each of the I2C ports with Periphery:

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

For usage examples, see the https://python-periphery.readthedocs.io/en/latest/i2c.html.

SPI

Edit config.txt to enable SPI function:

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

The following code shows how to instantiate each of the SPI ports with Periphery:

```
# SPI1, SS0, Mode 0, 10MHz
spi1_0 = SPI("/dev/spidev1.0", 0, 10000000)
# SPI5, SS0, Mode 0, 10MHz
spi1_1 = SPI("/dev/spidev5.0", 0, 10000000)
```

For usage examples, see the https://python-periphery.readthedocs.io/en/latest/spi.html.

UART

Note: fiq_debugger and uart0 use the same pin. Set fiq_debugger first while both on

Edit /boot/config.txt to enable UART function:

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

The following code shows how to instantiate each of the UART ports with Periphery:

```
# UART0, 115200 baud
uart1 = Serial("/dev/ttyS0", 115200)
# UART4, 9600 baud
uart3 = Serial("/dev/ttyS4", 9600)
```

For usage examples, see the https://python-periphery.readthedocs.io/en/latest/serial.html.

Sample Code

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()
```

Example (Run)

```
sudo python3 blink.py
```

How to check current hardware information

Current CPU frequency

To read current real-time CPU frequency:

Dual-core Cortex-A72(up to 1.8GHz)

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

Quad-core Cortex-A53(up to 1.4GHz)

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

Current GPU frequency

To read current GPU frequency

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

Current CPU & GPU Temperature

To monitor real-time SoC temperature

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

Advanced Script

Save below text as 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 / 10
00000}')

echo "SoC Temp=> $soc_temp degree C"

echo "CPU Freq=> $cpu_freq GHz"

echo "GPU Freq=> $gpu_freq MHz"
```

Example:

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