BeagleBone SBC Cheatsheet

A comprehensive cheat sheet covering the BeagleBone Single Board Computer, including setup, GPIO, and common commands. This guide provides a quick reference for developers and hobbyists using BeagleBone for embedded projects.

Getting Started with BeagleBone

Initial Setup

Connecting to BeagleBone:

Connect BeagleBone to your computer via USB. It should appear as a network drive.

Accessing via SSH:

Use an SSH client (e.g., PuTTY, Terminal) to connect to the BeagleBone. Default IP address is 192.168.7.2. The default username is `debian` and the password is `temppwd`.

Updating the System:

After logging in, update the system using:

```
sudo apt update
sudo apt upgrade
```

Installing Essential Tools:

Install essential development tools:

```
sudo apt install git python3-pip build-essential
```

Configuring Network:

Edit `/etc/network/interfaces` to set up static IP address or other network configurations.

Basic Commands

`pwd`	Print working directory.
`ls`	List directory contents.
`cd`	Change directory.
`mkdir`	Create a new directory.
`rm`	Remove a file.
`nano` or `vim`	Text editors for file editing.

Boot Configuration

U-Boot:

BeagleBone uses U-Boot as its bootloader. Configuration files are located in `/boot/uEnv.txt`.

Kernel Configuration:

Kernel parameters can be modified via `uEnv.txt`. Changes require a reboot.

Device Tree Overlays:

Device tree overlays (`.dtbo`) are used to configure hardware peripherals. Enable overlays by adding lines to `uEnv.txt`.

Example Overlay Enable:

```
##Enable I2C1
dt overlay=BB-I2C1-00A0.dtbo
```

GPIO Programming

Accessing GPIO

GPIO Pins:

GPIO pins can be accessed via the command line or through programming languages like Python.

Using `config-pin`:

`config-pin` is a utility to configure pin modes.

```
sudo config-pin p9.12 gpio
sudo config-pin p9.12 out
```

Checking Pin State:

Read the value of the GPIO pin:

```
cat /sys/class/gpio/gpio60/value
```

Finding the GPIO Number:

Each pin has a GPIO number associated with it. Use the BeagleBone Black System Reference Manual to find the number.

Python GPIO Control

Installing `py-gpio`:

Install the `py-gpio` library for Python:

```
sudo pip3 install py-gpio
```

Python Code Example:

```
import gpio

pin = gpio.GPIO(60) # Example GPIO pin
pin.export()
pin.direction = 'out'
pin.value = 1  # Set pin high
```

Cleaning Up:

Always unexport the pin when done:

```
pin.unexport()
```

Alternative Library - Adafruit BBIO:

Another popular library is `Adafruit_BBIO`. Install with `sudo pip3 install Adafruit_BBIO`.

Device Tree Overlays for GPIO

Creating a Custom Overlay:

Create a device tree source (`.dts`) file. This defines the hardware configuration.

Compiling the Overlay:

Use the device tree compiler to create the `.dtbo` file:

```
dtc -O dtb -I dts -o my_overlay.dtbo
my_overlay.dts
```

Example DTS file:

```
/dts-v1/;
/plugin/;

/{
    compatible = "ti,beaglebone-black";

    fragment@0 {
        target = <&am33xx_pinmux>;
        __overlay__ {
            pinctrl_bonnet_gpio:
pinmux_bonnet_gpio {
                pinctrl-single,pins = <0x030
(PIN_OUTPUT_PULLUP | MUX_MODE7)>; /* P9_12,
GPIO1_28 */
            };
        };
    };

    fragment@1 {
        target = <&ocp>;
        __overlay__ {
            bonnet_gpio {
                compatible = "gpio-leds";

gpios = <&gpio1 28 GPIO_ACTIVE_HIGH>;
                status = "okay";
            };
        };
    };
};
```

Networking

Configuring Network Interfaces

Listing Network Interfaces:

Use `ifconfig` or `ip addr` to list available network interfaces.

Editing `/etc/network/interfaces`:

Manually configure network interfaces by editing `/etc/network/interfaces`.

```
auto eth0
iface eth0 inet static
address 192.168.1.100
netmask 255.255.255.0
gateway 192.168.1.1
dns-nameservers 8.8.8.8 8.8.4.4
```

Using `dhcpcd`:

For dynamic IP configuration, ensure `dhcpcd` is running.

```
sudo systemctl enable dhcpcd
sudo systemctl start dhcpcd
```

Wireless Setup

Connecting to Wi-Fi:

Use `iwconfig` and `wpa_supplicant` to connect to Wi-Fi networks. First, identify wireless interface using `iwconfig`

Configuring `wpa_supplicant`:

Create/edit `/etc/wpa_supplicant/wpa_supplicant.conf`:

```
network={
    ssid="YourNetworkName"
    psk="YourWiFiPassword"
}
```

Bringing up the interface:

```
sudo ifconfig wlan0 up
sudo wpa_supplicant -i wlan0 -c
/etc/wpa_supplicant/wpa_supplicant.conf
sudo dhclient wlan0
```

Firewall Configuration

Using `iptables`:

`iptables` is used for setting up firewall rules. Example: Allowing SSH traffic:

```
sudo iptables -A INPUT -p tcp --dport 22 -j
ACCEPT
```

Saving Rules:

Save iptables rules using `iptables-save`.

Using `ufw` (Uncomplicated Firewall):

A more user-friendly firewall configuration tool:

```
sudo apt install ufw
sudo ufw enable
sudo ufw allow ssh
```

Peripherals and Hardware

I2C Communication

Enabling I2C:

Ensure I2C is enabled in `uEnv.txt` via device tree overlay. Example:

```
dt overlay=BB-I2C1-00A0.dtbo
```

Detecting I2C Devices:

Use `i2cdetect` to scan the I2C bus for connected devices:

```
sudo i2cdetect -y 1
```

Using `i2cget` and `i2cset`:

Read and write to I2C devices using `i2cget` and `i2cset` respectively.

UART Serial Communication

Accessing UART Ports:

UART ports are available as `/dev/ttyO0`, `/dev/ttyO1`, etc.

Using `minicom`:

Use `minicom` or `screen` to communicate over UART. First, install minicom:

```
sudo apt install minicom
```

Then, configure minicom:

```
sudo minicom -s
```

Example `minicom` Configuration:

Set serial device to `/dev/ttyO1`, baud rate to 115200, and disable hardware flow control.

PWM

Accessing PWM:

PWM functionality is available via device tree overlays. Enable PWM overlay in `uEnv.txt`.

PWM Control:

Control PWM parameters such as period and duty cycle via `/sys/class/pwm/...`.

Example:

```
echo 1000000 >
/sys/class/pwm/pwmchip0/pwm0/period
echo 500000 >
/sys/class/pwm/pwmchip0/pwm0/duty_cycle
echo 1 > /sys/class/pwm/pwmchip0/pwm0/enable
```