BeagleBone Black - Driver Development Hardware Reference

System Info

- Kernel: 6.12.32-bone28
- Board: TI_AM335x_BeagleBone_Black
- Device Tree: am335x-boneblack.dts

Hardware Controllers Status

✅ I2C Controllers (Working)

Detected Hardware:

- 44e0b000.i2c → I2C-0 (bus 0 @ 400 kHz) - Internal/Cape EEPROM
- 4819c000.i2c → I2C-2 (bus 2 @ 100 kHz) - Exposed on headers

Available Devices:

- /dev/i2c-0 - I2C bus 0
- /dev/i2c-2 - I2C bus 2

Kernel Support:

```
CONFIG_I2C=y
CONFIG_I2C_CHARDEV=y
CONFIG_I2C_OMAP=y
```

✅ UART Controllers (Working)

Detected Hardware:

- 44e09000.serial → UART0 (Console)
- Additional UARTs detected as ttyS0-ttyS5

Available Devices:

- /dev/ttyS0 - UART0 (Debug console)
- /dev/ttyS1 - UART1
- /dev/ttyS2 - UART2
- /dev/ttyS3 - UART3
- /dev/ttyS4 - UART4
- /dev/ttyS5 - UART5

Kernel Support:

```
CONFIG_SERIAL_8250=y
CONFIG_SERIAL_8250_OMAP=y
CONFIG_SERIAL_8250_CONSOLE=y
CONFIG_SERIAL_8250_NR_UARTS=6
```

⚠️ SPI Controllers (Hardware Present, Not Configured)

Hardware Status:

- SPI controllers exist but NO /dev/spidev* devices
- No SPI platform devices detected in /sys/bus/platform/devices/
- Pins are in GPIO mode (need mux configuration)

Kernel Support (Available but not active):

```
CONFIG_SPI=y
CONFIG_SPI_MASTER=y
CONFIG_SPI_OMAP24XX=y
CONFIG_SPI_SPIDEV=m
```

Pin Mappings for Driver Development

I2C-2 Pins (Already Configured)

Physical Pin	Function	Pinmux Register	Mode
P9.19	I2C2_SCL	0x44e1097C	Mode 3
P9.20	I2C2_SDA	0x44e10978	Mode 3

UART Pin Mappings

UART	Physical Pin	Function	Pinmux Register	Mode
UART1	P9.24	TX	0x44e10984	Mode 0
UART1	P9.26	RX	0x44e10980	Mode 0
UART2	P9.21	TX	0x44e10954	Mode 1
UART2	P9.22	RX	0x44e10950	Mode 1
UART4	P9.13	TX	0x44e10974	Mode 6
UART4	P9.11	RX	0x44e10970	Mode 6
UART5	P8.37	TX	0x44e108C0	Mode 4
UART5	P8.38	RX	0x44e108C4	Mode 4

SPI0 Pin Mappings (Needs Configuration)

Physical Pin	Function	Pinmux Register	Mode	Current State
P9.17	SPI0_CS0	0x44e1095C	Mode 0	GPIO (0x37)
P9.18	SPI0_MOSI (D1)	0x44e10958	Mode 0	GPIO (0x37)
P9.21	SPI0_MISO (D0)	0x44e10954	Mode 0	GPIO (0x37)
P9.22	SPI0_SCLK	0x44e10950	Mode 0	GPIO (0x37)

SPI1 Pin Mappings (Needs Configuration)

Physical Pin	Function	Pinmux Register	Mode	Current State
P9.31	SPI1_SCLK	0x44e10990	Mode 3	GPIO (0x00)
P9.29	SPI1_MOSI (D1)	0x44e10994	Mode 3	GPIO (0x10)
P9.30	SPI1_MISO (D0)	0x44e10998	Mode 3	GPIO (0x27)
P9.28	SPI1_CS0	0x44e1099C	Mode 3	GPIO (0x02)

Pin Multiplexing (Pinmux) Details

Understanding Pinmux Values

The pin-mux registers use the format: `0x0000003X`
Bit Layout:

- Bit 0-2: MUX_MODE (0-7) - Selects pin function
- Bit 3: PULLUP_EN (0=disabled, 1=enabled)
- Bit 4: PULLTYPE (0=pull-down, 1=pull-up)
- Bit 5: RX_ACTIVE (0=output, 1=input)
- Bit 6: SLEWCTRL (slew rate control)

Common Values:

- `0x37` = Mode 7 (GPIO), Pull-up enabled, Input
- `0x30` = Mode 0 (Peripheral), Pull-up enabled, Input
- `0x20` = Mode 0 (Peripheral), Input only
- `0x00` = Mode 0 (Peripheral), Output only

Current Pin States

All GPIO pins currently configured as: `0x27` or `0x37` (GPIO mode with pull-up)

For Driver Development

What You Need to Do:

1. I2C Driver ✅ Ready

- Hardware is already configured
- Use `/dev/i2c-2` for testing
- Pins: P9.19 (SCL), P9.20 (SDA)
- No additional configuration needed

2. UART Driver ✅ Ready

- Hardware is already configured
- Use any `/dev/ttySX` for testing
- Common choice: `/dev/ttyS1` (P9.24/P9.26)
- No additional configuration needed

3. SPI Driver ⚠️ Needs Pin Configuration

Two approaches:

A. Device Tree Overlay Method (Recommended) Create a custom device tree overlay that:

- Configures pinmux registers
- Enables SPI controller
- Creates `/dev/spidev` nodes

B. Runtime Pin Configuration in Driver Your driver can configure pins using:

```
#include <linux/pinctrl/consumer.h>

struct pinctrl *pinctrl;
struct pinctrl_state *pins_default;

pinctrl = devm_pinctrl_get(dev);
pins_default = pinctrl_lookup_state(pinctrl, "default");
pinctrl_select_state(pinctrl, pins_default);
```

Hardware Base Addresses (AM335x)

I2C Controllers

- I2C0: `0x44E0B000`
- I2C1: `0x4802A000`
- I2C2: `0x4819C000`

UART Controllers

- UART0: `0x44E09000`
- UART1: `0x48022000`
- UART2: `0x48024000`
- UART3: `0x481A6000`
- UART4: `0x481A8000`
- UART5: `0x481AA000`

SPI Controllers

- SPI0: `0x48030000`
- SPI1: `0x481A0000`

Pinmux Control

- Base Address: `0x44E10800`
- Size: 568 bytes (142 pins × 4 bytes)

Testing Commands

Test I2C

```
# Scan for devices
sudo i2cdetect -y 2

# Read from device (example)
sudo i2cget -y 2 0x50 0x00
```

Test UART

```
# Loopback test (connect TX to RX)
# Terminal 1
cat /dev/ttyS1

# Terminal 2
echo "test" > /dev/ttyS1
```

Test SPI (After Configuration)

```
# Will work once /dev/spidev* exists
ls /dev/spidev*
```

Next Steps for SPI

Option 1: Create Device Tree Overlay

1. Write `.dts` file with pinmux and SPI configuration
2. Compile to `.dtbo`
3. Load via `/boot/uEnv.txt`
4. Reboot

Option 2: Configure in Your Driver

Your driver's probe function should:

1. Request pinctrl
2. Configure pins for SPI mode
3. Register SPI master/slave
4. Create character device

Option 3: Runtime Configuration

```
# If config-pin is available (need to install tools)
config-pin P9.17 spi_cs
config-pin P9.18 spi
config-pin P9.21 spi
config-pin P9.22 spi_sclk
```

Useful Kernel Documentation

- `/usr/src/linux/Documentation/devicetree/bindings/spi/`
- `/usr/src/linux/Documentation/devicetree/bindings/i2c/`
- `/usr/src/linux/Documentation/devicetree/bindings/serial/`

TI AM335x Technical Reference Manual

Chapter references for driver development:

- Chapter 19: I2C

- Chapter 19: UART
- Chapter 24: McSPI (SPI)
- Chapter 9: Pin Multiplexing