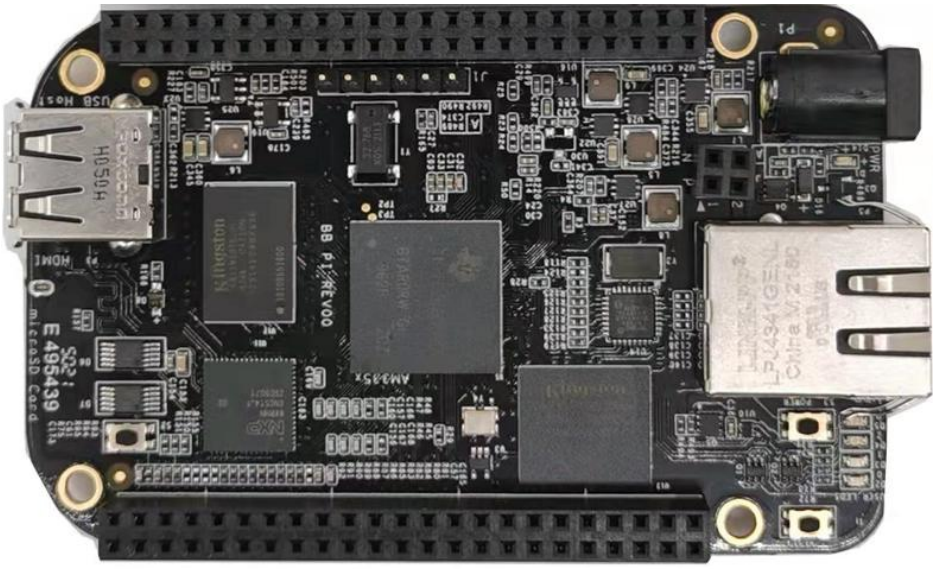
BB Pi



Product Migration Instructions

Version 0.0 – Mar.15, 2022

Revision History:

Version	Date	Editor	Description
0.0	2022-03-15	Jake	Initial Version

Purpose:

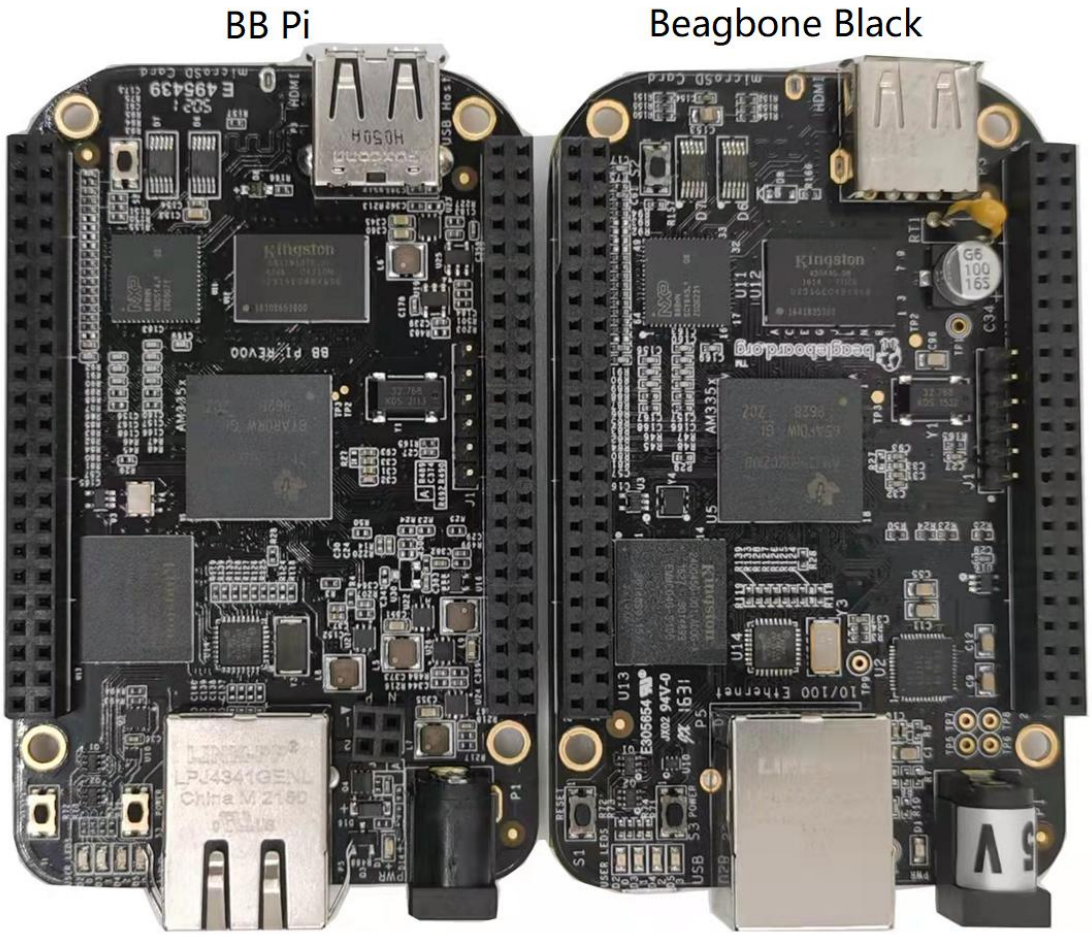
This document is compiled to clarify the differences between the boards Beagbone Black and BB Pi.

Table of Contents

1. Purpose

The BB Pi is design to replace as much as possible Beagbone Black. It has the most same placement as like Beagbone Black, so the user can migrate their projects to the BB Pi directly without any changes for their shells or accessories.

The appearance comparison is as follows:



2. Changed elements

This chapter describe the compements that changed between Beagblone Black and BB Pi.

2.1 Modify position 1: PMIC

Cause: The PMIC cannot be purchased without delivery date

	Manufacturer	Part Number	Temperature Range
Beagbone Black	TI	TPS65217C	Industrial grade
BB Pi	SGMICRO Richtek	2* SGM2019 5*RT8010	Industrial grade

Note:

a) There is no battery management chip on board so BB Pi can not support battery function and the test points TP5, TP6, TP7, TP8 are removed from BB Pi

b) Since the DCDC regulator is fixed output so that BB Pi cannot be adjusted power consumption dynamically, and it is design to get the best performance running.

c) The PWR_BUT only can be used as a NMI Key.

d) Firmware should be updated to support these changes or some features will fail.

2.2 Modify position 2: The ALPS/ SCHA5B0200 is EOL

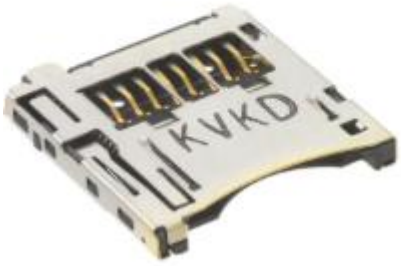
Cause: The PMIC cannot be purchased without delivery date

	Manufacturer	Part Number	Temperature Range
Beagbone Black	ALPS	SCHA5B0200	
BB Pi	Molex	5027740891	

Note:

The appearance of the two products is as follows

BB Pi



Beagbone Black

2.3 Modify position 3: RJ45 Connector

Cause: The PMIC cannot be purchased without delivery date

	Manufacturer	Part Number	Temperature Range
Beagbone Black	Link PP	LPJ0011BBNL	
BB Pi	Link PP	LPJ4341GENL	

3. Features comparison

The features comparison is as follows:

Feature	BB Black	BB PI
MPU	AM335x	AM335x
SDRAM Memory	512M DDR3	512M DDR3
Onboard Flash	4GB EMMC	4GB EMMC
PMIC	TPS65217C	×
Debug UART0	✓	✓
Debug Optional Onboard 20Pin CTI JTAG	✓	✓
Power Source	✓	✓
Indicators	✓	✓
HS USB2.0 Client Port	✓	✓
HS USB2.0 Host Port	✓	✓
Ethernet 10/100M RJ45	✓	✓, with POE
Micro SD	✓	✓
Reset Button	✓	✓
Boot Button	✓	✓
Power Button	✓, long press will cause hardware reset	Only intterput input, long press can not cause hardware reset
Video Out MicroHDMI	✓	✓
GPIO on P8	✓	✓
5V IN through P9	✓	✓
5V OUT through P9	✓	✓
3.3V OUT through P9	✓,500mA shared	✓, 1A independent
PWR_BUT through P9	✓, long press will cause hardware reset	Only intterput input, long press can not cause hardware reset
Reset through P9	✓	✓
UART through P9	✓	✓
I2C through P9	✓	✓
SPI through P9	✓	✓
PWM through P9	✓	✓
ADC through P9	✓	✓

4. Firmware migration

4.1 Create Debian Image

- Go to website **https://beagleboard.org/latest-images** to download Debian image. For example, the latest image is "**AM3358 Debian 10.3 2020-04-06 4GB SD IoT**" nowadays.
- Write the downloaded image **bone-debian-10.3-iot-armhf-2020-04-06-4gb.img** into MicroSD card with **Win32diskimager** tool.
- Mount the MicroSD card under Ubuntu system:
  - root@Ubuntu:~# **mount /dev/sdX1 /mnt**
  - root@Ubuntu:~# **cp -f <YOUR_PATH>/AM335X-BONEBLACK-D.dtbo /mnt/lib/firmware/**
  - root@Ubuntu:~# **vi /mnt/boot/uEnv.txt**

```
#Docs: http://elinux.org/Beagleboard:U-boot_partitioning_layout_2.0

uname_r=4.19.94-ti-r42
#uuid=
#dtb=

###U-Boot Overlays###
###Documentation: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian#U-Boot_Overlays
###Master Enable
enable_uboot_overlays=1
###
###Overide capes with eeprom
#uboot_overlay_addr0=/lib/firmware/<file0>.dtbo
#uboot_overlay_addr1=/lib/firmware/<file1>.dtbo
#uboot_overlay_addr2=/lib/firmware/<file2>.dtbo
#uboot_overlay_addr3=/lib/firmware/<file3>.dtbo
###
###Additional custom capes
#uboot_overlay_addr4=/lib/firmware/<file4>.dtbo
#uboot_overlay_addr5=/lib/firmware/<file5>.dtbo
```

```
#uboot_overlay_addr6=/lib/firmware/<file6>.dtbo
#uboot_overlay_addr7=/lib/firmware/<file7>.dtbo
###
###Custom Cape
dtb_overlay=/lib/firmware/AM335X-BONEBLACK-D.dtbo
###
###Disable auto loading of virtual capes (emmc/video/wireless/adc)
#disable_uboot_overlay_emmc=1
#disable_uboot_overlay_video=1
#disable_uboot_overlay_audio=1
#disable_uboot_overlay_wireless=1
#disable_uboot_overlay_adc=1
###
###PRUSS OPTIONS
###pru_rproc (4.14.x-ti kernel)
#uboot_overlay_pru=/lib/firmware/AM335X-PRU-RPROC-4-14-TI-00A0.dtbo
###pru_rproc (4.19.x-ti kernel)
uboot_overlay_pru=/lib/firmware/AM335X-PRU-RPROC-4-19-TI-00A0.dtbo
###pru_uio (4.14.x-ti, 4.19.x-ti & mainline/bone kernel)
#uboot_overlay_pru=/lib/firmware/AM335X-PRU-UIO-00A0.dtbo
###
###Cape Universal Enable
enable_uboot_cape_universal=1
###
###Debug: disable uboot autoload of Cape
#disable_uboot_overlay_addr0=1
#disable_uboot_overlay_addr1=1
#disable_uboot_overlay_addr2=1
#disable_uboot_overlay_addr3=1
###
###U-Boot fdt tweaks... (60000 = 384KB)
#uboot_fdt_buffer=0x60000
###U-Boot Overlays###

cmdline=coherent_pool=1M net.ifnames=0 lpj=1990656 rng_core.default_quality=100 qu
iet rw

#In the event of edid real failures, uncomment this next line:
#cmdline=coherent_pool=1M net.ifnames=0 lpj=1990656 rng_core.default_quality=100 q
uiet video=HDMI-A-1:1024x768@60e

##enable Generic eMMC Flasher:
```

```
##make sure, these tools are installed: dosfstools rsync
#cmdline=init=/opt/scripts/tools/eMMC/init-eMMC-flasher-v3.sh
```

Note:

① Uncomment **dtb_overlay** line and make it point to **AM335X-BONEBLACK-D.dtbo**.

② Modify **cmdline** line and append **'rw'** to the end. Otherwise, the SD card will be readonly under system after ARM board booting up.

Now, you can install the bootable SD card into the ARM board, power on and check:

```
U-Boot SPL 2019.04-00019-g1f71ed3256-dirty (Feb 20 2022 - 23:04:22 +0800)
Trying to boot from MMC1
Loading Environment from EXT4... ** File not found /boot/uboot.env **
** Unable to read "/boot/uboot.env" from mmc0:1 **
U-Boot 2019.04-00019-g1f71ed3256-dirty (Feb 20 2022 - 23:04:22 +0800)
CPU  : AM335X-GP rev 2.1
I2C:   ready
DRAM:  512 MiB
No match for driver 'omap_hsmmc'
No match for driver 'omap_hsmmc'
Some drivers were not found
Reset Source: Global external warm reset has occurred.
Reset Source: Power-on reset has occurred.
RTC 32KCLK Source: External.
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
Loading Environment from EXT4... ** File not found /boot/uboot.env **

** Unable to read "/boot/uboot.env" from mmc0:1 **
Board: BeagleBone Black
<ethaddr> not set. Validating first E-fuse MAC
BeagleBone Black:
BeagleBone Cape EEPROM: no EEPROM at address: 0x54
BeagleBone Cape EEPROM: no EEPROM at address: 0x55
BeagleBone Cape EEPROM: no EEPROM at address: 0x56
BeagleBone Cape EEPROM: no EEPROM at address: 0x57
Net:   eth0: MII MODE
cpsw, usb_ether
Press SPACE to abort autoboot in 0 seconds
board_name=[A335BNLT] ...
board_rev=[00C0] ...
switch to partitions #0, OK
mmc0 is current device
SD/MMC found on device 0
```

```
switch to partitions #0, OK
mmc0 is current device
Scanning mmc 0:1...
gpio: pin 56 (gpio 56) value is 0
gpio: pin 55 (gpio 55) value is 0
gpio: pin 54 (gpio 54) value is 0
gpio: pin 53 (gpio 53) value is 1
switch to partitions #0, OK
mmc0 is current device
gpio: pin 54 (gpio 54) value is 1
Checking for: /uEnv.txt ...
Checking for: /boot.scr ...
Checking for: /boot/boot.scr ...
Checking for: /boot/uEnv.txt ...
gpio: pin 55 (gpio 55) value is 1
2075 bytes read in 29 ms (69.3 KiB/s)
Loaded environment from /boot/uEnv.txt
Checking if uname_r is set in /boot/uEnv.txt...
gpio: pin 56 (gpio 56) value is 1
Running uname_boot ...
loading /boot/vmlinuz-4.19.94-ti-r42 ...
10095592 bytes read in 657 ms (14.7 MiB/s)
debug: [enable_uboot_overlays=1] ...
debug: [enable_uboot_cape_universal=1] ...
debug: [uboot_base_dtb_univ=am335x-boneblack-uboot-univ.dtb] ...
uboot_overlays: [uboot_base_dtb=am335x-boneblack-uboot-univ.dtb] ...
uboot_overlays: Switching too: dtb=am335x-boneblack-uboot-univ.dtb ...
loading /boot/dtbs/4.19.94-ti-r42/am335x-boneblack-uboot-univ.dtb ...
162266 bytes read in 53 ms (2.9 MiB/s)
uboot_overlays: [fdt_buffer=0x60000] ...
uboot_overlays: loading /lib/firmware/BB-ADC-00A0.dtbo ...
867 bytes read in 143 ms (5.9 KiB/s)
uboot_overlays: loading /lib/firmware/BB-BONE-eMMC1-01-00A0.dtbo ...
1584 bytes read in 288 ms (4.9 KiB/s)
uboot_overlays: loading /lib/firmware/BB-HDMI-TDA998x-00A0.dtbo ...
4915 bytes read in 222 ms (21.5 KiB/s)
uboot_overlays: loading /lib/firmware/AM335X-PRU-RPROC-4-19-TI-00A0.dtbo ...
3801 bytes read in 391 ms (8.8 KiB/s)
uboot_overlays: [dtb_overlay=/lib/firmware/AM335X-BONEBLACK-D.dtbo] ...
uboot_overlays: loading /lib/firmware/AM335X-BONEBLACK-D.dtbo ...
1311 bytes read in 567 ms (2 KiB/s)
loading /boot/initrd.img-4.19.94-ti-r42 ...
```

```
6589689 bytes read in 439 ms (14.3 MiB/s)
debug: [console=ttyS0,115200n8 bone_capemgr.uboot_capemgr_enabled=1 root=/dev/
mmcblk0p1 ro rootfstype=ext4 rootwait coherent_pool=1M net.ifnames=0 lpj=1990656 r
ng_core.default_quality=100 quiet rw] ...
debug: [bootz 0x82000000 0x88080000:648cf9 88000000] ...
## Flattened Device Tree blob at 88000000
   Booting using the fdt blob at 0x88000000
   Loading Ramdisk to 8f9b7000, end 8ffffcf9 ... OK
   Loading Device Tree to 8f92b000, end 8f9b6fff ... OK

Debian GNU/Linux 10 beaglebone ttyS0
BeagleBoard.org Debian Buster IoT Image 2020-04-06
Support: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian
default username:password is [debian:temppwd]

beaglebone login:
```

4.2 Install Cross-Compiler

➢ Download the cross-compiler:

Host: Ubuntu18/20 - 64bit:

- root@Ubuntu:~# **wget -c https://releases.linaro.org/components/toolchain/binaries/4.9-2017.01/arm-linux-gnueabihf/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf.tar.xz**

Host: Ubuntu18/20 - 32bit:

- root@Ubuntu:~# **wget -c https://releases.linaro.org/components/toolchain/binaries/4.9-2017.01/arm-linux-gnueabihf/gcc-linaro-4.9.4-2017.01-i686_arm-linux-gnueabihf.tar.xz**

➢ Install

- root@Ubuntu:~# **tar -xvf gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf.tar.xz**

➢ Check

- root@Ubuntu:~# **./gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gcc -v**

```
Using built-in specs.
COLLECT_GCC=./arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin/../l
ibexec/gcc/arm-linux-gnueabihf/4.9.4/lto-wrapper
Target: arm-linux-gnueabihf
... ...
Thread model: posix
gcc version 4.9.4 (Linaro GCC 4.9-2017.01)
```

4.3 Build DTB Source Code

Actually, we only need to build AM335X-BONEBLACK-D.dtb. But, it is neccessary to build with complete kernel source code.

- Download kernel source code
  - root@Ubuntu:~# **git clone -b 4.19 https://github.com/beagleboard/linux**
  - root@Ubuntu:~# **cd linux**
  - root@Ubuntu:~/linux # **cp -f <YOUR_PATH>/AM335X-BONEBLACK-D.dts arch/arm /boot/dts/overlays/**
  - root@Ubuntu:~/linux # **vi arch/arm/boot/dts/overlays/Makefile**

```
dtbo-$(CONFIG_ARCH_OMAP2PLUS) += \
......
    BW-ICE40Cape-00A0_LKM.dtbo  \
    M-BB-BBG-00A0.dtbo  \
    M-BB-BBGG-00A0.dtbo \
    AM335X-BONEBLACK-D.dtbo

targets += dtbs dtbs_install
targets += $(dtbo-y)
```

- Start to build
  - root@Ubuntu:~/linux # **export PATH=/root/gcc-linaro-4.9.4-2017.01-x86_64_arm-linux-gnueabihf/bin:$PATH**
  - root@Ubuntu:~/linux # **export ARCH=arm**
  - root@Ubuntu:~/linux # **export CROSS_COMPILE=arm-linux-gnueabihf-**
  - root@Ubuntu:~/linux # **make bb.org_defconfig**

- root@Ubuntu:~/linux # **make dtbs**

```
.....
  DTCO      arch/arm/boot/dts/overlays/M-BB-BBG-00A0.dtbo
  DTCO      arch/arm/boot/dts/overlays/M-BB-BBGG-00A0.dtbo
  DTCO      arch/arm/boot/dts/overlays/AM335X-BONEBLACK-D.dtbo
```

4.4 Support NMI Key

If you want to use the BOOT button under system, please follow the steps below to support it.

➢ Connect the ARM board to internet

- root@beaglebone:~# **apt-get update**
- root@beaglebone:~# **apt-cache search linux-headers-$(uname -r)**

```
linux-headers-4.19.94-ti-r42 - Linux kernel headers for 4.19.94-ti-r42 on armhf
```

- root@beaglebone:~# **apt-get install -y linux-headers-4.19.94-ti-r42**

- root@beaglebone:~# **tar -xvf <YOUR_PATH>/nmi-key.tar.xz**
- root@beaglebone:~# **cd nmi-key**
- root@beaglebone:~/nmi-key # **make KERNELDIR=/lib/modules/4.19.94-ti-r42/build**

```
make -C /lib/modules/4.19.94-ti-r42/build M=/root/nmi-key modules ARCH=arm
make[1]: Entering directory '/usr/src/linux-headers-4.19.94-ti-r42'
  CC [M]  /root/nmi-key/nmi-key.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /root/nmi-key/nmi-key.mod.o
  LD [M]  /root/nmi-key/nmi-key.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.19.94-ti-r42'
```

Now we get the target ko: nmi-key.ko

➢ Load and Test:

- root@beaglebone:~/nmi-key # **apt-get install -y evtest**

- root@beaglebone:~/nmi-key # **insmod nmi-key.ko**
- root@beaglebone:~/nmi-key # **dmesg | grep input**

```
input: nmi_key as /devices/platform/nmi_key/input/input0
```

- root@beaglebone:~/nmi-key # **evtest /dev/input/event0**

```
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "nmi_key"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 102 (KEY_HOME)
Properties:
Testing ... (interrupt to exit)
Event: time 1645378696.427025, type 1 (EV_KEY), code 102 (KEY_HOME), value 1
Event: time 1645378696.427025, ---------- SYN_REPORT ------------
Event: time 1645378696.804741, type 1 (EV_KEY), code 102 (KEY_HOME), value 0
Event: time 1645378696.804741, ---------- SYN_REPORT ------------
```

The key value: 102 [KEY_HOME] is defined in **AM335X-BONEBLACK-D.dts**:

```
    nmi_key {
        ... ...
        status = "okay";

        nmi {
            label = "nmi";
            linux,code = <KEY_HOME>;
        };
    };
```

Technical Support and Warranty

Technical Support

EMTOP TECHNOLOGY provides its product with one-year free technical support including:

- Providing software and hardware resources related to the embedded products of EMTOP TECHNOLOGY;
- Helping customers properly compile and run the source code provided by EMTOP TECHNOLOGY;
- Providing technical support service if the embedded hardware products do not function properly under the circumstance that customers operate according to the instructions in the documents provided by EMTOP TECHNOLOGY;
- Helping customers troubleshoot the products.

The following conditions will not be covered by our technical support service. We will take appropriate measures accordingly:

- Customers encounter issues related to software or hardware during their development process;
- Customers encounter issues caused by any unauthorized alter to the embedded operating system;
- Customers encounter issues related to their own applications;
- Customers encounter issues caused by any unauthorized alter to the source code provided by EMTOP TECHNOLOGY;

Warranty Conditions

1) 12-month free warranty on the PCB under normal conditions of use since the sales of the product;

2) The following conditions are not covered by free services; EMTOP TECHNOLOGY will charge accordingly:

A. Customers fail to provide valid purchase vouchers or the product identification tag is damaged, unreadable, altered or inconsistent with the products.

B. Products are damaged caused by operations inconsistent with the user manual;

C. Products are damaged in appearance or function caused by natural disasters (flood, fire, earthquake, lightning strike or typhoon) or natural aging of components or other force majeure;

D. Products are damaged in appearance or function caused by power failure, external forces, water, animals or foreign materials;

E. Products malfunction caused by disassembly or alter of components by customers or, products disassembled or repaired by persons or organizations unauthorized by EMTOP TECHNOLOGY, or altered in factory specifications, or configured or expanded with the components that are not provided or recognized by EMTOP TECHNOLOGY and the resulted damage in appearance or function;

F. Product failures caused by the software or system installed by customers or inappropriate settings of software or computer viruses;

G. Products purchased from unauthorized sales;

H. Warranty (including verbal and written) that is not made by EMTOP TECHNOLOGY and not included in the scope of our warranty should be fulfilled by the party who committed. EMTOP TECHNOLOGY has no any responsibility;

3) Within the period of warranty, the freight for sending products from customers to EMTOP TECHNOLOGY should be paid by customers; the freight from Embest to customers should be paid by us. The freight in any direction occurs after warranty period should be paid by customers.

4) Please contact technical support if there is any repair request.

Note:

📖 EMTOP TECHNOLOGY will not take any responsibility on the products sent back without the permission of the company.

Contact Information

Pre-sales: sales@emtop-tech.com

After-sales: support@emtop-tech.com

Website: www.emtop-tech.com

WiKi: www.emtop-tech.com/wiki