SITARA™ ARM® PROCESSORS

BOOt camp

Texas Instruments

U-Boot & Linux Kernel Board Port

In this session we will cover fundamentals necessary to port a TI Linux-based EVM platform to a custom target platform. We will introduce the necessary steps needed to port the following components: secondary program loader, u-boot and Linux kernel.

LAB: http://processors.wiki.ti.com/index.php/Sitara_Linux_Training

July 2012

Creative Commons Attribution-ShareAlike 3.0 (CC BY-SA 3.0)

creative commons

You are free:

to Share – to copy, distribute and transmit the work

to Remix – to adapt the work

to make commercial use of the work

Under the following conditions:

Attribution – You must give the original author(s) credit

Share Alike - If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

With the understanding that:

Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.

Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.

Other Rights — In no way are any of the following rights affected by the license:

Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

CC BY-SA 3.0 License:
http://creativecommons.org/licenses/by-sa/3.0/us/legalcode

CC BY-SA

Texas Instruments

Pre-work Check List

- ❑ Installed and configured VMWare Player v4 or later
- ❑ Installed Ubuntu 10.04
- ❑ Installed the latest Sitara Linux SDK and CCSv5
- ❑ Within the Sitara Linux SDK, ran the setup.sh (to install required host packages)
- ❑ Using a Sitara EVM, followed the QSG to connect ethernet, serial cables, SD card and 5V power
- ❑ Booted the EVM and noticed the Matrix GUI application launcher on the LCD
- ❑ Pulled the ipaddr of your EVM and ran remote Matrix using a web browser
- ❑ Brought the USB to Serial cable you confirmed on your setup (preferable)

Agenda

- Board Port Overview
- Porting U-Boot to an AM335x Target
- U-Boot Board Port Labs
- Porting the Linux Kernel to a AM335x Target
- Linux Kernel Board Port Labs

Board Port Overview

Presentation Overview

- Goal is to gain an understanding of the components of a board port for both U-Boot and Linux
- The board or target portion is the last part of a three step method (Architecture/SOC/Target Board)
- Explain how the SDK will support board ports going forward

Linux Board Background Assumptions

- Already Familiar with :
  - SPL/U-Boot/Linux (☺)
  - SPL/U-Boot/Linux boot sequence
  - U-Boot/Linux build process (kernel configuration)
  - Minicom setup
  - Root File Systems
- Very limited time,
  - Really only have time to show the tip of the iceberg, not going to all inclusive or discuss every facet of board porting, this is a starting place
  - we'll have to take extended question/answer after the class in the foyer or later over email. (or in the bar…. You buy ☺)
- This information is good for today only…… always in flux…..
- What's presented here today may not be the only way of implementation
- Standard disclaimer of "You can and should use what others have done as a method on what to do to move forward"

Things not covered today..

- Not covering all of the board port steps
  - Limited time today, so we will just be focusing on the code portion of the port
  - Directory setup
  - Machine ID discussion
  - Makefile modifications
  - Git Setup
  - Other Processors

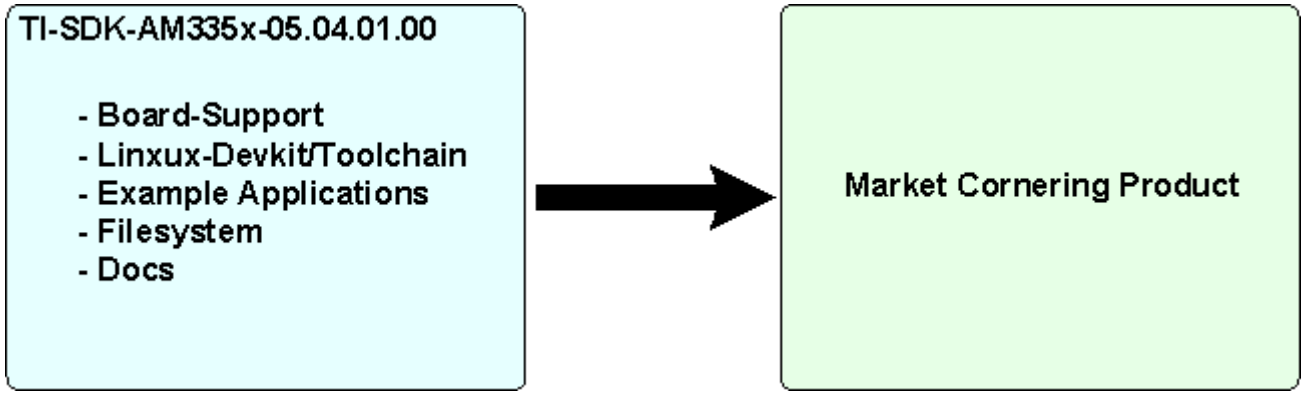
Linux Board Port Workshop Agenda

- The Mission/ “So…what’s a board port?”
- Look at the System Block Diagram of the target board being used
- Stages of a port
- Pin Mux Utility Tool Overview
- U-Boot Port
  - source tree
  - introduce the target board file
  - Perform two labs that use an already ported example (the code added by with each lab will be discussed)
- Linux Kernel Port
  - source tree
  - introduce the target board file
  - Perform four labs that use an already ported example (the source additions for each lab will be discussed)

The Mission

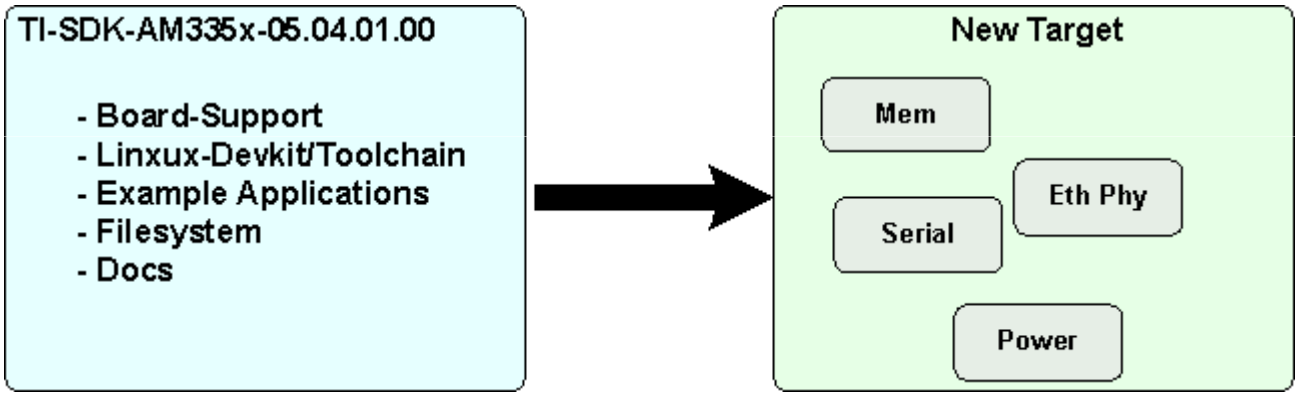
"Good Morning … the AM335x has been chosen as the processor for your new exciting market cornering product. Your job (no choice but to accept it ☺) is to get U-Boot and the Linux kernel running on this new platform as soon as possible.

To accomplish this you will take the board design from your HW team and use the AM335x EVM and accompanying Sitara Linux SDK and port U-Boot and the Linux kernel to your new Hardware. "

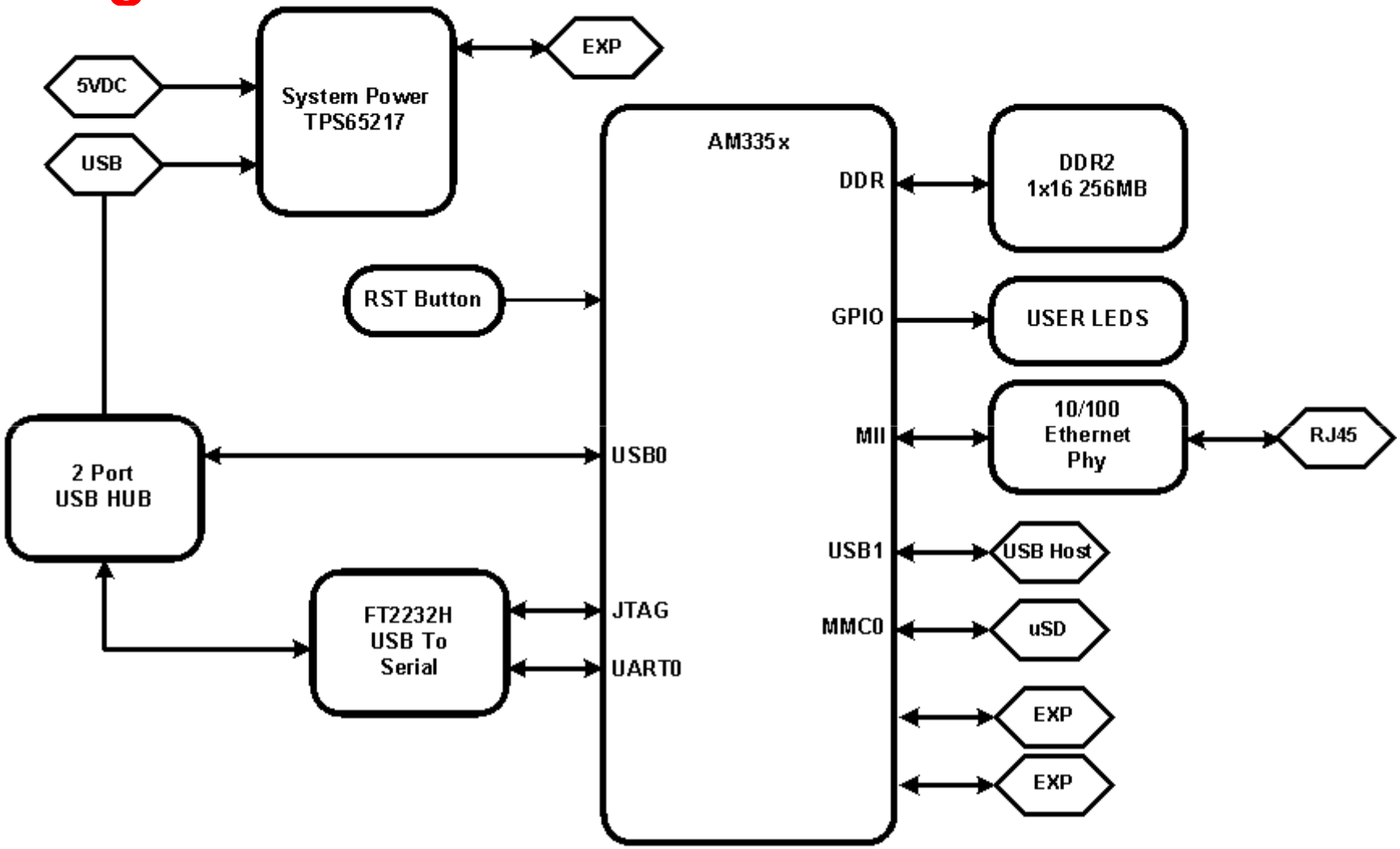


So….What's a board port?

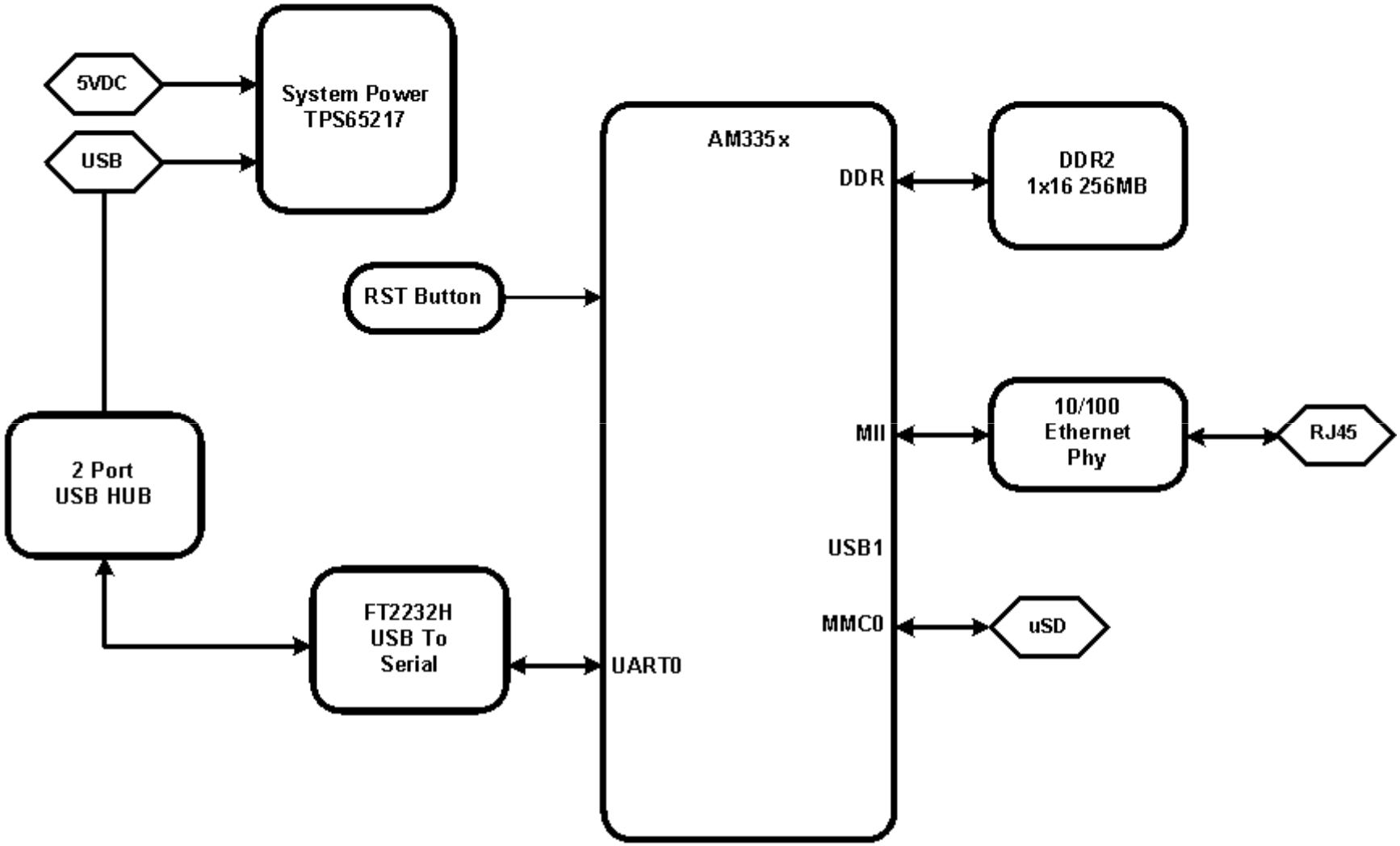
- It is taking the Sitara Linux SDK that is working on a known platform and moving it to a new target platform that is based on the same TI AM335x processor



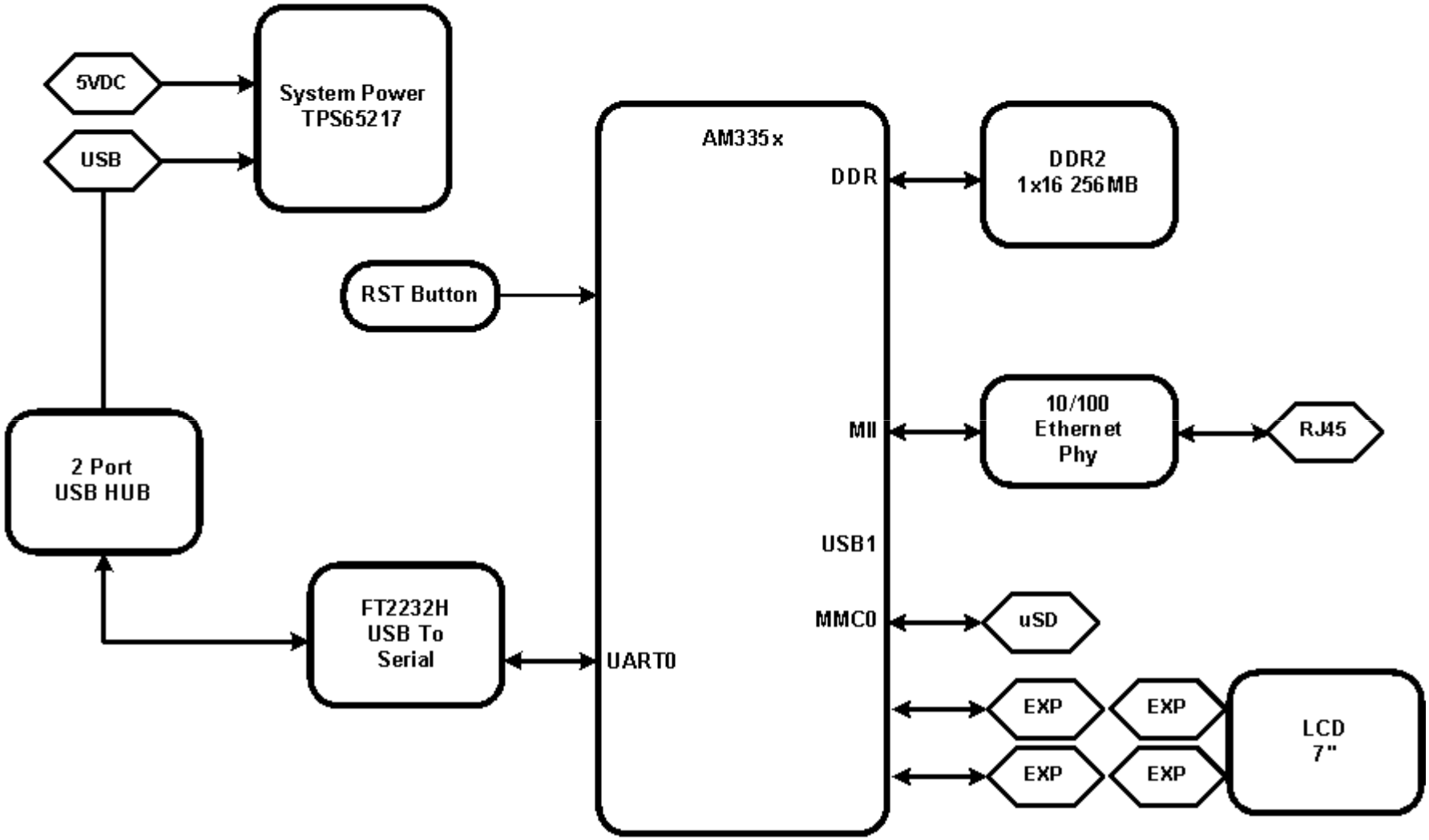
Target Board for this Exercise….
Beagle Bone



Target Board Port Configuration Example



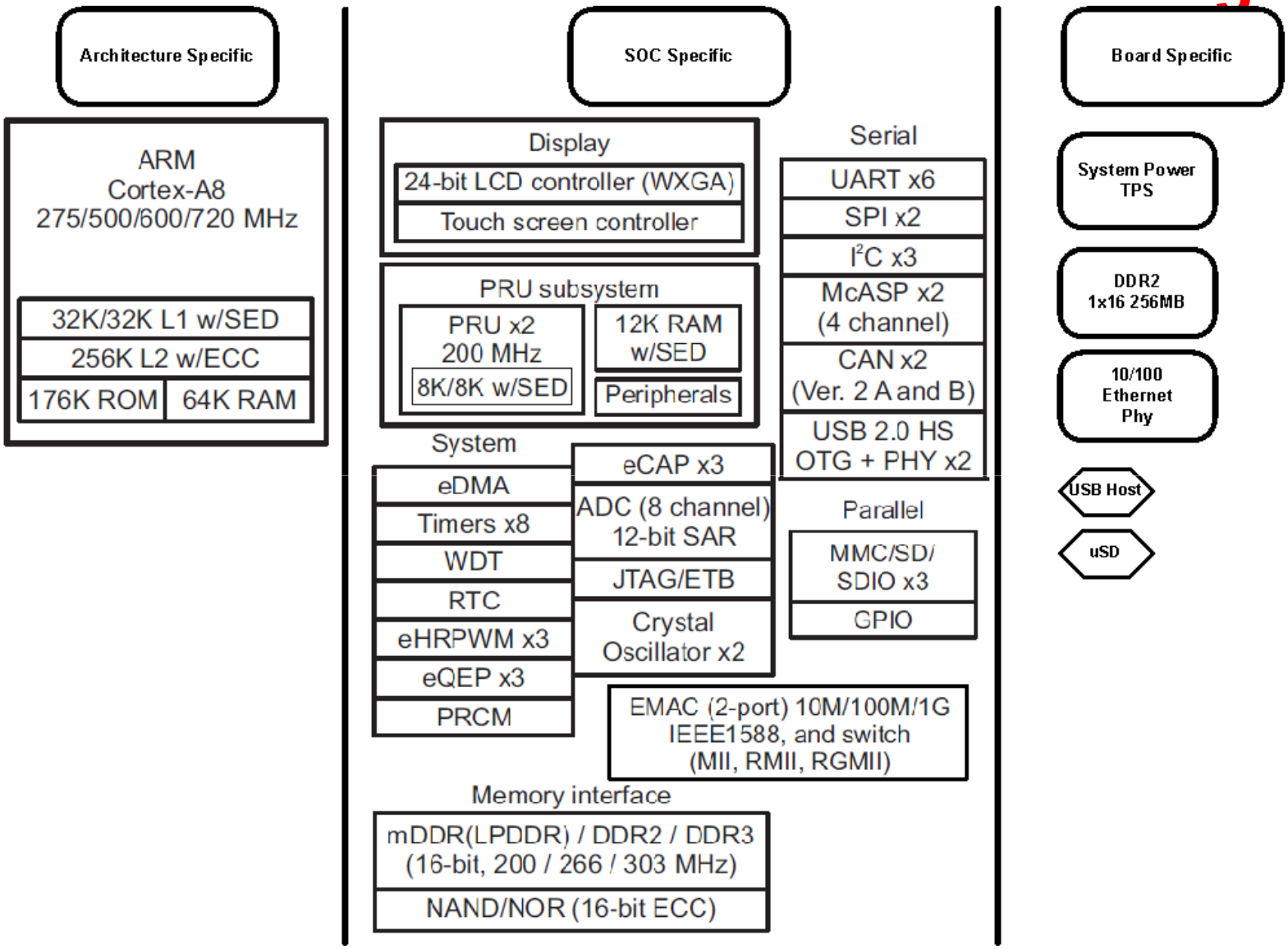
Will be adding an LCD to the system…..



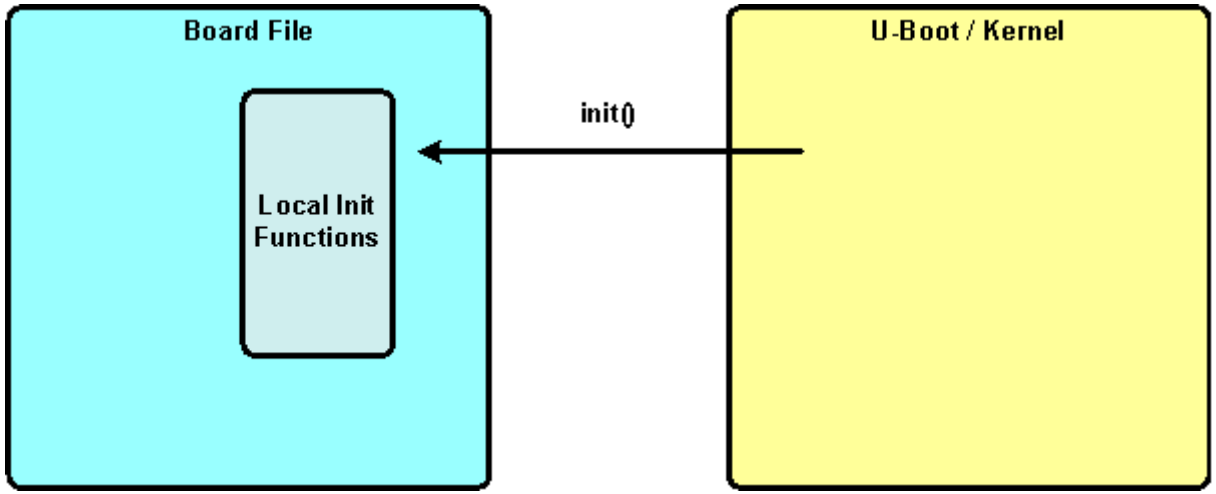
Board Port…. Tip of the iceberg

Used to show the balance of work necessary

Architecture vs. SOC vs. Board Porting



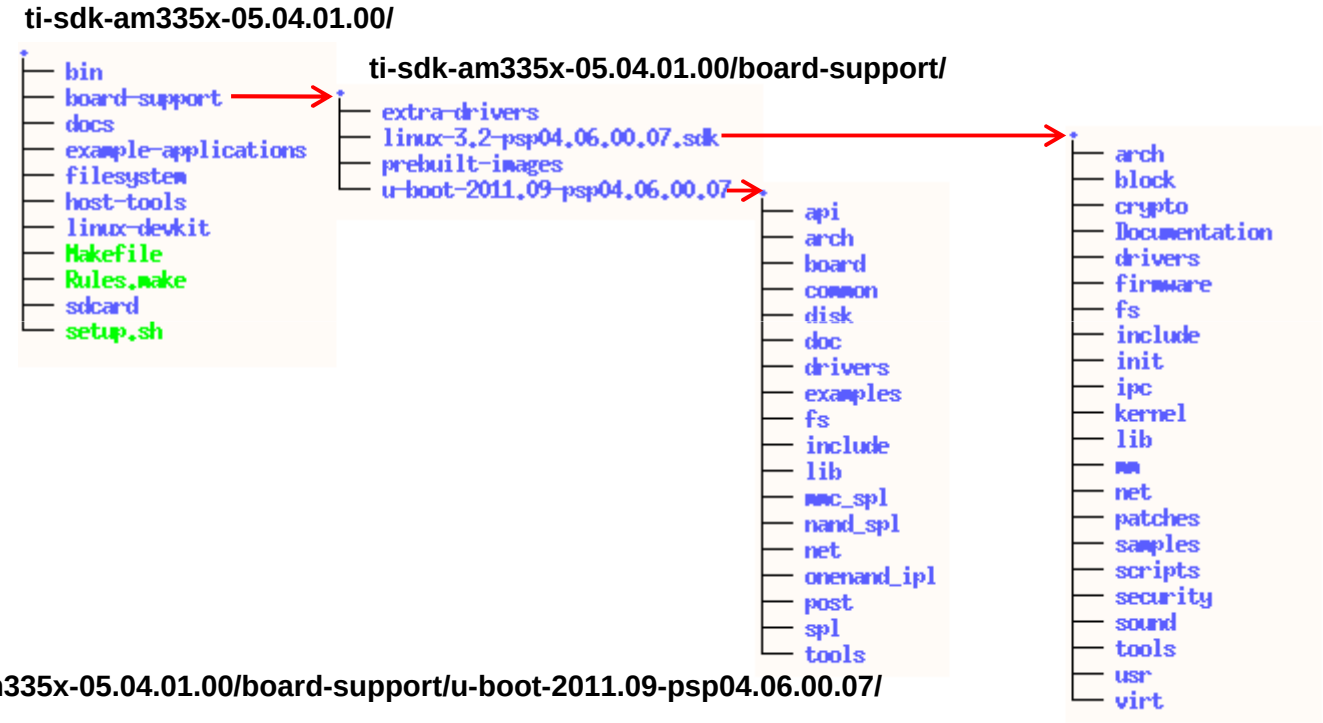
A Tale of Two Board Files



- Both U-boot and Linux follow a similar board file abstraction approach
- The Core Architecture is ported first
- The SOC supporting functions are ported next
- The last part to tie U-Boot/Kernel to the target is the Board file that defines “well known” initialization or entry functions that U-Boot and the Linux Kernel will call to handle “a priori” type board knowledge

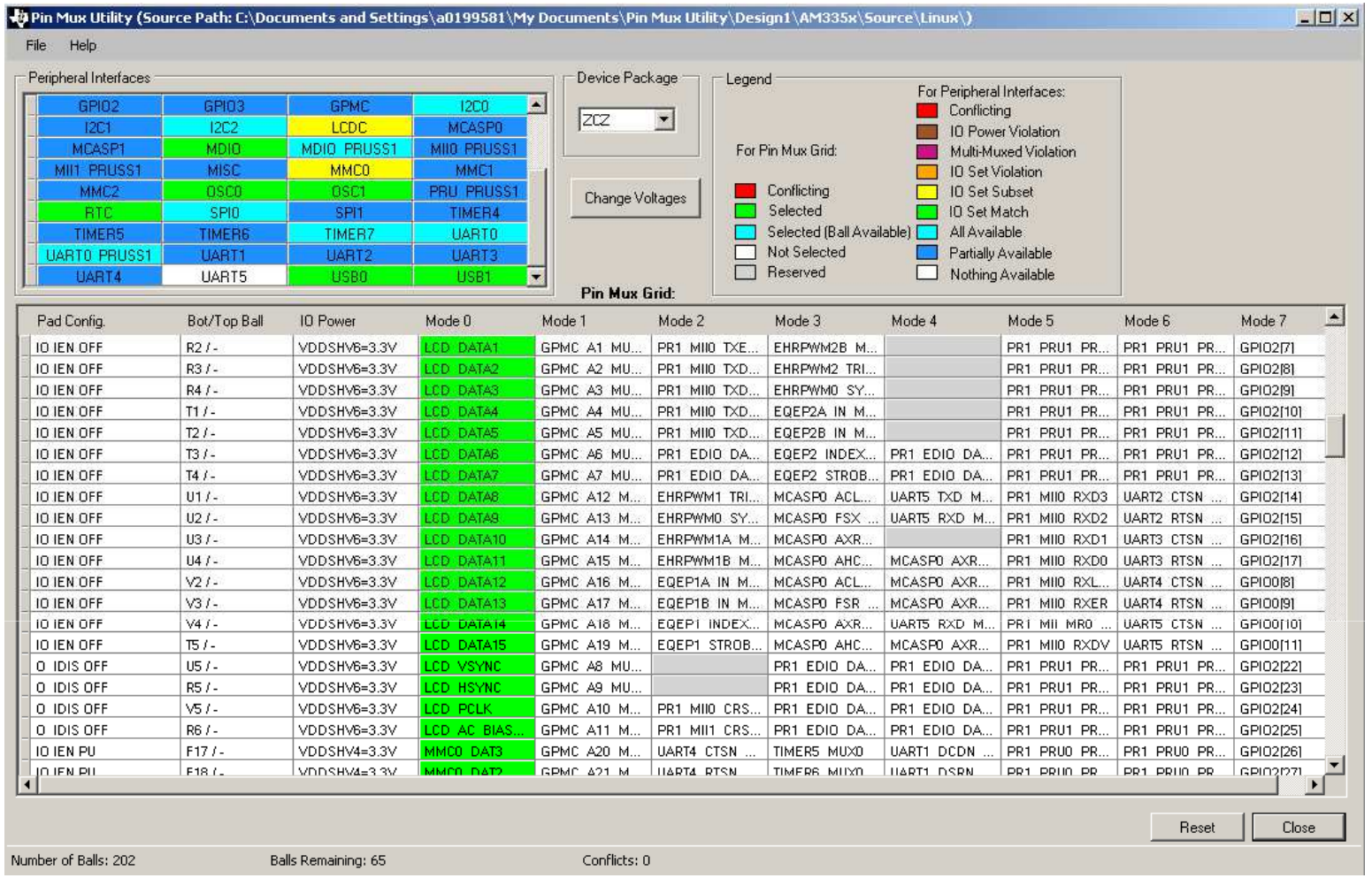
Where the U-boot and Kernel Sources are after TI-SDK-AM335x-05.04.01.00 installation

- Both the U-Boot and the Linux Kernel Sources are found in the installed TI-SDK-AM335x-05.04.01.00 directory



- Later in the presentation you will see references to just the specific sub-tree that has the respective source such as U-Boot or Linux

Pin Mux Utility



- GPIO Signals are “muxed” with peripheral interfaces. These can be configured into one of several modes either supporting the peripheral or remaining in a GPIO mode.

Selecting a mode using Pin Mux Utility

- Each Pin has a mode selection, using UART0 as an example here
- UART0 RXD Mode 0 is selected and GPIO 1.9 is de-selected

Pad Config.	Bot/Top Ball	IO Power	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
IO IEN OFF	E15 / -	VDDSHV6=3.3V	UART0_RXD	SPI1_CS0_M...	DCAN0_TX_...	I2C2_SDA_M...	ECAP2_IN_P...	PR1_PRU1_P...	PR1_PRU1_P...	GPIO1[10]
IO IEN OFF	E16 / -	VDDSHV6=3.3V	UART0_TXD	SPI1_CS1_M...	DCAN0_RX_...	I2C2_SCL_M...	ECAP1_IN_P...	PR1_PRU1_P...	PR1_PRU1_P...	GPIO1[11]

- UART0 RXD Mode 0 is selected and GPIO 1.9 is de-selected, notice Pad config changed too.

IO IEN OFF	E17 / -	VDDSHV6=3.3V	UART0_RTSN	UART4_TXD_...	DCAN1_RX_M...	I2C1_SCL_MU...	SPI1_D1_MUX0	SPI1_CS0_MU...	PR1_EDC_SY...	GPIO1[9]
I_IEN OFF	E15 / -	VDDSHV6=3.3V	UART0_RXD	SPI1_CS0_MU...	DCAN0_TX_M...	I2C2_SDA_MU...	ECAP2_IN_PW...	PR1_PRU1_P...	PR1_PRU1_P...	GPIO1[10]
IO IEN OFF	E16 / -	VDDSHV6=3.3V	UART0_TXD	SPI1_CS1_MU...	DCAN0_RX_M...	I2C2_SCL_MU...	ECAP1_IN_PW...	PR1_PRU1_P...	PR1_PRU1_P...	GPIO1[11]

- Utlitity helps find conflicts, two pins are simultaneously selected

IO IEN OFF	E17 / -	VDDSHV6=3.3V	UART0_RTSN	UART4_TXD_...	DCAN1_RX_M...	I2C1_SCL_MU...	SPI1_D1_MUX0	SPI1_CS0_MU...	PR1_EDC_SY...	GPIO1[9]
I_IEN OFF	E15 / -	VDDSHV6=3.3V	UART0_RXD	SPI1_CS0_MU...	DCAN0_TX_M...	I2C2_SDA_MU...	ECAP2_IN_PW...	PR1_PRU1_P...	PR1_PRU1_P...	GPIO1[10]
Conflict	E16 / -	VDDSHV6=3.3V	UART0_TXD	SPI1_CS1_MU...	DCAN0_RX_M...	I2C2_SCL_MU...	ECAP1_IN_PW...	PR1_PRU1_P...	PR1_PRU1_P...	GPIO1[11]

- Each Pin has a mode selection, using UART0 as an example here

IO IEN OFF	E17 / -	VDDSHV6=3.3V	UART0_RTSN	UART4_TXD_...	DCAN1_RX_M...	I2C1_SCL_MU...	SPI1_D1_MUX0	SPI1_CS0_MU...	PR1_EDC_SY...	GPIO1[9]
I_IEN OFF	E15 / -	VDDSHV6=3.3V	UART0_RXD	SPI1_CS0_MU...	DCAN0_TX_M...	I2C2_SDA_MU...	ECAP2_IN_PW...	PR1_PRU1_P...	PR1_PRU1_P...	GPIO1[10]
O_IDIS OFF	E16 / -	VDDSHV6=3.3V	UART0_TXD	SPI1_CS1_MU...	DCAN0_RX_M...	I2C2_SCL_MU...	ECAP1_IN_PW...	PR1_PRU1_P...	PR1_PRU1_P...	GPIO1[11]

- Pin Mux Utility User Guide

http://processors.wiki.ti.com/index.php/Pin_Mux_Utility_for_ARM_MPU_Processors_v2

Porting U-Boot to an AM335x Target

U-Boot Port Agenda

- What are the different stages of a Port
- Introduce the board file, where it fits in the Port Picture, where it is in the source tree
- What is the anatomy of the board file
- Introduce the Board File Template that can be used to port u-boot
- Labs Introduction

U-Boot Board Port Exercises and Source Links

- Link to the U-Boot Labs
  - http://processors.wiki.ti.com/index.php/Sitara_Linux_Training:_UBoot_Board_Port
- Link to the U-Boot Template Source tree (clone this tree)
  - git://gitorious.org/sitara-board-port/sitara-board-port-uboot.git
- PSP U-boot Repo
  - http://arago-project.org/git/projects/?p=u-boot-am33x.git;a=summary

SPL and U-Boot Builds

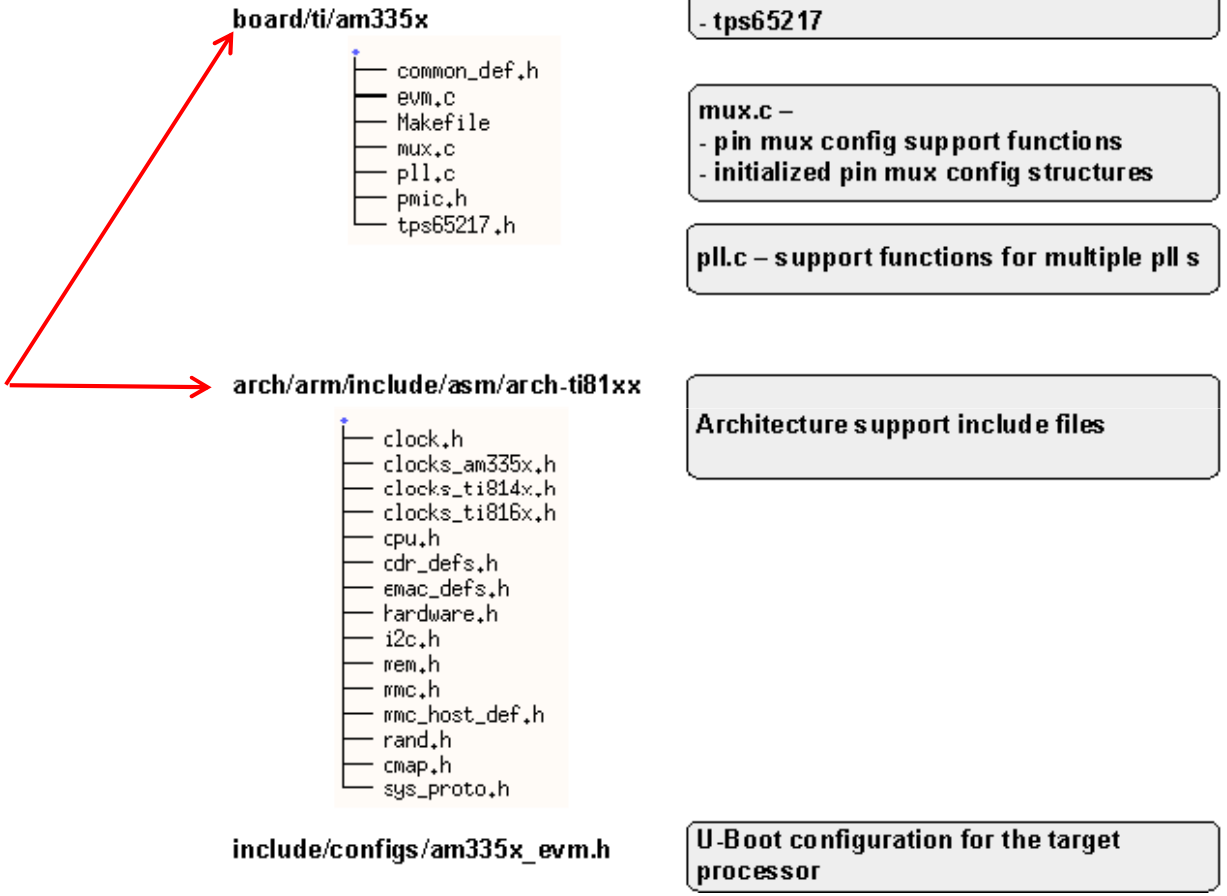
- “Dude……. Where’s my X-Loader?”
  - It has left the building…. Been replaced by SPL
- The same code base is used to build U-Boot (u-boot.img) and the SPL (still called MLO). Since the same code base is used pre-processor flags are used to isolate the code between the two builds. For example, you do not want the DDR and MPU clock init code in both builds. Also of merit is that one build yields both images.
- Below are examples of the pre-processor flags used:

#ifdef CONFIG_SPL_BUILD

#ifndef CONFIG_SPL_BUILD

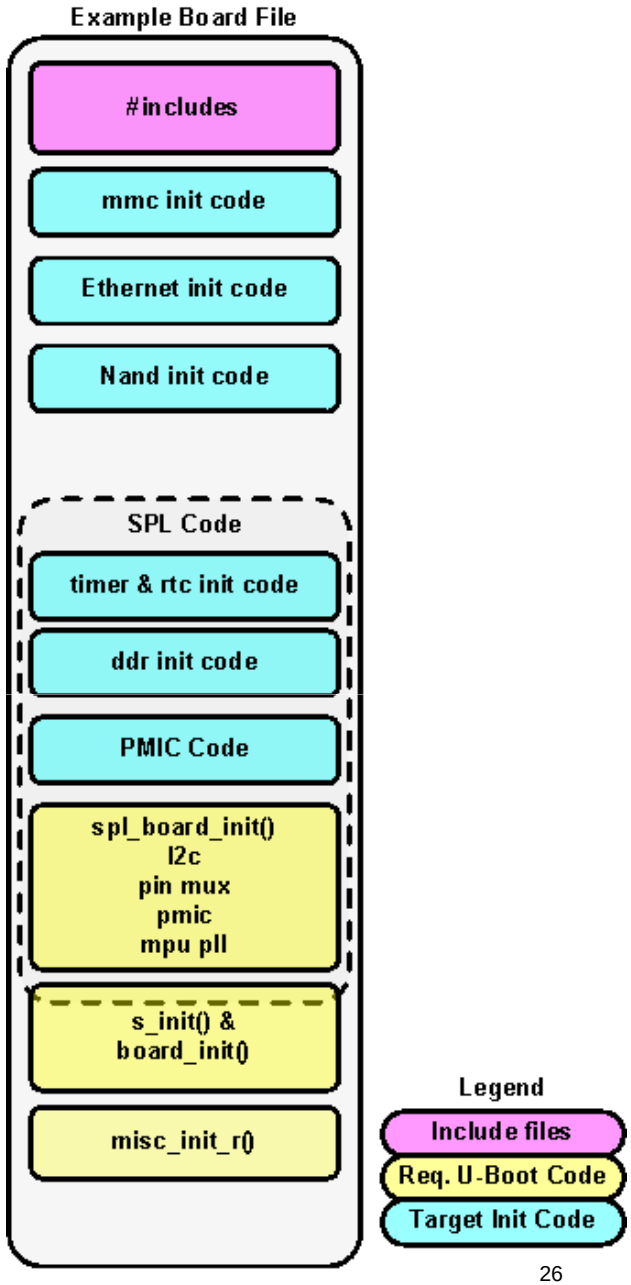
U-Boot Source Directory

- Using the existing am335x source directory
- The developer will be concentrating on one source directory and for the most part one include directory



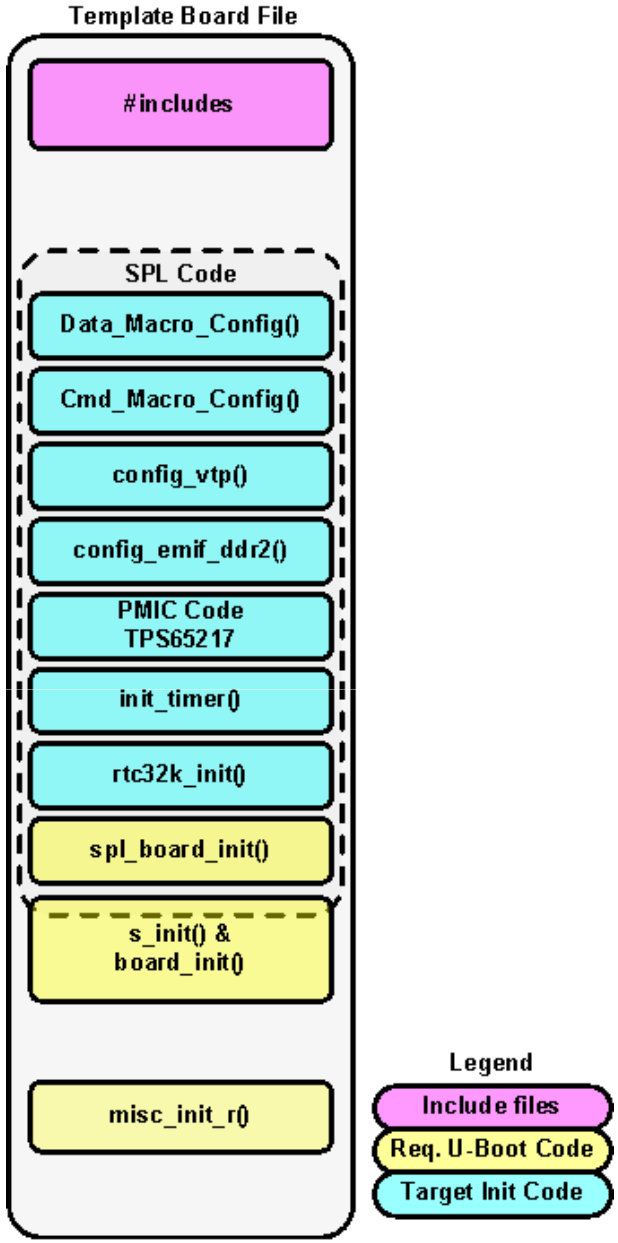
U-Boot Anatomy of a board File

- Defines Required interface functions for SPL and U-Boot
- One source file contains the code for both SPL and U-Boot and are separated by pre-processor flags
- SPL handles the initialization of clocks, DDR, Serial Port and PMIC
- Some functions are defined twice in both an SPL context and then again in a U-Boot context (s_init & board_init)
- The board file is where the developer will spend most of their effort for a port



U-Boot/SPL Board Template File

- The board file (evm.c) used here today is different from the one provided in the SDK
- Contains the code for both SPL and U-Boot
- This Board Template only enables MPU Clock, DDR and the Serial Port
- It's up to developer to decide how much functionality they choose to put into the board file and hence the u-boot.img. If the target board supports more peripherals but only one or two is needed to boot into the kernel why add that code?



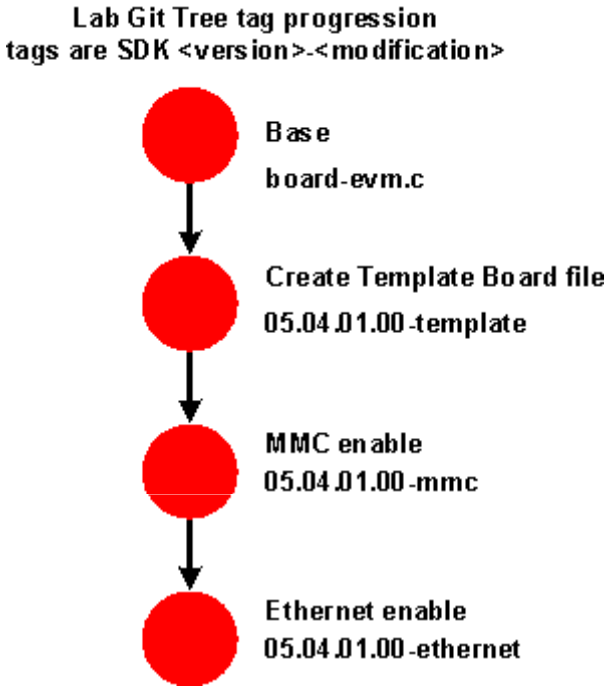
U-Boot Board Port Labs

Board Port Labs

- Lab 1
  - Introduce the template board file and how SPL and u-boot.img are built
- Lab 2
  - Build on the template file demonstrating how to add the MMC and Ethernet peripherals

Board Port Source Tree being used

- Currently Source is derived from AM-SDK-05.04.01.00, the Port Tree will follow or track each SDK release
- A git tree has been setup for these labs on the host machines
- Using existing board file name and build methods
- Using the default U-Boot configuration supplied with the SDK



U-Boot Board Port Exercise 1 - Overview

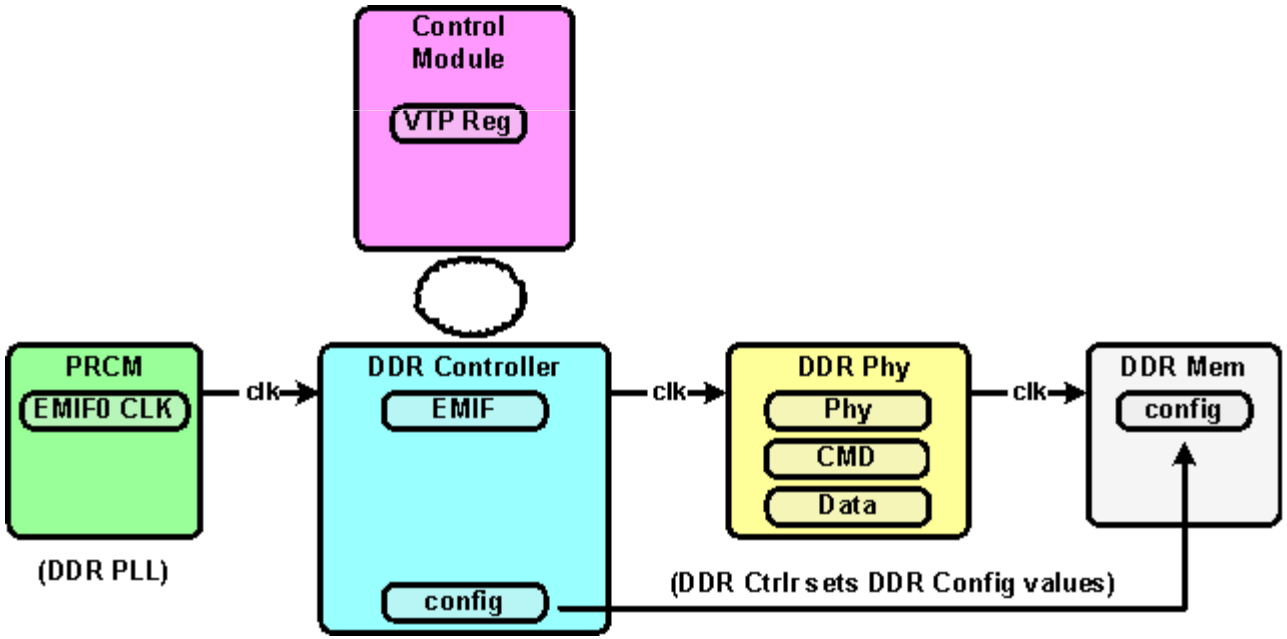
- Goal : Introduce workshop attendees to a board template file that can be used later for a U-Boot Board port
- How this is Demonstrated
  - Build both an SPL and u-boot.img using provided AM335x board template file, which has:
    - Base processor configuration for u-boot, ddr, clocks and a serial console are initialized
- What is being done:
  - Examine the board file to see what is being initialized
- Perform the Lab

First Burning Question:

So… where are the DDR timings and the Clock set?

First Burning Question: So… where are the clock and DDR timings set? DDR First

- DDR Setup requires portions of 4 functional blocks to be setup. (Block Diagram)
- EMIF , CMD, DATA and EMIF0 CLK are dependent on Memory selected



First Burning Question: So… where are the clock and DDR timings set? DDR First

```
Data_Macro_Config()
{
   raw reg write to DDR Phy control registers
}
```

```
CMD_Macro_Config()
{
   raw reg write to DDR Phy control registers
}
```

```
config_vtp()
{
   raw reg write to Control Module registers
}
```

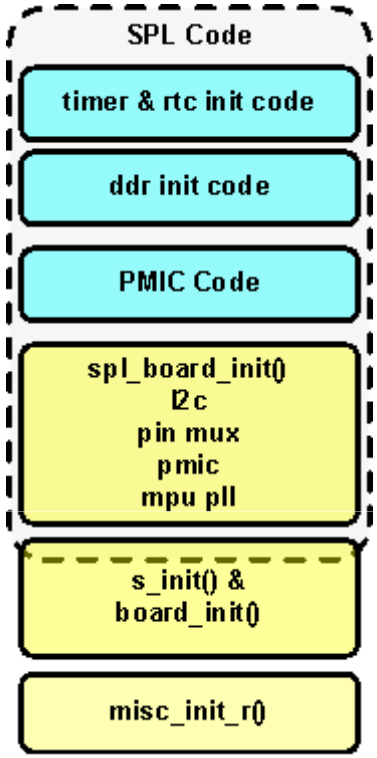
```
config_emif_ddr2()
{
   raw reg write to EMIF Timing control registers
}
```

```
static void config_am335x_ddr(void)
{
      enable_ddr_clocks();
      config_vtp();
      Cmd_Macro_Config();
      Data_Macro_Config(data_macro_0);
      Data_Macro_Config(data_macro_1);

      /* Several Raw Reg writes to DDR IO control */

      config_emif_ddr2();
}
```

board/ti/am335x/evm.c

All Register values are found in:
(change here to support mem changes)
arch/arm/include/asm/arch-ti81xx/ddr_defs.h

All Register offsets are found in:
arch/arm/include/asm/arch-ti81xx/cpu.h



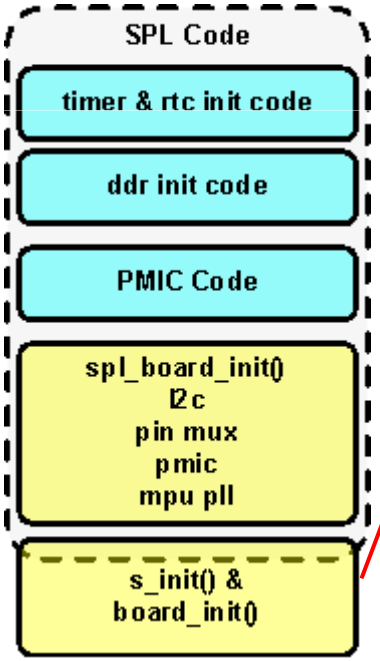
- The DDR is set up within the SPL context
- enable_ddr_clocks in pll.c,
- ddr_defs.h and cpu.h

Here is link to a Tool that can be used to generate necessary values to configure DDR

- Spread Sheet Tool can be found here
  - http://processors.wiki.ti.com/index.php/AM335x_EMIF_Configuration_tips

The SPL entry function

- s_init is called from lowlevel_init.S to setup system PLL, RTC, UART, timer and finally configures DDR



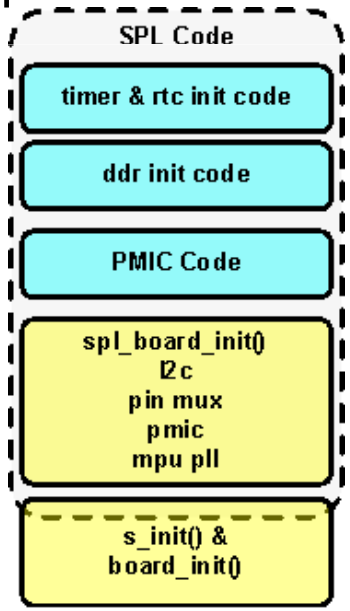

```
/*
 * early system init of muxing and clocks.
 */
void s_init(void)
{
  /* u-boot context */

#ifdef CONFIG_SPL_BUILD
      /* Setup the PLLs and the clocks for the peripherals */
      pll_init();

      /* Enable RTC32K clock */
      rtc32k_enable();

      /* UART softreset */

      enable_uart0_pin_mux();

      /* Disable smart idle */

      /* Initialize the Timer */
      init_timer();

      preloader_console_init();

      config_am335x_ddr();
#endif
}
```

board/ti/am335x/evm.c

This function has both SPL and u-boot contexts

And now to Set the MPU Clock Rate….

- SPL Context Function
- Before setting the MPU PLL the voltage and current are increased using I2C commands to the tps65217.




```
void spl_board_init(void)
{
   enable_i2c0_pin_mux();
   i2c_init(,);

   /* BeagleBone PMIC Code */
   i2c_probe(TPS65217_CHIP_PM)

   /* Increase USB current limit to 1300mA */
   tps65217_reg_write(, ,USB_INPUT_CUR_LIMIT_1300MA,
                      USB_INPUT_CUR_LIMIT_MASK)
   /* Set DCDC2 (MPU) voltage to 1.275V */
   tps65217_voltage_update(,DCDC_VOLT_SEL_1275MV)

   /* Set LDO3, LDO4 output voltage to 3.3V */
   tps65217_reg_write(,,LDO_VOLTAGE_OUT_3_3,)
   tps65217_reg_write(,,LDO_VOLTAGE_OUT_3_3, LDO_MASK)

   /* Set MPU Frequency to 720MHz */
   mpu_pll_config(MPUPLL_M_720);
}
(Representative code, simplified for the point of discussion)
```

- Called from arch/arm/cpu/armv7/start.S
- If you have a different PMIC you will most likely need a different code base than what is shown here

Board File Template for u-boot.img

- Within the u-boot context this is the entry function
- Same source file as used for SPL
- Pin Mux config is setup for i2c, uart (already done in SPL) and

```
int board_init(void)
{
	/* Configure the i2c0 pin mux */
	enable_i2c0_pin_mux();

	i2c_init(CONFIG_SYS_I2C_SPEED, CONFIG_SYS_I2C_SLAVE);

	board_id = BONE_BOARD;

	configure_evm_pin_mux(board_id);

#ifndef CONFIG_SPL_BUILD
	board_evm_init();
#endif

	gpmc_init();

	return 0;
}
```

board/ti/am335x/evm.c

Do LAB 1.....

U-Boot Board Port Exercise 2 - Overview

- Goal : Take the board template file (evm.c) and add both MMC and Ethernet support
- How this is Demonstrated
  - Using the supplied git tree checkout a Ethernet tagged branch, this has both the MMC and Ethernet support code. Build the kernel.
    - This adds Pin Mux support for both Ethernet and MMC
    - Adds the init functions for Ethernet and MMC.
- What is being done:
  - Examine the code changes necessary to implement Ethernet and MMC
- Perform the Lab

Steps to adding MMC and Ethernet to the target board file

- Review system info to see how peripheral is attached
- Pin Mux
  - Use the Pin Mux Utility to configure Pin Init data
- Create Device Init function
  - If device is supported in U-Boot, set the desired include in include/configs
- Add Device Init Function to board file

Pin Mux Utility

- Pin Mux tool capture for MII interface
- While the tool shows GMII this is the MII interface, doc bug in tool

```
static struct module_pin_mux mii1_pin_mux[] = {
        {OFFSET(mii1_rxerr), MODE(0) | RXACTIVE},       /* MII1_RXERR */
        {OFFSET(mii1_txen), MODE(0)},                   /* MII1_TXEN */
        {OFFSET(mii1_rxdv), MODE(0) | RXACTIVE},        /* MII1_RXDV */
        {OFFSET(mii1_txd3), MODE(0)},                   /* MII1_TXD3 */
        {OFFSET(mii1_txd2), MODE(0)},                   /* MII1_TXD2 */
        {OFFSET(mii1_txd1), MODE(0)},                   /* MII1_TXD1 */
        {OFFSET(mii1_txd0), MODE(0)},                   /* MII1_TXD0 */
        {OFFSET(mii1_txclk), MODE(0) | RXACTIVE},       /* MII1_TXCLK */
        {OFFSET(mii1_rxclk), MODE(0) | RXACTIVE},       /* MII1_RXCLK */
        {OFFSET(mii1_rxd3), MODE(0) | RXACTIVE},        /* MII1_RXD3 */
        {OFFSET(mii1_rxd2), MODE(0) | RXACTIVE},        /* MII1_RXD2 */
        {OFFSET(mii1_rxd1), MODE(0) | RXACTIVE},        /* MII1_RXD1 */
        {OFFSET(mii1_rxd0), MODE(0) | RXACTIVE},        /* MII1_RXD0 */
        {OFFSET(mdio_data), MODE(0) | RXACTIVE | PULLUP_EN}, /* MDIO_DATA */
        {OFFSET(mdio_clk), MODE(0) | PULLUP_EN},        /* MDIO_CLK */
        {-1},
};
```

Pad Config.	Bot/Top Ball	IO Power	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
I IEN PD	J15 / -	VDDSHV5=3.3V	GMII1_RXER	RMII1_RXER	SPI1_D1_M...	I2C1_SCL_M...	MCASP1_FS...	UART5_RTSN...	UART2_TXD ...	GPIO3[2]
O IDIS PD	J16 / -	VDDSHV5=3.3V	GMII1_TXEN	RMII1_TXEN	RGMII1_TCTL	TIMER4_MUX0	MCASP1_AX...	EQEP0_IND...	MMC2_CMD...	GPIO3[3]
I IEN PD	J17 / -	VDDSHV5=3.3V	GMII1_RXDV	LCD_MEMO...	RGMII1_RCTL	UART5_TXD ...	MCASP1_AC...	MMC2_DAT0...	MCASP0_AC...	GPIO3[4]
O IDIS PD	J18 / -	VDDSHV5=3.3V	GMII1_TXD3	DCAN0_TX ...	RGMII1_TD3	UART4_RXD...	MCASP1_FS...	MMC2_DAT1...	MCASP0_FS...	GPIO0[16]
O IDIS PD	K15 / -	VDDSHV5=3.3V	GMII1_TXD2	DCAN0_RX ...	RGMII1_TD2	UART4_TXD ...	MCASP1_AX...	MMC2_DAT2...	MCASP0_AH...	GPIO0[17]
O IDIS PD	K16 / -	VDDSHV5=3.3V	GMII1_TXD1	RMII1_TXD1	RGMII1_TD1	MCASP1_FS...	MCASP1_AX...	EQEP0A_IN ...	MMC1_CMD...	GPIO0[21]
O IDIS PD	K17 / -	VDDSHV5=3.3V	GMII1_TXD0	RMII1_TXD0	RGMII1_TD0	MCASP1_AX...	MCASP1_AC...	EQEP0B_IN ...	MMC1_CLK ...	GPIO0[28]
I IEN PD	K18 / -	VDDSHV5=3.3V	GMII1_TXCLK	UART2_RXD...	RGMII1_TCLK	MMC0_DAT7	MMC1_DAT0...	UART1_DCD...	MCASP0_AC...	GPIO3[9]
I IEN PD	L18 / -	VDDSHV5=3.3V	GMII1_RXCLK	UART2_TXD ...	RGMII1_RCLK	MMC0_DAT6	MMC1_DAT1...	UART1_DSR...	MCASP0_FS...	GPIO3[10]
I IEN PD	L17 / -	VDDSHV5=3.3V	GMII1_RXD3	UART3_RXD...	RGMII1_RD3	MMC0_DAT5	MMC1_DAT2...	UART1_DTR...	MCASP0_AX...	GPIO2[18]
I IEN PD	L16 / -	VDDSHV5=3.3V	GMII1_RXD2	UART3_TXD ...	RGMII1_RD2	MMC0_DAT4	MMC1_DAT3...	UART1_RIN ...	MCASP0_AX...	GPIO2[19]
I IEN PD	L15 / -	VDDSHV5=3.3V	GMII1_RXD1	RMII1_RXD1	RGMII1_RD1	MCASP1_AX...	MCASP1_FS...	EQEP0_STR...	MMC2_CLK ...	GPIO2[20]
I IEN PD	M16 / -	VDDSHV5=3.3V	GMII1_RXD0	RMII1_RXD0	RGMII1_RD0	MCASP1_AH...	MCASP1_AH...	MCASP1_AC...	MCASP0_AX...	GPIO2[21]
IO IEN PD	H18 / -	VDDSHV5=3.3V	RMII1_REFC...	XDMA_EVE...	SPI1_CS0_M...	UART5_TXD ...	MCASP1_AX...	MMC0_POW...	MCASP1_AH...	GPIO0[29]
IO IEN PU	M17 / -	VDDSHV5=3.3V	MDIO_DATA	TIMER6_MUX2	UART5_RXD...	UART3_CTSN...	MMC0_SDC...	MMC1_CMD...	MMC2_CMD...	GPIO0[0]
O IDIS PU	M18 / -	VDDSHV5=3.3V	MDIO_CLK	TIMER5_MUX2	UART5_TXD ...	UART3_RTSN...	MMC0_SDW...	MMC1_CLK ...	MMC2_CLK ...	GPIO0[1]

Adding MMC to the U-Boot Board file

- Find the pre-processor flags in the am335x_evm.h config file that control inclusion of MMC
- Use the name found for a weak alias to define in the board file
- Create the init function in the board file

```
/* HSMMC support */
#ifdef CONFIG_MMC
#define CONFIG_GENERIC_MMC
#define CONFIG_OMAP_HSMMC
#define CONFIG_CMD_MMC
#define CONFIG_DOS_PARTITION
#define CONFIG_CMD_FAT
#define CONFIG_CMD_EXT2
#endif
```

`#define CONFIG_MMC` — Define in the config file include/configs/am335x_evm.h

drivers/mmc/mmc.c — In the driver file look for a weak alias definition, the name defined here is the one to name the init function in the board file

```
int board_mmc_init(bd_t *bis) __attribute__((weak, alias("__def_mmc_init")));
```

```
#ifdef CONFIG_GENERIC_MMC
int board_mmc_init(bd_t *bis)
{
        omap_mmc_init(0);
        omap_mmc_init(1);
        return 0;
}
#endif
```

Define in the board file and U-boot will call to initialize

board/ti/am335x/evm.c

Adding Ethernet to the U-Boot Board File

- Use the name found for a weak alias to define in the board file, in net/eth.c
- Create the init functions in the board file
  - 2 functions are created one to init the phy (local) and the board_eth_init definition for u-boot network driver to call
- There are additional supporting structures define in the board file

net/eth.c — In the driver file look for a weak alias definition, the name defined here is the one to name the init function in the board file

```
*
 * CPU and board-specific Ethernet initializations.  Aliased function
 * signals caller to move on
 */
static int __def_eth_init(bd_t *bis)
{
	return -1;
}

int board_eth_init(bd_t *bis) __attribute__((weak, alias("__def_eth_init")));
```

```
static void evm_phy_init(char *name, int addr)
{
/* Large function.... */

}
```

board/ti/am335x/evm.c

```
int board_eth_init(bd_t *bis)
{
	eth_getenv_enetaddr(,)

	__raw_writel(MII_MODE_ENABLE, MAC_MII_SEL);

	return cpsw_register(&cpsw_data);
}
```

board/ti/am335x/evm.c

git diff – Code Difference between template and mmc commit

- “git tag” is used to list tags on the git tree

```
05.04.01.00-base
05.04.01.00-ethernet
05.04.01.00-mmc
05.04.01.00-template
```

- “git diff” this is used to isolate code between git commits.

```
schuyler@morpheus:~/bp_uboot/sitara-board-port-uboot$ git diff 05.04.01.00-template..05.04.01.00-mmc
diff --git a/board/ti/am335x/evm.c b/board/ti/am335x/evm.c
index 1635871..b4d7e55 100644
--- a/board/ti/am335x/evm.c
+++ b/board/ti/am335x/evm.c
@@ -487,3 +487,15 @@ int board_late_init(void)
        return 0;
 }
 #endif
+
+#ifndef CONFIG_SPL_BUILD
+#ifdef CONFIG_GENERIC_MMC
+int board_mmc_init(bd_t *bis)
+{
+       omap_mmc_init(0);
+       omap_mmc_init(1);
+       return 0;
+}
+#endif
+#endif /* CONFIG_SPL_BUILD */
+
schuyler@morpheus:~/bp_uboot/sitara-board-port-uboot$
```

- Do not be concerned about knowing git at this point, here we are using this for illustration purposes.

git diff – Code Difference between mmc and ethernet commit

- “git diff” commands goes across several screens
- Type “q” to quit command at any point
- Note the plus sign on the edge of the diagram, code addition

```
diff --git a/board/ti/am335x/evm.c b/board/ti/am335x/evm.c
index b4d7e55..863e4cb 100644
--- a/board/ti/am335x/evm.c
+++ b/board/ti/am335x/evm.c
@@ -499,3 +499,133 @@ int board_mmc_init(bd_t *bis)
 #endif
 #endif /* CONFIG_SPL_BUILD */

+#ifdef CONFIG_DRIVER_TI_CPSW
+/* TODO : Check for the board specific PHY */
+static void evm_phy_init(char *name, int addr)
+{
+       unsigned short val;
+       unsigned int cntr = 0;
+       unsigned short phyid1, phyid2;
+
+
+       /* Enable Autonegotiation */
+       if (miiphy_read(name, addr, MII_BMCR, &val) != 0) {
+               printf("failed to read bmcr\n");
+               return;
+       }
+       val |= BMCR_FULLDPLX | BMCR_ANENABLE | BMCR_SPEED100;
+
+       if (miiphy_write(name, addr, MII_BMCR, val) != 0) {
+               printf("failed to write bmcr\n");
+               return;
+       }
+       miiphy_read(name, addr, MII_BMCR, &val);
+
+       /* Setup general advertisement */
+       if (miiphy_read(name, addr, MII_ADVERTISE, &val) != 0) {
+               printf("failed to read anar\n");
+               return;
+       }
+
+
:
```

git diff – Code Difference between mmc and ethernet commit (cont)

- Code continuation for Ethernet PHY setup
- This code was extracted from Beagle Bone specific code from the SDK release.

```
+       val |= (LPA_10HALF | LPA_10FULL | LPA_100HALF | LPA_100FULL);
+
+       if (miiphy_write(name, addr, MII_ADVERTISE, val) != 0) {
+               printf("failed to write anar\n");
+               return;
+       }
+       miiphy_read(name, addr, MII_ADVERTISE, &val);
+
+       /* Restart auto negotiation*/
+       miiphy_read(name, addr, MII_BMCR, &val);
+       val |= BMCR_ANRESTART;
+       miiphy_write(name, addr, MII_BMCR, val);
+
+       /*check AutoNegotiate complete - it can take upto 3 secs*/
+       do {
+               udelay(40000);
+               cntr++;
+               if (!miiphy_read(name, addr, MII_BMSR, &val)) {
+                       if (val & BMSR_ANEGCOMPLETE)
+                               break;
+               }
+       } while (cntr < 250);
+
+       if (cntr >= 250)
+               printf("Auto negotitation failed\n");
+
+       return;
+}
+
+static void cpsw_control(int enabled)
+{
+       /* nothing for now */
+       /* TODO : VTP was here before */
+       return;
+}
+
:
```

git diff – Code Difference between mmc and ethernet commit (cont)

- Code continuation for Ethernet setup
- This code was extracted from Beagle Bone specific code from the SDK release.

```
+static struct cpsw_slave_data cpsw_slaves[] = {
+       {
+               .slave_reg_ofs  = 0x208,
+               .sliver_reg_ofs = 0xd80,
+               .phy_id         = 0,
+       },
+       {
+               .slave_reg_ofs  = 0x308,
+               .sliver_reg_ofs = 0xdc0,
+               .phy_id         = 1,
+       },
+};
+
+static struct cpsw_platform_data cpsw_data = {
+       .mdio_base              = AM335X_CPSW_MDIO_BASE,
+       .cpsw_base              = AM335X_CPSW_BASE,
+       .mdio_div               = 0xff,
+       .channels               = 8,
+       .cpdma_reg_ofs          = 0x800,
+       .slaves                 = 2,
+       .slave_data             = cpsw_slaves,
+       .ale_reg_ofs            = 0xd00,
+       .ale_entries            = 1024,
+       .host_port_reg_ofs      = 0x108,
+       .hw_stats_reg_ofs       = 0x900,
+       .mac_control            = (1 << 5) /* MIIEN */,
+       .control                = cpsw_control,
+       .phy_init               = evm_phy_init,
+       .gigabit_en             = 1,
+       .host_port_num          = 0,
+       .version                = CPSW_CTRL_VERSION_2,
+};
+
+int board_eth_init(bd_t *bis)
+{
+       uint8_t mac_addr[6];
:
```

git diff – Code Difference between mmc and ethernet commit (cont)

- Code continuation for Ethernet setup
- This code was extracted from Beagle Bone specific code from the SDK release.
- How is board_eth_init(..) called?

```
+       .version                = CPSW_CTRL_VERSION_2,
+};
+
+int board_eth_init(bd_t *bis)
+{
+       uint8_t mac_addr[6];
+       uint32_t mac_hi, mac_lo;
+       u_int32_t i;
+
+       if (!eth_getenv_enetaddr("ethaddr", mac_addr)) {
+               debug("<ethaddr> not set. Reading from E-fuse\n");
+               /* try reading mac address from efuse */
+               mac_lo = __raw_readl(MAC_ID0_LO);
+               mac_hi = __raw_readl(MAC_ID0_HI);
+               mac_addr[0] = mac_hi & 0xFF;
+               mac_addr[1] = (mac_hi & 0xFF00) >> 8;
+               mac_addr[2] = (mac_hi & 0xFF0000) >> 16;
+               mac_addr[3] = (mac_hi & 0xFF000000) >> 24;
+               mac_addr[4] = mac_lo & 0xFF;
+               mac_addr[5] = (mac_lo & 0xFF00) >> 8;
+
+               if (is_valid_ether_addr(mac_addr))
+                       eth_setenv_enetaddr("ethaddr", mac_addr);
+               else {
+                       printf("Caution: Using hardcoded mac address. "
+                              "Set <ethaddr> variable to overcome this.\n");
+               }
+       }
+
+       __raw_writel(MII_MODE_ENABLE, MAC_MII_SEL);
+               /* No gigabit */
+       cpsw_data.gigabit_en = 0;
+
+       return cpsw_register(&cpsw_data);
+}
+#endif
(END)
```

Do LAB 2.....

U-Boot Board Port Summary

- Introduced a board port template file with a minimal feature set. Discussed the components in this file. This file could be used for actual board ports.
- Performed two labs demonstrating the template file in action.

Porting the Linux Kernel to a AM335x Target

Linux Port Agenda

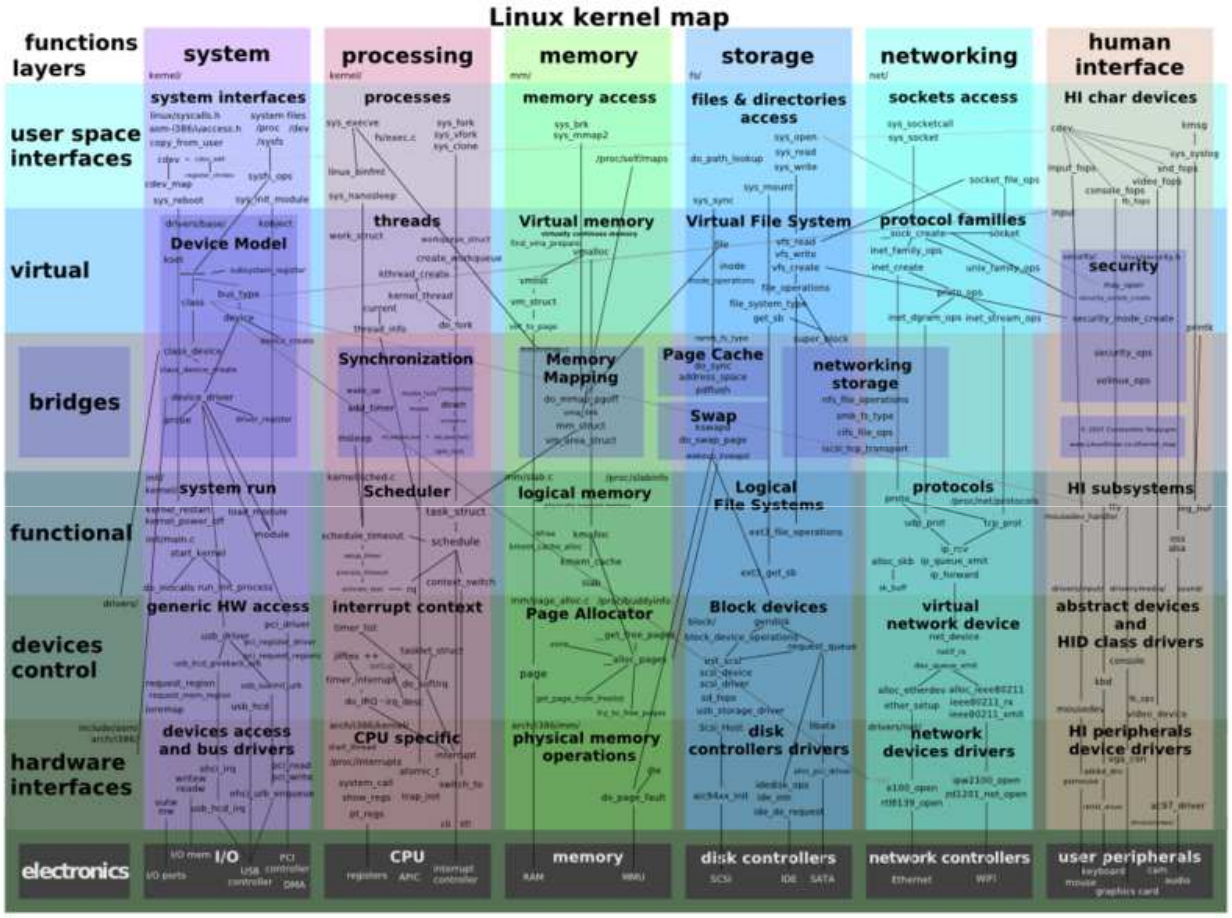
- What are the different stages of a Port
- Introduce the board file, where it fits in the Port Picture, where it is in the source tree
- Discuss the OMAP2+ Machine Shared Common Code
- Labs Introduction

Linux Board Port Exercises and Source Links

- Link to the U-Boot Labs
  - http://processors.wiki.ti.com/index.php/Sitara_Linux_Training:_Linux_Board_Port
- Link to the Linux Template Source tree (clone this tree)
  - git://gitorious.org/sitara-board-port/sitara-board-port-linux.git
- PSP Linux Kernel Repo –
  - http://arago-project.org/git/projects/?p=linux-am33x.git;a=summary

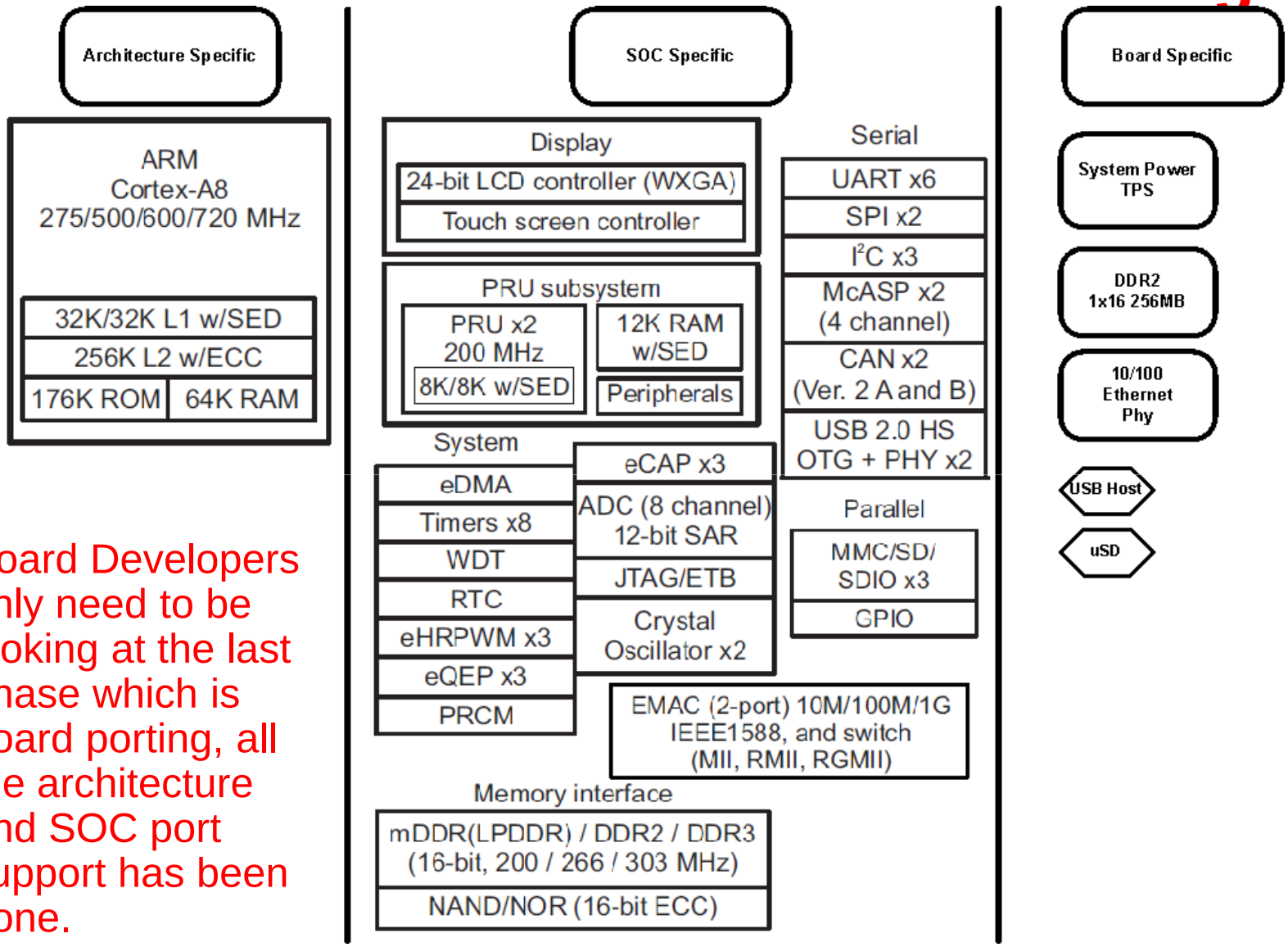
Linux Kernel Overview (AHHHHH…. The Kernel…)

- A very complex and overwhelming kernel block diagram, this is just to make you aware of what's below the waterline…..
- With a target port the architecture and SOC port has already been done. Therefore, the majority of this block diagram has been taken care of for the target port developer.



Source is: http://en.wikipedia.org/wiki/File:Linux_kernel_map.png

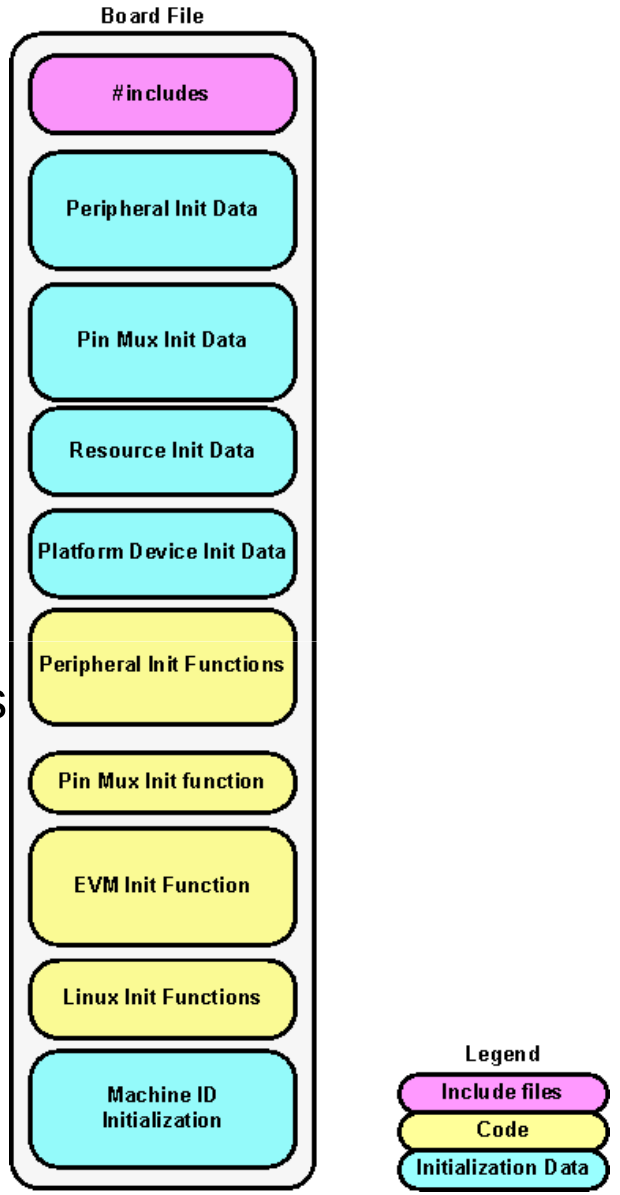
Architecture vs. SOC vs. Board Porting



- Board Developers only need to be looking at the last phase which is board porting, all the architecture and SOC port support has been done.

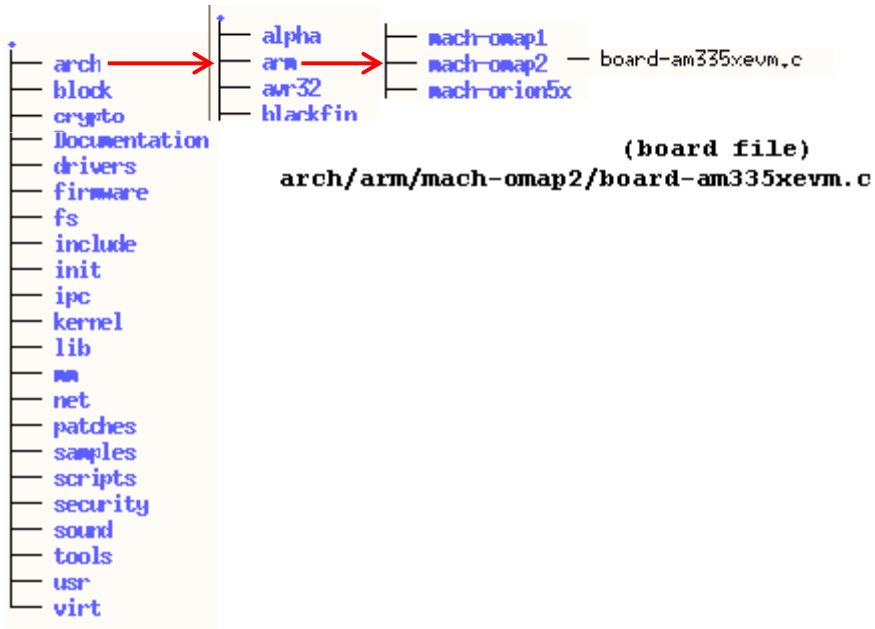
The Target Port Starts with a Board File

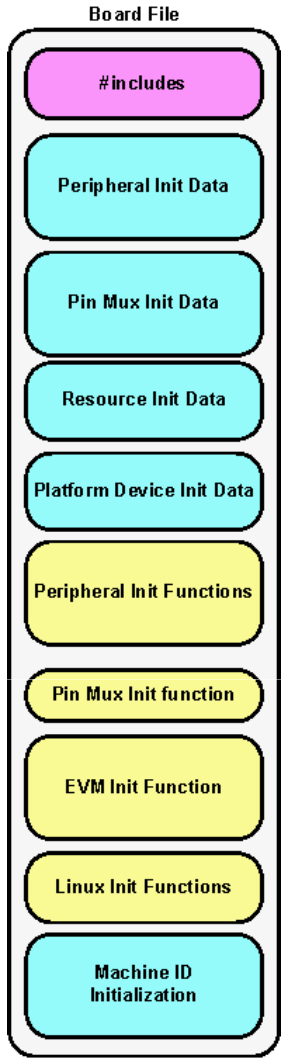
- Defines the Machine Name
- Declares Initialization Data for Peripherals being used
- Declare Pin Mux initialization Data
- Defines Initialization functions
- Provides required Machine Initialization functions
- Calls Common Initialization functions
- Summary is that this file defines several required elements required to boot a Linux kernel, one of several bricks in the wall so to speak.



Linux Kernel Source Tree Overview (Where is the Board file.)

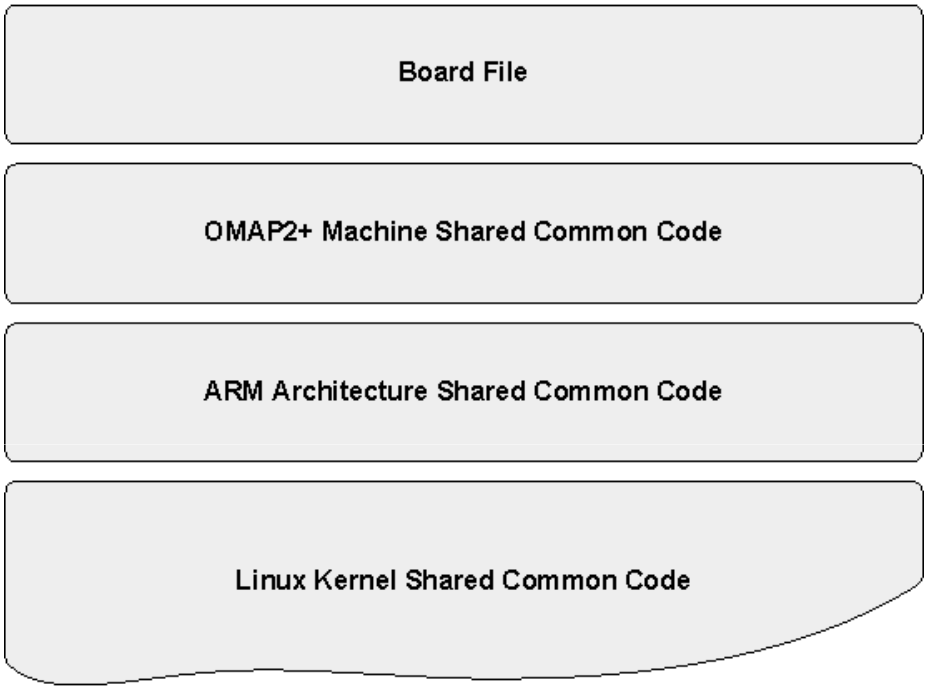
- The board file is located in a source directory called arch/arm/mach-omap2/ where all other board files are located of the same machine type.





How the Board File fits in the stack

- Board Developer will spend most of their time in the Board file.
- The Board file makes use of the machine shared common code
- The underlying port to the ARM Architecture Shared common code is already done and does not need to be looked at
- Finally everything rests on the Linux Kernel Shared Common Code.
- The lower in the stack you go the less direct interaction the board developer will or need to have.



OMAP2+ Machine Shared Common Code

- There are several board files in the mach-omap2 directory. These board files typical use the support functions defined within this directory. Below is a sampling of some of the supporting common code, not all are mentioned here.

OMAP2 Machine Shared Common Code – arch/arm/mach-omap2
Not a complete listing of the interfaces, just a few are highlighted and to explain how they are used, review this directory to see additional interfaces

serial	Sets up UARTs including pin mux	gpio	Initilization function
devices	Init calls, platform registration for most peripherals	i2c	Reset and Mux functions
common	Init calls to define global address range for select interfaces	mux	Defines a Pin Mux abstraction with supporting functions
clocks	Define clock domain mgmt functions	hsmmc	Init functions, hw and platform data
control	OMAP2 control registers	sdrc	Init function for SDRC and SMS
display	Display init calls, handles the differences between OMAP2,3 and 4	voltage	Voltage domain support functions

OMAP2+ Machine Shared Common Code

- Provided as means to provide a common interface to the SOC peripherals to reduce the time necessary to implement a board port
- This interface is not always a clear dividing between maintainers and board developers.
- This is not a documented interface and due to the changing nature of the Linux kernel will almost always be in flux. Maintainers in the end have the authority to accept reject code for their particular tree.

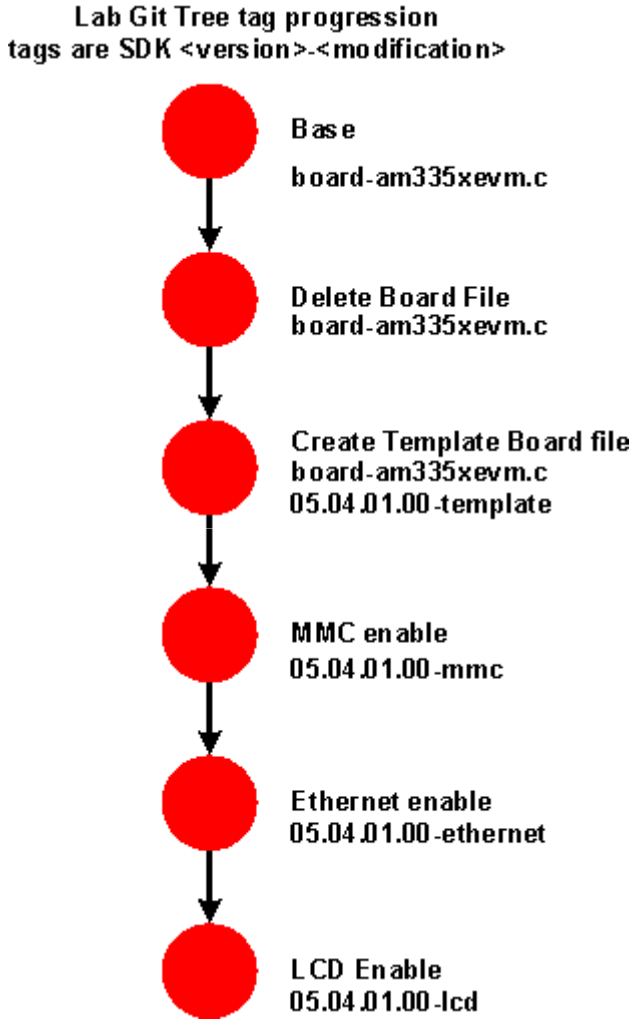
Linux Board Port Labs

Board Port Labs

- Lab 1
  - Introduce the template board file
- Lab 2
  - Build on the template file demonstrating how to add the MMC peripheral to provide a Root file system
- Lab 3
  - Build onto template file again this time adding Ethernet for network connectivity
- Lab 4
  - Demonstrate how to add an LCD panel to the board file

Board Port Source Tree being used

- Currently Source is derived from AM-SDK-05.04.01.00, the Port Tree will follow or track each SDK release
- A git tree has been setup for these labs on the host machines
- Using existing board file name and build methods
- Using the default kernel configuration supplied with the SDK

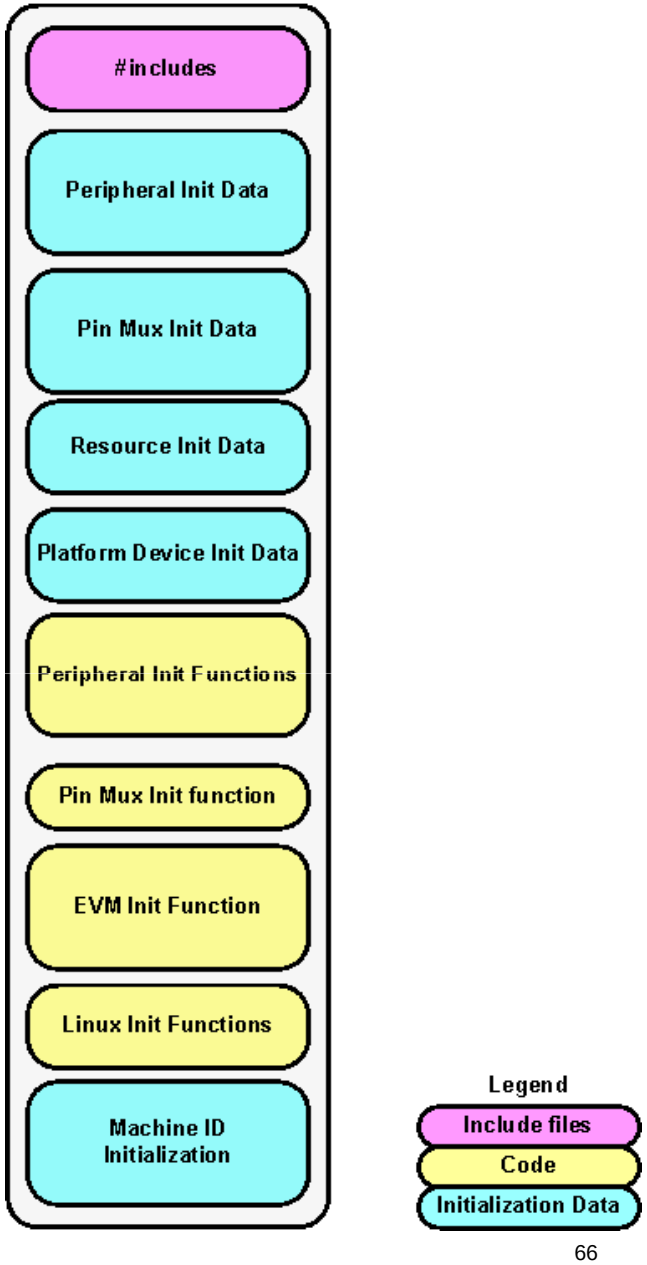


Linux Board Port Exercise 1 - Overview

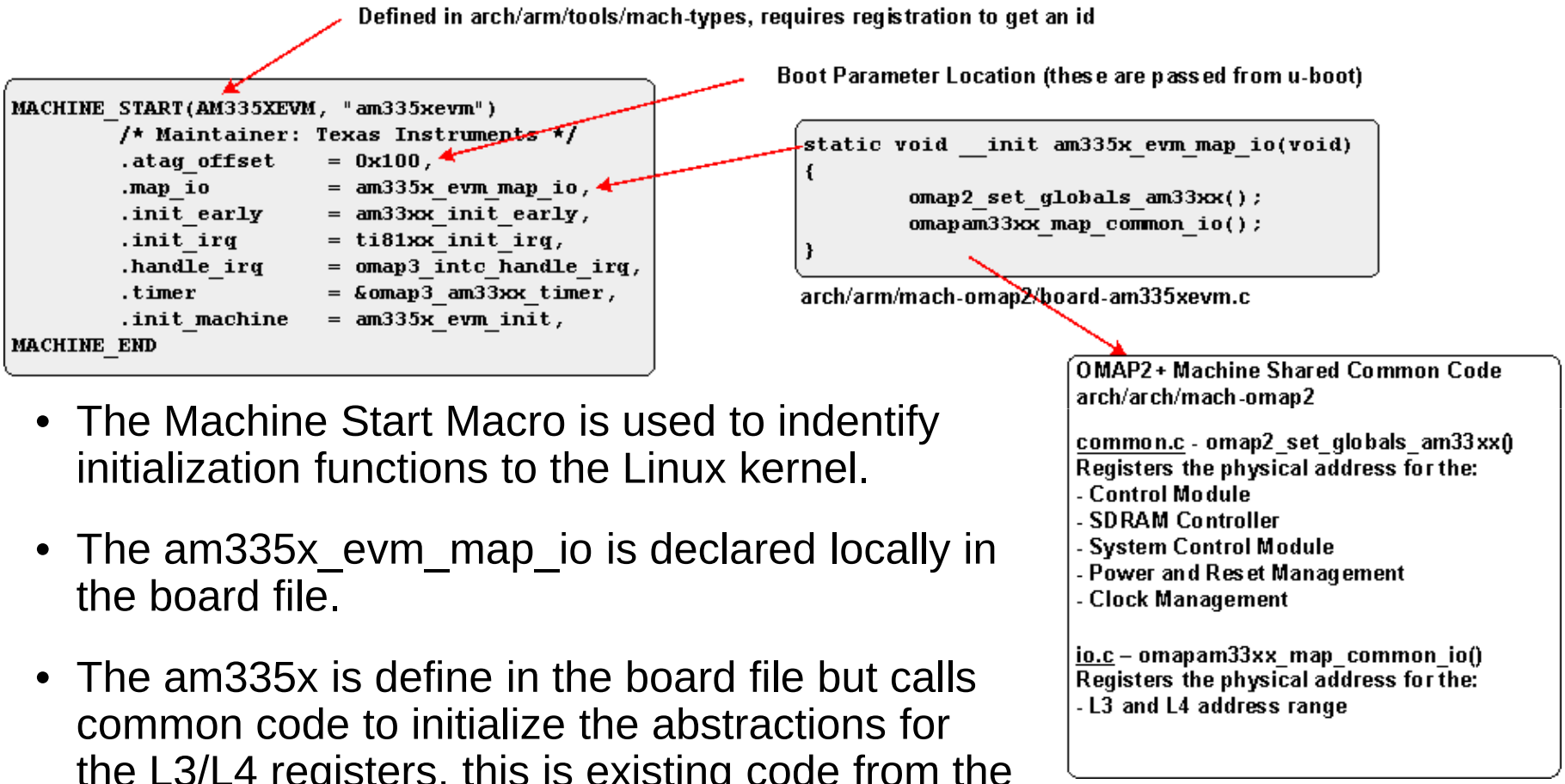
- Goal : Introduce workshop attendees to a board template file that can be used later for a Linux board port
- How this is Demonstrated
  - Build a kernel using provided AM335x board template file, which has:
    - Base processor configuration for Linux, just serial console peripheral is initialized
    - This board will not completely boot… no peripheral is defined for a Root File System
- What is being done:
  - Examine the board file to see what is being initialized
- Perform the Lab

Template Board File Anatomy

- Binds Linux to a particular target
- Interfaces with the OMAP2+ Machine Shared Common Code.
- Defines pin mux configuration
- The file contains device initialization functions and data.
- Defines the Machine ID and identifies to the Linux Kernel initialization functions



Template Board File Elements
MACHINE_START – Key Interface To Kernel



- The Machine Start Macro is used to indentify initialization functions to the Linux kernel.
- The am335x_evm_map_io is declared locally in the board file.
- The am335x is define in the board file but calls common code to initialize the abstractions for the L3/L4 registers, this is existing code from the OMAP2+ Shared Common Code, no need to modify.

Template Board File Elements – (cont.)
MACHINE_START – Key Interface To Kernel

```
MACHINE_START(AM335XEVM, "am335xevm")
        /* Maintainer: Texas Instruments */
        .atag_offset    = 0x100,
        .map_io         = am335x_evm_map_io,
        .init_early     = am33xx_init_early,
        .init_irq       = ti81xx_init_irq,
        .handle_irq     = omap3_intc_handle_irq,
        .timer          = &omap3_am33xx_timer,
        .init_machine   = am335x_evm_init,
MACHINE_END
```

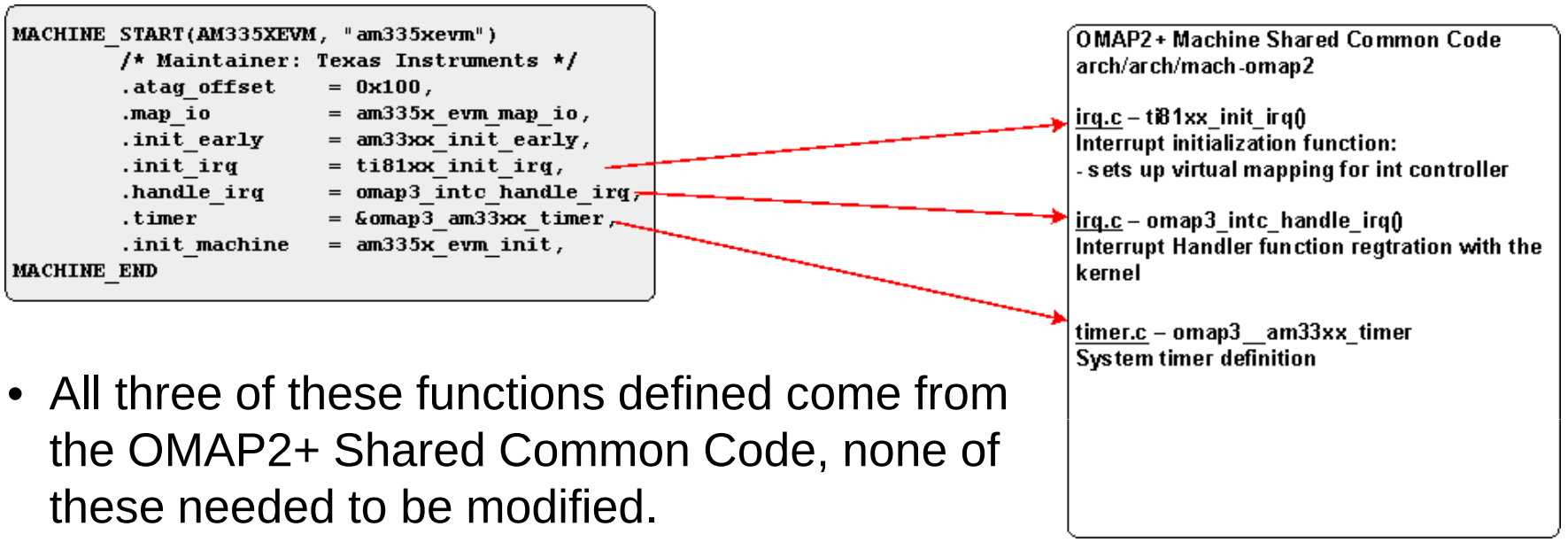
OMAP2 Shared Common Code
arch/arch/mach-omap2

io.c – am335x_init_early()
Several SOC initialization functions:
- global mapping
- revision checking
- feature checking
- common init
- voltage domains
- prm init
- power domains
- clock mgmt instance init
- hwmod init
- hwmod init post setup
- clock init

- The am33xx_init_early is a function within the OMAP2+ Shared common code.
- This is called directly from the common code without modification

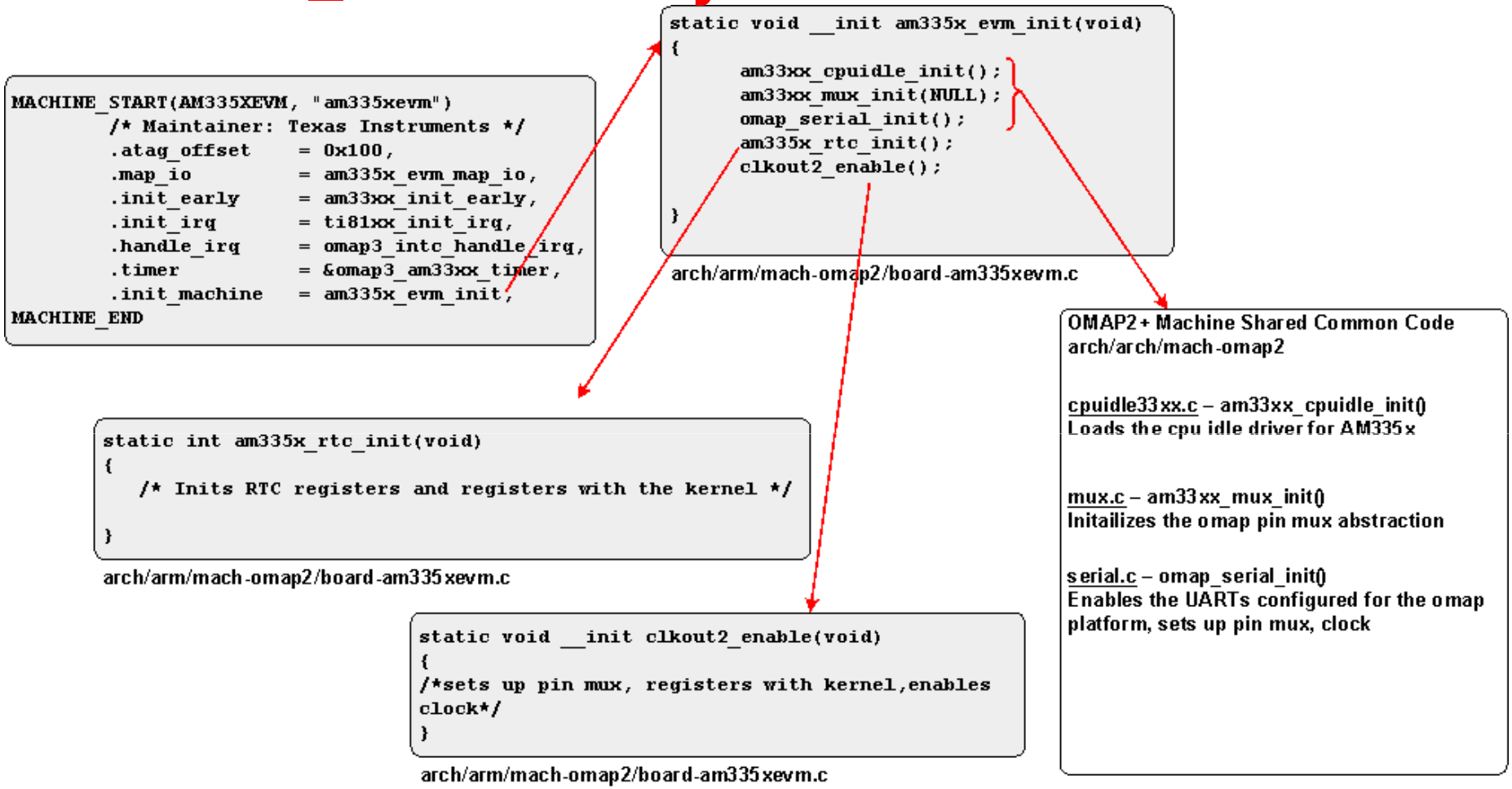
Board Template File Elements – (cont)
MACHINE_START – Key Interface To Kernel

```
MACHINE_START(AM335XEVM, "am335xevm")
        /* Maintainer: Texas Instruments */
        .atag_offset    = 0x100,
        .map_io         = am335x_evm_map_io,
        .init_early     = am33xx_init_early,
        .init_irq       = ti81xx_init_irq,
        .handle_irq     = omap3_intc_handle_irq,
        .timer          = &omap3_am33xx_timer,
        .init_machine   = am335x_evm_init,
MACHINE_END
```



- All three of these functions defined come from the OMAP2+ Shared Common Code, none of these needed to be modified.

Template Board File Elements – (cont)
MACHINE_START – Key Interface To Kernel



- The am335x_evm_init() is defined by the developer, but uses several functions from the OMAP2 Common Code without modification.

Question

Within the kernel source, where is the am335xevm board file located?

arch/arm/mach-omap2

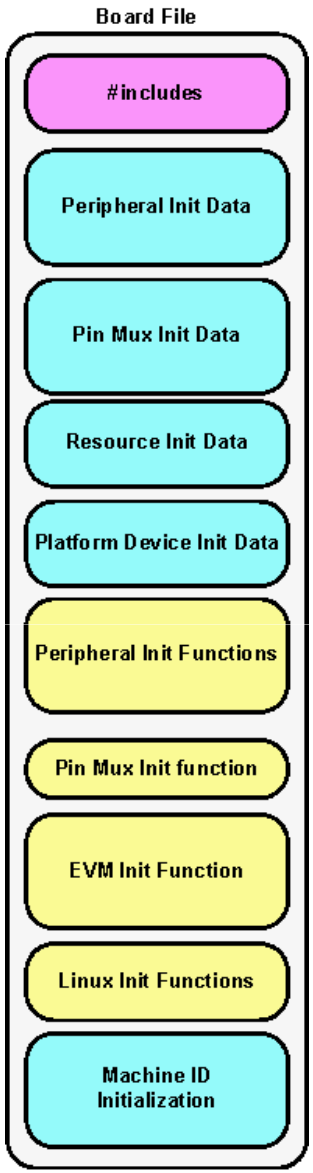
Do LAB 1.....

Linux Board Port Exercise 2 - Overview

- Goal: Build on the template file demonstrating how to add the MMC peripheral to provide a Root file system
- How this is demonstrated:
  - Using the provided lab git tree branch that has the code additions necessary to enable MMC
  - With MMC enabled the root file system can now be mounted
- What is being done:
  - Explaining the code addition components
- Perform the Lab

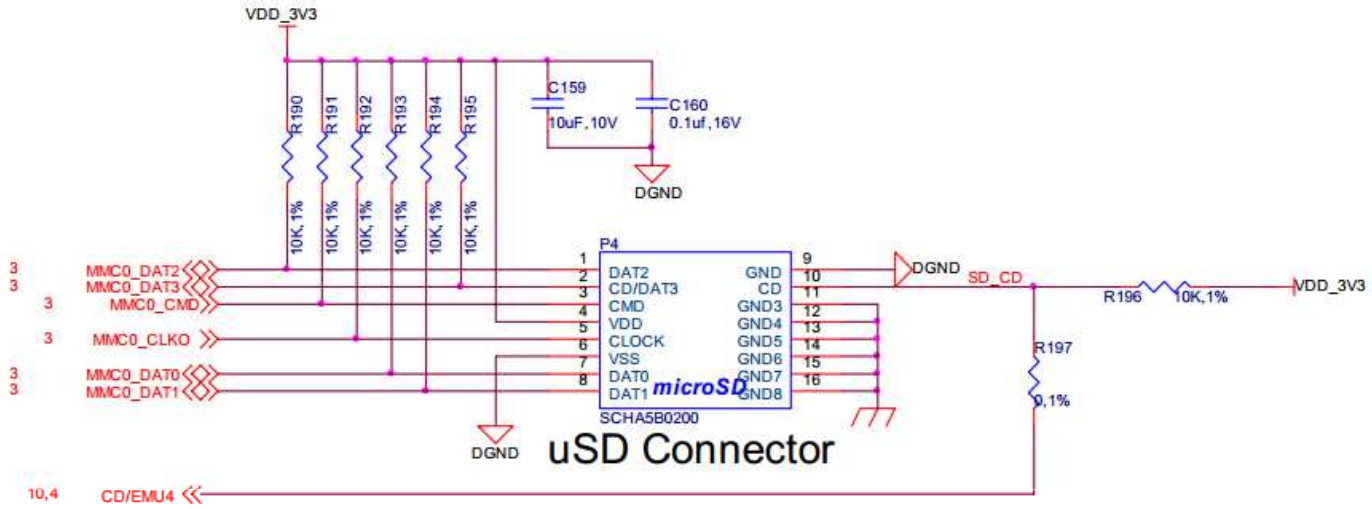
Steps to adding an MMC interface to target board file

- Review system info to see how peripheral is attached
- Pin Mux
  - Use the Pin Mux Utility to configure Pin Init data
- Device/Platform Initialization data
  - Some peripherals may not require init data
- Create Device Init function
- Add Device Init function to EVM Init Function



How is the peripheral attached? – Schematic to Pin Mux Utility

- Beagle Bone Schematic



- Pin Mux Tool Capture
- Beagle Bone does not use the WP pin

Using the pin mux tool to isolate the pin necessary for mmc0

Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
MMC0_DAT3	GPMC_A20 ...	UART4_CTSN...	TIMER5_MUX0	UART1_DCD...	PR1_PRU0_...	PR1_PRU0_...	GPIO2[26]
MMC0_DAT2	GPMC_A2* ...	UART4_RTSN...	TIMER6_MUX0	UART1_DSR...	PR1_PRU0_...	PR1_PRU0_...	GPIO2[27]
MMC0_DAT1	GPMC_A22 ...	UART5_CTSN...	UART3_RXD...	UART1_DTR...	PR1_PRU0_...	PR1_PRU0_...	GPIO2[28]
MMC0_DAT0	GPMC_A23 ...	UART5_RTSN...	UART3_TXD ...	UART1_RIN ...	PR1_PRU0_...	PR1_PRU0_...	GPIO2[29]
MMC0_CLK	GPMC_A24 ...	UART3_CTSN...	UART2_RXD...	DCAN1_TX ...	PR1_PRU0_...	PR1_PRU0_...	GPIO2[30]
MMC0_CMD	GPMC_A25 ...	UART3_RTSN...	UART2_TXD ...	DCAN1_RX ...	PR1_PRU0_...	PR1_PRU0_...	GPIO2[31]
SPI0_CS1	UART3_RXD...	ECAP1_IN_P...	MMC0_POW...	XDMA_EVE...	MMC0_SDC...	EMU4_MUX1	GPIO0[6]
MCASP0_AC...	EQEP0A_IN ...	MCASP0_AX...	MCASP1_AC...	MMC0_SDW...	PR1_PRU0_...	PR1_PRU0_...	GPIO3[18]

Have simplified the pin mux tool to show the pins necessary for the mmc0 interface
MMC0 pins for data, clk, cmd are being used from mode 0
GPIO 0.6 and GPIO 3.18 are being used for card detect and write protect respectively mode 7

Lab 2 Board File Additions – Pin Mux Initialization Data

- Capture from the Pin Mux tool, AM3358 ZCZ package

Using the pin mux tool to isolate the pin necessary for mmc0

Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
MMC0_DAT3	GPMC_A20 ...	UART4_CTSN...	TIMER5_MUX0	UART1_DCD...	PR1_PRU0 ...	PR1_PRU0 ...	GPIO2[26]
MMC0_DAT2	GPMC_A2[1] ...	UART4_RTSN...	TIMER6_MUX0	UART1_DSR...	PR1_PRU0 ...	PR1_PRU0 ...	GPIO2[27]
MMC0_DAT1	GPMC_A22 ...	UART5_CTSN...	UART3_RXD...	UART1_DTR...	PR1_PRU0 ...	PR1_PRU0 ...	GPIO2[28]
MMC0_DAT0	GPMC_A23 ...	UART5_RTSN...	UART3_TXD ...	UART1_RIN ...	PR1_PRU0 ...	PR1_PRU0 ...	GPIO2[29]
MMC0_CLK	GPMC_A24 ...	UART3_CTSN...	UART2_RXD...	DCAN1_TX ...	PR1_PRU0 ...	PR1_PRU0 ...	GPIO2[30]
MMC0_CMD	GPMC_A25 ...	UART3_RTSN...	UART2_TXD ...	DCAN1_RX ...	PR1_PRU0 ...	PR1_PRU0 ...	GPIO2[31]
SPI0_CS1	UART3_RXD...	ECAP1_IN_P...	MMC0_POW...	XDMA_EVE...	MMC0_SDC...	EMU4_MUX1	GPIO0[6]
MCASP0_AC...	EQEP0A_IN ...	MCASP0_AX...	MCASP1_AC...	MMC0_SDW...	PR1_PRU0 ...	PR1_PRU0 ...	GPIO3[18]

Have simplified the pin mux tool to show the pins necessary for the mmc0 interface
MMC0 pins for data, clk, cmd are being used from mode 0
GPIO 0.6 and GPIO 3.18 are being used for card detect and write protect respectively mode 7

- Use existing pinmux_config struct to create pin mux initialization data for mmc0
- Number of pins has to match

Pin Mux definition for MMC 0

```
/* Module pin mux for mmc0 */
static struct pinmux_config mmc0_pin_mux[] = {
        {"mmc0_dat3.mmc0_dat3", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
        {"mmc0_dat2.mmc0_dat2", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
        {"mmc0_dat1.mmc0_dat1", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
        {"mmc0_dat0.mmc0_dat0", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
        {"mmc0_clk.mmc0_clk",   OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
        {"mmc0_cmd.mmc0_cmd",   OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
        {"mcasp0_aclkr.mmc0_sdwp", OMAP_MUX_MODE7 | AM33XX_PIN_INPUT_PULLUP},
        {"spi0_cs1.mmc0_sdcd",  OMAP_MUX_MODE7 | AM33XX_PIN_INPUT_PULLUP},
        {NULL, 0},
};
```

```
{"mmc0_dat3.mmc0_dat3", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP}
```

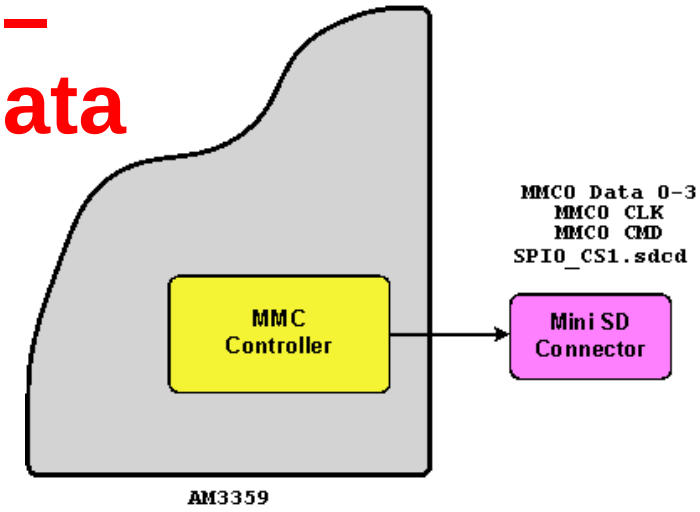
pin name – mmc0_dat3 <arch/arm/mach-omap2/mux33xx.c >

pin value and type < arch/arm/mach-omap2/mux.h >

Lab 2 Board File Additions – MMC Device Initialization Data

Init Data for MMC 0

```
/* Convert GPIO signal to GPIO pin number */
#define GPIO_TO_PIN(bank, gpio) (32 * (bank) + (gpio))

static struct omap2_hsmmc_info am335x_mmc[] __initdata = {
        {
                .mmc            = 1,
                .caps           = MMC_CAP_4_BIT_DATA,
                .gpio_cd        = GPIO_TO_PIN(0, 6),
                .gpio_wp        = GPIO_TO_PIN(3, 18),
                .ocr_mask       = MMC_VDD_32_33 | MMC_VDD_33_34, /* 3V3 */
        },
        {
                .mmc            = 0,    /* will be set at runtime */
        },
        {
                .mmc            = 0,    /* will be set at runtime */
        },
        {}      /* Terminator */
};
```



- MMC initialization structure to enable interface #1
- This init data is from EVM, BB does not use WP signal

- OMAP 2 mmc structure definition
- Only the elements used are shown, several more

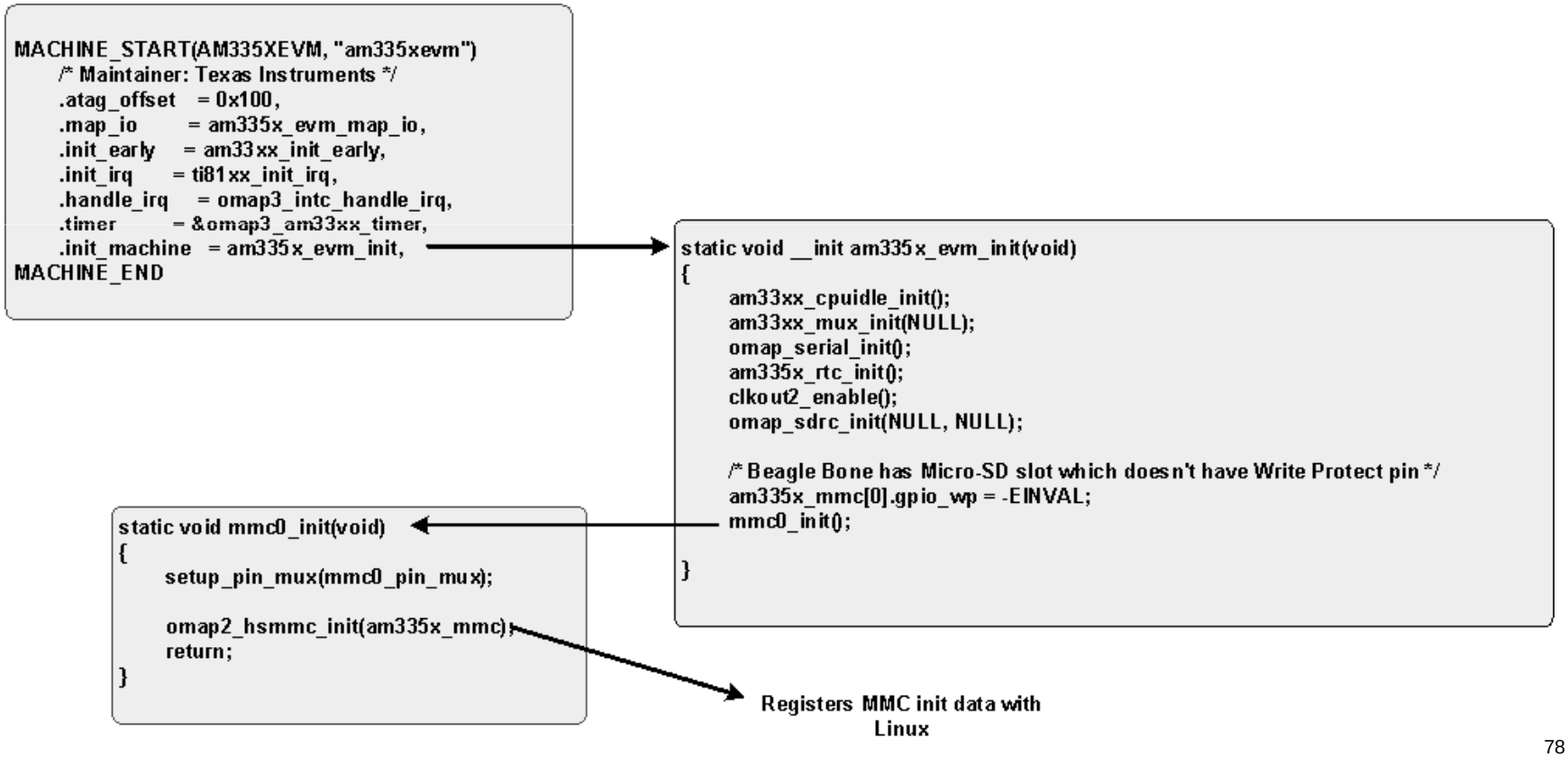
```
struct omap2_hsmmc_info {
  u8      mmc;      /* controller 1/2/3 */
  u32     caps;     /* 4/8 wires and any additional host
                     * capabilities OR'd (ref. linux/mmc/host.h) */
  .
  .
  int     gpio_cd;  /* or -EINVAL */   <Card Detect>
  int     gpio_wp;  /* or -EINVAL */   <Write Protect>
  .
  int     ocr_mask; /* temporary HACK */ <voltage range for slot>
  .
};
```

arch/arm/mach-omap2/hsmmc.h – omap2_hsmmc_info

include/linux/mmc/host.h – capabilities definitions and voltage range definitions

Initialization Function Call Sequence for MMC Enabling

- This sequence of code is adding in the MMC initialization code to the template file.



mmc0 initialization – did it work?

Did mmc0 messages show up in the console log or dmesg log?

```
[    1.040191] Waiting for root device /dev/mmcblk0p2...
[    1.078430] mmc0: host does not support reading read-only switch, assuming write-enable.
[    1.089355] mmc0: new high speed SDHC card at address 1234
[    1.095764] mmcblk0: mmc0:1234 SA04G 3.63 GiB
[    1.102752]  mmcblk0: p1 p2
[    1.153137] kjournald starting.  Commit interval 5 seconds
```

Did mmc0 show up in sysfs?

```
root@am335x-evm:~# ls -la /sys/devices/platform/omap/omap_hsmmc.0/
drwxr-xr-x    4 root     root            0 Dec 28 09:46 .
drwxr-xr-x   29 root     root            0 Dec 28 09:46 ..
lrwxrwxrwx    1 root     root            0 Dec 28 09:46 driver -> ../../../../bus/platform/drivers/omap_hsmmc
drwxr-xr-x    3 root     root            0 Dec 28 09:46 mmc_host
-r--r--r--    1 root     root         4096 Dec 28 09:52 modalias
drwxr-xr-x    2 root     root            0 Dec 28 09:52 power
lrwxrwxrwx    1 root     root            0 Dec 28 09:46 subsystem -> ../../../../bus/platform
-rw-r--r--    1 root     root         4096 Dec 28 09:46 uevent
```

Just for curiousity sake... did the root file system mount to mmc?

```
root@am335x-evm:~# mount
rootfs on / type rootfs (rw)
/dev/root on / type ext3 (rw,relatime,errors=continue,barrier=1,data=ordered)
proc on /proc type proc (rw,relatime)
tmpfs on /mnt/.splash type tmpfs (rw,relatime,size=40k)
sysfs on /sys type sysfs (rw,relatime)
none on /dev type tmpfs (rw,relatime,size=1024k,nr_inodes=8192,mode=755)
/dev/mmcblk0p2 on /media/mmcblk0p2 type ext3 (rw,relatime,errors=continue,barrier=1,data=ordered)
/dev/mmcblk0p1 on /media/mmcblk0p1 type vfat (rw,relatime,fmask=0022,dmask=0022,codepage=cp437,iocharset=iso8859-1,shortname=mixed,errors=remount-ro)
devpts on /dev/pts type devpts (rw,relatime,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw,relatime)
tmpfs on /var/volatile type tmpfs (rw,relatime,size=16384k)
tmpfs on /dev/shm type tmpfs (rw,relatime,mode=777)
tmpfs on /media/ram_type tmpfs (rw,relatime,size=16384k)
```

git diff – Code Difference between template and mmc commit

- Code for MMC setup
- This code was extracted from Beagle Bone specific code from the SDK release.
- git tag result for linux board port tree
- git diff command for this commit

```
schuyler@morpheus:~/bp_linux/sitara-board-port-linux$ git tag
05.04.01.00-backlight
05.04.01.00-base
05.04.01.00-ethernet
05.04.01.00-lcd
05.04.01.00-mmc
05.04.01.00-remove-board-file
05.04.01.00-template
05.04.01.00-touchscreen
```

```
diff --git a/arch/arm/mach-omap2/board-am335xevm.c b/arch/arm/mach-omap2/board-am335xevm.c
index 22c2d71..c94e063 100644
--- a/arch/arm/mach-omap2/board-am335xevm.c
+++ b/arch/arm/mach-omap2/board-am335xevm.c
@@ -65,6 +65,25 @@
 /* Header for code common to all OMAP2+ machines. */
 #include "common.h"

+/* Convert GPIO signal to GPIO pin number */
+#define GPIO_TO_PIN(bank, gpio) (32 * (bank) + (gpio))
+
+static struct omap2_hsmmc_info am335x_mmc[] __initdata = {
+       {
+               .mmc            = 1,
+               .caps           = MMC_CAP_4_BIT_DATA,
+               .gpio_cd        = GPIO_TO_PIN(0, 6),
+               .gpio_wp        = GPIO_TO_PIN(3, 18),
+               .ocr_mask       = MMC_VDD_32_33 | MMC_VDD_33_34, /* 3V3 */
+       },
+       {
+               .mmc            = 0,    /* will be set at runtime */
+       },
+       {
+               .mmc            = 0,    /* will be set at runtime */
+       },
+       {}      /* Terminator */
+};

 /* module pin mux structure */
@@ -73,6 +92,19 @@ struct pinmux_config {
        int val; /* Options for the mux register value */
 };

+/* Module pin mux for mmc0 */
+static struct pinmux_config mmc0_pin_mux[] = {
:
```

```
git diff 05.04.01.00-template..05.04.01.00-mmc
```

git diff – Code Difference between template and mmc commit (cont)

- Code for MMC setup
- Pin mux was started on previous page
- This code was extracted from Beagle Bone specific code from the SDK release.

```
+       {"mmc0_dat3.mmc0_dat3", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
+       {"mmc0_dat2.mmc0_dat2", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
+       {"mmc0_dat1.mmc0_dat1", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
+       {"mmc0_dat0.mmc0_dat0", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
+       {"mmc0_clk.mmc0_clk",   OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
+       {"mmc0_cmd.mmc0_cmd",   OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
+       {"mcasp0_aclkr.mmc0_sdwp", OMAP_MUX_MODE7 | AM33XX_PIN_INPUT_PULLUP},
+       {"spi0_cs1.mmc0_sdcd",  OMAP_MUX_MODE7 | AM33XX_PIN_INPUT_PULLUP},
+       {NULL, 0},
+};
+
 /*
 * @pin_mux - single module pin-mux structure which defines pin-mux
 *                       details for all its pins.
@@ -92,6 +124,14 @@ static struct pinmux_config clkout2_pin_mux[] = {
        {NULL, 0},
 };

+static void mmc0_init(void)
+{
+       setup_pin_mux(mmc0_pin_mux);
+
+       omap2_hsmmc_init(am335x_mmc);
+       return;
+}
+
 static void __init clkout2_enable(void)
 {
        struct clk *ck_32;
@@ -233,6 +273,10 @@ static void __init am335x_evm_init(void)
     clkout2_enable();
     omap_sdrc_init(NULL, NULL);

+   /* Beagle Bone has Micro-SD slot which doesn't have Write Protect pin */
+    am335x_mmc[0].gpio_wp = -EINVAL;
+      mmc0_init();
:
```

git diff – Code Difference between template and mmc commit (cont)

- Code for MMC setup
- Note this looks like a repeat from previous page, only these lines are different…
- How is mmc0_init() called?
- This code was extracted from Beagle Bone specific code from the SDK release.
- use “q” to quit

```
+       {"mmc0_clk.mmc0_clk",   OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
+       {"mmc0_cmd.mmc0_cmd",   OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
+       {"mcasp0_aclkr.mmc0_sdwp", OMAP_MUX_MODE7 | AM33XX_PIN_INPUT_PULLUP},
+       {"spi0_cs1.mmc0_sdcd",  OMAP_MUX_MODE7 | AM33XX_PIN_INPUT_PULLUP},
+       {NULL, 0},
+};
+
 /*
 * @pin_mux - single module pin-mux structure which defines pin-mux
 *                       details for all its pins.
@@ -92,6 +124,14 @@ static struct pinmux_config clkout2_pin_mux[] = {
        {NULL, 0},
 };

+static void mmc0_init(void)
+{
+       setup_pin_mux(mmc0_pin_mux);
+
+       omap2_hsmmc_init(am335x_mmc);
+       return;
+}
+
 static void __init clkout2_enable(void)
 {
        struct clk *ck_32;
@@ -233,6 +273,10 @@ static void __init am335x_evm_init(void)
     clkout2_enable();
     omap_sdrc_init(NULL, NULL);

+   /* Beagle Bone has Micro-SD slot which doesn't have Write Protect pin */
+    am335x_mmc[0].gpio_wp = -EINVAL;
+       mmc0_init();
+
 }

 static void __init am335x_evm_map_io(void)
(END)
```

Do LAB 2.....

Lab 2 Summary

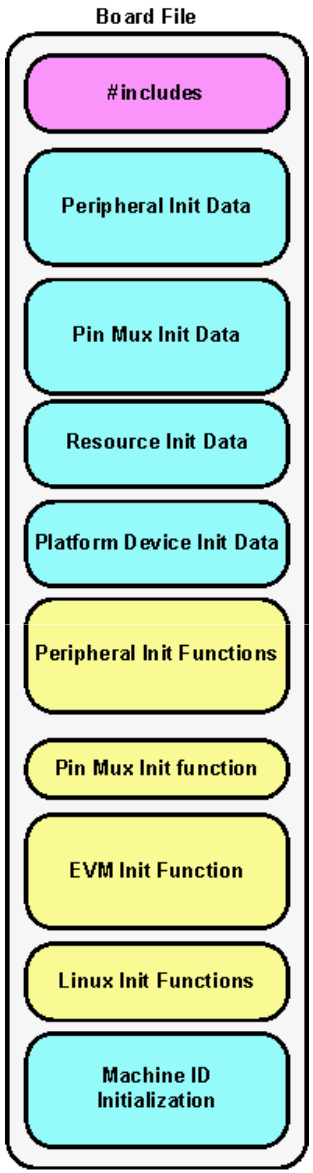
- Added code to the board port template file to handle pin mux, MMC controller initialization and evm initialization function.
- All changes happened within the board file

Linux Board Port Exercise 3 - Overview

- Goal: Build onto the template file again adding Ethernet for Network connectivity
- How this is demonstrated:
  - Using the lab git tree branch with the code additions necessary to enable Ethernet
  - With Ethernet enabled Remote Matrix will be brought up on the browser on the Host machine
- What is being done:
  - Explaining the code addition components (in multiple files this time)
- Perform the Lab

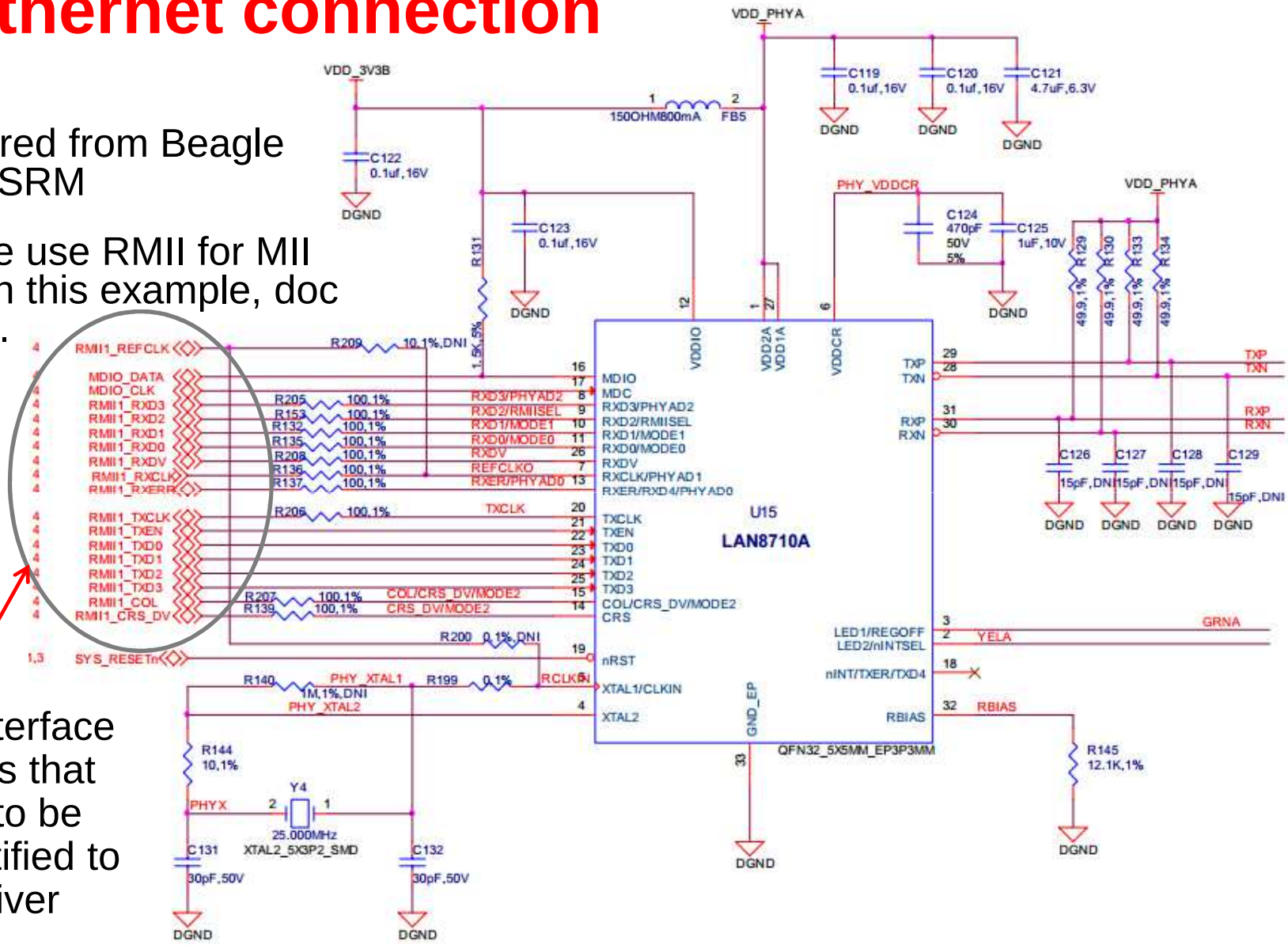
Steps to adding Ethernet to target board file

- Review system info to see how peripheral is attached
- Pin Mux
  - Use the Pin Mux Utility to configure Pin Init data
- Device/Platform Initialization data
  - None required for this integration
- Create Device Init function
  - Additional Init code required outside the board file
- Add Device Init Function to EVM Init Function



MII Ethernet connection

- Captured from Beagle Bone SRM
- Please use RMII for MII here in this example, doc bug….
- MII Interface signals that need to be indentified to the driver



Pin Mux Utility and pinmux config struct

- Pin Mux tool capture for MII interface
- While the tool shows GMII this is the MII interface, doc bug in tool

Pad Config.	Bot/Top Ball	IO Power	Mode 0	Mode 1
I IEN PD	H16 / -	VDDSHV5=3.3V	GMII1_COL	RMII2_REF
I IEN PD	H17 / -	VDDSHV5=3.3V	GMII1_CRS	RMII1_CRS
I IEN PD	J15 / -	VDDSHV5=3.3V	GMII1_RXER	RMII1_RXE
O IDIS PD	J16 / -	VDDSHV5=3.3V	GMII1_TXEN	RMII1_TXEN
I IEN PD	J17 / -	VDDSHV5=3.3V	GMII1_RXDV	LCD_MEM(
O IDIS PD	J18 / -	VDDSHV5=3.3V	GMII1_TXD3	DCAN0_TX
O IDIS PD	K15 / -	VDDSHV5=3.3V	GMII1_TXD2	DCAN0_RX
O IDIS PD	K16 / -	VDDSHV5=3.3V	GMII1_TXD1	RMII1_TXD1
O IDIS PD	K17 / -	VDDSHV5=3.3V	GMII1_TXD0	RMII1_TXD(
I IEN PD	K18 / -	VDDSHV5=3.3V	GMII1_TXCLK	UART2_RXD
I IEN PD	L18 / -	VDDSHV5=3.3V	GMII1_RXCLK	UART2_TXD
I IEN PD	L17 / -	VDDSHV5=3.3V	GMII1_RXD3	UART3_RXD
I IEN PD	L16 / -	VDDSHV5=3.3V	GMII1_RXD2	UART3_TXD
I IEN PD	L15 / -	VDDSHV5=3.3V	GMII1_RXD1	RMII1_RXD
I IEN PD	M16 / -	VDDSHV5=3.3V	GMII1_RXD0	RMII1_RXD
IO IEN PD	H18 / -	VDDSHV5=3.3V	RMII1_REFCLK	XDMA_EVE
IO IEN PU	M17 / -	VDDSHV5=3.3V	MDIO_DATA	TIMER6_MU
O IDIS PU	M18 / -	VDDSHV5=3.3V	MDIO_CLK	TIMER5_MU

Pin Mux definition for Ethernet using the MII interface

```
/* Module pin mux for mii1 */
static struct pinmux_config mii1_pin_mux[] = {
        {"mii1_rxerr.mii1_rxerr", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLDOWN},
        {"mii1_txen.mii1_txen", OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT},
        {"mii1_rxdv.mii1_rxdv", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLDOWN},
        {"mii1_txd3.mii1_txd3", OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT},
        {"mii1_txd2.mii1_txd2", OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT},
        {"mii1_txd1.mii1_txd1", OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT},
        {"mii1_txd0.mii1_txd0", OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT},
        {"mii1_txclk.mii1_txclk", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLDOWN},
        {"mii1_rxclk.mii1_rxclk", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLDOWN},
        {"mii1_rxd3.mii1_rxd3", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLDOWN},
        {"mii1_rxd2.mii1_rxd2", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLDOWN},
        {"mii1_rxd1.mii1_rxd1", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLDOWN},
        {"mii1_rxd0.mii1_rxd0", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLDOWN},
        {"mdio_data.mdio_data", OMAP_MUX_MODE0 | AM33XX_PIN_INPUT_PULLUP},
        {"mdio_clk.mdio_clk", OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT_PULLUP},
        {NULL, 0},
};
```

- This demonstrates how the Pin Mux utility can assist in filling out the pinmux_config structure

devices.c - code addition outside of board file

- Reason - This is code added to devices.c to supplement existing am33x_cpsw_init, does not require eeprom support.
- Reads the MAC IDs
- Sets the PHY type
- Registers MDIO
- Register CPSW with Linux kernel

Added this function to devices.c to initialize ethernet peripheral, not a TI target board

```
void am33xx_cpsw_init_generic(unsigned int phy_type, unsigned int gigen)
{
	u32 mac_lo, mac_hi;

	mac_lo = omap_ctrl_readl(TI81XX_CONTROL_MAC_ID0_LO);
	mac_hi = omap_ctrl_readl(TI81XX_CONTROL_MAC_ID0_HI);
	am33xx_cpsw_slaves[0].mac_addr[0] = mac_hi & 0xFF;
	am33xx_cpsw_slaves[0].mac_addr[1] = (mac_hi & 0xFF00) >> 8;
	am33xx_cpsw_slaves[0].mac_addr[2] = (mac_hi & 0xFF0000) >> 16;
	am33xx_cpsw_slaves[0].mac_addr[3] = (mac_hi & 0xFF000000) >> 24;
	am33xx_cpsw_slaves[0].mac_addr[4] = mac_lo & 0xFF;
	am33xx_cpsw_slaves[0].mac_addr[5] = (mac_lo & 0xFF00) >> 8;

	mac_lo = omap_ctrl_readl(TI81XX_CONTROL_MAC_ID1_LO);
	mac_hi = omap_ctrl_readl(TI81XX_CONTROL_MAC_ID1_HI);
	am33xx_cpsw_slaves[1].mac_addr[0] = mac_hi & 0xFF;
	am33xx_cpsw_slaves[1].mac_addr[1] = (mac_hi & 0xFF00) >> 8;
	am33xx_cpsw_slaves[1].mac_addr[2] = (mac_hi & 0xFF0000) >> 16;
	am33xx_cpsw_slaves[1].mac_addr[3] = (mac_hi & 0xFF000000) >> 24;
	am33xx_cpsw_slaves[1].mac_addr[4] = mac_lo & 0xFF;
	am33xx_cpsw_slaves[1].mac_addr[5] = (mac_lo & 0xFF00) >> 8;

	__raw_writel(phy_type,
			AM33XX_CTRL_REGADDR(MAC_MII_SEL));

	memcpy(am33xx_cpsw_pdata.mac_addr,
			am33xx_cpsw_slaves[0].mac_addr, ETH_ALEN);
	platform_device_register(&am33xx_cpsw_mdiodevice);
	platform_device_register(&am33xx_cpsw_device);
	clk_add_alias(NULL, dev_name(&am33xx_cpsw_mdiodevice.dev),
			NULL, &am33xx_cpsw_device.dev);
}
```

Ethernet Device Init and EVM Init functions

- The MII init function – call pin mux setup.

Ethernet Initialization function

```
static void mii1_init(void)
{
        setup_pin_mux(mii1_pin_mux);
        return;
}
```

EVM Init function (simplified for discussion purposes, just the added part for Ethernet)

```
/* Called as part of board initialization, defined in MACHINE_START */
static void __init am335x_evm_init(void)
{
.
.
      mii1_init();
      am33xx_cpsw_init_generic(MII_MODE_ENABLE,gigabit_enable);

}
```

- The EVM init function – calls mii1_init and the cpsw init function.

Ethernet Initialization – Did it work?

Did the PHY message show in the console or dmesg log?

```
[   12.288146]
[   12.288177] CPSW phy found : id is : 0x7c0f1
[   12.294921] PHY 0:01 not found
eth0      no wireless extensions.

udhcpc (v1.13.2) started
Sending discover...
[   15.288574] PHY: 0:00 - Link is Up - 100/-ull
Sending discover...
Sending select for 128.247.107.4...
Lease of 128.247.107.4 obtained, lease time 28800
adding dns 128.247.5.10
adding dns 157.170.147.7
```

Was an IP address obtained?

```
root@am335x-evm:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 40:5F:C2:76:86:1A
          inet addr:128.247.107.4  Bcast:0.0.0.0  Mask:255.255.254.0
          UP BROADCAST RUNNING ALLMULTI MULTICAST  MTU:1500  Metric:1
          RX packets:14495 errors:0 dropped:5377 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1538756 (1.4 MiB)  TX bytes:1180 (1.1 KiB)
          Interrupt:40

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Can ping another machine on the network?

```
root@am335x-evm:~# ping 128.247.106.201
PING 128.247.106.201 (128.247.106.201): 56 data bytes
64 bytes from 128.247.106.201: seq=0 ttl=64 time=0.580 ms
64 bytes from 128.247.106.201: seq=1 ttl=64 time=0.244 ms
64 bytes from 128.247.106.201: seq=2 ttl=64 time=0.214 ms
64 bytes from 128.247.106.201: seq=3 ttl=64 time=0.183 ms
64 bytes from 128.247.106.201: seq=4 ttl=64 time=0.275 ms
```

Do LAB 3.....

Lab 3 summary

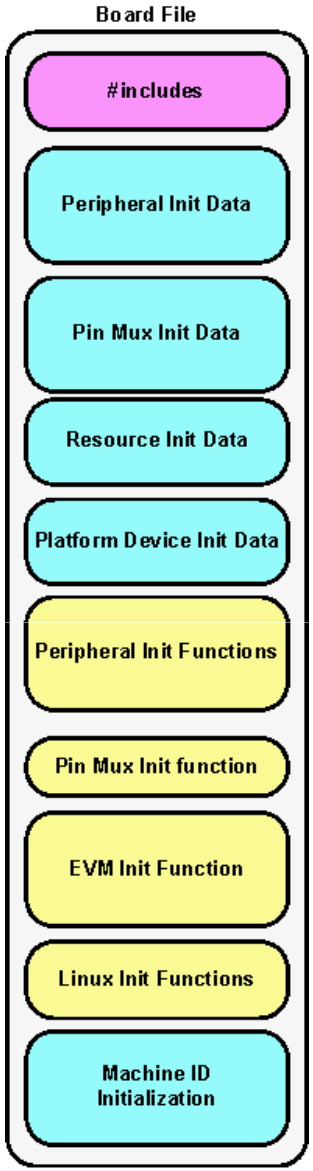
- Followed the steps of system attach review, pin mux config, device init to evm init
- Had to add additional code outside the board file to support initializing the cpsw for a generic case

Linux Board Port Exercise 4 - Overview

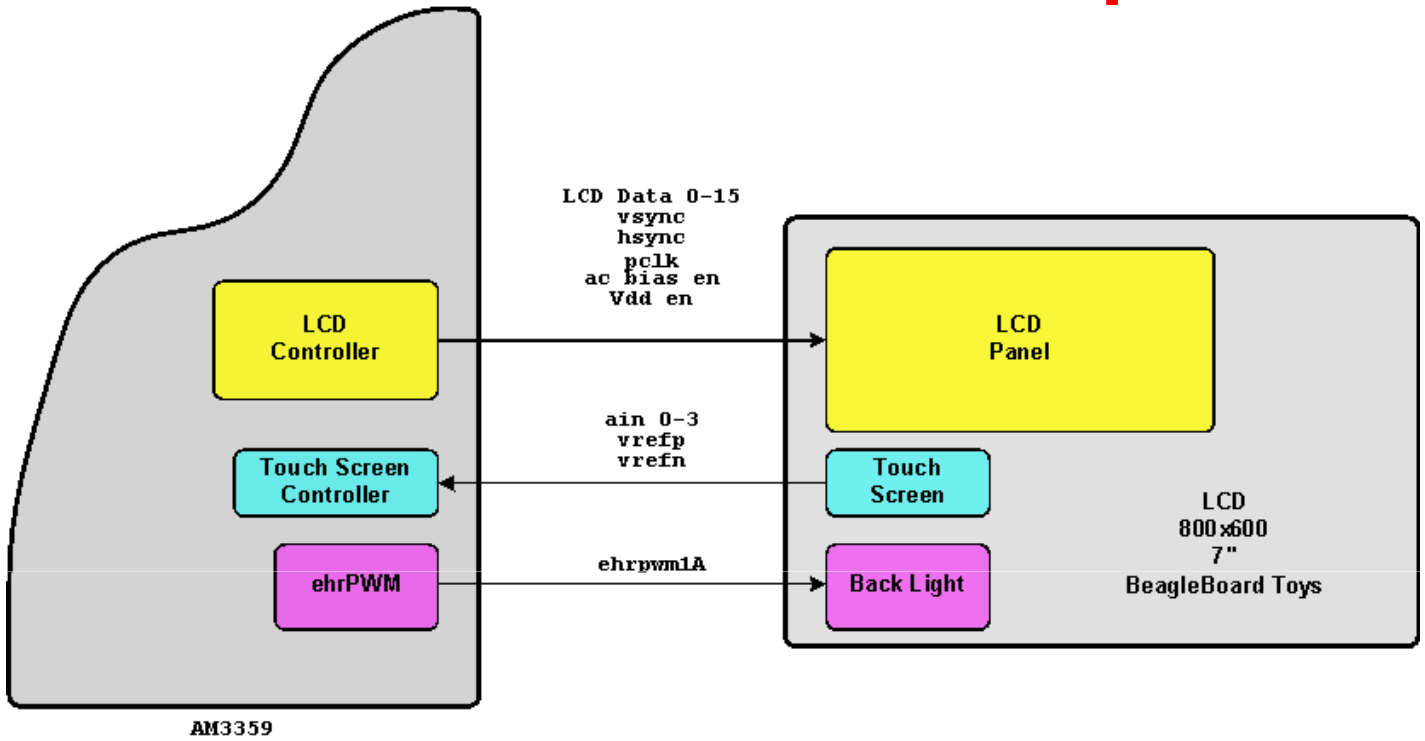
- Goal: Build onto the template file again adding support for an LCD panel
- How this is demonstrated:
  - Using the lab git tree tagged branch with code additions necessary to enable an LCD Panel
- What is being done:
  - Explaining the code addition components (multiple files this time)
- Perform the Lab

Steps to adding an LCD Panel to target board file

- Review the system
  - 3 interfaces used: PWM (backlight), LCD, Touch Screen
- Pin Mux
  - Use the Pin Mux Utility to configure Pin Init data
- Device/Platform Initialization data?
  - Backlight , LCD and Touch screen all have initialization data
- Create Device Init function initializes all 3 components
- Add Device init to board_init



LCD Panel Functional Components



- LCD is the same 7” panel currently found on the EVM
- The respective controllers require data initialization

LCD Panel Pin Mux Initialization

```
/* Module pin mux for Beagleboard 7" LCD cape */
static struct pinmux_config bbcape7_pin_mux[] = {
      {"lcd_data0.lcd_data0",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data1.lcd_data1",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data2.lcd_data2",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data3.lcd_data3",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data4.lcd_data4",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data5.lcd_data5",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data6.lcd_data6",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data7.lcd_data7",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data8.lcd_data8",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data9.lcd_data9",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data10.lcd_data10",     OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data11.lcd_data11",     OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data12.lcd_data12",     OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data13.lcd_data13",     OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data14.lcd_data14",     OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_data15.lcd_data15",     OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT
            | AM33XX_PULL_DISA},
      {"lcd_vsync.lcd_vsync",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT},
      {"lcd_hsync.lcd_hsync",       OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT},
      {"lcd_pclk.lcd_pclk",         OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT},
      {"lcd_ac_bias_en.lcd_ac_bias_en", OMAP_MUX_MODE0 | AM33XX_PIN_OUTPUT},
      {"ecap0_in_pwm0_out.gpio0_7", OMAP_MUX_MODE7 | AM33XX_PIN_OUTPUT}, // AVDD_EN
      {NULL, 0},
};
```

Pad Config.	Bot/Top Ball	IO Power	Mode 0
IO IEN OFF	R1 / -	VDDSHV6=3.3V	LCD_DATA0
IO IEN OFF	R2 / -	VDDSHV6=3.3V	LCD_DATA1
IO IEN OFF	R3 / -	VDDSHV6=3.3V	LCD_DATA2
IO IEN OFF	R4 / -	VDDSHV6=3.3V	LCD_DATA3
IO IEN OFF	T1 / -	VDDSHV6=3.3V	LCD_DATA4
IO IEN OFF	T2 / -	VDDSHV6=3.3V	LCD_DATA5
IO IEN OFF	T3 / -	VDDSHV6=3.3V	LCD_DATA6
IO IEN OFF	T4 / -	VDDSHV6=3.3V	LCD_DATA7
IO IEN OFF	U1 / -	VDDSHV6=3.3V	LCD_DATA8
IO IEN OFF	U2 / -	VDDSHV6=3.3V	LCD_DATA9
IO IEN OFF	U3 / -	VDDSHV6=3.3V	LCD_DATA10
IO IEN OFF	U4 / -	VDDSHV6=3.3V	LCD_DATA11
IO IEN OFF	V2 / -	VDDSHV6=3.3V	LCD_DATA12
IO IEN OFF	V3 / -	VDDSHV6=3.3V	LCD_DATA13
IO IEN OFF	V4 / -	VDDSHV6=3.3V	LCD_DATA14
IO IEN OFF	T5 / -	VDDSHV6=3.3V	LCD_DATA15
O IDIS OFF	U5 / -	VDDSHV6=3.3V	LCD_VSYNC
O IDIS OFF	R5 / -	VDDSHV6=3.3V	LCD_HSYNC
O IDIS OFF	V5 / -	VDDSHV6=3.3V	LCD_PCLK
O IDIS OFF	R6 / -	VDDSHV6=3.3V	LCD_AC_BIAS_EN

- Pin Mux Tool capture for the LCD Panel

LCD Touch Screen Pin Mux Initialization

- Pin Mux Capture of Pins used for Touch Screen
- 4 Wire Resistive touch
- 2 Wire for Voltage reference
- Pin connections are determined by schematic reference

Pad Config.	Bot/Top Ball	IO Power	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
I IEN OFF	A7 / -	VDDA ADC=	AIN3							
I IEN OFF	B7 / -	VDDA ADC=	AIN2							
I IEN OFF	C7 / -	VDDA ADC=	AIN1							
I IEN OFF	B6 / -	VDDA ADC=	AIN0							
I IEN OFF	B9 / -	VDDA ADC=	VREFP							
I IEN OFF	A9 / -	VDDA ADC=	VREFN							

```
/* Module pin mux for touchscreen controller */
static struct pinmux_config tsc_pin_mux[] = {
        {"ain0.ain0",           OMAP_MUX_MODE0 | AM33XX_INPUT_EN},
        {"ain1.ain1",           OMAP_MUX_MODE0 | AM33XX_INPUT_EN},
        {"ain2.ain2",           OMAP_MUX_MODE0 | AM33XX_INPUT_EN},
        {"ain3.ain3",           OMAP_MUX_MODE0 | AM33XX_INPUT_EN},
        {"vrefp.vrefp",         OMAP_MUX_MODE0 | AM33XX_INPUT_EN},
        {"vrefn.vrefn",         OMAP_MUX_MODE0 | AM33XX_INPUT_EN},
        {NULL, 0},
};
```

LCD Back Light Pin Mux Initialization

- Just a single pin used for the backlight.
- This is a pwm signal that is used to control brightness

```
/* Module pin mux for LCD backlight */
static struct pinmux_config ehrpwm_pin_mux[] = {
        {"gpmc_a2.ehrpwm1A", OMAP_MUX_MODE6 | AM33XX_PIN_OUTPUT},
        {NULL, 0},
};
```

Pad Config.	Bot/Top Ball	IO Power	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
0_IDIS PD	U14 / -	VDDSHV3=3.3V	GPMC_A2_MUX0	GMII2_TXD3	RGMII2_TD3	MMC2_DAT1_M...	GPMC_A18_MUX0	PR1_MII1_TXD2	EHRPWM1A_MU...	GPIO1[18]

Add LCD Panel – Data Initialization

```
/* Define display configuration */
static struct lcd_ctrl_config bbcape7_cfg = {
      &bbcape7_panel,
      .ac_bias          = 255,
      .ac_bias_intrpt         = 0,
      .dma_burst_sz           = 16,
      .bpp              = 16,
      .fdd              = 0x80,
      .tft_alt_mode           = 0,
      .stn_565_mode           = 0,
      .mono_8bit_mode         = 0,
      .invert_line_clock      = 1,
      .invert_frm_clock = 1,
      .sync_edge        = 0,
      .sync_ctrl        = 1,
      .raster_order           = 0,
};
```

- This configures the registers in the LCD Controller.
- The datasheet for LCD will provide information (to name a few)
  - BPP
  - Clock polarity
  - Data Format
  - DMA

```
struct lcd_ctrl_config {
      const struct display_panel *p_disp_panel;

      /* AC Bias Pin Frequency */
      int ac_bias;
      /* AC Bias Pin Transitions per Interrupt */
      int ac_bias_intrpt;
      /* DMA burst size */
      int dma_burst_sz;
      /* Bits per pixel */
      int bpp;
      /* FIFO DMA Request Delay */
      int fdd;
      /* TFT Alternative Signal Mapping (Only for active) */
      unsigned char tft_alt_mode;
      /* 12 Bit Per Pixel (5-6-5) Mode (Only for passive) */
      unsigned char stn_565_mode;
      /* Mono 8-bit Mode: 1=D0-D7 or 0=D0-D3 */
      unsigned char mono_8bit_mode;
      /* Invert line clock */
      unsigned char invert_line_clock;
      /* Invert frame clock  */
      unsigned char invert_frm_clock;
      /* Horizontal and Vertical Sync Edge: 0=rising 1=falling */
      unsigned char sync_edge;
      /* Horizontal and Vertical Sync: Control: 0=ignore */
      unsigned char sync_ctrl;
      /* Raster Data Order Select: 1=Most-to-least 0=Least-to-most */
      unsigned char raster_order;
      /* DMA FIFO threshold */
      int fifo_th;
};
```

LCD Panel Initialization data used by the LCDC

```
/* ThreeFive S9700RTWV35TR */
    [2] = {
        .name = "TFC_S9700RTWV35TR_01B",
        .width = 800,
        .height = 480,
        .hfp = 39,
        .hbp = 39,
        .hsw = 47,
        .vfp = 13,
        .vbp = 29,
        .vsw = 2,
        .pxl_clk = 30000000,
        .invert_pxl_clk = 0,
    },
```

drivers/video/da8xx-fb.c

```
struct da8xx_panel {
        const char      name[25];       /* Full name <vendor>_<model> */
        unsigned short  width;
        unsigned short  height;
        int             hfp;            /* Horizontal front porch */
        int             hbp;            /* Horizontal back porch */
        int             hsw;            /* Horizontal Sync Pulse Width */
        int             vfp;            /* Vertical front porch */
        int             vbp;            /* Vertical back porch */
        int             vsw;            /* Vertical Sync Pulse Width */
        unsigned int    pxl_clk;        /* Pixel clock */
        unsigned char   invert_pxl_clk; /* Invert Pixel clock */
};
```

drivers/video/da8xx-fb.c

- LCD Panel interfacing numbers have to be added in the da8xx-fb.c if they are not alreay defined.
- These numbers are derived from the datasheet for the panel (to name a few)
  - Screen resolution
  - Timings
  - Pixel Clock and Polarity

Backlight Initialization Data

- PWM is used to control the LCD Panel Brightness

```
/* LCD backlight platform Data */
#define AM335X_BACKLIGHT_MAX_BRIGHTNESS        100
#define AM335X_BACKLIGHT_DEFAULT_BRIGHTNESS    50
#define AM335X_PWM_PERIOD_NANO_SECONDS         (1000000 * 5)
```

```
/* Setup pwm-backlight for bbtoys7lcd */
static struct platform_device bbtoys7lcd_backlight = {
      .name           = "pwm-backlight",
      .id             = -1,
      .dev            = {
            .platform_data  = &bbcape7lcd_backlight_data,
      }
};
```

```
static struct platform_pwm_backlight_data bbcape7lcd_backlight_data = {
      .pwm_id         = BBCAPE7LCD_PWM_DEVICE_ID,
      .ch             = -1,
      .max_brightness = AM335X_BACKLIGHT_MAX_BRIGHTNESS,
      .dft_brightness = AM335X_BACKLIGHT_DEFAULT_BRIGHTNESS,
      .pwm_period_ns  = AM335X_PWM_PERIOD_NANO_SECONDS,
};
```

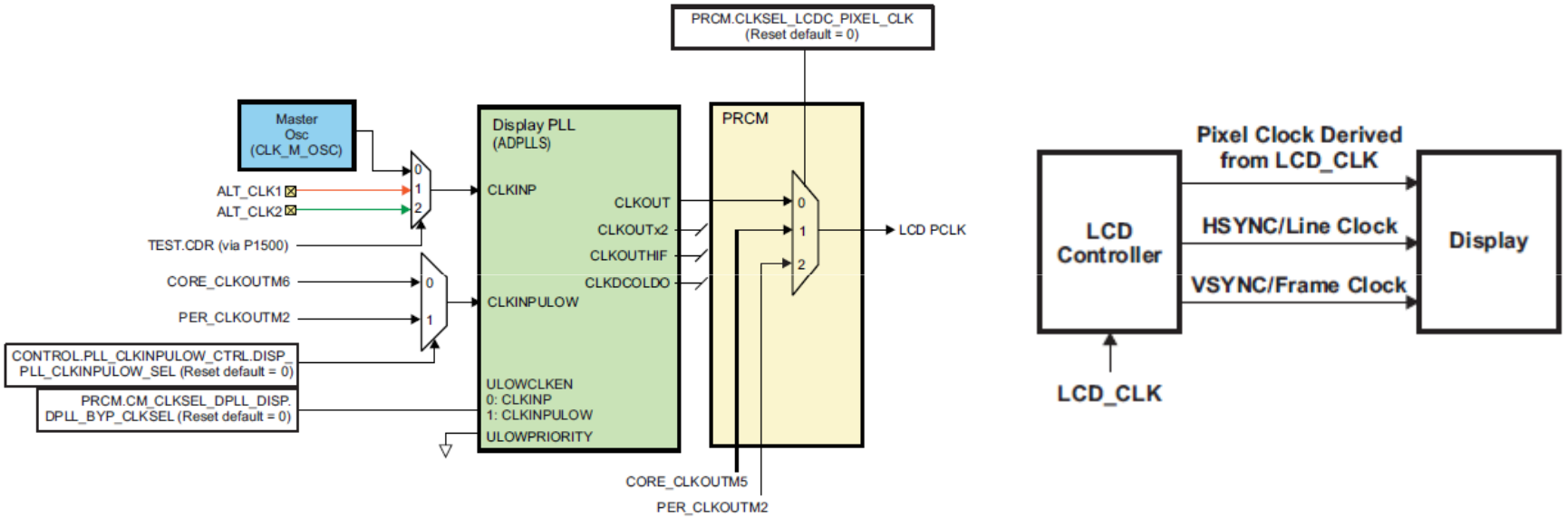
LCD Init Function

- The steps are:
  - Pin mux setup
  - Assign a GPIO to support VDD_en to the LCD
    - Refer to schematic on which to use
  - Define PLL value for the pixel clock
  - Register with the kernel

```
/* Configure display pll */
static int __init conf_disp_pll(int rate)
{
	struct clk *disp_pll;
	int ret = -EINVAL;

	disp_pll = clk_get(NULL, "dpll_disp_ck");
	if (IS_ERR(disp_pll)) {
		pr_err("Cannot clk_get disp_pll\n");
		goto out;
	}

	ret = clk_set_rate(disp_pll, rate);
	clk_put(disp_pll);
out:
	return ret;
}
```

```
/* Initialize and register lcdc device */
#define BEAGLEBONE_LCD_AVDD_EN GPIO_TO_PIN(0, 7)

static void bbcape7lcd_init(void)
{
	setup_pin_mux(bbcape7_pin_mux);
	gpio_request(BEAGLEBONE_LCD_AVDD_EN, "BONE_LCD_AVDD_EN");
	gpio_direction_output(BEAGLEBONE_LCD_AVDD_EN, 1);

	if (conf_disp_pll(300000000)) {
		pr_info("Failed to set pixclock to 300000000, not attempting to"
				"register LCD cape\n");
		return;
	}

	if (am33xx_register_lcdc(&bbcape7_pdata))
		pr_info("Failed to register Beagleboard LCD cape device\n");

	return;
}
```

LCD Clocking Layout



$$LCD\_PCLK = \frac{LCD\_CLK}{CLKDIV}$$

Touch Screen and Backlight Init Functions

- These init functions call the pin mux config function with the earlier defined initialized structures

```
/* Initialize and register tsc device */
static void tsc_init(void)
{
      int err;

      am335x_touchscreen_data.analog_input = 1;
      setup_pin_mux(tsc_pin_mux);
      err = am33xx_register_tsc(&am335x_touchscreen_data);
      if (err)
            pr_err("failed to register touchscreen device\
n");
}
```

```
/* Enable ehrpwm for backlight control */
static void enable_ehrpwm1(void)
{
      ehrpwm_backlight_enable = true;
      setup_pin_mux(ehrpwm_pin_mux);
}
```

LCD Init Sequence in the EVM Init function

- Calling three functions, initialization of
  - Backlight
  - LCD
  - touchscreen

```
/* Called as part of board initialization, defined in MACHINE_START */
static void __init am335x_evm_init(void)
{
.
.
      enable_ehrpwm1();
      bbcape7lcd_init();
      tsc_init();
.
.
}
```

Do LAB 4.....

Summary Lab 4

- LCD required 3 functions to be configured, Backlight, Touch Screen and LCDC
  - required device initialization data
  - required init functions
  - required pin mux configurations

- Made additions to the board file and the frame buffer support file

So…. does it work yet? Works Enough!

Thank You

Additional Information Sources for Post Workshop Review