MuntsOS Embedded Linux

Application Note #19: BeaglePlay Target Platform Notes

Revision 1
6 March 2025

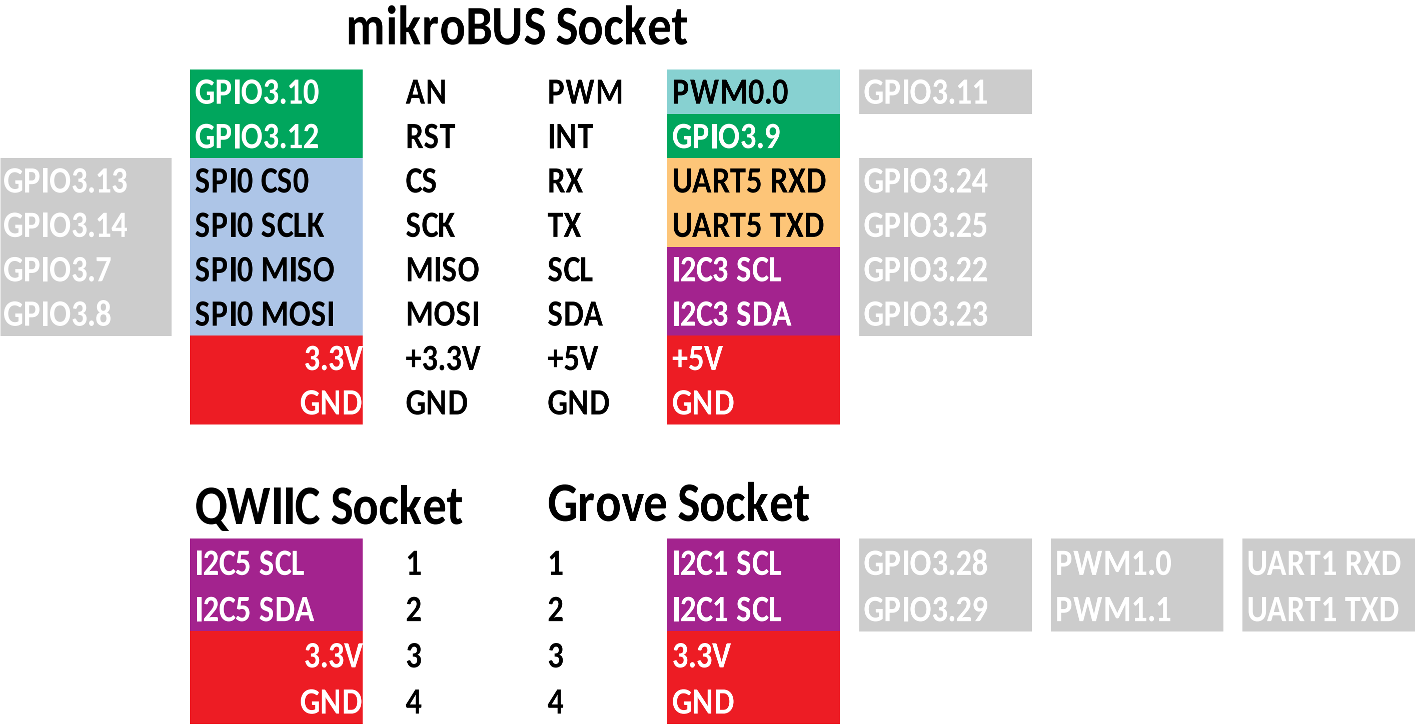
by Philip Munts
dba Munts Technologies
http://tech.munts.com

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.



mikroBUS Socket

	GPIO3.10	AN	PWM	PWM0.0	GPIO3.11
	GPIO3.12	RST	INT	GPIO3.9	
GPIO3.13	SPI0 CS0	CS	RX	UART5 RXD	GPIO3.24
GPIO3.14	SPI0 SCLK	SCK	TX	UART5 TXD	GPIO3.25
GPIO3.7	SPI0 MISO	MISO	SCL	I2C3 SCL	GPIO3.22
GPIO3.8	SPI0 MOSI	MOSI	SDA	I2C3 SDA	GPIO3.23
	3.3V	+3.3V	+5V	+5V	
	GND	GND	GND	GND	

QWIIC Socket

I2C5 SCL	1
I2C5 SDA	2
3.3V	3
GND	4

Grove Socket

1	I2C1 SCL	GPIO3.28	PWM1.0	UART1 RXD
2	I2C1 SCL	GPIO3.29	PWM1.1	UART1 TXD
3	3.3V			
4	GND			

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.

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`**.

`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-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-gpio.dtbo`

Configures all mikroBUS pins as GPIO *and* disables the corresponding SPI, I^{2}C, 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`**, normally the SPI slave select signal.

mikrobus-i2c-gpio.dtbo

Configures both of the mikroBUS I^2C pins as GPIO. Does *not* disable the corresponding I^2C 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 **RX** and **TX**
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 the time expiring 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 requirement 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`**.