# Table of contents

1 Introduction .......................................................... 3
   1.1 Detailed overview .................................................. 3
       1.1.1 AM67A SoC .................................................. 4
   1.2 Board components location ....................................... 4
       1.2.1 Front components .......................................... 4
       1.2.2 Back components ........................................... 5

2 BeagleY-AI Quick Start .............................................. 7
   2.1 What’s included in the box? .................................... 7
   2.2 Getting started ................................................... 7
       2.2.1 Boot Media .................................................. 7
       2.2.2 Power Supply .............................................. 8
       2.2.3 Board connection ........................................... 9
       2.2.4 USB Tethering ............................................. 9
       2.2.5 Using BeagleY-AI ............................................ 10
       2.2.6 Connecting to WiFi ......................................... 13
   2.3 Demos and Tutorials .............................................. 14

3 Design and specifications ........................................... 15
   3.1 Block diagram and overview ..................................... 15
   3.2 SoC ..................................................................... 15
   3.3 Boot modes .......................................................... 15
   3.4 Power sources ...................................................... 15
   3.5 PMIC .................................................................. 15
   3.6 General connectivity and expansion ............................. 15
   3.7 Buttons and LEDs .................................................. 15
   3.8 Networking .......................................................... 15
   3.9 Ethernet ................................................................ 15
   3.10 Memory, media, and storage ..................................... 15
   3.11 Multimedia I/O .................................................... 15
   3.12 Debug ports ........................................................ 15
   3.13 Mechanical Specifications ....................................... 17
       3.13.1 Dimensions & Weight ...................................... 17

4 Expansion .................................................................. 51
   4.1 PCIe .................................................................. 51

5 Demos and tutorials .................................................... 53
   5.1 Using GPIO ........................................................... 53
       5.1.1 Pin Numbering ............................................... 53
       5.1.2 Required Hardware .......................................... 53
       5.1.3 GPIO Write .................................................... 53
       5.1.4 Blink an LED .................................................. 55
       5.1.5 GPIO Read ..................................................... 56
       5.1.6 Read a Button ............................................... 56
       5.1.7 Combining the Two ........................................... 56
       5.1.8 Understanding Internal Pull Resistors ................. 57
       5.1.9 Troubleshooting ............................................. 58
BeagleY-AI is an open-source single board computer based on the Texas Instruments AM67A Arm-based vision processor.
Chapter 1

Introduction

BeagleY-AI is an open-source single board computer designed for edge AI applications.

1.1 Detailed overview

BeagleY-AI is based on the Texas Instruments AM67A Arm-based vision processor. It features a quad-core 64-bit Arm®Cortex®-A53 CPU subsystem at 1.4GHz, Dual general-purpose C7x DSP with Matrix Multiply Accelerator (MMA) capable of 4 TOPs each, Arm Cortex-R5 subsystem for low-latency I/O and control, a 50 GFlop GPU, video and vision accelerators, and other specialized processing capability.
## Table 1.1: BeagleY-AI features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Texas Instruments AM67A, Quad 64-bit Arm® Cortex®-A53 @1.4 GHz, multiple cores including Arm/GPU processors, DSP, and vision/deep learning accelerators</td>
</tr>
<tr>
<td>RAM</td>
<td>4GB LPDDR4</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>Beagleboard BM3301, 802.11ax Wi-Fi</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Bluetooth Low Energy 5.4 (BLE)</td>
</tr>
<tr>
<td>USB Ports</td>
<td>4 x USB 3.0 TypeA ports supporting simultaneous 5Gbps operation, 1 x USB 2.0 TypeC, supports USB 2.0 device mode</td>
</tr>
<tr>
<td>Ethernet</td>
<td>Gigabit Ethernet, with PoE+ support (requires separate PoE HAT)</td>
</tr>
<tr>
<td>Camera/Display</td>
<td>2 x 4-lane MIPI camera connector (one connector muxed with DSI capability)</td>
</tr>
<tr>
<td>Display Output</td>
<td>1 x HDMI display, 1 x OLDI display, 1 x DSI MIPI Display</td>
</tr>
<tr>
<td>Real-time Clock</td>
<td>Supports external coin-cell battery for power failure time retention</td>
</tr>
<tr>
<td>Debug UART</td>
<td>1 x 3-pin debug UART</td>
</tr>
<tr>
<td>Power</td>
<td>5V/3A DC power via USB-C</td>
</tr>
<tr>
<td>PCIe Interface</td>
<td>PCI-Express® Gen3 x 1 interface for fast peripherals (requires separate M.2 HAT or other adapter)</td>
</tr>
<tr>
<td>Expansion Connector</td>
<td>40-pin header</td>
</tr>
<tr>
<td>Fan connector</td>
<td>1 x 4-pin fan connector, supports PWM control and fan speed measurement</td>
</tr>
<tr>
<td>Storage</td>
<td>microSD card slot with UHS-1 support</td>
</tr>
<tr>
<td>Tag Connect</td>
<td>1 x JTAG, 1 x External PMIC programming port</td>
</tr>
</tbody>
</table>

### 1.1.1 AM67A SoC

### 1.2 Board components location

#### 1.2.1 Front components
Table 1.2: BeagleY-AI board front components location

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiFi/BLE</td>
<td>Beagleboard BM3301 with 802.11ax Wi-Fi &amp; Bluetooth Low Energy 5.4 (BLE)</td>
</tr>
<tr>
<td>RAM</td>
<td>4GB LPDDR4</td>
</tr>
<tr>
<td>Expansion</td>
<td>40pin Expansion header compatible with HATs</td>
</tr>
<tr>
<td>SoC</td>
<td>TI AM67A Arm®Cortex®-A53 4 TOPS vision SoC with RGB-IR ISP for 4 cameras, machine vision, robotics, and smart HMI</td>
</tr>
<tr>
<td>Fan</td>
<td>4pin Fan connector</td>
</tr>
<tr>
<td>USB-A</td>
<td>4 x USB 3 TypeA ports supporting simultaneous 5Gbps operation host ports</td>
</tr>
<tr>
<td>Network Connectivity</td>
<td>Gigabit Ethernet</td>
</tr>
<tr>
<td>PoE</td>
<td>Power over Ethernet HAT connector</td>
</tr>
<tr>
<td>Camera/Display</td>
<td>1 x 4-lane MIPI camera/display transceivers, 1 x 4-lane MIPI camera</td>
</tr>
<tr>
<td>Debug UART</td>
<td>1 x 3-pin JST-SH 1.0mm debug UART port</td>
</tr>
<tr>
<td>Display Output</td>
<td>1 x HDMI display</td>
</tr>
<tr>
<td>USB-C</td>
<td>1 x Type-C port for power, and supports USB 2 device</td>
</tr>
<tr>
<td>PMIC</td>
<td>Power Management Integrated Circuit for 5V/5A DC power via USB-C with Power Delivery support</td>
</tr>
<tr>
<td>Bicolor LED</td>
<td>Indicator LED</td>
</tr>
<tr>
<td>Power button</td>
<td>ON/OFF button</td>
</tr>
<tr>
<td>PCIe</td>
<td>PCI-Express® Gen3 x 1 interface for fast peripherals (requires separate M.2 HAT or other adapter)</td>
</tr>
</tbody>
</table>

1.2.2 Back components

Table 1.3: BeagleY-AI board back components location

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag-Connect</td>
<td>1 x JTAG &amp; 1 x Tag Connect for PMIC NVM Programming</td>
</tr>
<tr>
<td>Display output</td>
<td>1 x OLDI display</td>
</tr>
<tr>
<td>Storage</td>
<td>microSD card slot with support for high-speed SDR104 mode</td>
</tr>
</tbody>
</table>

1.2. Board components location
Chapter 2

BeagleY-AI Quick Start

2.1 What’s included in the box?

When you purchase a BeagleY-AI, you’ll get the following in the box:

1. BeagleY-AI
2. 2.4GHz antennas
3. Quick-start card

Tip: For board files, 3D model, and more, you can checkout the BeagleY-AI repository on OpenBeagle.

2.2 Getting started

To get started you need the following:

1. USB type-A to type-C cable or type-C to type-C cable
2. 5V - 3A power supply
3. MicroSD Card
4. Boot media

2.2.1 Boot Media

Download the boot media from https://www.beagleboard.org/distros/beagley-ai-debian-xfce-12-5-2024-03-25 and flash it on a micro SD Card using using Balena Etcher following these steps:

1. Select downloaded boot media
2. Select SD Card
3. Flash!
Once flashed, you can insert the SD card into your BeagleY-AI as shown in the image below:

### 2.2.2 Power Supply

To power the board you can either connect it to a dedicated power supply like a mobile charger or a wall adapter that can provide $5V \geq 3A$. Checkout the [docs power supply page](#) for power supply recommendations.
2.2.3 Board connection

There is only one USB type-C port on board, if you choose to use a dedicated power supply for first time setup, you may access the board via one of the following methods:

1. Connection to HDMI display, Keyboard and Mouse
2. UART using RPi debug probe or similar
3. Ethernet network connection

Another direct and easy option is to connect the board directly to your PC or Laptop using a USB type-C cable.

**Note:** If you are using the board with a fan or running a heavy task you should always power the board with a dedicated power supply that can supply $5V \geq 3A$.

2.2.4 USB Tethering

To initially test your board, you can connect the board directly to your computer using a type-A to type-C cable shown in the image below.

After connecting, you should see the power LED glow, and soon just like with other Beagles, you’ll see a virtual wired connection on your computer. To access the board you can use SSH as shown below.

**Note:** Here you must update the default password to something safer.
2.2.5 Using BeagleY-AI

To setup your BeagleY-AI for normal usage, connect the following:

1. 5V $\geq$ 3A power supply
2. HDMI monitor using micro HDMI to full-size HDMI cable
3. Ethernet cable from the board to your router
4. Wireless or wired keyboard & mice

If everything is connected properly you should see four penguins on your monitor.
When prompted, log in using the updated login credentials you updated during the USB tethering step.

**Note:** You can not update login credentials at this step, you must update them during USB tethering step!

Once logged in you should see the splash screen shown in the image below:
Test network connection by running ping 8.8.8.8

Explore and build with your new BeagleY-AI board!
2.2.6 Connecting to WiFi

Connect 2x antennas to your BeagleY-AI board if not pre-attached.

After successfully attaching the antenna, power up the board. Once booted you can follow the commands below to connect to any WiFi access point,

- To list the wireless devices attached, (you should see wlan0 listed)
  ```
  iwctl device list
  ```

- Scan WiFi using,
  ```
  iwctl station wlan0 scan
  ```

- Get networks using,
  ```
  iwctl station wlan0 get-networks
  ```

- Connect to your wifi network using,
  ```
  iwctl --passphrase "<wifi-pass>" station wlan0 connect "<wifi-name>"
  ```

- Check wlan0 status with,
  ```
  iwctl station wlan0 show
  ```

- To list the networks with connected WiFi marked you can again use,
  ```
  iwctl station wlan0 get-networks
  ```

- Test connection with ping command,
  ```
  ping 8.8.8.8
  ```
2.3 Demos and Tutorials

- Booting from NVMe Drives
Chapter 3

Design and specifications

Work in progress

If you want to know how BeagleY-AI is designed and the detailed specifications, then this chapter is for you. We are going to attempt to provide you a short and crisp overview followed by discussing each hardware design element in detail.

3.1 Block diagram and overview

3.2 SoC

3.3 Boot modes

3.4 Power sources

3.5 PMIC

3.6 General connectivity and expansion

3.7 Buttons and LEDs

3.8 Networking

3.9 Ethernet

3.10 Memory, media, and storage

3.11 Multimedia I/O

3.12 Debug ports
Fig. 3.1: BeagleY-AI block diagram
3.13 Mechanical Specifications

3.13.1 Dimensions & Weight

Table 3.1: Dimensions & weight

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>85 x 56 x 20 mm</td>
</tr>
<tr>
<td>Max height</td>
<td>20 mm</td>
</tr>
<tr>
<td>PCB Size</td>
<td>85 x 56 mm</td>
</tr>
<tr>
<td>PCB Layers</td>
<td>14 layers</td>
</tr>
<tr>
<td>PCB Thickness</td>
<td>1.6 mm</td>
</tr>
<tr>
<td>RoHS compliant</td>
<td>Yes</td>
</tr>
<tr>
<td>Gross Weight</td>
<td>110 g</td>
</tr>
<tr>
<td>Net Weight</td>
<td>50 g</td>
</tr>
</tbody>
</table>
Fig. 3.3: BeagleY-AI I2C tree
Fig. 3.4: BeagleY-AI SoC CSI1, CSI2, and CSI3

3.13. Mechanical Specifications
Fig. 3.5: BeagleY-AI SoC DDR0 connections

Fig. 3.6: BeagleY-AI SoC DSI0 TX connections
Fig. 3.7: BeagleY-AI SoC eFUSE, VMON, Debug, and RSVD

3.13. Mechanical Specifications 21
### Chapter 3. Design and specifications

#### Fig. 3.8: BeagleY-AI SoC GPMC0

<table>
<thead>
<tr>
<th>GPMC0</th>
<th>R22</th>
<th>BOOTO MODE0</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPMC0</td>
<td>R23</td>
<td>BOOTO MODE1</td>
</tr>
<tr>
<td>GPMC0</td>
<td>R26</td>
<td>BOOTO MODE2</td>
</tr>
<tr>
<td>GPMC0</td>
<td>T27</td>
<td>BOOTO MODE3</td>
</tr>
<tr>
<td>GPMC0</td>
<td>T28</td>
<td>BOOTO MODE4</td>
</tr>
<tr>
<td>GPMC0</td>
<td>T24</td>
<td>BOOTO MODE5</td>
</tr>
<tr>
<td>GPMC0</td>
<td>T21</td>
<td>BOOTO MODE6</td>
</tr>
<tr>
<td>GPMC0</td>
<td>T22</td>
<td>BOOTO MODE7</td>
</tr>
<tr>
<td>GPMC0</td>
<td>U27</td>
<td>BOOTO MODE8</td>
</tr>
<tr>
<td>GPMC0</td>
<td>U26</td>
<td>BOOTO MODE9</td>
</tr>
<tr>
<td>GPMC0</td>
<td>V27</td>
<td>BOOTO MODE10</td>
</tr>
<tr>
<td>GPMC0</td>
<td>V25</td>
<td>BOOTO MODE11</td>
</tr>
<tr>
<td>GPMC0</td>
<td>V24</td>
<td>BOOTO MODE12</td>
</tr>
<tr>
<td>GPMC0</td>
<td>V22</td>
<td>BOOTO MODE13</td>
</tr>
<tr>
<td>GPMC0</td>
<td>V23</td>
<td>BOOTO MODE14</td>
</tr>
<tr>
<td>GPMC0</td>
<td>V21</td>
<td>BOOTO MODE15</td>
</tr>
</tbody>
</table>

| GPMC0_ADV/NALE | T22/GPIO0_22/2C1_EN/3V3 | [20] |
| GPMC0_BEO_N_CLE | VOUT0_0ATA16 | [24] |
| GPMC0_BEO_N_CLE | VOUT0_0ATA17 | [24] |
| GPMC0_BEO_N_CLE | VOUT0_0ATA18 | [24] |
| GPMC0_BEO_N_CLE | VOUT0_0ATA19 | [24] |
| GPMC0_BEO_N_CLE | VOUT0_0ATA20 | [24] |
| GPMC0_BEO_N_CLE | VOUT0_0ATA21 | [24] |
| GPMC0_BEO_N_CLE | VOUT0_0ATA22 | [24] |
| GPMC0_BEO_N_CLE | VOUT0_0ATA23 | [24] |

| GPMC0_ADV/NALE | N21/GPIO0_22/USB_RST/3V3 | [21] |
| GPMC0_BEO_N_CLE | P27 | [24] |
| GPMC0_BEO_N_CLE | P26 | [27] |
| GPMC0_BEO_N_CLE | T23 | [26] |
| GPMC0_BEO_N_CLE | T23 | [26] |
| GPMC0_BEO_N_CLE | T23 | [26] |
| GPMC0_BEO_N_CLE | T23 | [26] |
| GPMC0_BEO_N_CLE | T23 | [26] |
| GPMC0_BEO_N_CLE | T23 | [26] |

| GPMC0_OEN_REN | N22/GPIO0_40/RTC_INT/3V3 | [27] |
| GPMC0_WAIT0_T1 | V21 | [27] |
| GPMC0_WAIT0_T1 | W28 | [27] |
| GPMC0_WREN | N23 | [27] |
| GPMC0_WPN | N24 | [27] |

[X722SSAAAW] | BGAS04_0c85_18X18mm | [24] |
| FCBGAS4 | [24] |

---
### 3.13. Mechanical Specifications

**Fig. 3.9: BeagleY-AI SoC ground connections**

<table>
<thead>
<tr>
<th>U26U</th>
<th>Ground/VSS</th>
<th>VSS</th>
<th>P5</th>
<th>F7</th>
<th>P8</th>
<th>P10</th>
<th>P13</th>
<th>P15</th>
<th>P17</th>
<th>P19</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A18</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A21</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A27</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C23</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D26</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E6</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F17</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F21</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G1</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G3</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G5</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G8</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G10</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G12</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G14</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G16</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G20</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H4</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H7</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H14</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G17</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H3</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J5</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J12</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J15</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J17</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J16</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J26</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K1</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K13</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K15</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K17</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K19</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L10</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L11</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L13</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L14</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L16</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L18</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M1</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M7</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M8</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M11</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M14</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M16</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M18</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N6</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N12</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N15</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N17</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N26</td>
<td>VSS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

XJ722SS5AMW
BGA594_0d65_18X18mm
FCEBGA594
Fig. 3.10: BeagleY-AI SoC MMC0, MMC1, and MMC2

Fig. 3.11: BeagleY-AI SoC OLDI
3.13. Mechanical Specifications
Chapter 3. Design and specifications
Fig. 3.18: BeagleY-AI SoC VOUT

Fig. 3.19: BeagleY-AI SoC analog power1

3.13. Mechanical Specifications

27
Fig. 3.20: BeagleY-AI AI SoC IO and DDR power2
Fig. 3.21: BeagleY-AI SoC digital power3
Chapter 3. Design and specifications

Fig. 3.22: BeagleY-AI SoC Reset, Cntrls, and Clk

Fig. 3.23: BeagleY-AI SoC RGMII1 RST

Fig. 3.24: BeagleY-AI VDD core hcp
3.13. Mechanical Specifications

Fig. 3.25: BeagleY-AI wkup reset cntrls osc

Fig. 3.26: BeagleY-AI boot modes

Fig. 3.27: BeagleY-AI VSYS 3V3
Fig. 3.28: BeagleY-AI 3V3/V5 to 1V1 LDO

Fig. 3.29: BeagleY-AI PMIC

Fig. 3.30: BeagleY-AI PMIC NVM programming interface
Fig. 3.31: BeagleY-AI user expansion connector

Fig. 3.32: BeagleY-AI RPI CSI

3.13. Mechanical Specifications
Fig. 3.33: BeagleY-AI RPI DSI/CSI

Fig. 3.34: BeagleY-AI dual USB1
3.13. Mechanical Specifications

Fig. 3.35: BeagleY-AI dual USB2

Fig. 3.36: BeagleY-AI dual USB current limiter

Ilimit is set to 2000mA
Fig. 3.37: BeagleY-AI fan connector

Fig. 3.38: BeagleY-AI general IO
### 3.13. Mechanical Specifications

**Fig. 3.39: BeagleY-AI MCU general IO**

![MCU General IO Diagram](image)

- **Fig. 3.40: BeagleY-AI USB3 hub**

![USB3 Hub Diagram](image)
Fig. 3.41: BeagleY-AI USB-C

Fig. 3.42: BeagleY-AI USB hub config
Fig. 3.43: BeagleY-AI USB VBUS resistor divider circuit

Note: USB VBUS Resistor divider circuit.

Fig. 3.44: BeagleY-AI I2C2 pull-up resistors
Fig. 3.45: BeagleY-AI I2C ext RTC

Fig. 3.46: BeagleY-AI voltage level translator
Fig. 3.47: BeagleY-AI LEDs
Fig. 3.48: BeagleY-AI WiFi module
Fig. 3.49: BeagleY-AI ethernet connector
Fig. 3.50: BeagleY-AI ethernet DP83867

Fig. 3.51: BeagleY-AI ethernet phy caps

Fig. 3.52: BeagleY-AI ethernet phy misc
3.13. Mechanical Specifications
Fig. 3.56: BeagleY-AI board id eeprom

Fig. 3.57: BeagleY-AI DDR caps
Fig. 3.58: BeagleY-AI DDR

Fig. 3.59: BeagleY-AI DDR power

3.13. Mechanical Specifications
Fig. 3.60: BeagleY-AI microSD card interface

Fig. 3.61: BeagleY-AI PCIE connector

Fig. 3.62: BeagleY-AI HDMI addr protection
Fig. 3.63: BeagleY-AI HDMI power

Fig. 3.64: BeagleY-AI HDMI reset

Fig. 3.65: BeagleY-AI RGB888 to HDMI

Review Note:
HDMI: VDD_1V2, Imax=46.01mA
Option 1: VDD_1V2 = VIDO2_1V2, from LDO2 of PMIC TPS6521910
Option 2: VDD_1V2 = VDD_LDO_1V2, from external LDO U4.
Fig. 3.66: BeagleY-AI Tag-Connect

Fig. 3.67: BeagleY-AI debug UART port
Chapter 4

Expansion

4.1 PCIe

For software reference, you can see how PCIe is used on NVMe HATs.

- Booting from NVMe Drives
- Using IMX219 CSI Cameras
- Using the on-board Real Time Clock (RTC)
Chapter 5

Demos and tutorials

5.1 Using GPIO

GPIO stands for General-Purpose Input/Output. It’s a set of programmable pins that you can use to connect and control various electronic components.

You can set each pin to either read signals (input) from things like buttons and sensors or send signals (output) to things like LEDs and motors. This lets you interact with and control the physical world using code!

A great resource for understanding pin numbering can be found at pinout.beagley.ai

**Note:** WARNING - BeagleY-AI GPIOs are 3.3V tolerant, using higher voltages WILL damage the processor!

5.1.1 Pin Numbering

You will see pins referenced in several ways. While this is confusing at first, in reality, we can pick our favorite way and stick to it.

The two main ways of referring to GPIOs is by their number, so GPIO 2, 3, 4 etc. as seen in the diagram below. This corresponds to the SoC naming convention. For broad compatibility, BeagleY-AI re-uses the Broadcom GPIO numbering scheme used by RaspberryPi.

The second (and arguably easier) way we will use for this tutorial is to use the actual pin header number (shown in dark grey)

So, for the rest of the tutorial, if we refer to hat-08-gpio we mean the 8th pin of the GPIO header. Which, if you referenced the image below, can see refers to GPIO 14 (UART TX)

If you are curious about the “real” GPIO numbers on the Texas Instruments AM67A SoC, you can look at the board schematics.

5.1.2 Required Hardware

For the simple blink demo, all that is needed is an LED, a Resistor (we use 2.2K here) and 2 wires.

Similarly, a button is used for the GPIO read example, but you can also just connect that pin to 3.3V or GND with a wire to simulate a button press.

5.1.3 GPIO Write

At it’s most basic, we can set a GPIO using the gpioset command.
Fig. 5.1: BeagleY-AI pinout
To set HAT Pin 8 to ON:

```bash
gpioset hat-08-gpio 0=1
```

To set HAT Pin 8 to OFF:

```bash
gpioset hat-08-gpio 0=0
```

### 5.1.4 Blink an LED

Let's write a script called `blinky.sh` that contains the following:

```bash
#!/bin/bash
while :
do
gpioset hat-08-gpio 0=1
sleep 1
gpioset hat-08-gpio 0=0
sleep 1
done
```

The script is quite simple, it's an infinite “while” loop in which we do the following:

1. set the HAT Pin 8 as 1 (HIGH)
2. Wait 1 Second
3. set the HAT Pin 8 as 0 (LOW)
4. Wait 1 Second

Now execute it by typing:

```
bash blinky.sh
```
You can exit by pressing Ctrl + c on your keyboard.

5.1.5 GPIO Read

Reading GPIOs can be done using the `gpioget` command:

```bash
gpioget hat-08-gpio 0
```

Results in 1 if the Input is held HIGH or 0 if the Input is held LOW

5.1.6 Read a Button

A push button simply completes an electric circuit when pressed. Depending on wiring, it can drive a signal either “Low” (GND) or “High” (3.3V).

We will connect our Button between HAT Pin 16 (GPIO23) and Ground (GND).

The cool part is since we have an internal pull-up resistor, we don’t need an external one! The pull resistor guarantees that the Pin stays in a known (HIGH) state unless the button is pressed, in which case it will go LOW.

Let’s write a script called `button.sh` to continuously read an input pin connected to a button and print out when it’s pressed! :

```bash
#!/bin/bash
while : do
  if (( $(gpioget hat-12-gpio 0) == 0 ))
    then
      echo ”Button Pressed!”
  fi
done
```

5.1.7 Combining the Two

Now, logically, let’s make an LED match the state of the button.

Let’s modify our script and call it `blinkyButton.sh`:

```bash
#!/bin/bash
while : do
  if (( $(gpioget hat-12-gpio 0) == 0 ))
    then
      gpioset hat-08-gpio 0=1
  else
    (continues on next page)
```

(continues on next page)
This means when we see HAT Pin 12 go LOW, we know the button is pressed, so we set HAT Pin 8 (our LED) to ON, otherwise, we turn it OFF.

Now execute it by typing:

```
bash blinkyButton.sh.sh
```

You can exit by pressing Ctrl + c on your keyboard.

### 5.1.8 Understanding Internal Pull Resistors

Pull-up and pull-down resistors are used in digital circuits to ensure that inputs to logic settle at expected levels.

- **Pull-up resistors**: Connect the input to a high voltage level (e.g., 3.3V) to ensure the input reads as a logic high (1) when no active device is pulling it low.
- **Pull-down resistors**: Connect the input to ground (GND) to ensure the input reads as a logic low (0) when no active device is pulling it high.

These resistors prevent floating inputs and undefined states.

By default, all GPIOs on the HAT Header are configured as **Inputs with Pull-up Resistors Enabled**.

This is important for something like a button, as without it, once a button is released, it goes in an “undefined” state!

To configure Pull-ups on a per-pin basis, we can use pass the following arguments within `gpioget` or `gpioset`:

```
-B, --bias=[as-is|disable|pull-down|pull-up] (defaults to 'as-is')
```

The “Bias” argument has the following options:

- **as-is** - This leaves the bias as-is... quite self explanatory
- **disable** - This state is also known as High-Z (high impedance) where the Pin is left Floating without any bias resistor
- **pull-down** - In this state, the pin is pulled DOWN by the internal 50KΩ resistor
- **pull-up** - In this state, the pin is pulled UP by the internal 50KΩ resistor

For example, a command to read an input with the Bias intentionally disabled would look like this:

```
gpioget --bias=disable hat-08-gpio 0
```

Pull resistors are a foundational block of digital circuits and understanding when to (and not to) use them is important.

This article from SparkFun Electronics is a good basic primer - [Link](#)
5.1.9 Troubleshooting

- **My script won’t run!**

Make sure you gave the script execute permissions first and that you’re executing it with a `./` before.

To make it executable:

```bash
chmod +X scriptName.sh
```

To run it:

```bash
./scriptName.sh
```

5.1.10 Bonus - Turn all GPIOs ON/OFF

Copy and paste this with the button on the right to turn all pins ON.

```bash
gpioset hat-03-gpio 0=1 ;
gpioset hat-05-gpio 0=1 ;
gpioset hat-08-gpio 0=1 ;
gpioset hat-10-gpio 0=1 ;
gpioset hat-11-gpio 0=1 ;
gpioset hat-12-gpio 0=1 ;
gpioset hat-13-gpio 0=1 ;
gpioset hat-14-gpio 0=1 ;
gpioset hat-15-gpio 0=1 ;
gpioset hat-16-gpio 0=1 ;
gpioset hat-18-gpio 0=1 ;
gpioset hat-19-gpio 0=1 ;
gpioset hat-20-gpio 0=1 ;
gpioset hat-21-gpio 0=1 ;
gpioset hat-22-gpio 0=1 ;
gpioset hat-23-gpio 0=1 ;
gpioset hat-24-gpio 0=1 ;
gpioset hat-26-gpio 0=1 ;
gpioset hat-27-gpio 0=1 ;
gpioset hat-28-gpio 0=1 ;
gpioset hat-29-gpio 0=1 ;
gpioset hat-30-gpio 0=1 ;
gpioset hat-31-gpio 0=1 ;
gpioset hat-32-gpio 0=1 ;
gpioset hat-33-gpio 0=1 ;
gpioset hat-34-gpio 0=1 ;
gpioset hat-35-gpio 0=1 ;
gpioset hat-36-gpio 0=1 ;
gpioset hat-37-gpio 0=1 ;
gpioset hat-38-gpio 0=1 ;
gpioset hat-39-gpio 0=1 ;
gpioset hat-40-gpio 0=1
```

Similarly, copy and paste this to turn all pins OFF.

```bash
gpioset hat-03-gpio 0=0 ;
gpioset hat-05-gpio 0=0 ;
gpioset hat-08-gpio 0=0 ;
gpioset hat-10-gpio 0=0 ;
gpioset hat-11-gpio 0=0 ;
gpioset hat-12-gpio 0=0 ;
gpioset hat-13-gpio 0=0 ;
gpioset hat-14-gpio 0=0 ;
gpioset hat-15-gpio 0=0 ;
gpioset hat-16-gpio 0=0 ;
gpioset hat-18-gpio 0=0 ;
gpioset hat-19-gpio 0=0 ;
gpioset hat-20-gpio 0=0 ;
gpioset hat-21-gpio 0=0 ;
gpioset hat-22-gpio 0=0 ;
gpioset hat-23-gpio 0=0 ;
gpioset hat-24-gpio 0=0 ;
gpioset hat-26-gpio 0=0 ;
gpioset hat-27-gpio 0=0 ;
gpioset hat-28-gpio 0=0 ;
gpioset hat-29-gpio 0=0 ;
gpioset hat-30-gpio 0=0 ;
gpioset hat-31-gpio 0=0 ;
gpioset hat-32-gpio 0=0 ;
gpioset hat-33-gpio 0=0 ;
gpioset hat-34-gpio 0=0 ;
gpioset hat-35-gpio 0=0 ;
gpioset hat-36-gpio 0=0 ;
gpioset hat-37-gpio 0=0 ;
gpioset hat-38-gpio 0=0 ;
gpioset hat-39-gpio 0=0 ;
gpioset hat-40-gpio 0=0
```

5.1.11 Going Further

- [pinout.beagley.ai](http://pinout.beagley.ai)
- [GPIOSet Documentation](#)
- [GPIOGet Documentation](#)
5.2 Pulse Width Modulation (PWM)

5.2.1 What is it

PWM, or Pulse Width Modulation, is a technique used to control the amount of power delivered to an electronic device by breaking up the power signal into discrete ON and OFF periods. The amount of time the signal spends ON during each cycle determines the output power level (brightness of the LED).

![PWM Signal Diagram]

5.2.2 How do we do it

First we unbind the pin as GPIO

```
echo hat-08-gpio > /sys/bus/platform/drivers/gpio-aggregator/unbind
```

Now we override the driver

```
echo gpio-aggregator > /sys/devices/platform/hat-08-pwm/driver_override
```

Then we bind the pin

```
echo hat-08-pwm > /sys/bus/platform/drivers/gpio-aggregator/bind
```

Let’s write a script called `fade.sh` that contains the following:

```
#!/bin/bash

PWMPIN="/sys/devices/platform/bus@f0000/23000000.pwm/pwm/pwmchip3/pwm1"

echo 1000 > $PWMPIN/period
echo 0 > $PWMPIN/duty_cycle
echo 0 > $PWMPIN/enable
sleep 1
for i in {1..500};
  do
echo $i > $PWMPIN/duty_cycle
  echo 1 > $PWMPIN/enable
  echo $i
  sleep 0.0005
done
for i in {500..1};
  do
  echo $i > $PWMPIN/duty_cycle
  echo 1 > $PWMPIN/enable
  echo $i
```

(continues on next page)
5.2.3 Troubleshooting

5.2.4 Going Further

5.3 Using the on-board Real Time Clock (RTC)

Real Time Clocks (RTCs) provide precise and reliable timekeeping capabilities, which are beneficial for applications ranging from simple timekeeping to complex scheduling and secure operations.

Without an RTC, a computer must rely on something called Network Time Protocol (NTP) to obtain the current time from a network source. There are many cases however where an SBC such as BeagleY-AI may not have a constant or reliable network connection. In situations such as these, an RTC allows the board to keep time even if the network connection is severed or the board loses power for an extended period of time.

Fortunately, BeagleY-AI comes with a built-in **DS1340** RTC for all your fancy time keeping needs!

5.3.1 Required Hardware

BeagleY provides a **1.00 mm pitch, 2-pin JST SH connector** for a coin cell battery to enable the RTC to keep time even if power is lost to the board.

These batteries are available from several vendors:

- Raspberry Pi 5 RTC Battery via Adafruit
- Raspberry Pi 5 RTC Battery via DigiKey
5.3.2 Uses for an RTC

1. **Maintaining Accurate Time:** RTCs provide an accurate clock that continues to run even when the SBC is powered down. This is crucial for maintaining the correct time and date across reboots.

2. **Timestamping:** Many applications need to know the current time for timestamping data, logs, or events. For example, IoT devices may need to log sensor data with precise timestamps.

3. **Scheduling Tasks:** In some applications, tasks need to be scheduled at specific times. An RTC allows the SBC to keep track of time accurately, ensuring that tasks are performed at the correct times.

4. **Network Synchronization:** If the SBC is part of a larger network, having an accurate time helps with synchronizing data and events across the network.

5. **Standby Power Efficiency:** Many RTCs operate with a very low power requirement and can keep time even when the rest of the board is in a low-power or sleep mode. This helps in reducing overall power consumption.

5.3.3 Reading time

**Note:** If you have not connected your BeagleY-AI to a network so it can get time from an NTP server, you must set the time before being able to read it. If you don’t do this first, you’ll see errors.

Reading the current time on the RTC is achieved using the `hwclock` command.

```
debian@BeagleY:~$ sudo hwclock
2024-05-10 00:00:02.224187-05:00
```

5.3.4 Setting time

You can set time manually by running the following command:

```
5.3. Using the on-board Real Time Clock (RTC)  61
```
5.3.5 Diving Deeper

There are actually two different "times" that your Linux system keeps track of.

- System time, which can be read using the `date` or `timedatectl` commands
- RTC (hardware) time which can be read using the `hwclock` command shown above.

Comparing the time, we see something interesting, they're different!

You can just type "date" but the format will be different, so we add some extra instructions to match the format.

```bash
debian@BeagleBone:~$ date +\%Y-%m-%d\ '\%H:\%M:\%S.\%N\%:z
2024-05-10 21:06:50.05859373+00:00
```

```bash
debian@BeagleBone:~$ sudo hwclock
2024-05-10 21:06:56.692874+00:00
```

But why? We see here that our system and hardware clock are over 9 seconds apart!

Ok, in this particular case we set the HW clock slightly ahead to illustrate the point, but in real life “drift” is a real problem that has to be dealt with. Environmental conditions like temperature or stray cosmic rays can cause electronics to become ever so slightly out of sync, and these effects only grow over time unless corrected. It’s why RTCs and other fancier time keeping instruments implement various methods to help account for this such as temperature compensated oscillators.

Let’s fix our hardware clock. We assume here that the system clock is freshly synced over NTP so it’s going to be our true time “source”.

```bash
debian@BeagleBone:~$ sudo hwclock --systohc
```

Let’s write a simple script to get the two times, we’ll call it `getTime.sh`:

```bash
HWTIME=${(sudo hwclock)
echo “RTC - ${HWTIME} ”
SYSTIME=${(date +\%Y-%m-%d\ ’\%H:\%M:\%S.\%N\%:z)
echo “SYS - ${SYSTIME} ”
```

Now let’s run it!

```bash
debian@BeagleBone:~$ sudo chmod +x getTime.sh
debian@BeagleBone:~$ ./getTime.sh
RTC - 2024-05-10 21:52:58.374954+00:00
SYS - 2024-05-10 21:52:59.048442940+00:00
```

As we can see, we’re still about a second off, but this is because it takes a bit of time to query the RTC via I2C.

If you want to learn more, the Going Further at the end of this article is a good starting point!

5.3.6 Troubleshooting

The most common error results from not having initialized the RTC at all. This usually happens if the system is powered on without an RTC battery and without a network connection.

In such cases, you should be able to read the time after setting the time as follows:
5.3.7 Going Further

Consider learning about topics such as time keeping over GPS and Atomic Clocks!

There are some good YouTube videos below to provide sources for inspiration.

Network Time Protocol - Computerphile

Nanosecond Clock Sync - Jeff Geerling

Using GPS with PPS to synchronize clocks over the network

Note: This page is a work in progress. Further drive testing and images will be added soon

5.4 Using PCA9685 Motor Drivers

There are several such “Motor and Servo Driver HATs” available on Amazon, Adafruit and other marketplaces. While different manufacturers implement them slightly differently, the operating principle remains the same.

This guide aims to show you examples for two, namely the Xicoolee and Adafruit variants and how you can modify the example Python userspace library for other variations.
Chapter 5. Demos and tutorials

Fig. 5.3: https://youtu.be/RvnG-ywF6_s

Fig. 5.4: https://youtu.be/7aTZ66ZL6Dk
5.4.1 Operating Principle

The NXP PCA9685 is a simple 16-channel, 12-bit PWM controller that communicates over I2C.

While originally designed as an LED driver, its ability to output PWM also makes it suitable as a Servo Motor driver.

In addition, to add the ability to drive DC motors, some board designers add one or two Toshiba TB6612FNG dual motor drivers as shown in the schematic below.

If we look at the Xicoolee board and compare it to the schematic, we see that indeed Servo Channels 3-8 on the PCB Silkscreen match pins 12 through 18 of the PWM Driver, while PWM1, PWM2, INA1/2 and INB1/2 are used in conjunction with the TB6612FNG.

Looking at the TB6612FNG Datasheet, we can see that the IN pins for Channels A and B (INAx, INBx) are used to control the direction or “mode” of the DC motor, while the PWM signal controls the rotation speed for that particular channel.
Thus, we can use the decoder table above to infer that to drive motor channel A at 50% speed clockwise, we would set the PCA9685 to output INA1 High, INA2 Low and PWM1 at a 50% duty cycle.

If we wanted to go counter-clockwise, we would simply swap things around so INA1 was Low, INA2 was High and assuming we want to keep the same rotation speed, PWM1 at a 50% duty cycle.

Lastly, we have the option for a “Short Brake” for the motors but please note that it is not recommended to keep motors in this state as that shorts the coils internally and will cause them to heat up over time. If you want to stop your motor, you should issue a “Short brake” state followed by a short delay to allow the motor to physically stop rotating and then leave the motor in the “Stop” state (which de-energizes the coils) by setting IN1 and IN2 to LOW.

But enough theory, let’s use some actual code to make things spin...

5.4.2 Using Adafruit ServoKit

If you are looking to drive Servo motors accurately and not particularly interested in driving DC motors, you may consider using the Adafruit ServoKit library which simplifies this type of use case. As with all python modules, make sure you do so inside a virtual environment as shown below!

```bash
mkdir project-name && cd project-name
python3 -m venv .venv
source .venv/bin/activate
sudo pip3 install --upgrade setuptools
sudo pip install --upgrade adafruit-python-shell
sudo python raspi-blinka.py
pip3 install adafruit-circuitpython-servokit adafruit-circuitpython-busdevice adafruit-circuitpython-register
```

From here, you should be able to run some example code such as the following:

```python
import time
from adafruit_servokit import ServoKit

# Set channels to the number of servo channels on your kit.
# 8 for FeatherWing, 16 for Shield/HAT/Bonnet.
```
5.4.3 Python User-space Driver

As mentioned before, the PCA9685 is a rather simple I2C device, so the driver for it is equally simple: PCA9685.py

Simply download this to the root of your project and you are most of the way there.

From there, you simply need an import statement and to define the driver instance:

```python
from PCA9685 import PCA9685
pwm = PCA9685(0x60, debug=False)  # Default I2C Address for the shield is 0x60
pwm.setPWMFreq(50)  # Most Servo Motors use a PWM Frequency of 50Hz
```

You can now drive LEDs or servo motors by issuing the following command (replacing pin and dutyCycle with your particular values):

```python
pwm.setDutycycle(pin, dutyCycle)
```

5.4.4 WaveShare Motor and Servo Driver HAT

Waveshare writes some of the better documentation for these types of Motor Driver HATs

5.4.5 XICOOLEE Motor and Servo Driver HAT
Looking at the schematic for the Xicoolee HAT, we see that we need to define our DC motor pins as follows:

```python
# Xicoolee TB6612FNG
self.PWMA = 0
self.AIN1 = 2
self.AIN2 = 1
self.PWMB = 5
self.BIN1 = 3
self.BIN2 = 4
```

We can then run some simple example code as shown below:

```python
#!/usr/bin/python
from PCA9685 import PCA9685
import time

Dir = ['forward', 'backward']
pwm = PCA9685(0x40, debug=False)
pwm.setPWMFreq(50)

class MotorDriver():
    def __init__(self):
        # Match these to your particular HAT!
        self.PWMA = 0
        self.AIN1 = 2
        self.AIN2 = 1
        self.PWMB = 5
        self.BIN1 = 3
        self.BIN2 = 4

    def MotorRun(self, motor, index, speed):
        if speed > 100:
            return
        if (motor == 0):
            pwm.setDutyCycle(self.PWMA, speed)
        if (index == Dir[0]):
            print ("1")
            pwm.setLevel(self.AIN1, 0)
            pwm.setLevel(self.AIN2, 1)
        else:
            print ("2")
            pwm.setLevel(self.AIN1, 1)
            pwm.setLevel(self.AIN2, 0)

        elif (index == Dir[1]):
            print ("3")
            pwm.setLevel(self.BIN1, 0)
            pwm.setLevel(self.BIN2, 1)
        else:
            print ("4")
            pwm.setLevel(self.BIN1, 1)
            pwm.setLevel(self.BIN2, 0)

    def MotorStop(self, motor):
        if (motor == 0):
            (continues on next page)
```
pwm.setDutycycle(self.PWMA, 0)
else:
pwm.setDutycycle(self.PWMB, 0)

print("this is a motor driver test code")
Motor = MotorDriver()

print("forward 2 s")
Motor.MotorRun(0, 'forward', 100)
time.sleep(2)

print("backward 2 s")
Motor.MotorRun(0, 'backward', 100)
time.sleep(2)

print("stop")
Motor.MotorStop(0)
Motor.MotorStop(1)

5.4.6 Adafruit DC & Stepper Motor HAT

Looking at the schematic for the Adafruit HAT, we see that we need to define our DC motor pins as follows:
5.5 Booting from NVMe Drives

Note: This page is a work in progress. Further drive testing and images will be added soon

BeagleY-AI supports a PCI-Express x1 interface which enables data rates of up to 1GB/s for high speed expansion.

Note: While the SoC supports PCI-e Gen 3, the flat-flex connector required by HATs is only rated for PCI-e Gen 2, so, as is the case with other similar boards in this form factor, actual transfer speeds may be limited to Gen 2, depending on a variety of layout and environmental factors

This enables it to take advantage of standard PC NVMe drives which offer exponentially higher random and sequential read/write speeds as well as improved endurance over SD cards or traditional eMMC storage.

While the boot-ROM on the AM67 SoC does not support direct boot-to-NVMe, we can use a method where we boot U-Boot from the SD Card and then use it to load the Linux filesystem from external NVMe storage.

5.5.1 Verified HATs and Drives

Most/All HATs and NVMe drives should work, but the following have been verified to work as part of writing this guide:

HATs:
1. Geekworm X1001 PCIe to M.2 Key-M
2. Geekworm X1000 PCIe M.2 Key-M

NVMe drives:
1. Kingston OM3PDP3512B (512GB 2230)
2. Kingston NV2 (512GB 2280)

Drive Adapters (3D Printable):
The X1000 above uses the slightly uncommon 2242 drive size, so, an adapter may be required to mount a 2230 drive.
1. A simple adapter from @eliasjonsson on Printables works great - https://www.printables.com/model/578236-m2-ssd-2230-to-2242

2. Similar adapters exist for 2230 to 2280 for example such as this one from @nzalog - https://www.printables.com/model/217264-2230-to-2280-m2-adapter-ssd

5.5.2 Step by step

Note: This article was written using the BeagleY-AI Debian XFCE 12.5 2024-03-25 image.

Step 1. Boot from SD Normally

Grab the latest BeagleY-AI SD Image from (BeagleBoard.org/distros.)

Once logged in and at the terminal, make sure your system is up to date (a reboot is also recommended after updating)

```
sudo apt-get update && sudo apt-get full-upgrade -y
sudo reboot
```

Step 2. Verify that your NVMe drive is detected

The command `lspci` will list the attached PCI Express devices on the system:

```
debian@BeagleY:~$ lspci
```

You should see an output similar to the following, where the first entrance is the SoC internal PCI Express bridge device and the second device listed is your NVMe drive, in this case, a Kingston OM3PDP3 drive.

```
00:00.0 PCI bridge: Texas Instruments Device b010
01:00.0 Non-Volatile memory controller: Kingston Technology Company, Inc.,→OM3PDP3 NVMe SSD (rev 01)
```

Now that we know the PCIe device is detected, let’s see if it’s recognized as a Storage Device:

The command `lsblk` will list the attached storage devices on the system:

```
debian@BeagleY:~$ lsblk
```

```
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
mmcblk1 179:0 0 29.7G 0 disk
├─mmcblk1p1 179:1 0 256M 0 part /boot/firmware
├─mmcblk1p2 179:2 0 4G 0 part [SWAP]
└─mmcblk1p3 179:3 0 25.5G 0 part /
nvme0n1 259:0 0 476.9G 0 disk
└─nvme0n1p1 259:1 0 476.9G 0 part
```

Here we see that two devices are connected, `mmcblk1` corresponds to our SD card, and `nvme0n1` corresponds to our NVMe drive, so everything is ready to go!

If your drives aren’t listed as expected, please check the Troubleshooting section at the end of this document.

Step 3. Copy your filesystem and modify extlinux.conf for NVMe boot

A variety of useful scripts are available in `/opt/`, one of them enables us to move our micro-sd contents to NVMe and make BeagleY-AI boot from there directly.

5.5. Booting from NVMe Drives 71
The following 3 commands will change your U-boot prompt to boot from NVMe by default, but the serial boot menu will still enable you to fall back to SD boot or other modes if something happens.

**Note:** This will copy the entire contents of your SD card to the NVMe drive, so expect it to take upwards of 15 minutes. This only needs to be run one time.

```
sudo cp -v /opt/u-boot/bb-u-boot-beagley-ai/beagley-ai-microsd-to-nvme-w-\→swap /etc/default/beagle-flasher
sudo beagle-flasher-mv-rootfs-to-nvme
sudo reboot
```

Enjoy NVMe speeds!

Now that we’ve run the scripts above, you should see that `lsblk` now reports that our `/` or root filesystem is on the `nvme0n1p1` partition, meaning we are successfully booting from the NVMe drive.

It’s subtle, but the change can be seen by running `lsblk` again.

```
debian@BeagleY:~$ lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
mmcblk1 179:0  0  29.7G  0  disk
├─mmcblk1p1 179:1  0  256M  0  part /boot/firmware
├─mmcblk1p2 179:2  0  4G   0  part
└─mmcblk1p3 179:3  0  25.5G  0  part
nvme0n1 259:0  0  476.9G  0  disk
└─nvme0n1p1 259:1  0  476.9G  0  part /
```

Congratulations!

### 5.5.3 Troubleshooting

While most setups should work, it is possible that a combination of Software, Hardware or both can result in minor issues. Here are some ideas for troubleshooting on your own:

**Check that your cables are plugged in and oriented correctly**

The flat-flex ribbon cable will only connect correctly one way, so ensure the orientation is correct with your expansion HAT manual and that the ribbon cable is correctly seated.

**A note on power-hungry drives**

While most drives can be powered as-is with only the ribbon cable, some drives, especially high end full-size 2280 drives may consume more power than normal for an M.2 connector. For such cases, some HAT expansions will provide a means of providing external supplemental power. If your drive is not detected, it may be worthwhile to try using a drive from a different manufacturer as a troubleshooting step.

As a side note, since 2230 drives are normally designed to run in Laptops, they tend to also consume less power than their desktop counterparts and as such, are a “safer” option.

**Check the Linux Kernel Logs for PCI:**

You should see something similar to below without further errors:
5.6 Using IMX219 CSI Cameras

Note: This page is a work in progress. Further drive testing and images will be added soon.

To enable an IMX219 CSI camera, modify the following file: /boot/firmware/extlinux/extlinux.conf

We can check the available list of Device Tree Overlays as such:

```
debian@BeagleBone:~$ ls /boot/firmware/overlays/ | grep "beagley"
```

```
k3-am67a-beagley-ai-csi0-imx219.dtbo
k3-am67a-beagley-ai-csi0-ov5640.dtbo
k3-am67a-beagley-ai-csi1-imx219.dtbo
k3-am67a-beagley-ai-dsi-rpi-7inch-panel.dtbo
k3-am67a-beagley-ai-lincolntech-185lcd-panel.dtbo
```

5.6.1 Using CSI Port 0

Then, add the following line to load the IMX219 CSI0 DTBO:

```
fdtoverlays /overlays/k3-am67a-beagley-ai-csi0-imx219.dtbo
```

Your /boot/firmware/extlinux/extlinux.conf file should look something like this:

```
label microSD (default)
    kernel /Image
    append console=ttyS2,115200n8 root=/dev/mmcblk0p2 ro rootfstype=ext4
    rootwait net.ifnames=0
    fdt /ti/k3-j722s-beagley-ai.dtb
    fdtoverlays /overlays/k3-am67a-beagley-ai-csi0-imx219.dtbo
    initrd /initrd.img
```

Now reboot...

```
debian@BeagleBone:~$ ls /dev/ | grep "video"
```

```
video0
video1
video2
```
5.6.2 Using CSI Port 1

5.6.3 Troubleshooting

```
Found /extlinux/extlinux.conf
Retrieving file: /extlinux/extlinux.conf
beagley-ai microSD (extlinux.conf)
  1: microSD Recovery
  2: microSD (RPI 7inch panel)
  3: microSD (lincolntech-1851cd panel)
  4: microSD (csi0 imx219)
  5: microSD (csi1 imx219)
  6: microSD (csi0 ov5640)
  7: microSD (default)
Enter choice: 4
  4: microSD (csi0 imx219)
```

5.7 Using the Arducam Dual V3Link Camera Kit

The Arducam Dual V3Link Camera Kit is an IMX219 based kit that leverages Texas Instruments’ FPDLInk technology to enable using two CSI cameras over a single port up to 15 meters away using twisted pair cables.
Note: Unlike the larger quad-camera kit, the dual camera kit aims to simplify the software stack and improve interoperability with the Raspberry Pi and other non-TI SBCs by forgoing the ability to support multi-stream CSI inputs. This means that it is limited to “switching” between the two FPDLink inputs but has the benefit of not requiring additional drivers beyond support for the base CSI camera driver (IMX219 in this case).

5.7.1 Initial Hardware Connection

Simply plug in the HAT into the BeagleY GPIO header and connect the CSI header as shown below. Either CSI header may be connected but make sure you use the corresponding CSI port DTS when enabling your “camera”.

5.7.2 Verify that the HAT is connected

The Arducam HAT should present itself as an I2C device on Bus 1. To check that the I2C Bus looks like we expect:
sudo i2cdetect -r -y 1

To verify actual communication with the FPDlink device, we issue the following command:
sudo i2ctransfer -f -y 4 w3@0x0c 0xff 0x55 0x01 r1

5.7.3 Switching CSI Channels

The channel numbering for FPDLink goes from 1 to 2 (as opposed to counting from 0 as is the case for CSI)
Thus, to select video output from channel 1:
sudo i2ctransfer -f -y 4 w3@0x0c 0xff 0x55 0x01

To switch to channel 2:
sudo i2ctransfer -f -y 4 w3@0x0c 0xff 0x55 0x02

5.7.4 Troubleshooting

For additional documentation and support, see the Arducam Docs.
Chapter 6

Support

All support for BeagleY-AI design is through BeagleBoard.org community at BeagleBoard.org forum.

6.1 Production board boot media

6.2 Certifications and export control

6.2.1 Export designations

• HS: 8471504090
• US HS: 8543708800
• UPC: 640265311062
• EU HS: 8471707000
• COO: CHINA

6.2.2 Size and weight

• Bare board dimensions: 85 x 56 x 20 mm
• Bare board weight: 50 g
• Full package dimensions: 140 x 100 x 40 mm
• Full package weight: 110g

6.3 Additional documentation

6.3.1 Hardware docs

For any hardware document like schematic diagram PDF, EDA files, issue tracker, and more you can checkout the BeagleY-AI design repository.

6.3.2 Software docs

For BeagleY-AI specific software projects you can checkout all the BeagleY-AI project repositories group.
6.3.3 Support forum

For any additional support you can submit your queries on our forum, https://forum.beagleboard.org/tag/beagley-ai

6.3.4 Pictures

6.4 Change History

**Note:** This section describes the change history of this document and board. Document changes are not always a result of a board change. A board change will always result in a document change.

6.4.1 Board Changes

For all changes, see https://openbeagle.org/beagley-ai/beagley-ai. Versions released into production are noted below.

Table 6.1: BeagleY-AI board change history

<table>
<thead>
<tr>
<th>Rev</th>
<th>Changes</th>
<th>Date</th>
<th>By</th>
</tr>
</thead>
</table>
