Introduction

The OSD3358 is a System-In-Package (SIP) that contains a Texas Instruments AM3358, 512MB of DDR3, and power management for both internal and external needs thus providing components common to many applications. For a complete list of features and benefits see the OSD3358 Data Sheet.

This document provides, along with usage notes for major subsystems, guidelines for getting started with your application specific OSD3358 based development, and notes on migration from a BeagleBone Black development platform to your OSD3358 based product. There are two development platforms referenced by this document; the AM3358 based BeagleBone Black Rev C and the OSD3358 Single-board Computer (SBC) Reference Design. The "Migration to an OSD3358 Based Design" section provides notes on the migration from the Beaglebone Black Rev C to the OSD3358 SBC Reference Design.

Also, your application may have requirements that differ from those of a standard BeagleBone Black design. The "General Applications Considerations" section provides notes on eliminating unused subsystems from your application, and notes on options and differences that may occur when migrating from a BeagleBone Black based design to your own product.

Note: The AM335x Datasheet references specific features that are supported by specific package types. The OSD3358 supports all the features of a revision 2.1 AM3358 ZCZ package.

Notice: The information provided within this document is for informational use only. Octavo Systems makes no guarantees or warranty to the information contained.

OCTAVO SYSTEMS

OSD3358 Application Guide

Rev.4 8/16/2016

Table of Contents

1 Revision History

Revision Number	Revision Date	Changes	Author
1.0	5/4/2016	First release	Doug Deao
2.0	5/27/2016	Fixed Links to OSD335x Data Sheet	Greg Sheridan
3.0	6/21/2016	Added NAND, Device Driver and Kernel Memory Footprint Reduction sections	Doug Deao
4.0	8/15/2016	Updated Device Drivers Section	Doug Deao

2 Documentation Reference

OSD3358 Data Sheet
OSD3358 SBC Reference Design Schematics
AM335x Data Sheet
AM335x Technical Reference Manual (TRM)
TPS65217x PMIC Datasheet
TL5209 LDO Datasheet
Beagle Bone Black System Reference Manual (SRM)
BeagleBone Black Schematics Rev C
AM335x Schematic Checklist
AM335x Silicon Errata

3 Migration to an OSD3358 Based Design

These sections discuss detail of migrating from a BeagleBone Black or OSD3358 SBC Reference Design to your own application specific OSD3358 based design. Your application may not require all the BeagleBone Black peripherals. The following sections provide guidelines for eliminating components where your requirements allow.

3.1 OSD3358 Single-board Computer (SBC) Reference Design

The OSD3358 SBC Reference Design is fully backward compatible with the BeagleBone Black Rev C design. There are design differences that are explained in the OSD3358 Power Design and Clocks sections. See the following links for OSD3358 SBC Reference Design details:

OSD3358 SBC Reference Design Schematics
OSD3358 SBC Reference Design Files

3.2 Differences from BeagleBone Black Rev C to OSD3358 SBC Reference Design

The OSD3358 integrates a number of the key components from the BeagleBone Black Rev C. This section focuses on components that are eliminated when migrating from the BeagleBone Black Rev C to the OSD3358 SBC Reference Design.

BeagleBone Black Rev C Schematics Sheet 1:

> Eliminate the TPS65217C PMIC and TL5209 LDO, and all discrete components associated with these devices. The discrete components associated with the TPS65217C PMIC and TL5209 that are included in the OSD3358 are defined in the OSD3358 Datasheet.
>
> **Important:** See the OSD3358 Power Design section for additional notes on the 3.3V power rails and differences between the BeagleBone Black Rev C design and the OSD3358 SBC Reference Design.

BeagleBone Black Rev C Schematic Sheet 3:

> **Important:** See the Clock section for recommended design changes from BeagleBone Black Rev C and the OSD3358 SBC Reference Design.

BeagleBone Black Rev C Schematic Sheet 5:

> Eliminate all passive discrete AM3358 power supply components. The discrete components required for the AM3358 power supply that are included in the OSD358 are defined in the OSD3358 datasheet.
>
> **Important:** See the OSD3358 Internal Power Supplies section for additional details on implementing an OSD3358 power supply.

BeagleBone Black Rev C Schematic Sheet 7:

> Eliminate the DDR and all discrete VDDS_DDR components.

3.3 Minimal BeagleBone OSD3358 Design

If your application does not require all the BeagleBone Black peripherals, many can be eliminated without impacting Linux. A minimal OSD3358 base platform that boots a BeagleBone Black Linux distribution consists of the following components:

- OSD3358 SIP, reset circuit and power supply
- MicroSD card for development, eMMC for production
- USB PC (Client) port – used as a power source and Linux console
- ID EEPROM – optional

- Boot configuration for MicroSD card

See the Reset section for notes on migrating from a development platform to a final product. When designing your base platform be careful to understand all the OSD3358 power design notes, constraints and warnings in the OSD3358 Power Design section.

During development a microSD card can be used to boot from and contain the file system. Once your development is complete and you have a better understanding of your specific application's space requirements, migration to an eMMC device for production is suggested. See the eMMC/microSD section for additional details. If your base platform does not include a microSD card, then you can boot over UART0 or USB0 for development and use the eMMC for the file system. There is a boot configuration that allows for booting from UART0 or USB0 (USB PC/Client port) if no boot information is found on MMC1 (eMMC) or MMC0 (microSD). See the Boot Configuration section for details.

A USB PC (Client) port is also suggested for development. See the next section for additional USB PC (Client) port usage considerations. The USB PC port can be removed for production if your application has no functional requirement for it.

The ID EPPROM is not required to boot Linux, but not including it requires BeagleBone Black U-Boot modifications that are covered in the Board ID EPPROM and U-Boot section.

If your base platform design is consistent with the BeagleBone Black Rev C and OSD3358 SBC Reference Design, then the standard BeagleBone Black Linux distribution will boot. Any deviation from the OSD3358 SBC Reference Design may require Device Tree and U-Boot modifications.

3.4 Application Specific Requirements

In addition to an OSD3358 base platform that boots Linux, you will most likely have additional application specific requirements. The following is a list of major BeagleBone Black components that may need to be added to your OSD3358 base platform to meet your application specific requirements:

- If a USB PC (Client) port is required, you can utilize it as a Linux console, a USB storage drive or/and a web server port. As a web server port you can provide an application configuration and/or a user interface that runs from a host's browser over USB, similar to the BeagleBone's start-up web server (see bone101 for additional details). See the USB section for design considerations.

- If a USB host port is required, then you may have to add a 5V supply to source enough power for your USB peripherals. See the Input Power section and USB section for design considerations.

- If remote access over Ethernet is required, then you will need to add either a wired Ethernet port or WIFI to your base platform. See the LAN section for design considerations.

- If your application requires a display, the OSD3358 supports both LCD and HDMI display types. See the Display section for LCD and HDMI design considerations.

Besides the BeagleBone Black standard components, you probably have your own components that need to be added. There are a few approaches to adding your application specific hardware to the base platform depending on your requirements:

- If your application specific hardware is currently implemented with a cape (or multiple capes), and you have no requirement to retain the expansion capability provided by the BeagleBone Black cape expansion connectors for your final product, you can integrate your cape designs directly into the base platform. This approach creates a single board without the expansion connectors. In this case you can continue to utilize your original Device Tree Overlays to enable the AM3358 peripherals used by your application and to setup the Pinmux, as if you were still using a cape. Since the Device Tree Overlay utilizes the expansion port pins designators, it may be desirable to retain the pin designators with the signal names in your schematics to make correlation between your schematics and the Device Tree Overlay easier.

- If your application requirements include system expandability provided by the cape expansion ports, then you will want to add the Beaglebone Black expansion connectors to your base platform. If you remain consistent with the OSD3358 SBC Reference Design your base system will be compatible with all BeagleBone Black Capes. This approach will most likely require the use of Device Tree Overlays to configure the system for existing capes or support your custom cape. See the BeagleBone Black SRM for details on cape design.

  You can also include both the cape expansion connectors and additional application specific functions directly on the base platform. In this case you can have peripherals, such as the LCD interface, drive both a base platform function and the expansion connector. You can also remove signals from the expansion ports for use on the base platform only, but this would limit compatibility with existing capes. In cases where a function is added to the base platform, the Device Tree or Device Tree Overlay (if the function is shared with the expansion port) will most likely need to be updated.

- If there is no requirement for your application to support further cape expansion, then you can reuse all the BeagleBone Black expansion connector pins and any OSD3358 functional signals not used by the base platform to implement your application specific requirements. Be careful you don't use OSD3358 signals that are shared by other functions you may need in your final product (like eMMc port 1 signals). With this approach you can retain utilization of the Device Tree Overlay (per the first bullet), but use of any OSD3358 peripheral signals that are not accessed through the expansion port designators may require Device Tree modifications.

Additional details can be found in the Expansion Capes and Pinmux sections.

Regardless of the approach, if the interface to your application is through GPIO or one of the Beaglebone Black supported standard communications types (UART, I2C, …), you may be able to implement control of your device through a user space program, rather than a device driver. A device driver is only required if your application needs interrupt service, has real-time deadlines or requires a higher level of security than a user space program provides.

Once your base platform is working, you can then eliminate the components that were only required for development, eliminate power options and test points not used by your application, and transition from a microSD to a final eMMC component.

4 General Application Considerations

The following sections provide additional information on a variety of standard subjects that should be considered for any design.

4.1 OSD3358 Power Design

The OSD3358 SIP includes a TPS65217C PMIC, a TL5209 LDO and all the discrete components required to support the OSD3358's internal and external power needs from any combination of three power sources; a 5 VDC supply (also referred to as the AC supply), a 5 VDC USB PC (Client) port or a Li-Ion Battery.

4.1.1 3.3VDC Supply Notes

There are two differences between the BeagleBone Black Rev C and OSD3358 SBC Reference Design power supplies. The OSD3358 SBC Reference Design includes a 3.3V Clamp circuit and the OSD3358 SBC Reference Design utilizes the VDD_3V3B power rail for all external components and the VDD_3V3A power rail for OSD3358 internal use only**.**

3.3V Clamp:

The OSD3358 SBC Reference Design schematic (bottom of sheet 2) has two 3.3V clamping circuits on VDD_3V3A and VDD_3V3B that are not in the BeagleBone Black design. These clamping circuits ensure the voltage difference between the 1.8-V and 3.3-V rails are never greater than 2 volts. The clamping circuit is required due to the following warning in the AM335x Data Sheet:

> "The 3.3-V IO power supplies may be ramped simultaneously with the 1.8-V IO power supplies if the voltage sourced by any 3.3-V power supplies does not exceed the voltage sourced by any 1.8-V power supply by more than 2 V. Serious reliability issues may occur if the system power supply design allows any 3.3-V IO power supplies to exceed any 1.8-V IO power supplies by more than 2 V."

See TI document **SLVU731A** "Use of a Clamping Circuit for Simultaneous Ramp Down" section for more details on the clamping circuit.

3.3V Usage Notes:

In the OSD3358 SBC Reference Design schematic (sheet 5) VDD_3V3A is derived from the OSD3358's VDDSHV voltage supply, provided by the TPS65217C's LD04, and VDD_3V3B is derived from the OSD3358's SYS_VDD1_3P3V voltage supply, supplied by the internal TL5209. VDD_3V3A (VDDSHV) is for OSD3358 internal use only. VDD_3V3B is used as a general purpose 3.3V board and cape supply.

Warning: The BeagleBone Black Rev C design uses VDD_3V3A as a reference voltage for a few pull-ups. In the OSD3358 case use of this power rail as a reference voltage or to power any external components is highly discouraged.

4.1.2 OSD3358 Output Power Supplies

The following notes apply to OSD3358 SBC Reference Design schematic sheet 5:

The balls of each rail (VDD_CORE, VDD_MPU, VDD_RTC, VDDSHV, VDDS_DDR and VDDS_PLL) are tied together internally and therefore are not required to be tied together externally and are normally left as no connects. It is recommended that for debug purposes each internal rail be made observable on a test point.

OSD3358 SBC Reference Design Power Supplies:

- SYS_5V (SYS_VOUT) – This is the switched output of the OSD3358's TPS65217C. This supply is typically used to source additional voltage regulation circuits in your design if one of the provided OSD3358's TPS65217C LDO outputs does not provide the required voltage or current requirements of your application.

- VDD_3V3B (SYS_VDD1_3P3V) – This is the output of the OSD3358's TL5209. It's dedicated to power external 3.3V circuitry.

- VDD_3V3AUX (SYS_VDD2_3P3V) - This is the LDO2 output of the OSD3358's TPS65217C. This is a second 3.3V supply that can be used in your application to power external circuitry if SYS_VDD1_3P3V is not sufficient for your application's requirements. In the BeagleBone Black and OSD3358 SBC Reference Design's it's simply used for the power LED. If the Power LED flashes once, the OSD3358's TPS65217C has encountered a problem that caused it to shut down.

- VDD_1V8 (SYS_VDD_1P8V) - This is the LDO3 output of the OSD3358's TPS65217C. It's a shared power source between internal and external components.

If your design uses one of these supplies to power more circuitry than what is in the BeagleBone Black Rev C or OSD3358 SBC Reference Design, you will need to check the current load against the OSD3358 Datasheet's "Recommended Operating Conditions" table.

The following OSD3358 Output Power Supplies not used in the OSD3358 SBC Reference Design can be used in your design:

- VDD_RTC (SYS_RTC_1P8V) – This is the LDO1 output of the OSD3358's TPS65217C. It provides power to the AM3358's RTC and is the first rail to come up in the power sequence. It can be used in your application for additional 1.8V power if SYS_VDD_1P8V supply is not sufficient to meet your application's requirements. If your application requires operation in RTC-only mode, external use should be limited to circuits that must be powered while all other power rails are in SLEEP mode. SYS_RTC_1P8V is an always on power domain, but if the OSD3358's TPS65217C is put in SLEEP mode there is a current limit of typically 1 mA. (See the AM3358 RTC section for additional RTC information).

- VDD_ADC (SYS_ADC_1P8V) - This is the LDO3 output of the OSD3358's TPS65217C filtered for analog applications. It is used to power the OSD3358's AM3358's ADC and can also be used to power external analog components.

There is additional information on power rails and the power-up process in the BeagleBone Black SRM.

4.1.3 OSD3358 Input Power Supplies

The OSD3358 input power balls, VIN_AC, VIN_USB and VIN_BAT are connected directly to the corresponding power inputs of the OSD3358's TPS65217C PMIC. The TPS65217C selects the input power type that is used for the OSD3358 internal components and OSD3358 power output balls. See the TPS65217C Data Sheet's "Detailed Description" section for a complete description of all TPS65217C features. Also note that many of the TPS65217C's operating parameters can be modified and tuned for your application through the TPS65217C's I2C port.

If you are using a USB PC (Client) port to provide VIN_USB and your application includes a USB host port, you will need to take care that the total application power (including the power consumed by any USB host peripherals) does not exceed the 500 mA limit on the USB PC port power.

Warning: The absolute maximum ratings (voltage and current) for VIN_AC, VIN_USB and VIN_BAT can be found in the OSD3358 Data Sheet's."OSD335x Absolute Maximum Ratings" table. If these ratings are exceeded the OSD3358 SIP can be damaged. The recommended operating conditions can be found in the OSD3358 Data Sheet's "Recommended Operating Conditions" table.

Warning: VIN_AC, VIN_USB and VIN_BAT are the only OSD3358 power inputs. VDD_CORE, VDD_MPU, VDD_RTC, VDDSHV, VDDS_DDR and VDDS_PLL are all OSD3358 power output pins and should normally be left as no connects.

4.1.4 General OSD3358 Power Design Notes

The following are general notes for OSD3358 power supply design:

- The OSD3358 incorporates power supply decoupling caps for all internal components. Details can be found in the OSD3358 Data Sheet's "Passives" section. The OSD3358 SIP does not require any additional external decoupling capacitors.
- The OSD3358 Data Sheet's "Output Power" section provides a list of output power supply balls, their use internally to the OSD3358 SIP (if used internally), the voltage source (TL5209 or TPS65217C), and maximum current that can be provided at the OSD3358 balls. The OSD3358 SBC Reference Design only uses a subset of these power supplies.
- The TPS65217x PMIC Datasheet's "Feature Description" section provides detail on power rail wake-up and power-up sequencing.
- The BeagleBone Black Rev C and OSD3358 SBC Reference Design both support the AM3358's RTC-only mode which utilizes the AM3358's internal RTC LDO. To modify the RTC operating mode see the AM3358 RTC section.

Note: The OSD3358 SIP does not pin out the TPS65217C PMIC's WLED Driver.

4.1.5 AM3358 Dual-Voltage Power Domains

The AM3358 has a set of dual-voltage IO power domains. In many cases the AM3358 ball map defines multiple signal options for a single ball, and in some cases a signal is dupliucated on multiple balls. In the AM3358 case this allows moving the signal from one IO voltage domain to another. The OSD3358 SIP does not support this feature and thus all AM3358 VDDSHV1-VDDSHV6 dual-voltage domains are supplied by the OSD3358's TPS65217C's LD04 at 3.3 volts.

4.1.6 Power Test Points

If your design requires test points for power supply checkout, these are the recommended power supply test points:

Ball Signal Name	Note
VIN_AC	External Input Power
VIN_USB	USB Input Power
VIN_BAT	Battery Input Power
BAT_VOLT	Battery voltage sense
BAT_TEMP	Battery temperature sense
VDD_CORE	Internal Power Domain
VDD_MPU	Internal Power Domain
VDDS_DDR	Internal Power Domain

VDDS_PLL	Internal Power Domain
VDDSHV	Internal Power Domain
SYS_VOUT	External Power Domain
SYS_VDD1_3P3V	External Power Domain
SYS_VDD2_3P3V	External Power Domain
SYS_RTC_1P8V	Shared Power Domain
SYS_VDD_1P8V	Shared Power Domain
SYS_ADC_1P8V	Shared Analog Power Domain

4.1.7 Battery Power Source/Charger

The OSD3358's internal TPS65217C provides a linear battery charger for single-cell Li-Ion and Li-Polymer batteries. See TPS65217C Data Sheet's "Battery Charger and Power Path" section for battery charging and power supply prioritization details.

Octavo Systems recommends only the use of single-cell Li-Ion and Li-Polymer batteries with a built-in NTC (negative temperature coefficient) thermistor. In this case the battery will have a sense (temp) terminal that should be connected to the OSD3358's BAT_TEMP signal. If the battery has a built-in protection circuit and no sense terminal, then a 10K ohm resistor can be used in place of the thermistor. BAT_VOLT is the OSD3358 TPS65217C's BAT_SENSE input. BAT_VOLT is normally connected to the OSD3358's VIN_BAT.

Note 1: The OSD3358's TPS65217C PMIC will supply supplemental current from the battery if USB or AC maximum current is exceeded. See the TPS65217C Data Sheet (section 9.3.9) for details.

Note 2: Per a warning in the TPS65217C Datasheet (section 9.3.9.1) it is not recommended to use both the AC and USB inputs of the OSD3358 when the battery is absent.

Warning: Use of a 10K ohm resistor to defeat the OSD3358's TPS65217C charging safety circuits for battery packs that do not include a thermistor or built-in protection circuits is highly discouraged.

4.1.8 Power Button

The power button allows for powering-up the TPS65217C from either the OFF or SLEEP states, and can be used to power cycle the system (switch must be held for 8 seconds). It also provides the ability to alert the processor before powering down to provide an orderly shutdown. For more detail on the OSD3358's TPS65217C power button input (PMIC_IN_PB_IN) see the TPS65217C Datasheet's "Push Button Monitor (PB_IN)" section. Also see the Power Button discussion in the BeagleBone Black SRM's "Power Button" section.

For AM3358 Linux power management features see:

- Linux Core Power Management User's Guide
- AM335x Linux Power Management User Guide

Please note the previous link includes the minimum Linux revisions required for each of the power management features.

Note: The power button on the BeagleBone Black and OSD3358 SBC Reference Design is meant for evaluation purposes as a simple means to wake up the TPS65217C. If your application puts the TPS65217C in the low power state, you will need to design a wakeup mechanism that is appropriate for your application.

4.1.9 AM335x Schematic Checklist Power Sections

There is additional information in the AM335x Schematic Checklist that covers unused power rails, low power considerations, and general power notes that apply to OSD3358 designs. For OSD3358 specific conflicts with the AM335x Schematic Checklist see the following warnings and notes.

Warning: The AM335x Schematic Checklist indicates that VDDA_ADC should be tied to ground if the analog interface is not used. In the OSD3358 case VDDA_ADC (SYS_ADC_1P8V ball) is an output voltage source supplied by the OSD3358's internal TPS65217C LDO3. Connecting to ground will most likely damage the OSD3358 module.

Note 1: The AM335x Schematic Checklist recommends small shunt resistors on the voltage rail paths to allow for power consumption measurements. In the OSD3358 case these resistors will only work on the input voltage balls, VIN_AC, VIN_USB and VIN_BAT. Also, keep in mind that in the OSD3358 case VDD_MPU, VDD_CORE, VDDS_DDR, VDDSHV, VDD_RTC cannot be used to measure the current loads internal to the OSD3358. If shunt resistors are used on OSD3358 output voltage balls, you will only measure the current load of your applications circuitry external to the OSD3358.These resistors, if used, must be sized properly for the current load of the supply.

Note 2: The AM335x Schematic Checklist indicates that in cases where the AM3358 USB Phy is not used, the AM3358 USB power terminals (VDDA1P8V_USB and VDDA3P3V_USB) should be connected to a proper power rail and software used to power down the USB Phy. Since the OSD3358 supplies this power internally, you will only need to power down the unused USB Phys with your software.

4.2 Clocks

The BeagleBone Black crystal circuits have some minor deviations from the AM335x Data Sheet's "Clock Specifications" section. It's recommended the AM335x Data Sheet Clock Specifications be followed exactly, including the recommendation for optional R_{bias} (not populated) and R_d (0 ohms) resistors in prototypes to evaluate production crystal components. You may need to use these resistors to tune the oscillator circuit, which can be highly sensitive to layout.

It is also acceptable to utilize a LVCMOS Digital Clock Source as defined in the AM335x Data Sheet's "Clock Specifications" section.

Note 1: Per the AM335x Silicon Errata Advisory 1.0.30 and AM335x Schematic Checklist Clocking section, the OSD3358 has internally connected the OSC0_GND and OSC1_GND balls to the OSD3358's digital ground to improve noise immunity.

Note 2: The 32KHz crystal for OSC1 is only required if your application uses the AM3358's RTC timer feature. If the AM3358's RTC is disable (see the AM3358 RTC section for details) the OSD3358's OSC1_IN and OSC1_OUT balls are no connects and OSC1_GND connected to digital ground.

See the AM335x Schematic Checklist Clocking section for additional notes on clocks.

4.3 AM3358 Real Time Clock (RTC)

The BeagleBone Black Rev C and OSD3358 SBC Reference Design are both implemented to support the AM3358's RTC-only mode. For this mode RTC_KALDO_ENN is pulled down (see sheet 5 of the OSD3358 SBC Reference Design schematic) to enable the AM3358's RTC power domain supplied by an AM3358 internal LDO.

In the AM335x Schematic Checklist RTC section there is a table that shows the correct hookup for each of the AM3358's RTC operating modes. The OSD3358 internally connects SYS_RTC_1P8V to the AM3358's VDDS_RTC pin. This is an always on supply provided by the OSD3358's TPS65217C LDO1. The OSD3358 also includes internally the 1 uF decoupling capacitor on CAP_VDD_RTC for RTC-only mode, but this will not prevent the other modes from operating properly. The following hookup notes should be followed to modify the OSD3358's AM3358's RTC mode:

RTC-Only (keeps RTC power on while the AM3358 is powered down):

- CAP_VDD_RTC – leave ball open
- RTC_KALDO_ENN – Pull down to digital ground (VSS).
- RTC_PWRONRSTN – Connect to OSD3358's PMIC_OUT_LDO_PGOOD ball.
- PMIC_POWER_EN – Connect to OSD3358's PMIC_IN_PWR_EN ball
- EXT_WAKEUP – Connect to OSD3358's PMIC_OUT_NWAKEUP ball

RTC timer functionality but no RTC-only mode (saves power usage by the RTC LDO):

- CAP_VDD_RTC – connect to OSD3358's VDD_CORE
- RTC_KALDO_ENN – Pull up to SYS_RTC_1P8V
- RTC_PWRONRSTN – Connect to OSD3358's PMIC_OUT_LDO_PGOOD ball.
- PMIC_POWER_EN – Connect to OSD3358's PMIC_IN_PWR_EN ball
- EXT_WAKEUP – Pull down to digital ground (VSS).

RTC disabled (RTC'd digital core is powered but held in reset):

- CAP_VDD_RTC – connect to OSD3358's VDD_CORE
- RTC_KALDO_ENN – Pull up to SYS_RTC_1P8V
- RTC_PWRONRSTN – Pull down to digital ground (VSS).
- PMIC_POWER_EN – Connect to OSD3358's PMIC_IN_PWR_EN ball
- EXT_WAKEUP – Pull down to digital ground (VSS).
- See the Clocks section for a note on the RTC OSC1 signals when the RTC is disabled

Note: PMIC_POWER_EN is an output from the AM3358's RTC and is only required to be connected to the OSD3358's PMIC_IN_PWR_EN ball regardless of the RTC mode.

4.4 Reset

The BeagleBone Black and OSD3358 SBC Reference Design reset circuit consists of the following components:

- A SN74LVC1G07 open drain line driver from PMIC_OUT_PGOOD to eliminate random glitches on the SYS_RESETn signal (OSD3358's WARMRSTN ball).
- A 2.2uF capacitor to maintain an active SYS_RESETn for sufficient time to reset all the OSD3358's internal components.
- A reset switch.

The BeagleBone Black and OSD3358 SBC Reference Design also utilize the SYS_RESETn signal to reset the Ethernet Phy, and this signal is also connected to the expansion connector for use by BeagleBone capes.

The OSD3358 has three functional reset signals (PWRONRSTN, RTC_PWRONRSTN, and WARMRSTN) that all correspond to AM3358 reset IO pins. See the AM335x Data Sheet and the AM335X TRM's "Reset Management" section for reset details.

When migrating to your own product design, it may be a requirement to eliminate the reset switch. As long as all other reset circuitry is retained, the switch can be eliminated from the design, in which case a power cycle is required to reset the OSD3358.

Note: The SN74LVC1G07 open drain circuit must be retained for OSD3358 designs.

Note: The AM335x Schematic Checklist Warm Reset section indicates that WARMRSTn should be used as an output or an input, but should not be used for both at the same time because of a debounce issue. Additional detail can be found in the AM335x TRM's "External Warm Reset" section, which indicates the issue only effects reset circuits that utilize an external push button circuit due to inadequate debounce time.

Note: The BeagleBone Black and OSD3358 SBC Reference Design utilize a 2.2uF capacitor to maintain an active SYS_RESETn for sufficient time to reset all OSD335x components.

4.5 Boot Configuration

Both the BeagleBone Black and the OSD3358 SBC Reference Design utilize a switch to select one of two boot sequences. See the AM335X TRM's "SYSBOOT Configuration Pins" section for all SYSBOOT Configuration options. For a production OSD3358 design it may be desirable to eliminate the BeagleBone Black switch and select a single boot sequence. Each boot sequence can test up to four boot devices. Along with booting from the eMMC or SD card, the AM3358 can also boot over USB, UART, SPIO, Ethernet, NAND, or XIP devices. Each boot device has specific requirements defined in the AM335X TRM. It's common during development to boot over a peripheral (USB, UART, SPIO or Ethernet) and then switch to eMMC or microSD card booting once development is complete.

See the BeagleBone Black SRM for more details on its boot configuration, design, and default boot options.

Note 1: The BeagleBone Black and OSD3358 SBC Reference Designs share the SYS_BOOT signals with the LCD_DATA bus. The BeagleBone Black SRM documents that it's also possible to override the switch settings via the expansion bus, but recommends gating these signals with SYS_RESETn to insure these signals are removed from the expansion pins after reset. If your application requires a different boot design, and also uses the LCD_DATA bus for a display interface, care must be taken to size the SYS_BOOT and LCD_DATA bus loads properly and to eliminate reflections on the LCD_DATA bus that could adversely affect the display system. See the AM335x Data Sheet's DC Electrical Characteristics section for details on signal loading.

Note 2: See the AM335x Silicon Errata for boot usage notes and advisory issues (only those that affect revision 2.1 AM335x devices should be considered).

4.6 LEDs

If your application requires more LEDs than are provided by the BeagleBone Black and OSD3558 Reference Design, you can utilize any OSD3358 GPIO to control an LED as long as you utilize an appropriate transistor/buffer to drive the LED current. If your application does not require all the BeagleBone Black LEDs, the GPIOs may be reused for other functions.

See the BeagleBone Black SRM for more details on its User and Power LEDs.

4.7 Communications

The BeagleBone Black and OSD3358 SBC Reference Design support identical communication types provided by the AM3358. These include I2C, UART, SPI, USB, and Ethernet. The following sections provide notes on the BeagleBone Black and OSD3358 SBC Reference Design implementations and links to additional documentation specific to each communication type.

4.7.1 I2C

In the BeagleBone Black and OSD3358 SBC Reference Design the I2C0 bus is used for communication with the TPS65217C PMIC, HDMI Framer and the EEPROM. I2C1 and I2C2 are connected to the BeagleBone Black expansion connectors. See the BeagleBone Black SRM for details on I2C expansion connector signal mapping.

See additional design notes in the AM335x Schematic Checklist I2C section.

I2C electrical data and usage information can be found in the “I2C” sections of the AM335x Data Sheet and AM335X TRM.

Note: The OSD3358 does not connect the AM3358’s I2C0 bus to the TPS65217C PMIC I2C bus internally. This connection must be made externally between the OSD3358’s I2C0 and PMIC_IN_I2C balls.

4.7.2 UART

The BeagleBone Black and OSD3358 SBC Reference Design utilize a 6-pin header for UART0. The design includes an SN74LVC2G241 buffer that isolates the UART’s IOs during power-down and protects the OSD3358 from serial peripherals running at voltages higher than 3.3Volts (up to 5.5 volts). In addition to UART0 there are 5 additional AM3358 UARTs that are connected to the BeagleBone Black expansion connectors. See the BeagleBone Black SRM for details on UART expansion connector mapping.

If your application requires a specific serial port type (such as RS-232, RS-422, RS-485, …) an adapter (such as a MAX232) that provides the correct voltage levels and drive current is required. You can also utilize a UART for USB communication with a FTDI USB bridge device such as the FT230X.

UART electrical data and usage information can be found in the “Universal Asynchronous Receiver Transmitter (UART)” sections of the AM335x Data Sheet and AM335x TRM.

4.7.3 SPI

The OSD3358 has two SPI ports provide by the AM3358. In the BeagleBone Black and OSD3358 SBC Reference Design SPI0 is connected to the expansion connector and SPI1 is connected to the expansion connector and the TDA1998 (HDMI Framer) to provide a serial audio port (see the HDMI section for more details). SPI1 can only be used over the expansion connector or for a SPI function in your own design if your design does not require those signals for a HDMI audio port. See the BeagleBone Black SRM for details on SPI expansion connector mapping.

SPI Electrical data and usage information can be found in the “Multichannel Serial Port Interface (McSPI)” sections of the AM335x Data Sheet and AM335x TRM.

4.7.4 USB

The BeagleBone Black and OSD3358 SBC Reference Design include both a PC (Client) and Host USB ports. The design of both USB ports utilizes TPD4S012 USB ESD devices to provide ESD protection of the USB data pins.

- USB Host – The host port (USB1) is USB 2.0 HS compatible and provides up to 500mA of power for hosted devices (WIFI dongles, usb drives, …). The actual amount of power available to a host device may be less than 500mA, like in the case the BeagleBone power is supplied by the USB PC port. To protect the PMIC, limit the current provided, and provide additional ESD protection for the USB power pin, a TPS2051 is utilized to provide USB host power.

- USB PC (Client) – The PC port (USB0) can be used as the Linux boot console, a USB storage drive and/or a web server port. It provides power from the host PC USB port to the PMIC.

See the BeagleBone Black SRM for details on USB operations.

The OSD3358 USB ports can be configured as host or client ports. The BeagleBone Black and OSD3358 SBC Reference Design have simply chosen USB0 as the client port and USB1 as the host port. If your design requires changing a USB port type (from host to client or client to host) the Device Tree will need to be updated. If you change the USB design, see the AM335x Schematic Checklist USB section for details and notes. USB usage information can be found in the AM335x TRM’s “Universal Serial Bus” section. For AM3358 specific detailed USB Layout guidelines see TI document SPRABT8.

Note 1: If your application requires booting from USB, see the AM335x Silicon Errata usage note section 3.1.3. This note indicates that PCB's required to support USB boot must be designed to use the default DATAPOLARITY USB feature.

Note 2: See the AM335x Silicon Errata advisory 1.0.33 and 1.0.34 that document additional USB issues.

Note 3: The AM335x Schematic Checklist indicates that in cases where the AM3358 USB Phy is not used, the AM3358 USB power terminals (VDDA1P8V_USB and VDDA3P3V_USB) should be connected to a proper power rail and software used to power down the USB Phy. Since the OSD3358 supplies this power internally, you will only need to power down the unused USB Phys which can be accomplished from the Device Tree.

4.7.5 LAN

The BeagleBone Black and OSD3358 SBC Reference Design implements a MII and MDIO interface to the LAN8710A Ethernet Phy. See the BeagleBone Black System Reference Manual (SRM) for Ethernet implementation details.

If your application requires a different Ethernet Phy or a Gigabit EtherNet Phy, see the AM335x Data Sheet's "Ethernet Media Access Controller (EMAC) and Switch" section for EMAC and Switch electrical data and the AM335x TRM's "Ethernet Subsystem" chapter for GMII and RGMII interface details.

The OSD3358's AM3358 supports two EtherNet MACs and a switch. In the BeagleBone Black and OSD3358 SBC Reference Design MAC 1 is used for the wired port and MAC 2 is not used. Note that many of the MAC 2 interface signals are not connected to the BeagleBone Black expansion connectors. If your application requires a second Ethernet port, a second Ethernet Phy must be added to your design. The BeagleBone Black Device Tree includes entries for both MACs with MII ports. If you change the port type or Phy type, Device Tree modifications will be required.

If your application requires support for WIFI, most WIFI modules include a MAC and therefore typically would not utilize the OAS3358's MMI/RMII/RGMII MAC interfaces. In the TI WL18xx WIFI module case a MMC port is utilized for host communication, a UART is used for communications with other subsystems the module may support (such as Bluetooth), and a few GPIOs are required. See the WL18xx Platform Integration Guide for details.

Note: The AM335x Schematic Checklist Ethernet section suggests the inclusion of series termination resistors on the MII/RMII/RGMII interface. The LAN8710A device utilized in the BeagleBone Black and OSD8838 Reference Design has specific MII termination requirements that were followed for both designs. See the LAN8710 Schematic Checklist for details. It's recommended you follow all the schematic checklist guidelines for the Ethernet Phy device your application utilizes.

4.8 eMMC/microSD

The BeagleBone Black and OSD3358 SBC Reference Design utilize a 4GB 8-bit eMMC 4.5 standard device and a 4-bit microSD Card connector. During development you may have utilized a larger microSD card to expand the space available in the eMMC device. For production it may be desirable to consolidate to a single larger eMMC card. If your application requires a larger eMMC device, devices are available up to 128 GB

The BeagleBone Black and OSD3358 SBC Reference Design utilize the MMC1 interface as an 8-bit port for the eMMC while MMC0 is utilized as a 4-bit port for the microSD card interface. MMC0 was selected for the 4-bit microSD port to avoid a pinmux conflict with the Ethernet signals. See the BeagleBone Black System Reference Manual (SRM) for all eMMC and microSD implementation details, use of MMC1 signals on the expansion connectors and potential conflicts during the boot process.

If your application requirements deviate from those of the BeagleBone Black and OSD3358 SBC Reference Design, see the AM335X TRM's "Multimedia Card (MMC)" section for additional details on the AM3358 MMC interfaces. If your application requires MMC port modifications from the BeagleBone Black and OSD3358 SBC Reference Design implementation, the Linux Device Tree and U-Boot pinmux settings (see U-Boot's board/ti/am335x/mux.c) may need to be updated with the new MMC characteristics.

Note 1: The AM335x Schematic Checklist MMC section calls for a termination resistor on the MMCx_CLK signals. This termination resistor is not included in the BeagleBone Black or the ODS335x Reference Design, so if your layout deviates from the BeagleBone Black Rev C/OSD3358 SBC Reference Design layout it's recommended to include this termination resistor in the design.

The BeagleBone Black U-boot board/ti/am335x/mux.c file sets RXACTIVE to a 1 for the mmc0_clk and mmc1_clk signals per the AM3358 Schematic Checklist note.

Also per the AM335x Schematic Checklist all eMMC RST, CLK CMD and DATA signals should be pulled up to the 3.3V power rail.

Note 2: There is a MMC IO restriction not defined in the Pinmux Utility. See the AM335x Silicon Errata's "Pin Multiplexing: Valid IO Sets and Restrictions" section for usage details.

Warning: The AM335x Datasheet indicates support for both 1.8V and 3.3V MMC/SD devices. The OSD3358 only supports 3.3V MMC/SD devices directly. To use 1.8V devices voltage translation to 3.3V is required.

4.9 NAND

The BeagleBone Black platform does not utilize NAND memory, but the AM335x, it's Boot ROM, U-Boot, and Linux all support NAND SLC and MLC devices. NAND is less expensive than eMMC and microSD devices, but typically slower. When choosing a NAND device, design considerations include but are not limited to:

- Cost
- Compatibility with the AM335x GPMC NAND support
- Compatibility with the AM335x Boot ROM support
- Compatibility with the AM335x U-boot support
- NAND size requirements
- NAND read/write performance
- GPMC bus width (8 or 16 bits) and conflicts with other functions muxed with the GPMC signals

The AM335x General Purpose Memory Controller (GPMC) provides a wide range of ECC support (BCH4, BCH8, BCH16 and Hamming code for 8-bit and 16-bit devices). The AM335x Boot ROM narrows the supported ECC types to BCH8 and BCH16. U-Boot's most recent release only supports BCH8 ECC. TI's NAND driver provides support for 1-bit Hamming, BCH4, BCH8 and BCH16. See the Linux Core User's Guide NAND Driver ECC Schemes Support section for a list of ECC modes, Device Tree bindings, and Kconfig settings.

The AM3358's Boot ROM ECC can also be disabled with a pull-up on SYSBOOT[9] for support of NAND with internal ECC. See the Boot Configuration section for additional SYSBOOT information. See the AM335x TRM's Initialization section for NAND Boot ROM details.

Warning: When selecting a NAND device that Linux will be booted from and used as the Linux file system, the ECC for that device must be supported by the AM3358's Boot ROM, U-Boot and the Linux NAND device driver. In current BeagleBone Black Linux distributions BCH8 is the only ECC supported across all three.

Warning: U-Boot as provided by current Linux distributions does not provide support for BCH16 devices.

Warning: The OMAP NAND driver may need to be configured as a built-in module. See the Device Drivers section for more information on modules. Depending on your NAND type it may also be required to enable hardware ECC when configuring the NAND driver, and disable other driver features such as Software BCH ECC. Also note the NAND driver requires Device Tree modifications to select the specific ECC scheme. BeagleBone Black Device Tree modifications are also required to disable the eMMC device (mmc2) and enable the ELM (Error Location Module) and GPMC Device Tree nodes. See the Linux Core NAND User's Guide Configurations section for details.

Any NAND device selected for integration in an OSD3358 application must be identifiable by one of the following methods.

NAND Device Identification:

The AM335x Boot ROM first attempts to identify the NAND device characteristics using the ONFI signature. If the NAND device's parameter page does not have an ONFI signature, the AM335x's Boot ROM will read the device ID. If the device ID matches a supported device, the device parameters are extracted from the internal ROM code table. There is also a NANDI2C boot mode for use with NAND devices for cases where the previous two methods don't work. If the Boot ROM detects this boot mode it attempts to read the NAND geometry from an I2C EEPROM (using I2C0). Additional details on NAND identification can be found in the AM335x TRM's Memory Booting section.

NAND Device Integration:

For NAND integration details see the AM335x TRM's GPMC section. Also note that the AM3358's GPMC port is muxed with signals that are connected to the BeagleBone Black expansion connectors.

For additional information see the following links:

Linux Core NAND User's Guide
Raw NAND ECC
Managed NAND

4.10 Display

The BeagleBone Black and OSD3358 SBC Reference Design provide for both HDMI and LCD displays. The HDMI and LCD interfaces share the LCD_DATA bus. If your application requires a display, see the next two sections for details. If your application does not require a display, the LCD_DATA bus signals can be redefined through the pinmux to select other functional signals.

4.10.1 LCD

The BeagleBone Black and OSD3358 SBC Reference Design provide access to the 16-bit LCD interface through the cape expansion connectors. The LCD data pins are also used to read the boot configuration (SYS_BOOT[0:15]) at power-up and for the HDMI Framer. See the BeagleBone Black SRM for details on LCD expansion connector mapping, conflicts and potential loading issues.

If your application requires an LCD interface, and you are not using an existing cape, see the AM335x Schematic Checklist's LCD section and the AM335x Silicon Errata's "LCD: Color Assignments of LCD_DATA Terminals" usage note for proper LCD pinouts. Also, LCD expansion capes utilize a Device Tree overlay. If your application requires an integrated LCD panel (in place of the BeagleBone Black HDMI Framer), Device Tree modifications are required. The Sitara Linux LCDC Porting Guide EVM section has examples for TI's AM335x EVM.

For applications that utilize a touch screen see AM335x Schematic Checklist Touchscreen section for notes on the touch screen's analog interface that may apply to your design.

Note 1: See the AM335x Silicon Errata advisories 1.0.27 and 1.0.28 for LCDC issues and AM335x Silicon Errata advisories 1.0.31 and 1.0.32 for Touch Screen Controller (TSC) issues.

4.10.2 HDMI

The BeagleBone Black and OSD3358 SBC Reference Design HDMI interface utilizes a TDA19988 HDMI Framer (Transmitter) that is protected with ESD devices that are designed specifically for high-speed HDMI applications. The TDA19988 is interfaced to the OSD3358 through the LCD_DATA, I2C0 and SPI1 interfaces. The SPI1 balls are pinmux selected for the McASP (Multichannel Audio Serial Port) and are used as the audio interface between the AM3358 and the TDA19988. See the BeagleBone Black SRM for HDMI design details.

The TDA19988, the ESD devices and HDMI connector can be eliminated if your application does not include a HDMI requirement and the LCD_DATA and SPI1 OSD3358 signals reused for other functions. These pins are also connected to the BeagleBone Black cape expansion connector.

U-Boot is used to setup the SPI1 pinmux settings for HDMI audio (also see the Pinmux section for U-Boot pinmux details). If HDMI is not a requirement or you change the HDMI audio design, the U-Boot file may need to be updated.

Also see the TI Linux LCDC Porting Guide HDMI section for information on BeagleBone Bone Black HDMI Device Tree support. If you modify the HDMI Framer (Transmitter) device type when migrating to your own application or the interface to the OSD3358, the Linux Device Tree may need to be updated with the new HDMI characteristics. Also, if your application does not include a HDMI device, you may need to remove or modify the Device Tree's HDMI components for your applications display type.

4.11 Board ID EEPROM and U-Boot

U-Boot reads the Board ID EEPROM to determine the specific board U-Boot is running on. When migrating to your own application from the BeagleBone Black or OSD3358 SBC Reference Design, before you can boot a BeagleBone Black Linux distribution, the ID EEPROM on your card must be programmed (see discussion below on eliminating the Board ID EEPROM). If your design does not deviate from the BeagleBone Black or OSD3358 SBC Reference Design, then you can use the existing BeagleBone Black EEPROM contents available at:

https://github.com/beagleboard/image-builder

You can also use U-Boot to read the existing BeagleBone Black ID EEPROM and then modify the board id structure contained in the EEPROM to create a unique board id. This is normally

only required if you modified the BeagleBone Black's U-Boot code to match your application requirements.

See the U-Boot section for additional information on U-Boot usage.

The BeagleBone Black U-Boot id structure is defined in uboot/board/ti/am335x/board.h, and the BeagleBone Black board level configuration code (for I2C, DDR, Voltage and frequency OPP setup, …) is in uboot/board/ti/am335x/board.c. The initial BeagleBone Black pinmux settings are in uboot/board/ti/am43xx/mux.c. Keep in mind that these are the initial settings required to boot Linux, and that additional pinmux changes (See the Pinmux section) are made in the BBB Device Tree or a Device Tree overlay for expansion capes.

Note: The OSD3358 SBC Reference Design is backward compatible with BeagleBone Black Rev C EEPROM programming. This means the OSD3358 SBC Reference Design is backward compatible with any BeagleBone Black Linux distribution.

The ID EEPROM is not mandatory, but eliminating it from your base platform design will require U-Boot modifications. There are two ways to remove the EEPROM dependency from the BeagleBone Black U-Boot code's board.c; you can hardcode the board id header (fastest appraoch) or you can remove all the EEPROM header data checks (most readable approach). If you eliminate the ID EEPROM, you may need to employ some other method of determining version differences for future releases.

4.12 Expansion Capes

The BeagleBone Black can accept up to four cape expansion boards simultaneously. If your application requires a cape that is operational with up to three other caps, then to avoid conflicts you will want to design your application to the cape standards provided in the BeagleBone Black SRM. The BeagleBone Black cape design only exposes a subset of AM3358 signals to the expansion connectors, and the specific signal selected is dependent on the pinmux (see the Pinmux section). See the BeagleBone Black SRM for details on expansion connector signal mapping.

If you create your own cape then you will most likely also have to provide a Device Tree overlay file that describes your cape. BeagleBone Black compatible expansion capes also normally include an I2C EEPROM (must be on I2C2 in the address range of 0x54 to 0x57), that provides information such as the name of the cape, cape version, part number, and other relevant run-time information. The information required to load the correct Device Tree Overlay for a specific cape is read from the I2C EEPROM by the Cape Manager. If you plan to support multiple versions of a cape, it's recommended to include the I2C EEPROM in your cape design. See the BeagleBone Black SRM "Cape Board Support" section for details. Device Tree Overlays can also be loaded, configuring the pinmux and enabling the device, or unloaded (disabling the device) at run-time through the bone_capemgr /sys interface. If your cape design does not include an I2C EEPROM, this method can be employed to load your Device Tree Overlay. Also, see the Device Tree Overlay section for an overview and a link to additional information.

Both the eMMC and HDMI devices utilize Device Tree Overlays since the MMC1 and LCD interfaces are shared between the base platform and the expansion ports. You may need to disable the HDMI or Audio functions in the HDMI Device Tree Overlay if their pin usage conflicts with your cape's pin usage, assuming your application does not use HDMI.

If your application does not include the expansion connectors, then you can avoid the BeagleBone Black cape requirements and Device Tree overlays, but may need to modify the BeagleBone Black Device Tree to support any modifications to the base design.

Also note that when you design your application specific card, if you integrate a cape design into your card but eliminate the expansion connectors, the Device Tree overlay still defines the pins used by the cape in terms of the expansion connector pins. If you want to utilize OSD3358 signals not available to the expansion connector pins in your design, you may need to modify the BeagleBone Black Device Tree to support your modifications or replace the BegaleBone Black Device Tree with an application specific Device Tree.

Note: In cases where your application requires a reduced mechanical footprint for the expansion connectors, It's possible to design an application specific expansion port that is consistent with the BeagleBone Black Device Tree Overlays, but only utilizes a subset of the BeagleBone Black expansion port signals.

4.13 Pin Mux

Many of the AM3358 signals exposed at the balls of the OSD3358 support multiple functions that are enabled through the AM3358's pinmux control registers. For example, the signal UART0_TXD can also be used for SPI, I2C, PWM, and PRU functions. Along with the pin mode (function), users can also select and enable pull-up/pull-down resistors, enable the pin as a receiver (input), and select the slew rate for the pin. Keep in mind these selections must be compatible with your board's electrical design.

Also when you transition to an application specific design, If you are not using a BeagleBone Black design feature, in many cases (but not all), those unused signals may have additional functionally your design can utilize that can be enabled through the pinmux.

If you are making significant changes from the BeagleBone Black and OSD3358 SBC Reference Design, use of TI's Pinmux Tool is recommended to the verify valid IO sets have been selected. See the AM335x Schematic Checklist Pinmux section for additional information and a link to TI's Pinmux tool.

If changes are required to a signal's pinmux definition, there are multiple places in a BeagleBone Black Linux distribution this may be defined depending on the function:

- BeagleBone Black U-Boot uboot/board/ti/am43xx/mux.c
- BeagleBone Black Device Tree

- BeagleBone Black Device Tree overlay (for expansion capes)

Note 1: When using the Pinmux Tool remember to configure the AM3358 voltages supplies in terms of the OSD3358. See the OSD2258 Data Sheet's "OSD335x Passives" table for a list AM3358 power supplies and the corresponding OSD3358 supply.

Note 2: There are some IO restrictions not defined in the Pinmux Utility. See the AM3358 Silicon Errata section 3.1.5 for usage details.

4.14 Debug

4.14.1 JTAG

Both the BeagleBone Black and the OSD3358 SBC Reference Design include a JTAG connector for debug purposes. In both designs the placement of the connector is such that signal routes are minimized (less than a few inches). If it's necessary to move the relative placement of the connector and thus increase the routing distance of the JTAG signals or if your application includes multiple JTAG devices, please see TI's XDS Target Connection Guide for detailed instructions that are dependent on your routing distance and load requirements.

Note: The BeagleBone Black and the OSD3358 SBC Reference Design utilize the CTI 20-pin JTAG header that is not populated on production boards. Part numbers for the TI 20-pin connector can be found in the XDS Target Connection Guide Header Part Numbers section

4.14.2 Clocks

Per the AM335x Schematic Checklist General Debug section the XDMA_EVENT_INTR0 and XDMA_EVENT_INTR1 signals can be configured (through the pinmux) as clock outputs (CLKOUT1 and CLKOUT2) that can be brought out to test points for debug purposes. CLKOUT1 is a replica of the OSC0 input clock, CLKOUT2 can be configured to output the OSC1 input clock or one of four other internal clocks. See the AM335x Data Sheet's "Output Clock Specifications" and "Output Clock Characteristics" sections for details.

In the BeagleBone Black and OSD3358 SBC Reference Design CLKOUT1 is used as the clock input to the HDMI Framer (CEC clock). CLKOUT2 is connected to the expansion connector for use in capes.

4.14.3 Power

See the Power Test Points section for recommended voltage test points.

4.15 OSD3358 PWB Layout

The OSD3358 SIP eliminates many of the AM3358 and DDR PWB layout requirements.

Achieving a 2-layer PWB stack-up is possible when not using any OSD3358 high-speed signals. In this case a PWB stack-up with a signal layer and a split power/ground layer can be used.

Most high-speed designs will require at a minimum a 4-layer stackup with full ground and power planes on the inside two layers. When utilizing OSD3358 high-speed signals, see the following layout guidelines:

- USB - TI's AM335x and AM43xx USB Layout Guidelines
- Ethernet – See the manufacturer's schematic checklist and layout guidelines. The BeagleBone Black Rev C and OSD3358 SBC Reference Design utilize a LAN8710 that has a schematic checklist and layout guidelines.
- HDMI/LCD – See the manufacturer's schematic checklist and layout guidelines.

Also see TI's Sitara Layout Checklist (ignore the DDR sections) for general PWB Layout guidelines if layout guidelines for your specific high-speed components are not available.

Also note that many AM3358 signals are assigned to the same OSD3358 ball locations. See the OSD3358 Pin Assignment and Application Differences from Texas Instruments AM3358 for details on differences between the two devices.

4.16 AM335x Silicon Errata

The only AM335x Silicon Errata usage notes and advisory notices that apply to the OSD3358 are those for revision 2.1, except for advisory 1.0.30 (see Note 2 below). Please examine these advisories carefully to determine if they apply to your application.

Note 1: The AM335x Schematic Checklist External Interrupt section references AM335x Silicon Errata Advisory 1.0.6. This Advisory does not apply to the OSD3358 since the module is implemented with AM3358 revision 2.1.

Note 2: Per the AM335x Silicon Errata Advisory 1.0.30 and AM3358 Schematic Checklist Clocking section, the OSD3358 has internally connected the OSC0_GND and OSC1_GND balls to the OSD3358's digital ground to improve noise immunity.

4.17 Software

This document references U-Boot, the Device Tree and Device Tree Overlays. The following sections provide links to additional documentation on these subjects.

4.17.1 U-Boot

The AM3358 has a built-in ROM based boot sequence (ROM based) that configures clocks, initializes the boot hardware, and copies U-Boot from the boot device to memory for execution. U-Boot in turn then boots Linux. For additional details on U-Boot and its use in the BeagleBone Black see the Board ID EEPROM and U-Boot section.

If your base platform is identical to a BeagleBone Black Rev C, ready to flash Linux distributions are available.

U-boot can also be used to flash the boot image. Each boot device has specific requirements and procedures:

- microSD/eMMC – If your base platform supports microSD and eMMC see the instructions at the BeagleBone Software Updates page.
- eMMC - To program and boot from eMMC without using the microSD card see the Sitara Linux Program the eMMC on Beaglebone Black.

For generic Texas Instruments U-Boot documentation see the AM335x_U-Boot User's Guide. This document includes information on using U-Boot with different peripherals and media types. Even though this is a generic AM335x boot page based on the AM335X EVM, many of the processes are applicable to BeagleBone Black based systems.

There are alternatives to using U-Boot (beyond the scope of this document).

4.17.2 Device Tree

The device tree provides a structured description of static hardware to the Linux kernel and device drivers, abstracting application specific properties of the CPU (such as operating points) and peripherals (such as memory mapping, bus widths, interrupt assignments and the pinmux). The Device Tree is also used by Linux to associate peripherals with specific device drivers.

In the BeagleBone Black case the device tree is used to describe CPU and peripheral properties dedicated to the base platform. Device Tree Overlays are used to describe hardware that is shared through the expansion ports and can change at run-time.

See the Linux Device Tree documentation for general device tree information and Device Tree Usage.

4.17.3 Device Tree Overlays

As pointed out in the previous section, the Device Tree describes the static system properties of the CPU and peripherals to the kernel, and Device Tree Overlays are used to describe dynamic properties that can be changed at run-time as in the case of adding expansion capes. This makes the addition of new capes to a system, and combining multiple capes in a single system much easier since there is no requirement to modify the Device Tree. Each expansion cape simply requires a Device Tree Overlay. Device Tree Overlays can be loaded automatically, if the expansion cape includes an I2C EEPROM or loaded at run-time through the bone_capemgr /sys interface. See the Expansion Capes section for more details.

See the Introduction to the BeagleBone Black Device Tree for additional background and details on Device Tree Overlays.

4.17.4 Device Drivers

One of the major advantages to using a BeagleBone Black based Linux distribution is that device drivers are built-in for many of the AM335x subsystems. This provides significant savings, reducing development time and potentially maintenance costs. Many drivers also provide user space interfaces enabling implementation of non-real-time application code as a user space program or script, allowing you to take advantage of all the services provided by Linux.

If you start development with a BeagleBone Black, or if your OSD3358 application specific card is backward compatible with the BeagleBone Black, typically the quickest method to get started with the latest drivers and kernel image is to utilize a BeagleBone Black Debian pre-built image. The latest BeagleBone Black Debian image and the Getting Started Guide that has instructions for updating your BeagleBone Black's kernel image are at:

https://beagleboard.org/latest-images.

The pre-built kernel image includes a fixed set of device drivers. Some are built-in to the kernel and loaded at boot time, while others are built as dynamically loaded modules that are loaded after Linux boots (using Device Tree Overlays, insmod or modprobe utilities).

Removing drivers not used by your application, in most cases, will provide the greatest Kernel memory footprint reduction. To add or remove a device driver from a Linux distribution requires both the source for the Linux distribution and the proper toolchain. Once you have installed the toolchain and downloaded the kernel source, the Linux kernel must be configured prior to building the kernel. The result of configuring the kernel is a .config file specific to your system. Configuring the kernel includes selecting kernel options, features and drivers to be included (or excluded) as built-in or dynamic modules in your Linux build. There are multiple configuration tools and methods available, all of which are launched with "make". See your Linux distribution's README file's "CONFIGURING the kernel:" section for a list of available "make" configuration commands.

For manually configuring the kernel, the most commonly used configuration tool is menuconfig (see below for an example of launching menuconfig with "make"). Menuconfig provides a simple text based menu-driven user interface to navigate between kernel options, features and drivers. After saving your changes and exiting, menuconfig updates the .config file. It is not recommended that you attempt to manually edit the .config file because of dependency issues that can be difficult to resolve. You should also make a backup copy of your original .config file before you modify it. Once the kernel's .config file has been modified the kernel must be re-built, also using "make" (see below for an example). It is also a very good idea to have a bootable backup of your Linux distribution just in case a configuration modification causes Linux to not complete the boot process or crash.

BeagleBoardDebian provides instructions on installing and running Debian Linux built with the ARM EABihf (arm-linux-gnueabihf-) toolchain. See the BeagleBoardDebian's BeagleBone Black section for a link to the complete Instructions for generating the BeagleBone Black Debian kernel image from source on a Linux host machine through the use of a cross compiler. These instructions utilize a build script that is downloaded from a git clone. Once the repository has

been cloned, you must select the version of the kernel that is appropriate for your requirements. The command “git branch -r” can be used to generate a list of the kernel versions available for checkout. The build script (./build_kernel.sh) is then executed to clone the kernel source, install the toolchain, run menuconfig and build the kernel, modules and device tree binaries. To complete the process, you must also build U-Boot and download a root file system. Once you have built and downloaded all the required components, instructions are provided for setting up and copying the images to a microSD card from your Linux host. The uEnv.txt file must also be created and loaded onto the microSD card and the /etc/network/interfaces file updated per the instructions.

From the KERNEL directory you can modify the kernel configuration (add or delete drivers) with menuconfig, build driver modules, or build the entire kernel image with “make”. The kernel build script needs to be ran at least once to get the default BeagleBone Black Debian .config file and install the toolchain, before trying to use “make” manually. Also, the resulting build files are not copied to the “./bb-kernel/deploy” directory, as with the build script, but can be found in the “KERNEL/arch/arm/boot” directory. For example:

```
make ARCH=arm menuconfig        - Run menuconfig to create .config
make ARCH=arm modules           - build the driver modules
make ARCH=arm uImage            - build the kernel image
make ARCH=arm dtbs              - Build the device tree binaries
```

If you are building on a host Linux machine you will also have to set the CROSS_COMPILE environment variable either on the “make” command line or with the export utility to:

CROSS_COMPILE=../dl/gcc-linaro-*arm-linux-gnueabihf/bin/arm-linux-gnueabihf-

Additional build options that can be included on the make command line or as environment variables are defined in the kernel documentation directory at:

kernel/Documentation/kbuild/kbuild.txt.

Note: Many AM335x sub-systems have been ported from a previous generation OMAP platform. In these cases, the menuconfig elements will typically be referenced using the OMAPn naming convention.

4.17.5 Kernel Memory Footprint Reduction

There are two primary considerations to make when attempting to reduce the kernel footprint.

- Flash verses RAM footprint
- Reductions that affect performance

Since the BeagleBone Black Linux Kernel distributions support use of the platform as a single board computer the distribution’s kernel image must support all the board level subsystems, expansion capes, and plug-and-play components (such as USB devices). Also, there are many drivers that are built as dynamically loaded modules and only loaded at run-time if the hardware

is detected or if they are loaded manually. Eliminating drivers not used by your application that are built as dynamically loaded modules will have a significant effect on the flash footprint, but may not have a significant effect on the RAM footprint. If your application requires dynamic loading of driver modules, the RAM footprint may be reduced by stripping the modules of symbols if there are no external dependencies. Eliminating built-in drivers that are not used by your application will affect both the flash and RAM footprints. See the Device Driver section and the discussion below in the Driver Reduction topic for additional details.

Linux also has built-in debug and profiling capabilities that are generally enabled by default and are needed during application development but are normally not needed or desirable in final production systems. Eliminating the built-in debug and profiling support will have an impact on both the flash and RAM footprints. See the Debug Reduction topic below for more details.

Eliminating unused drivers and the debug capabilities will have very little effect on the performance of your application, although eliminating unused built-in drivers may improve the boot-time of the system. You can also reduce the flash and RAM footprints of your Linux system by modifying the build optimization level to optimize for size, but at some performance cost. This is only recommended in cases where eliminating unused drivers and disabling the debug and profiling capabilities did not achieve the footprint reduction required for your application. See the Optimization Levels topic below for more details**.**

The Kernel Size Tuning Guide provides instructions for measuring the kernel size. The previous recommendations will typically provide the greatest memory footprint reductions. If the previous recommendations are not sufficient to achieve your application's required memory footprint, the Kernel Size Tuning Guide's Kernel Configuration Options section provides additional settings that will reduce the kernel footprint. Not all recommendations in the Kernel Size Tuning Guide's Kernel Configuration Options section may be applicable to your specific application, so it's recommended you examine each option carefully and try them one at a time to verify the impact on your system. For example, CONFIG_MODULES cannot be disabled if your system loads any modules dynamically at runtime, like when using Device Tree Overlays at runtime or when using insmod or modprobe.

In all cases you will need the Linux distribution source and toolchain to re-build the kernel image after making configuration modifications. See the Device Drivers section for details on where to get the BeagleBoard Linux source which includes the menuconfig configuration tool referenced below.

Driver Footprint Reduction:

For specific applications, many subsystems that are required for a single board computer may not be required by your application and their drivers eliminated from your Linux build. In addition to eliminating unused drivers, in many cases, the RAM footprint of the drivers that are dynamically loaded can be reduced by eliminating the symbols that are retained by default.

The menuconfig kernel configuration tool allow you to build driver modules in one of three states:

- [ ] – Empty, will not be compiled or included in the kernel build (select this for drivers and configuration options not needed by your application.
- [M] – Module, build a dynamically loaded module. The module can be loaded on demand by the kernel or loaded manually with modprobe or insmod.
- [*] – Built-in, the module is built-in to the kernel image.

The bracket type ([], <>, {}, --) provides dependency information, which can limit which drivers can be built-in or built as a dynamically loaded module. See the menuconfig Symbols section for details.

Note: The organization of drivers and configuration options in menuconfig can change between kernel versions.

To modify driver support using menuconfig, use the arrow buttons to navigate to the “Device Drivers --->” configuration category and hit return to select.

```
Generic Driver Options --->
Bus Devices --->
…
```

The Device Driver configuration category presents a series of drivers and driver categories. Each driver can have options and each category (for example “Bus Devices --->”) can have additional drivers and support options. Each should be examined to determine which drivers and options should be included in your build. If a driver or option is not required for your application and the selection is not empty, modify it to empty (using the space bar) to eliminate the driver or option from your build. Once you have completed making modifications to a screen, use the left and right arrow buttons to navigate the menu at the bottom of the menuconfig screen to exit back to the previous screen or save the configuration modifications.

Warning: Do not remove the I2C driver since this is used for PMIC communications.

The RAM footprint of loaded modules can also be reduced by striping modules of debug symbols using the INSTALL_MOD_STRIP=1 make command line option.

Debug Reduction:

Using the menuconfig kernel configuration tool, you can disable support for the kernel’s printk messages, bug support, and profiling. Eliminating printk will reduce the kernel size by approximately 5 to 10%. See Debugging by printing for details on printk topics. It’s not recommended removing the printk messages or BUG() support until your application is ready for final testing and is normally the final step for tuning a kernel for size.

For basic instructions on using menuconfig see the Driver Footprint Reduction topic.

To remove printk support set the following menuconfig element to empty:

```
General Setup --->
    Configure standard kernel features (expert users) --->
        [] Enable support for printk
```

To remove kernel warning and bug support set the following menuconfig element to empty:

```
General Setup --->
    Configure standard kernel features (expert users) --->
        [] BUG() support
```

BUG() support uses printk and is used by the kernel and drivers to assert an error (kernel panic) or warning (logged and viewable by dmesg) when an event occurs the software does not know how to deal with. Generally, this is a feature used during driver development and may be disabled for final production builds to save space.

To remove data symbols from the kernel set the following menuconfig element to empty:

```
General Setup --->
    Configure standard kernel features (expert users) --->
        [ ] Include all symbols in Kallsyms
```

Note that when this option is not selected, Kallsyms only contains symbols of functions (text section symbols) for OOPs messages and backtraces. Selecting this option adds the data section symbols that are normally only needed for debug purposes.

To remove profile support from the kernel set the following menuconfig elements to empty:

```
General Setup --->
    [ ] Profiling support
    <>OProfile system profiling
    [ ] Kprobes
```

Optimization Levels:

Most Linux distributions use the –O2 optimization level by default, which will result in a larger footprint over –Os(optimize for size). If your application requires a reduced kernel footprint, and eliminating unused drivers and debug capabilities did not achieve your reduction goals, to utilize –Os optimization your application must also be compatible with the performance level achieved with –Os optimization.

To change the optimization level from –O2 to –Os set the following menuconfig element:

```
General Setup --->
    [*] Optimize for size
```

End of Document