MuntsOS Embedded Linux

Application Note #19: BeaglePlay Target Platform Notes

Revision 1
13 March 2025

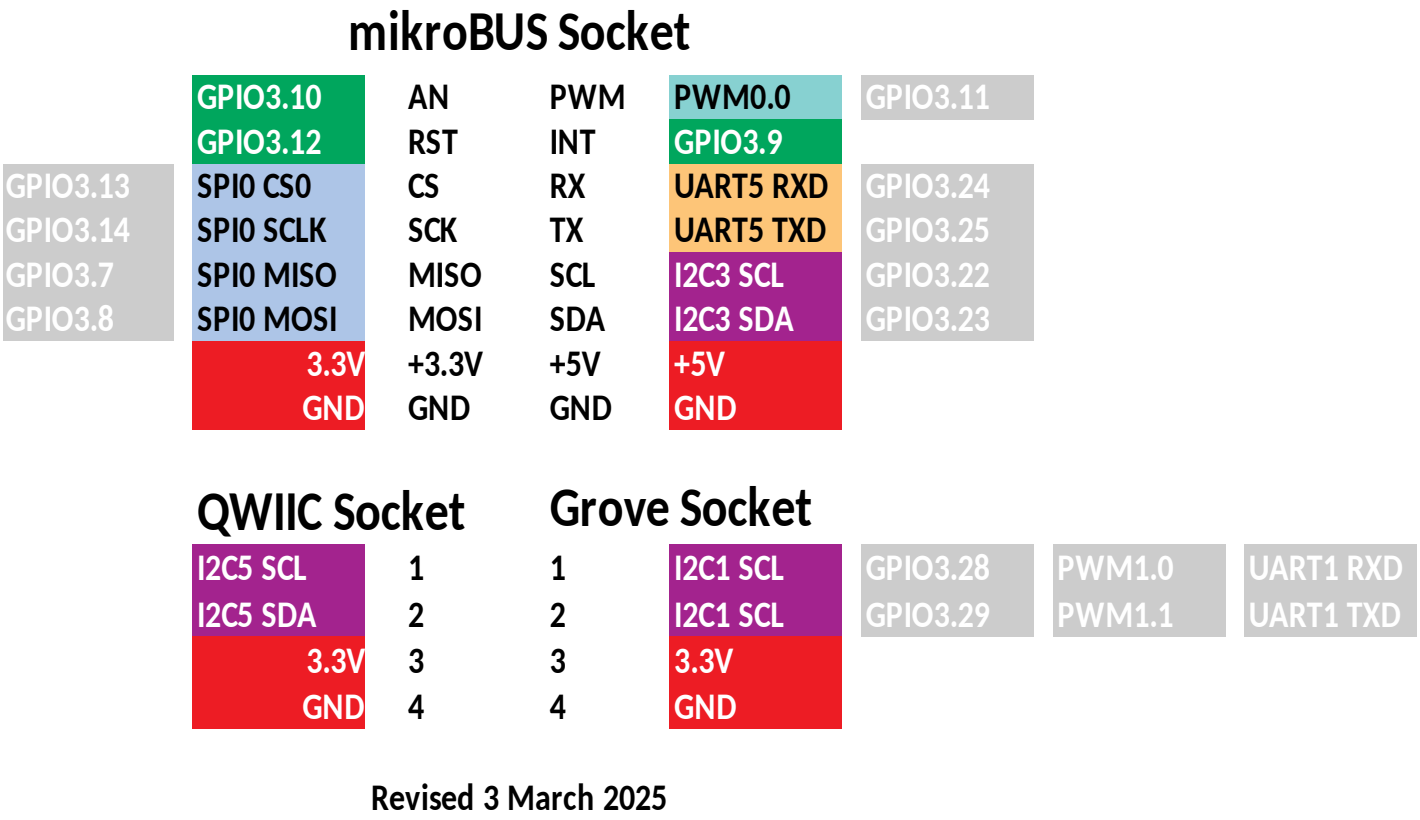
by Philip Munts
dba Munts Technologies
http://tech.munts.com

Introduction

The BeaglePlay is a small Linux microcomputer board with industry standard interfaces for add-on I/O modules (a mikroBUS socket, a QWIIC socket, and a Grove socket) instead of a general purpose expansion header. It has a Texas Instruments AM6254 ARMv8 Cortex-A53 quad core CPU and comes with 2 GB of RAM. The BeaglePlay has one USB-A receptacle for peripheral devices and one USB-C receptacle for power and tethering. It has 10/100/1000BASE-T and 10BASE-T1L wired Ethernet interfaces, a WL1807MOD wireless Ethernet interface, and a CC1352P7 wireless microcontroller capable of supporting a wide variety of radio networks.

Standard Hardware Configuration

After installing a **MuntsOS Embedded Linux** Thin Server, the BeaglePlay will have all of the mikroBUS hardware subsystems (I^2C bus master, PWM output, SPI bus master, and UART) active and the Grove and QWIIC sockets configured as I^2C bus masters.



Revised 3 March 2025

`GPIOX.Y` denotes `/dev/gpiochipX` line `Y`.

`PWMX.Y` denotes `/sys/class/pwm/pwmchipX` output `Y`.

Alternate GPIO pin functions (shown as gray in the above diagram) can be configured with device tree overlays described in the following section.

Boot Loader

MuntsOS Embedded Linux for the BeaglePlay includes a customized U-Boot bootloader. The boot loader deliverables consist of:

- Three binary code files: **`tiboot3.bin`**, **`tispl.bin`**, and **`u-boot.img`**.
- Three compiled U-Boot boot scripts: **`boota.scr`**, **`boot0.scr`**, and **`boot1.scr`**.
- Some license and **`README`** text files that are included for information only.

A *greatly* simplified description of the boot process flow is:

1. The AM6254 microprocessor ROM boot loader loads **`tiboot3.bin`** from eMMC or microSD.
2. **`tiboot3.bin`** loads **`tispl.bin`**.
3. **`tispl.bin`** loads **`u-boot.img`**.
4. **`u-boot.img`** loads **`boot.scr`**.
5. **`boot.scr`** imports some U-Boot environment variables from the text file **`config.txt`**.
6. **`boot.scr`** loads the device tree blob **`BeaglePlay.dtb`**.
7. **`boot.scr`** loads and applies device tree overlays specified by the **`OVERLAYS`** environment variable imported from **`config.txt`**. The value the **`OVERLAYS`** environment variable is a possibly empty space separated list of device tree overlay names without the **`.dtbo`** file suffix *e.g.* **`OVERLAYS=`** or **`OVERLAYS=disable-spe`** or **`OVERLAYS=grove-gpio mikrobus-gpio`**.
8. **`boot.scr`** loads the Linux kernel image **`BeaglePlay.img`**.
9. **`boot.scr`** boots the Linux kernel, passing it the device tree and information from the **`BOARDNAME`** and **`CMDLINE`** environment variables imported from **`config.txt`**.

Boot Scripts

boota.scr

This boot script loads **`config.txt`**, **`BeaglePlay.dtb`**, and **`BeaglePlay.img`** from DOS partition **`/dev/mmcblk1p1`** (microSD) if the user button is pressed and from DOS partition **`/dev/mmcblk0p1`** (eMMC) otherwise.

boot0.scr

This boot script loads **`config.txt`**, **`BeaglePlay.dtb`**, and **`BeaglePlay.img`** from DOS partition **`/dev/mmcblk0p1`** (eMMC).

boot1.scr

This boot script loads **`config.txt`**, **`BeaglePlay.dtb`**, and **`BeaglePlay.img`** from DOS partition **`/dev/mmcblk1p1`** (microSD).

Notes on microSD Cards

The AM6254 microprocessor ROM boot loader seems to be *extremely* picky about loading **`tiboot3.bin`** from a microSD card. The microSD card *must* be partitioned with a type **`0xC`** (Win95 FAT32 LBA) partition starting at sector 2048 with the boot flag set. Furthermore, U-Boot requires the partition to be formatted with no more than 32 sectors per cluster.

According to my extensive testing, with the user button pressed and held until MuntsOS has started, the ROM boot loader and U-Boot will boot from some (but not all!) 16 and 32 GB cards in my inventory and some (but not all!) 64 GB cards if you create a 32 GB or smaller partition or force the cluster size to 32 sectors per cluster. They will boot from no card smaller than 16 GB nor any card larger than 64 GB no matter what partition size or cluster size.

The only way around how picky the ROM boot loader and U-Boot are is to load U-Boot from eMMC and Linux from a microSD card (the third boot loader installation flavor below). Both old 1 GB and new 400 GB cards work in this mode.

Boot Loader Installation

The following three flavors of boot loader installation make sense.

All of MuntsOS Installed to microSD

1. Unpack a **MuntsOS** Thin Server *e.g.* muntsos-BeaglePlay.zip to a microSD card.
2. Insert the microSD card into the socket.
3. Press and hold the user button and then apply power to the BeaglePlay.
4. Release the user button *only* after **MuntsOS** has finished booting.

Unfortunately this requires holding the user button *every* time you want to boot into **MuntsOS**. The two following boot loader installation flavors are more convenient, but they each require this one at least one time to install.

All of MuntsOS Installed to eMMC

1. Prepare a microSD card and boot **MuntsOS** as described in the previous section.

2. Install **`tiboot3.bin`** to eMMC boot partition **`/dev/mmcblk0boot0`** with the following commands:

```
mount /boot
echo 0 > /sys/block/mmcblk0boot0/force_ro
dd if=/dev/zero of=/dev/mmcblk0boot0 bs=512 count=4096
dd if=/boot/tiboot3.bin of=/dev/mmcblk0boot0
echo 1 > /sys/block/mmcblk0boot0/force_ro
```

3. Partition the eMMC with **`fdisk -u /dev/mmcblk0`**. First remove any existing partitions then create a new data partition starting at sector 2048. Change the data partition type to **`0xC`** (Win95 FAT32 LBA). Mark the data partition as bootable. Then exit **`fdisk`** with a **`w`** to write changes to the eMMC.

4. Format the eMMC data partition with **`mkdosfs /dev/mmcblk0p1`**.

5. Copy everything from the microSD card to the eMMC with the following commands:

```
mount /dev/mmcblk0p1 /mnt
cp -p -r /boot/*     /mnt
cp -p /mnt/boot0.scr /mnt/boot.scr
umount /boot
umount /mnt
```

6. Remove the microSD card and reboot. The BeaglePlay should boot into **MuntsOS Embedded Linux**.

Save the microSD card for system recovery, in case you accidentally corrupt the files in the eMMC data partition to the point the system will no longer boot.

Boot Loader Installed to eMMC / Linux installed to microSD

1. Perform steps 1 to 4 in the previous section.

2. Install U-Boot and friends to the eMMC data partition with the following commands:

```
mount /dev/mmcblk0p1      /mnt
cp -p /boot/LICENSE.*     /mnt
cp -p /boot/README.boot   /mnt
cp -p /boot/*.scr         /mnt
cp -p /boot/bootfiles.md5 /mnt
cp -p /boot/tiboot3.bin   /mnt
cp -p /boot/tispl.bin     /mnt
cp -p /boot/u-boot.img    /mnt
cp -p /mnt/boot1.scr      /mnt/boot.scr
umount /boot
umount /mnt
```

3. Now reboot. The BeaglePlay should load U-Boot from eMMC and the Linux kernel from the microSD card.

Device Tree Overlays

MuntsOS includes the following device tree overlays for altering the BeaglePlay hardware configuration. Device tree overlays are applied by the boot loader and selected by editing `/boot/config.txt`, changing `OVERLAYS=` to *e.g.* `OVERLAYS=grove-gpio mikrobus-gpio`.

Network Interface Overlays

disable-ethernet.dtbo

Disables the 10/100/1000BASE-T wired Ethernet interface.

disable-spe.dtbo

Disables the 10BASE-T1L Single Pair Ethernet interface.

disable-wifi.dtbo

Disables the wireless Ethernet interface, by disabling the SDIO interface `mmc2`.

Grove Connector Overlays

grove-gpio.dtbo

Disables hardware subsystem `I2C1` and configures the Grove socket `D0` pin as `GPIO3.28` and `D1` pin as `GPIO3.29`.

grove-motor1.dtbo

Disables hardware subsystem `I2C1` and configures the Grove socket `D0` pin as `PWM1.0` and `D1` pin as `GPIO3.29`.

grove-motor2.dtbo

Disables hardware subsystem `I2C1` and configures the Grove socket `D0` pin as `PWM1.0` and `D1` pin as `PWM1.1`.

grove-motor3.dtbo

Disables hardware subsystem `I2C1` and configures the Grove socket `D0` pin as `GPIO3.28` and `D1` pin as `PWM1.1`.

grove-serial.dtbo

Disables hardware subsystem `I2C1` and configures the Grove socket as serial port `/dev/ttyS1`.

mikroBUS Socket Overlays

mikrobus-gpio.dtbo

Configures all mikroBUS pins as GPIO *and* disables the corresponding SPI, I²C, PWM, and UART hardware subsystems to conserve power consumption. This is the preferred overlay for supporting unusual Click Boards such as the Relay Click which drives one of the relays with **`CS`** (SPI slave select) and the other with **`PWM`**.

mikrobus-i2c-gpio.dtbo

Configures both of the mikroBUS I²C pins as GPIO. Does *not* disable the corresponding I²C hardware subsystem.

mikrobus-pwm-gpio.dtbo

Configures the mikroBUS **`PWM`** pin as GPIO. Does *not* disable the corresponding PWM hardware subsystem.

mikrobus-spi-gpio.dtbo

Configures all of the mikroBUS SPI pins as GPIO. Does *not* disable the corresponding SPI hardware subsystem.

mikrobus-uart-gpio.dtbo

Configures both of the mikroBUS serial port pins as GPIO. Does *not* disable the corresponding UART hardware subsystem.

Serial Ports

Unlike the Debian distribution for the BeaglePlay, there is a direct and sane mapping between the hardware serial ports and their device nodes (**`UART0`** is **`/dev/ttyS0`** etc.).

Hardware Subsystem	Device Node	
`UART0`	**`/dev/ttyS0`**	Console Serial Port Header **`J6`**
`UART1`	**`/dev/ttyS1`**	Grove Socket (requires **`grove-serial.dtbo`**)
`UART5`	**`/dev/ttyS5`**	mikroBUS socket
`UART6`	**`/dev/ttyS6`**	CC1352 wireless microcontroller

Watchdog Timer

The hardware watchdog timers within the AM6254 CPU on the BeaglePlay board are fundamentally broken for the purposes of **MuntsOS Embedded Linux**, because of their "Windowed Watchdog Timer" design.

For any watchdog timer, you must execute a particular I/O operation (colloquially called "kick" or "pet") to reset the watchdog timer before it expires and forcibly resets the CPU. A windowed watchdog timer imposes a further requirement: The kick operation must occur within a constrained time window. A kick too early (before the time window opens) will be ignored and a kick too late (after the time window closes) results in timer expiration and CPU reset. The windowed watchdog timers in the AM6254 CPU impose an even more draconian constraint: *An early kick causes an immediate CPU reset*.

In my opinion, this is an unacceptable constraint for a Linux embedded system. A typical embedded system running **MuntsOS** would place the watchdog kick in whatever main event loop exists in the main application program. The hardware watchdog timer period should be set to either a resonable recovery time (*e.g.* 5 seconds) or perhaps a value that is a small multiple of the maximum event loop cycle time.

The **MuntsOS** Linux kernel for the BeaglePlay configures a kernel software watchdog timer for **`/dev/watchdog`** instead of using one of the AM6254 hardware watchdog timers.

WiFi Antennas

The BeaglePlay WiFi interface works best with two **`2.4G/5G`** antennas, connected to coax sockets **`J3`** and **`J5`**. Using only a single antenna seems to greatly impare WiFi performance.