Chapter Outline

- Workshop Goals
- Workshop Outline
- Where To Go For More Info
Workshop Goal and Objectives

“Get an introduction to Linux running on the AM3517 EVM.”

- Describe three basic parts of Linux
- List various Linux distro’s available for TI processors
- Describe where to find Linux distributions for TI processors (such as the AM3517)
- Chart boot process for TI processors running Linux
- Boot AM3517 to Gnome desktop; both local & remote
- Examine Linux networking
  - Set Linux network configurations
  - Setup and transfer via NFS and Samba
  - Host (and access) an HTTP server

Chapter Outline

- Workshop Goals
- Workshop Outline
- Where To Go For More Info
Lab Exercises

Lab 1:  Connect and boot the AM3517 EVM
Boot the AM3517 from an SD/MMC card.

Lab 2:  Start Gnome (Linux GUI) desktop running on the AM3517 EVM
There are four parts to this lab which correspond to the four different ways that you can invoke Gnome:
   a.  Automatically start GNOME on the AM3517 EVM's LCD
   b.  Manually start GNOME on the AM3517 EVM's LCD
   c.  Display the GNOME GUI remotely on the host PC via secure SSH connection
   d.  Export the GNOME GUI onto the host PC

Lab 3:  Running video (and audio) on the AM3517
   a.  Play a .wmv video movie trailer
   b.  (Take Home Lab) View video from a USB Webcam
   c.  (Take Home Lab) Listen to audio via USB speakers

Lab 4:  Investigating Linux Networking with the AM3517
   a.  Configure Linux Ethernet connection, both temporarily and permanently
   b.  Configure a Samba Server
   c.  Configure an HTML Server
Chapter Outline

- Workshop Goals
- Workshop Outline

Where can I get additional skills? (from TI)

Texas Instruments Curriculum

<table>
<thead>
<tr>
<th>Building Linux based Systems (ARM or ARM+DSP processors)</th>
<th>DaVinci / OMAP / Sitara System Integration Workshop using Linux (4-days)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="http://www.ti.com/training">www.ti.com/training</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Building BIOS based Systems (DSP processors)</th>
<th>System Integration Workshop using DSP/BIOS (4-days)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="http://www.ti.com/training">www.ti.com/training</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Developing Algo's for C6x DSP's (Are you writing/optimizing algorithms for latest C64x+ or C674x DSP's CPU's)</th>
<th>C6000 Optimization Workshop (4-days)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="http://www.ti.com/training">www.ti.com/training</a></td>
</tr>
</tbody>
</table>

Online Resources:

- OMAP / Sitara / DaVinci Wiki
  - [http://processors.wiki.ti.com](http://processors.wiki.ti.com)

- TI E2E Community (videos, forums, blogs)
  - [http://e2e.ti.com](http://e2e.ti.com)

- This workshop presentation & exercises

- DaVinci Open-Source Linux Mail List

- Gstreamer and other projects
  - [http://linux.davincidsp.com](http://linux.davincidsp.com) or [https://gforge.ti.com/gf/](https://gforge.ti.com/gf/)

- TI Software
Find These Workshop Materials


Texas Instruments
**Getting Started with Linux Workshop**

**Welcome**
- Workshop Goals
- Agenda
- Where to Get More Info

**Device/Tools Overview**
- TI Devices with Linux Support
- Hardware Development Platforms
- Software Development Kits

**Linux**
- What Is It?
- Where Do I Get It?
- How Do I Build It?

**Booting Linux**
- Tracing the Boot Process
- Customizing Boot for Your Board
- Das U-Boot

**Working In Linux**
- X-Windows
- GNOME
- Networking

---

**Outline**

- **TI Embedded Processors Portfolio**
- **What Processing Do You Need?** (CPU, Accel)
- **Peripherals**
- **Device Families : Sitara, DaVinci, C6L/C6A8**
- **Choosing a Device : Web Tool**
- **Tools Overview**
**Outline**

- **TI Embedded Processors Portfolio**
- **What Processing Do You Need? (CPU, Accel)**
  - ARM Core
  - DSP Core
  - Accelerators
- **Peripherals**
- **Device Families**: Sitara, DaVinci, C6L/C6A8
- **Choosing a Device**: Web Tool
- **Tools Overview**
What Types of Processing Do You Need?

For example, in an Audio/Video application, what needs to be done?

- User Controls, GUI, OSD
- Peripheral Drivers
- Ethernet (other system comm)
- Video processing decoding, encoding, etc.
- Audio processing decoding, encoding, etc.

Key System Blocks

An integrated solution that reduces System complexity, Power consumption, and Support costs

- **Low Power**
  - No heat sink or fan required. Ideal for end equipment that require air-tight, sealed enclosures

- **ARM Core**
  - High performance processors (375MHz - 1GHz) drive complex applications running on Linux, WinCE, or Android systems

- **Graphics Accelerator**
  - Provides rich image quality, faster graphics performance and flexible image display options for advanced user interfaces

- **‘C6x DSP Core**
  - Off-load algorithmic tasks from the ARM, freeing it to perform your applications more quickly
  - Allows real-time multi-media processing expected by users of today’s end-products
  - Think of the DSP as the ultimate, programmable hardware accelerator
  - Video Accelerators – either stand-alone or combined with the DSP provide today’s meet today’s video demands with the least power required

- **Peripherals**
  - Multiplicity of integrated peripheral options tailored for various wired or wireless applications – simplify your design and reduce overall costs

- **Display Subsystem**
  - Off-loads tasks from the ARM, allowing development of rich “iPhone-like” user interfaces including graphic overlays and resizing without the need for an extra graphics card

**NOTE**
Features not available on all devices
## Outline

- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
  - ARM Core
  - DSP Core
  - Accelerators
- Peripherals
- Device Families: Sitara, DaVinci, C6L/C6A8
- Choosing a Device: Web Tool
- Tools Overview

## TI ARM CPU Processor Cores

- Classic ARM Processors
- Embedded Cortex Processors
- Application Cortex Processors


TI's ARM core's supporting Linux...
ARM Processors: ARM+DSP

- ARM9 and Cortex-A8 provide the horsepower required to run high-level operating systems like: Linux, WinCE and Android
- ARM926 processor (375 – 450MHz) is the most popular and widely used processor in the market
- ARM Cortex™-A8 processor (600 MHz – 1.5 GHz) is ideal for high compute and graphic intense applications


TI Application Processors

ARM® & ARM+ Portfolio

Advanced GUIs & Displays, Connectivity, Video Acceleration

Advanced GUIs & Displays, Connectivity, Intensive Signal Processing

- SITARA™
- INTEGRA™
- DAVINCI™

ARM
- Point of Sale
- Portable Data Terminals
- Human Machine Interface

DSP + ARM
- Networked Audio
- Test and Measurement
- Power & Energy Management
- Industrial Vision & Control

- Networked Audio & Video
- Media Servers
- Video Security
- Video Conferencing
Outline

- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
  - ARM Core
  - DSP Core
  - Accelerators
- Peripherals
- Device Families: Sitara, DaVinci, C6L/C6A8
- Choosing a Device: Web Tool
- Tools Overview

C6000 DSP Family CPU Roadmap

Fixed Point

- C6x
  - L1 RAM/Cache
  - Compact Instructions
  - EDMA3
- C62x
- C621x
- C67x
- C671x
- C64x
  - Video/Imaging Enhanced
  - EDMA2
- C64x+
  - Fixed and Floating Point
  - Lower power
  - EDMA3
  - PRU

Floating Point

Available on the most recent releases
Outline

- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
  - ARM Core
  - DSP Core
  - Accelerators
- Peripherals
- Device Families: Sitara, DaVinci, C6L/C6A8
- Choosing a Device: Web Tool
- Tools Overview

Video ≠ Graphics

<table>
<thead>
<tr>
<th>Video</th>
<th>Graphics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Captured</td>
<td>Generated</td>
</tr>
<tr>
<td>Motion</td>
<td>Animation</td>
</tr>
<tr>
<td>• Integrated image</td>
<td>• Multi-layered image</td>
</tr>
<tr>
<td>• Base image + motion</td>
<td>• New delta images</td>
</tr>
<tr>
<td>YUV/YCbCr format</td>
<td>RGB format</td>
</tr>
<tr>
<td>• Convert to RGB for display</td>
<td>• Ready for display</td>
</tr>
<tr>
<td>Video codec: block-based</td>
<td>• Graphics: pixel-based</td>
</tr>
<tr>
<td>• MPEG, WMV, JPG, etc.</td>
<td>• BMP, GIF, etc.</td>
</tr>
<tr>
<td>Natural attributes</td>
<td>Calculated attributes</td>
</tr>
<tr>
<td>• Digitized analog image</td>
<td>• Synthesized digital image</td>
</tr>
</tbody>
</table>

- With modern technology one can approach video performance and resolution, but the integrated effect of video is yet to be simulated in graphics.
- With increased HD video resolution, hardware acceleration is required; higher resolution – as well as demand for smooth 3D rendering – drives need for graphics acceleration.
### Simple Graphics/Video Example

![Simple Graphics/Video Example Image](image)

- **Video:** Remotely captured/stored; locally decoded
- **Graphics:** Locally generated

### Hardware Accelerator Summary

<table>
<thead>
<tr>
<th>Devices</th>
<th>3D Graphics</th>
<th>Video/Imaging/Audio</th>
<th>Video Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM355</td>
<td>MJCP: JPEG, MPEG4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM365/368</td>
<td>MJCP &amp; HCP: JPEG, MPEG4, H264, VC1, Facial, MP3, AAC, WMA, AEC</td>
<td>OSD, Resizer, Previewer, H3A</td>
<td></td>
</tr>
<tr>
<td>AM3517</td>
<td>PowerVR SGX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AM3715</td>
<td>PowerVR SGX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DM6446</td>
<td>VICP: VidEnc, Image, Other</td>
<td>OSD, Resizer, Previewer, H3A</td>
<td></td>
</tr>
<tr>
<td>DM6467</td>
<td>HDVICP: Video Encoding</td>
<td>TSIF</td>
<td></td>
</tr>
<tr>
<td>OMAP3515</td>
<td>PowerVR SGX</td>
<td>IVA2.2 : Video Encoding</td>
<td>OSD, Resizer, Overlay Mgr</td>
</tr>
<tr>
<td>OMAP3525</td>
<td>PowerVR SGX</td>
<td>IVA2.2 : Video Encoding</td>
<td>OSD, Resizer, Overlay Mgr</td>
</tr>
<tr>
<td>OMAP3530</td>
<td>PowerVR SGX</td>
<td>IVA2.2 : Video Encoding</td>
<td>OSD, Resizer, Overlay Mgr</td>
</tr>
<tr>
<td>DM3725</td>
<td>PowerVR SGX</td>
<td>IVA2.2 : Video Encoding</td>
<td>OSD, Resizer, Overlay Mgr</td>
</tr>
<tr>
<td>DM3730</td>
<td>PowerVR SGX</td>
<td>IVA2.2 : Video Encoding</td>
<td>OSD, Resizer, Overlay Mgr</td>
</tr>
</tbody>
</table>

Note: “Hardware accelerated” algorithms require software support – which is provided by TI and TI 3P’s
Outline

- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
- Peripherals
  - Device Families
  - Choosing a Device
  - Tools Overview

### Peripherals

- Video/Display
- ARM
- Graphics Accelerator
- C6x DSP
- Video Accelerator(s)

### Peripherals

<table>
<thead>
<tr>
<th>Serial</th>
<th>Storage</th>
<th>Master</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>McBSP</td>
<td>DDR2</td>
<td>USB 1.1</td>
<td>Timers</td>
</tr>
<tr>
<td>McASP</td>
<td>SDRAM</td>
<td>USB 2.0</td>
<td>Watch</td>
</tr>
<tr>
<td>ASP</td>
<td>Async</td>
<td>EMAC</td>
<td>PWM</td>
</tr>
<tr>
<td>UART</td>
<td>SD/MMC</td>
<td>uPP</td>
<td>eCAP</td>
</tr>
<tr>
<td>SPI</td>
<td>ATA/CF</td>
<td>HPI</td>
<td>RTC</td>
</tr>
<tr>
<td>I2C</td>
<td>SATA</td>
<td>EDMA3</td>
<td></td>
</tr>
<tr>
<td>CAN</td>
<td>RAM</td>
<td>SCR</td>
<td>GPIO</td>
</tr>
</tbody>
</table>

### PRU (Soft Periperal)

- CAN
- UART
- What's Next?
- DIY...

### Video/Display Subsystem

- Capture
- Analog Display
- Digital Display
- LCD Controller
Outline

- TI Embedded Processors Portfolio
- System Examples
- What Processing Do You Need? (CPU, Accel)
- Peripherals
- Final Considerations
  - Device Families: Sitara, Integra, DaVinci
  - Choosing a Device: Web Tool

Processor Families Evolution

Processors share common Foundation Software and Tools to ensure Portability and Compatibility.
**Outline**

- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
- Peripherals
- Device Families: Sitara, DaVinci, C6L/C6A8
- Choosing a Device: Web Tool
- Tools Overview
Chapter 1: Devices and Tools

Outline

- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
- Peripherals
- Device Families: Sitara, DaVinci, C6L/C6A8
- Choosing a Device: Web Tool
- Tools Overview
  - Development Kits
  - Software Development Kits (DVSDK, SDK)
Outline

- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
- Peripherals
- Device Families: Sitara, DaVinci, C6L/C6A8
- Choosing a Device: Web Tool
- Tools Overview
  - Development Kits
  - Software Development Kits (DVSDK, SDK)

DM6446 DVEVM
Modular EVM Kits – AM3517 Example

SOM Module
AM3517 SOM-M2
Price: < $100
SW Development
1.6” x 2”
Features:
- 256 MB DDR2 SDRAM
- 512 MB NAND flash
- Wired Ethernet
- Wireless 802.11b/g/n* 
- Bluetooth 2.1 + EDR IF*
Self-boot Linux image
Purchase – Logic via Arrow, Avnet, Digikey
Support – Logic

eXperimenter Kit
SDK-XAM3517-10-256512R
Price: $199
S/W and H/W Dev’t
5” x 6”
Features SOM features +
- HDMI (video only)
- MMC/SD card slot
- Network/USB/Serial/JTAG/Logic-LCD Connectors
- Built-in XDS100 emulation
- Purchase – Logic via Arrow, Avnet, Digikey
- Support – Logic
- SW: Linux, WinCE

EVM
TMDXEVM3517
Price: $999
Full Development Platform
- EVM additionally includes:
  - LCD
  - Multimedia In/Out
  - KeyPad
  - Connect: CAN, RJ45, USB, UART, stacked SD
- Channel – TI & distribution
- Support – TI & Logic
- Linux and WinCE SDK’s (from TI); Android SDK is in development

Hardware Development Environments

4 Types of Hardware Development Tools

System-on- Module
Use Case
- Simplify system board design
- Medium for Prototype or Production end equipment

Community Board
Use Case
- Evaluation of processor functionality
- Application development with limited peripheral access

eXperimenter Kit
Use Case
- Evaluation of processor functionality
- Application development with limited peripheral access

Evaluation Module
Use Case
- Touch-screen application development with full peripheral access
- Application specific development
### Evaluation and Development Kits

**Development Kit Contents:**
- Evaluation board and documentation
- Software Development Kits
- Development Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Part Number</th>
<th>Price</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM37x EVM</td>
<td>TMDXEVM3715</td>
<td>$1495</td>
<td>TI / Mistral</td>
</tr>
<tr>
<td>AM/DM37x Eval Module</td>
<td>TMDX3730EVM</td>
<td>$1495</td>
<td>TI / Mistral</td>
</tr>
<tr>
<td>OMAP35x EVM</td>
<td>TMDSEVM3530</td>
<td>$1495</td>
<td>TI / Mistral</td>
</tr>
<tr>
<td>AM3517 EVM</td>
<td>TMDXEVM3517</td>
<td>$999</td>
<td>TI / Logic</td>
</tr>
<tr>
<td>AM18x EVM</td>
<td>TMDXEVM1808L</td>
<td>$1150</td>
<td>TI</td>
</tr>
<tr>
<td>OMAP-L138 EVM</td>
<td>TMDXOSKL138BET</td>
<td>$849</td>
<td>TI / Logic</td>
</tr>
<tr>
<td>AM17x EVM</td>
<td>TMDXEVM1707</td>
<td>$845</td>
<td>TI</td>
</tr>
<tr>
<td>AM18x Experimenter Kit</td>
<td>TMDXEXP1808L</td>
<td>$445</td>
<td>TI</td>
</tr>
</tbody>
</table>

### Community Boards & Modules

**To Access:** Contact TI partners for more information or click link to buy now

<table>
<thead>
<tr>
<th>Tool</th>
<th>Part Number</th>
<th>Price</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beagle Board (OMAP35x)</td>
<td>Beagle</td>
<td>$149</td>
<td>Community</td>
</tr>
<tr>
<td>Hawkboard (OMAP-L138)</td>
<td>ISSPLHawk</td>
<td>$89</td>
<td>Community</td>
</tr>
<tr>
<td>OMAP35x System on Module</td>
<td>OMAP35x SOM-LV</td>
<td>$99</td>
<td>Logic</td>
</tr>
<tr>
<td>Overo OMAP35x Computer on Module</td>
<td>Overo</td>
<td>$149-$219</td>
<td>Gumstix</td>
</tr>
<tr>
<td>KBOC OMAP35x System on Module</td>
<td>KBOC</td>
<td>$139</td>
<td>KwikByte</td>
</tr>
</tbody>
</table>

* Prices subject to change
Outline

- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
- Peripherals
- Device Families: Sitara, DaVinci, C6L/C6A8
- Choosing a Device: Web Tool
- Tools Overview
  - Development Kits
    - Software Development Kits (DVSDK, SDK)

Software Development Kits

<table>
<thead>
<tr>
<th>S/W Dev’l Kit</th>
<th>Description</th>
<th>Processor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux PSP SDK</td>
<td>Small Linux Distro supporting TI ARM devices</td>
<td>OMAP35, AM35, AM18, OMAP-L1, DM644x, DM6467, DM3xx</td>
</tr>
<tr>
<td>“DVSDK”</td>
<td>TI provided libraries, examples, demos</td>
<td>All TI SOC’s: ARM, DSP, ARM+DSP</td>
</tr>
<tr>
<td></td>
<td>Codec Engine (ViSA), DSPlink,</td>
<td>Obviously, not all devices require all the s/w components</td>
</tr>
<tr>
<td></td>
<td>Codecs/Algos (XDM), BIOS, XDC,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Linux utilities, etc.</td>
<td></td>
</tr>
<tr>
<td>Code Gen Tools</td>
<td>Linux GNU Compiler (CodeSourcery)</td>
<td>All TI ARM and DSP devices where appropriate</td>
</tr>
<tr>
<td>(not really “kits” per se)</td>
<td>C6000 DSP Compiler (TI)</td>
<td></td>
</tr>
<tr>
<td>Graphics SDK</td>
<td>Graphix SVSGX development kit</td>
<td>OMAP3515, OMAP3530</td>
</tr>
<tr>
<td></td>
<td>OPENGL ES / VG demos, drivers, targetfs,</td>
<td>AM3517</td>
</tr>
<tr>
<td></td>
<td>Getting Started Guide</td>
<td></td>
</tr>
</tbody>
</table>

* PSP is a TI specific acronym that represents the name of the group inside of Texas Instruments which “owns” the kernel and driver development activities: Platform Support Package team
* Wireless SDK is available independently of these other kits to support the TI WLxxxx Bluetooth/WiFi devices
Wireless SDK: Getting started with WL1271 on OMAP35x EVM

Software
- Pre-integrated with TI's SDK
- WLAN and Bluetooth® software support (FM support not included)
- Pre-tested against WiFi and Bluetooth® specifications
- Open Source Linux drivers
  - Kernel 2.6.x
  - TI WLAN driver
  - BlueZ Bluetooth® stack
- Windows® CE 6.0 drivers
  - Available in mid 2010
  - Microsoft WiFi and Bluetooth® stacks
  - Adeneo’s Bluetooth Manager

Hardware
- Wireless Connectivity Card
  - WL1271 module with integrated TCXO
  - 2.4GHz chip antenna (default configuration)
  - U.FL antenna connector (optional configuration)
  - Plugs into EVM's Expansion Connector (supported on EVM Rev G)
- Evaluated 802.11b/g/n and Bluetooth® capability, and begin SW development
- Included in EVM box: Mar 2010
- Standalone Connectivity Card upgrade available from Mistral

Development tools and partners
- Compatible with EVM’s toolchain
- Wireless Connectivity Card reference schematics
- Command Line Interface (CLI) to configure and exercise WLAN & Bluetooth® applications
- Partners
  - LS Research: WL1271 module
  - Mistral: Linux System Integrator
  - Adeneo: WinCE Syst. Integrator

Code Generation Tools: CodeSourcery
- TI SDK’s recommend the CodeSourcery Lite version, although you may want to upgrade to a more complete solution
- See Code Sourcery ARM/GNU Linux datasheet
  http://www.codesourcery.com/spip/datasheet?target_arch=ARM&amp;target_os=GNU%2FLinux
- Check SDK/DVSDK or Linux PSP release notes for tool versions

<table>
<thead>
<tr>
<th>Feature</th>
<th>Lite</th>
<th>Personal</th>
<th>Professional</th>
</tr>
</thead>
<tbody>
<tr>
<td>30-day Installation Support</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Unlimited Support</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Priority Defect Correction</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Access to Updates, Knowledge Base</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Big Endian, Neon support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GNU C/C++ Compiler</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GNU Debugger (GDB)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Eclipse IDE</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Annual Subscription Price (per Host)</td>
<td>Free</td>
<td>$399</td>
<td>$2799</td>
</tr>
</tbody>
</table>
## Getting Started with Linux Workshop

<table>
<thead>
<tr>
<th>Workshop Goals</th>
<th>Agenda</th>
<th>Where to Get More Info</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device/Tools Overview</strong></td>
<td><strong>TI Devices with Linux Support</strong></td>
<td><strong>Hardware Development Platforms</strong></td>
</tr>
<tr>
<td><strong>Software Development Kits</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Linux</strong></td>
<td><strong>What Is It?</strong></td>
<td><strong>Where Do I Get It?</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>How Do I Build It?</strong></td>
</tr>
<tr>
<td><strong>Booting Linux</strong></td>
<td><strong>Tracing the Boot Process</strong></td>
<td><strong>Customizing Boot for Your Board</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Das U-Boot</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Working In Linux</strong></td>
<td><strong>X-Windows</strong></td>
<td><strong>GNOME</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Networking</strong></td>
<td></td>
</tr>
</tbody>
</table>

Outline

- Linux : What is it?
- Linux : Where do I get it?
- TI Customer Support
- (Optional) How Do I Build Linux – and Why?
Linux in Three Parts

1. **Bootloader**
   - Provides rudimentary hw init
   - Calls Linux kernel and passes boot arguments

2. **Kernel**
   - Initializes the system (and device)
   - Manages system resources
   - Provides services for user programs

3. **Filesystem**
   - Single filesystem (/ root)
   - Stores all system files
   - After init, kernel looks to filesystem for "what's next"
   - bootarg tells linux where to find root filesystem

---

**Linux Boot Process**

- **Boot Loader**
  - U-Boot
    - ARM assembly code
    - Passes args to Linux (bootargs)

- **Linux Kernel**
  - Initialize hardware

- **File System**
  - Init Process
    - /sbin/init – 1st process exe by kernel
      - Login console
      - Usually one of first prog’s to run

Looking more closely at the Kernel ...

---

**Chapter 2 : Linux - What, Where and How to Build It**
What’s in the Linux Kernel

User Mode
- Custom User Application
- Gstreamer Framework
- 3D Graphics Library
- HTTP Server

Kernel Mode
- Process Management
- Memory Management
- Virtual File System
- Network Stack
- Device Management

- CPU support code
- CPU/MMU support code
- Storage device drivers
- Network device drivers
- Character device drivers

CPU | RAM | MMC | EMAC | UART

A few more details about the filesystem...

Red Hat / Ubuntu: Root File System

Some folders common to Linux:

/dev – Common location to list all device drivers

/home - Storage for user’s files
- Each user gets their own folder (e.g., /home/user)
- Similar to “My Documents” in Windows
- DVS/SDK GSG directory for TI tools, examples, working directory
- “root” user is different, that user’s folder is at /root

/media – Usually find CDROM drive(s) mounted here

/imnt – Common location to mount other file systems
- Linux only allows one filesystem
- Add other disks (physical, network, etc.) by mounting them to an empty directory in the root filesystem
- Windows adds new filesystems (C:, D:, etc.) rather than using a single one

/usr – Storage for user binaries
- X86 Compiler for Ubuntu programs (gcc) is stored in here
Filesystems: Red Hat vs. Montavista

Red Hat (PC)
- bin
- boot
- dev
- etc
- home
- initrd
- lib
- lost+found
- media
- mnt
- opt
- proc
- run
- srv
- sys
- tmp
- usr
- var

MontaVista (ARM)
- /target
  - bin
  - boot
  - dev
  - etc
  - home
  - lib
  - media
  - mnt
  - opt
  - proc
  - root
  - sbin
  - srv
  - sys
  - tmp
  - usr
  - var

- **Tools/Host** filesystem: location our dev't tools
- **Target** filesystem: filesystem to run on TI processors
- Notice the similarities between the two different Linux filesystems
- When releasing to production, it’s common to further reduce the target filesystem to eliminate cost

Outline

- **Linux**: What is it?
- **Linux**: Where do I get it?
  - What are Distributions?
  - O/S Choices
  - Community Options
  - Commercial Options
  - Commercial vs Community
- **TI Customer Support**
- **(Optional)** How Do I Build Linux – and Why?
Build It Yourself?

Quote from kernel.org:

If you’re new to Linux, you don’t want to download the kernel, which is just a component in a working Linux system. Instead, you want what is called a distribution of Linux, which is a complete Linux system.

There are numerous distributions available for download on the Internet as well as for purchase from various vendors; some are general-purpose, and some are optimized for specific uses.

- This may be a bit of an understatement – even experienced users usually use a distribution
- Creating a distribution takes a lot of effort
- Maintaining a distribution ... takes even more effort
- In fact, using a distribution even takes quite a bit of effort

What Is a ‘Linux Distribution’

A ‘Linux distribution’ is a combination of the components required to provide a working Linux environment for a particular platform:

1. **Linux kernel port**
   - A TI LSP or Linux PSP is a Linux kernel port to a device, not just a set of device drivers

2. **Bootloader**
   - Uboot is the standard bootloader for ARM Linux

3. **Linux ‘file system’**
   - This does NOT mean a specific type of file system like FAT file system or flash file system … rather, it more like the “C:" drive in Windows
   - It refers to all the ‘user mode’ software that an application needs such as graphics libraries, network applications, C run-time library (glibc, uclibc), codec engine, dynamically-loaded kernel modules (CMEM, DSPLINK)

4. **Development tools**
   - Code Sourcery - GCC, GDB
   - MV DevRocket, CCSv5 (beta), GHS Multi, etc.
Linux isn't complete without a distribution
- MontaVista and Timesys, for example, provide commercial (i.e. production) distribution for TI's DaVinci / OMAP processors
- A few distributions supporting the open-source BeagleBoard (OMAP35x-based) include: OpenEmbedded, Ubuntu, Fedora, Android, Gentoo, ARMedslack and ALIP

Outline
- Linux : What is it?
- Linux : Where do I get it?
  - What are Distributions?
    - O/S Choices
    - Community Options
    - Commercial Options
    - Commercial vs Community
- TI Customer Support
- (Optional) How Do I Build Linux – and Why?
O/S Choices

- Linux
- Android (a.rowboat.com)
- WinCE
- Others...
  - QNX
  - Nucleus
  - Etc.

Build It Yourself

Commercial

Community

Linux Distributions

<table>
<thead>
<tr>
<th>Commercial</th>
<th>Community</th>
<th>Custom (Build it Yourself)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Timesys</td>
<td>TI SDK (PSP)</td>
<td>Open Embedded (OE)</td>
</tr>
<tr>
<td>• MontaVista</td>
<td></td>
<td>Custom from Sources</td>
</tr>
<tr>
<td>• Etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Binary</td>
<td>Ángström</td>
<td></td>
</tr>
<tr>
<td>(Update patches)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ease of Use

- Easy
- Tested

Experienced User

Latest

"GIT" from kernel.org, and others
Outline

- Linux: What is it?
- Linux: Where do I get it?
  - What are Distributions?
  - O/S Choices
    - Community Options
  - Commercial Options
  - Commercial vs Community
- TI Customer Support
- (Optional) How Do I Build Linux – and Why?

Community Options

- TI Linux SDK (PSP)
  - Pre-built snapshot of Linux tested against specific version of
    TI Software Development Kits
  - Updated at each new SDK/DVSDK release
  - PSP = Platform Support Package (name of TI team)
  - Currently, a “BusyBox-based” bare-bones distro (“lean/mean”)
  - Arago open-source OE project
    - Advantage of OE – recipes can be reused by Angstrom
      (or custom OE) users
    - In general, users shouldn’t (re)build using OE; no reason to,
      because if you want more features, we recommend you go with
      Angstrom (also built using OE)

- Ångström
  ...
Community Options

- **TI Linux SDK (PSP)**
  - Pre-built snapshot of Linux tested against specific version of TI Software Development Kits
  - Updated at each new SDK/DVSDK release
  - PSP = Platform Support Package (name of TI team)
  - Currently, a "BusyBox-based" bare-bones distro ("lean/mean")
  - Arago open-source OE project
    - Advantage of OE – recipes can be reused by Angstrom (or custom OE) users
    - In general, users shouldn’t (re)build using OE; no reason to, because if you want more features, we recommend you go with Angstrom (also built using OE)

- **Ångström**
  - Open-source, full-featured Linux distro targeted for embedded systems
  - Get it from:
    - User-compiled binaries widely available for many targets
    - [Narcissus](http://www.angstrom-distribution.org/narcissus)
      - Web-based tool to create binary version (with your own package selection)
    - Built using OE (user community can re-use TI OE recipes)
DIY Options

- **Open-Embedded (OE)**
  - Build Linux from source using OE’s Bit-Bake recipe(s)
  - Many recipes available for various architectures, including many variations for a given device
  - Builds full-up distro including Kernel and Filesystem
  - TI builds its PSP Linux distro’s via OE

- **Build from Sources (roll your own)**
  - Build directly from sources, such as kernel.org
  - Use GIT, as well as SVN and others to get sources from repo’s
  - Are you nuts? Either you want to waste your life redoing what OE did, or you’re so advanced you don’t need this presentation.
TI’s New Linux Strategy

TI’s New Linux Strategy

- **uboot**
- **Linux application packages on internet**
- **TI Linux SDK**
- **TI Customers**

**TI DSPS Developers**

- **Code Sourcery**
- **Sourcery G++**

**OpenEmbedded**

- **Recipes, config files**
- **Commercial Linux OS and tools vendors**
- **TI Customers**

**kernel.org**

**Uboot**

- **Uboot ports**
- **Recipes, config files**

**Commercial Linux OS and tools vendors**

- **TI Customers**

**TI Device Linux Kernel Patching Methodology**

**Arago “Staging” Area**

http://arago-project.org/git/projects

- ‘TI ‘work’ area
- OMAP3x Linux patches
- OMAP-L1x Linux patches
- DaVinci Linux patches
- U-Boot patches
- Test scripts and framework
- Earliest customer access to patches

**DaVinci and OMAP Linux staging trees**

- Current Linux kernels plus recently accepted TI and third-party patches
- Small temporary deltas from mainline kernel

**Mainline Trees**

- ‘Official’ Linux kernels rel’s
- [mach-omap2]
- [mach-davinci]
- ‘Official’ U-Boot releases

**Accepted Patches**
Outline

◆ Linux: What is it?
◆ Linux: Where do I get it?
  • What are Distributions?
  • O/S Choices
  • Community Options
  • Commercial Options
  • Commercial vs Community
◆ TI Customer Support
◆ (Optional) How Do I Build Linux – and Why?

Commercial O/S Vendors

◆ Linux
  • TimeSys
  • MontaVista
  • Wind River
  • Mentor
  • Ridgerun

◆ WinCE
  • Adeneo
  • Mistral
  • MPC Data
  • BSQUARE

◆ RTOS
  • Green Hills
  • Wind River (VxWorks)
  • ELogic (ThreadX)
  • QNX
  • Mentor (Nucleus)

Linux Partner Strategy

◆ Commercial: provide support, off-the-shelf Linux distributions or GNU tools
◆ Consultants: provide training, general embedded Linux development expertise, or specific expertise for developing drivers or particular embedded applications
Outline

◆ Linux : What is it?
◆ Linux : Where do I get it?
  ◆ What are Distributions?
  ◆ O/S Choices
  ◆ Community Options
  ◆ Commercial Options
    ◆ Commercial vs Community
◆ TI Customer Support
◆ (Optional) How Do I Build Linux – and Why?

Commercial vs Community

◆ Commercial
  ◆ Less effort – another team does the work of porting new kernel to the distribution ... and then laboriously testing it over-and-over again
  ◆ More robust – many customers generating more inputs/errata to team testing an maintaining the distribution
  ◆ More secure – among other reasons, many homebrew distributions don’t get around to adding security patches due to effort and time
  ◆ Latest features? Many vendors backport new features into their kernels – thus, you get the stability of a known kernel but with new features
  ◆ Good choice if: you don’t need the absolute latest features; you have a many projects to ammortize the costs; you’re a Linux wiz who really knows what they’re doing.
  ◆ Bottom Line – Commercial distributions trade cost (and the bleeding edge features) for robustness and less-effort. What is it worth, if your company depends on this product?

◆ Community (to Git or not)
  ◆ Access to latest improvements in Linux kernel
  ◆ Want to know exactly how it is all put together
  ◆ Maximum involvement in Linux community
  ◆ No costs .. (unless you count your labor)
  ◆ Bottom Line – Choose this option if you have the Linux expertise and labor is cheaper than NRE; or, you need access to the very latest features

Looking at MVL vs GIT example...
Example Comparison: **MVL Pro 5.0 vs GIT**

<table>
<thead>
<tr>
<th></th>
<th>MVL 5.0 Pro</th>
<th>Community Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kernel Version</strong></td>
<td>Uses 2.6.18, which is almost 3 years old</td>
<td>Uses latest available kernel</td>
</tr>
<tr>
<td><strong>Kernel bug-fixes</strong></td>
<td>Applied to 2.6.18, so no need to change kernel versions</td>
<td>Applied to current release, which changes every few months. User may need new kernel to get a fix.</td>
</tr>
<tr>
<td><strong>File System</strong></td>
<td>Comprehensive host and target file systems with GUI tools for optimization.</td>
<td>Not part of kernel. TI is addressing through Arago. Initially may be less user-friendly than MVL.</td>
</tr>
<tr>
<td><strong>Linux run-time Licensing</strong></td>
<td>Demo copy and LSP open source, but original licensing has created confusion.</td>
<td>TI offering is clearly free as GIT Linux distributions are open source.</td>
</tr>
<tr>
<td><strong>Tools licensing</strong></td>
<td>GNU Tools free. IDE requires annual subscription.</td>
<td>GNU Tools free. IDE requires annual subscription.</td>
</tr>
<tr>
<td><strong>Third-party support</strong></td>
<td>MV and its partners</td>
<td>Multi-vendor, including MV</td>
</tr>
</tbody>
</table>

You've come to a fork in the road ...

**TI Customers Can CHOOSE a Linux Path: Community or Commercial**

**Community first path**
- TI delivers LSP/DVSDK to community
- Smaller set of applications
- Customer builds up solution
- Open source assets
- Customer assets
- Faster access, newer kernels
- More customer responsibility
  - Invest own resources vs. $$

**Commercial complement path**
- Commercial Linux partner pulls from community
- Partner adds value: production testing, tools integration, support, application bundles, etc. for customers
- Service and subscription sales
- Executing with MontaVista, Timesys...
- Opportunities for other commercial vendors
Outline

- Linux: What is it?
- Linux: Where do I get it?
- TI Customer Support
- (Optional) How Do I Build Linux – and Why?

Linux Kernel Support from TI

- TI can ONLY provide support for the drivers it develops
  - TI has limited expertise in generic OS parts of the kernel such as network stacks and file systems
- TI does NOT support Linux application code
TI and Community Support

<table>
<thead>
<tr>
<th></th>
<th>TI</th>
<th>Community Forums</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-Boot</td>
<td>Only supports parts specific to TI devices</td>
<td><a href="mailto:u-boot@lists.denx.de">u-boot@lists.denx.de</a></td>
</tr>
<tr>
<td>Linux Kernel</td>
<td>Supports drivers specific to TI devices, not generic stack, scheduler, …</td>
<td><a href="http://www.kernel.org/pub/linux/docs/lkml/">http://www.kernel.org/pub/linux/docs/lkml/</a> for generic kernel &amp; many specific forums such as <a href="http://www.linux-usb.org/mailing.html">http://www.linux-usb.org/mailing.html</a> and <a href="https://www.redhat.com/mailman/listinfo/">https://www.redhat.com/mailman/listinfo/</a> video4linux-list</td>
</tr>
<tr>
<td>GNU tools</td>
<td>Does not support</td>
<td><a href="http://gcc.gnu.org/lists.html">http://gcc.gnu.org/lists.html</a> covers numerous gcc- and gdb-related lists</td>
</tr>
<tr>
<td>OpenEmbedded</td>
<td>Support limited to specific questions about TI devices or packages</td>
<td><a href="http://wiki.openembedded.net/index.php/MailingLists">http://wiki.openembedded.net/index.php/MailingLists</a></td>
</tr>
<tr>
<td>User-mode applications</td>
<td>TI does NOT support Linux application code, except for applications provided in the DVSDK</td>
<td>Applications have their own mailing lists: <a href="http://gstreamer.freedesktop.org/lists/">http://gstreamer.freedesktop.org/lists/</a>; <a href="http://httpd.apache.org/lists.html">http://httpd.apache.org/lists.html</a></td>
</tr>
</tbody>
</table>

Outline

- Linux: What is it?
- Linux: Where do I get it?
- TI Customer Support
- (Optional) How Do I Build Linux – and Why?
  - Don't ...
  - Build Default Linux
  - Build 'Custom' Linux
  - Static vs Dynamic
How Do I Build It, Let Me Count the Ways...

1. Don’t … find a pre-built Linux uImage
2. Build Default Linux
   a. make defconfig
   b. make

Why Re-Build Linux Kernel?
- TI SDK’s often support various ARM CPU’s, thus GSG directs you to specify target processor and rebuild kernel
- You made changes to a Linux source file (i.e. driver, etc.)
- ...

Change to Kernel’s Directory (TI/MontaVista LSP Example)
> cd ti-davinci/linux-2.6.18_pro500

Configure the Kernel
> make ARCH=arm CROSS_COMPILE=arm_v5t_le- davinci_dm644x_defconfig

Build the Kernel
> make ARCH=arm CROSS_COMPILE=arm_v5t_le- uImage
How Do I Build It, Let Me Count the Ways...

1. **Don’t** … find a pre-built Linux uImage

2. **Build Default Linux**
   a. make defconfig
   b. make uImage

3. **Build ‘Custom’ Linux**
   a. make defconfig
   b. make menuconfig
   c. make uImage

Outline

- Linux : What is it?
- Linux : Where do I get it?
- TI Customer Support
- *(Optional)* How Do I Build Linux – and Why?
  - Don't ...
  - Build Default Linux
  - Build ‘Custom’ Linux
    - Static vs Dynamic
Kernel Object Modules

How to add modules to Linux Kernel:

1. **Static** (built-in)

   - Linux Kernel source code is broken into individual modules
   - Only those parts of the kernel that are needed are built in

<table>
<thead>
<tr>
<th>Kernel Module Examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td>fbdev</td>
</tr>
<tr>
<td>v4l2</td>
</tr>
<tr>
<td>nfsd</td>
</tr>
<tr>
<td>dsp</td>
</tr>
<tr>
<td>audio</td>
</tr>
</tbody>
</table>

   - **fbdev**
   - **v4l2**
   - **nfsd**
   - **dsp**
   - **audio**

   **Kernel Module Examples**

   - **fbdev**
   - **v4l2**
   - **nfsd**
   - **dsp**
   - **audio**

   **How to add modules to Linux Kernel:**

   1. Static (built-in)

   **Static Linux Kernel Configuration**

   ```
   #> make ARCH=arm CROSS_COMPILE=arm_v5t_le- menuconfig
   ```

   **Code maturity level options**
   - General setup
   - Loadable module support
   - System type and features
   - Device Drivers
   - Ntavista System tools
   - Kernel hacking
   - Security options
   - Cryptographic options
   - Library routines

   ```make ARCH=arm CROSS_COMPILE=arm_v5t_le- menuconfig```
## Kernel Object Modules

### How to add modules to Linux Kernel:

1. **Static** (built-in)
   - **Linux Kernel**
     - oss
     - fbdev
     - httpd
     - v4l2
     - nfsd
     - dsp
     - ext3
   - Linux Kernel source code is broken into individual modules
   - Only those parts of the kernel that are needed are built in

2. **Dynamic** (insmod)
   - ```
     # insmod <mod_name>.ko [mod_properties]
   ```
   - Use `insmod` (short for insert module) command to dynamically add modules into the kernel
   - Keep statically built kernel small (to reduce size or boot-up time), then add functionality later with insmod
   - Insmod is also handy when developing kernel modules
   - Later we’ll insert two modules (cmem.ko, dsplink.ko) using a script: loadmodules.sh

### Kernel Module Examples:
- fbdev: frame buffer dev
- v4l2: video for linux 2
- nfsd: network file server dev
- dsp: oss digital sound proc.
- audio: alsa audio driver

---

*Chapter 2: Linux - What, Where and How to Build It*
### Getting Started with Linux Workshop

| Welcome                      | Workshop Goals  
|                             | Agenda          
|                             | Where to Get More Info |
| Device/Tools Overview       | TI Devices with Linux Support |
|                             | Hardware Development Platforms |
|                             | Software Development Kits |
| Linux                       | What Is It? |
|                             | Where Do I Get It? |
|                             | How Do I Build It? |
| Booting Linux               | Tracing the Boot Process |
|                             | Customizing Boot Process |
|                             | Das U-Boot |
| Working In Linux            | X-Windows |
|                             | GNOME |
|                             | Networking |

### Outline

- **Booting Linux - ROM to Kernel Sequence**
  - RBL
  - Xloader
  - Das U-Boot
- **Boot Image Location(s)**
- **Configuring U-Boot**
### Booting Linux – ROM to Kernel

1. **RBL (ROM)**
2. **x-loader (i.e. MLO)**
3. **U-Boot**

#### Device
- RBL
- x-loader

#### Flash
- U-Boot
- Linux Kernel

#### DDR2
- U-Boot
- Linux Kernel

### Bootloader Components

<table>
<thead>
<tr>
<th>Boot stage</th>
<th>Operations</th>
<th>User Config’d</th>
<th>DaVinci</th>
<th>Sitara</th>
</tr>
</thead>
<tbody>
<tr>
<td>First-level</td>
<td>This is ROM’d code for detecting desired boot type (NAND, UART, …) and loading executable code of second-level bootloader from selected peripheral/interface</td>
<td>No</td>
<td>RBL</td>
<td>RBL</td>
</tr>
<tr>
<td>Second-level</td>
<td>The primary function of this bootloader is to initialize external memory and system clocks so that a larger, more advanced boot loader (in this case U-boot) can be loaded.</td>
<td>Board Designer</td>
<td>UBL</td>
<td>XLDR</td>
</tr>
<tr>
<td>Linux boot</td>
<td>“Das U-boot” is the standard open-source Linux boot loader for ARM. It supports networking for TFTP/NFS booting. It is used to locate, load and execute the Linux kernel in ulmage format and is also responsible for passing arguments to the kernel</td>
<td>Yes</td>
<td>U-boot</td>
<td>U-Boot</td>
</tr>
</tbody>
</table>

**Customizing UBL / XLDR**
1. Configure system clocks
2. Setup memory interfaces

---

*Chapter 3: Booting Linux*
Outline

- Booting Linux - ROM to Kernel Sequence
- Boot Image Location(s)
- Configuring U-Boot

To Boot Linux, You Need...

1. Bootloader (U-Boot)
   - At reset, U-Boot bootloader is executed
2. Linux Kernel
   - U-Boot loads O/S kernel into DDR2 memory; then,
   - Connects to the root filesystem
     If you don’t know what this is, think of it as the ‘c:\’ drive of in Windows PC
3. Filesystem

Where are these 3 located?
Where Do You Find …

<table>
<thead>
<tr>
<th>Where located:</th>
<th>DM6446 EVM Default</th>
<th>AM3517 1-day Wkshp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a. UBL or Xloader/MLO</td>
<td>Flash</td>
<td>MMC</td>
</tr>
<tr>
<td>1b. Bootloader (U-Boot)</td>
<td>Flash</td>
<td>MMC</td>
</tr>
<tr>
<td>2. Linux Kernel</td>
<td>Flash</td>
<td>MMC</td>
</tr>
<tr>
<td>3. Filesystem</td>
<td>Hard Drive</td>
<td>MMC</td>
</tr>
</tbody>
</table>

- By default, the DM6446 DVEVM ships in “HDD boot” mode; this allows the demo applications to run "out-of-the-box"
- OMAP3530 & AM3517 ship with boot code in NAND Flash. Also, the SDK provides an MMC image you can burn to a card.

The MMC card would look like ...

SD / MMC Boot

- Partition 1 (FAT32)
  - XLDR (i.e. MLO)
  - U-Boot
  - UImage (i.e. Kernel)
- Partition 2 (EXT3)
  - Root Filesystem

A 3rd boot option is ...
### Where Do You Find …

<table>
<thead>
<tr>
<th>Where located:</th>
<th>DM6446 EVM Default</th>
<th>AM3517 1-day Wkshp</th>
<th>Good for Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a. UBL or Xloader/MLO</td>
<td>Flash</td>
<td>MMC</td>
<td>Flash or MMC</td>
</tr>
<tr>
<td>1b. Bootloader (U-Boot)</td>
<td>Flash</td>
<td>MMC</td>
<td>Flash or MMC</td>
</tr>
<tr>
<td>2. Linux Kernel</td>
<td>Flash</td>
<td>MMC</td>
<td>TFTP from Ubuntu</td>
</tr>
<tr>
<td>3. Filesystem</td>
<td>Hard Drive</td>
<td>MMC</td>
<td>NFS from Ubuntu</td>
</tr>
</tbody>
</table>

- “HDD boot”
- “MMC boot”
- “NFS boot”

- By default, the DM6446 DVEVM ships in “HDD boot” mode; this allows the demo applications to run "out-of-the-box"
- OMAP3530 & AM3517 ship with boot code in NAND. An MMC card demo also ships with the EVM's. Also, the SDK provides an MMC image
- “NFS boot” (network boot) is good for application development

### NFS Boot

- RS-232 is physical connection to U-Boot
- Use to stop DVEVM from stand-alone boot
- Configure U-Boot’s modes by setting/saving environment variables
- Ethernet provides physical connection to Linux PC
- Root filesystem is accessed via NFS protocol
- Don’t need to ‘flash’ DVEVM after compiling new program

Note: `~/targetfs = /home/user/targetfs`
Outline

- Booting Linux - ROM to Kernel Sequence
  - RBL
  - Xloader
  - Das U-Boot
- Boot Image Location(s)
  - Configuring U-Boot

Das U-Boot

- The Linux PSP SDK board is delivered with the open-source boot loader: Das U-Boot (U-Boot)

- At runtime, U-Boot is usually loaded in on-board Flash or an SD/MMC card

- In general, U-Boot performs the functions:
  1. Initializes the DaVinci EVM hardware
  2. Provides boot parameters to the Linux kernel
  3. Starts the Linux kernel
Configuring U-Boot and Starting Linux (5 Steps)

1. Connect an RS232 serial cable and start a Tera Term
2. Power on the DVEVM and press any key in TeraTerm to abort the boot sequence
3. Set U-Boot variables to select how Linux will boot (save changes to flash to retain settings after power cycle)
4. Boot Linux using either:
   - the U-Boot “boot” command
   - power-cycle the DVEVM
5. After Linux boots, log in to the DVEVM target as “root”

- Note, login with: “user” for the Tools Linux PC “root” for the DVEVM target
- You can use any RS-232 comm application (Linux or Win), we use Tera Term for its macro capability

What U-Boot param's do I need?

**Configuring U-Boot**

**Common Uboot Commands:**
- `printenv` - prints one or more uboot variables
- `setenv` - sets a uboot variable
- `saveenv` - save uboot variable(s)
- `run` - evaluate a uboot variable expression
- `ping` - (debug) use to see if Uboot can access NFS server

**Common Uboot Variables:**
- You can create whatever variables you want, though some are defined either by Linux or common style
  - `boots` - where Linux kernel should boot from
  - `bootargs` - string passed when booting Linux kernel
    e.g. tells Linux where to find the root filesystem
  - `serverip` - IP address of root file system for NFS boot
  - `nfspath` - Location on serverip for root filesystem
## Boot Variations

<table>
<thead>
<tr>
<th>Mode</th>
<th>IP</th>
<th>Linux Kernel</th>
<th>Root Filesystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>dhcp</td>
<td>Flash</td>
<td>HDD</td>
</tr>
<tr>
<td>2.</td>
<td>dhcp</td>
<td>Flash</td>
<td>NFS</td>
</tr>
<tr>
<td>3.</td>
<td>dhcp</td>
<td>TFTP</td>
<td>HDD</td>
</tr>
<tr>
<td>4.</td>
<td>dhcp</td>
<td>TFTP</td>
<td>NFS</td>
</tr>
<tr>
<td>5.</td>
<td>dhcp</td>
<td>MMC</td>
<td>MMC</td>
</tr>
</tbody>
</table>

U-Boot's `bootcmd` variable specifies the root filesystem.

### Flash

- `setenv bootcmd "mmc init; fatload mmc 0 ${loadaddr} uImage; run mmcargs; bootm ${loadaddr}"`
- `setenv bootcmd bootm 0x2050000`

### MMC

- `setenv bootcmd "mmc init; fatload mmc 0 ${loadaddr} uImage; run mmcargs; bootm ${loadaddr}"
- `ip=dhcp` root=/dev/mmcblk0p2
  - `setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/mmcblk0p2 rootfstype=ext3 rootwait nolock mem=120M`

### TFTP

- `setenv bootcmd 'dhcp;bootm'`

## Boot Variations (filesystem)

### HDD

- `ip=dhcp` root=/dev/hda1
  - `setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/hda1, nolock mem=120M`

### MMC

- `ip=dhcp` root=/dev/mmcblk0p2
  - `setenv bootargs console=ttyS0,115200n8 noinitrd root=/dev/mmcblk0p2 rootfstype=ext3 rootwait nolock mem=120M`

### NFS

- `ip=dhcp` root=/dev/nfs
  - `setenv bootargs console=ttyS0,115200n8 noinitrd root=/dev/nfs nfsroot=$(serverip):$(nfspath),nolock mem=120M`

U-Boot's `bootargs` variable specifies the root filesystem.
Using Tera Term Macros

- U-boot strings can be very precise, one wrong character could prevent booting
- TeraTerm Pro (freeware) provides a simple macro language
- Very convenient for setting up U-Boot, if you change settings frequently
## Getting Started with Linux Workshop

<table>
<thead>
<tr>
<th>Welcome</th>
<th>Workshop Goals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Agenda</td>
</tr>
<tr>
<td></td>
<td>Where to Get More Info</td>
</tr>
<tr>
<td>Device/Tools Overview</td>
<td>TI Devices with Linux Support</td>
</tr>
<tr>
<td></td>
<td>Hardware Development Platforms</td>
</tr>
<tr>
<td></td>
<td>Software Development Kits</td>
</tr>
<tr>
<td>Linux</td>
<td>What Is It?</td>
</tr>
<tr>
<td></td>
<td>Where Do I Get It?</td>
</tr>
<tr>
<td></td>
<td>How Do I Build It?</td>
</tr>
<tr>
<td>Booting Linux</td>
<td>Tracing the Boot Process</td>
</tr>
<tr>
<td></td>
<td>Customizing Boot for Your Board</td>
</tr>
<tr>
<td></td>
<td>Das U-Boot</td>
</tr>
<tr>
<td>Working In Linux</td>
<td>X-Windows</td>
</tr>
<tr>
<td></td>
<td>GNOME</td>
</tr>
<tr>
<td></td>
<td>Networking</td>
</tr>
</tbody>
</table>

### x11 Without Desktop

<table>
<thead>
<tr>
<th>Device</th>
<th>AM3517</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver</td>
<td>Linux</td>
</tr>
<tr>
<td></td>
<td>FBDEV framebuffer driver</td>
</tr>
<tr>
<td>Display Server</td>
<td>X server</td>
</tr>
<tr>
<td>Window Manager</td>
<td>x11 window manager</td>
</tr>
<tr>
<td>Display Manager</td>
<td>XDM (x11 Display Manager)</td>
</tr>
<tr>
<td>Session Manager</td>
<td>XSM (x11 session manager)</td>
</tr>
</tbody>
</table>
### Gnome Desktop Environment

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop Environment</td>
<td>Presents a desktop environment.</td>
</tr>
<tr>
<td>Session Manager</td>
<td>Manages a login session to a Window Manager. Allows a user to maintain the settings and state of a windowing environment.</td>
</tr>
<tr>
<td>Display Manager</td>
<td>Manages one or more local and/or remote Display servers (usually via window managers) for a login session.</td>
</tr>
<tr>
<td>Window Manager</td>
<td>Exposes independent, resizable virtual displays (windows) to applications.</td>
</tr>
<tr>
<td>Display Server</td>
<td>Network-transparent interface to display driver.</td>
</tr>
<tr>
<td>Driver</td>
<td>Exposes a bitmap-type interface (framebuffer) to the display hardware.</td>
</tr>
</tbody>
</table>

### KDE vs. Gnome

<table>
<thead>
<tr>
<th>KDE ... as a quick comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device</strong></td>
</tr>
<tr>
<td><strong>AM3517</strong></td>
</tr>
<tr>
<td><strong>Driver</strong></td>
</tr>
<tr>
<td><strong>Linux</strong></td>
</tr>
<tr>
<td><strong>FBDEV framebuffer driver</strong></td>
</tr>
<tr>
<td><strong>Display Server</strong></td>
</tr>
<tr>
<td><strong>X server</strong></td>
</tr>
<tr>
<td><strong>Window Manager</strong></td>
</tr>
<tr>
<td><strong>x11 window manager</strong></td>
</tr>
<tr>
<td><strong>Display Manager</strong></td>
</tr>
<tr>
<td><strong>GDM</strong></td>
</tr>
<tr>
<td><strong>Session Manager</strong></td>
</tr>
<tr>
<td><strong>Gnome Session Manager</strong></td>
</tr>
<tr>
<td><strong>Desktop Environment</strong></td>
</tr>
<tr>
<td><strong>Gnome Desktop</strong></td>
</tr>
</tbody>
</table>

Looking at Gnome graphically ...
K Desktop Environment

- KDE (K Desktop Environment)
- ksmserver (session mgr)
- KDM (KDE Display manager)
- kwm (K window mgr)
- QT embedded (QT / e) (No Display Server, direct framebuffer access via widget lib)
- Linux FBDEV Framebuffer Driver

- KDE (K Desktop Environment)
- ksmserver (session mgr)
- KDM (KDE Display manager)
- kwm (K window mgr)
- QT over x11 (QT / x11)
- X server
- Linux FBDEV Framebuffer Driver

x11 (Window Manager)

- xterm
- mplayer

- Also known as X, or X Window
- x11 is the most common Linux display manager
- x11 also manages Human Interface Devices (HIDs)
- Various applications send graphical data to x11, which displays each in an independent, resizable window
Desktop Managers

- Gnome is one of many Linux display managers
- Desktop managers provide a “Desktop” or similar environment (i.e. for handheld device)
- Applications (such as mplayer or xterm) typically interface directly to x11 and are therefore independent of the Display Manager
- Gnome display manager is “just another client” to x11
- A display manager isn’t required to run Linux
- x11 isn’t required to run Linux, though it is required by many graphical applications

Terminal vs. Desktop

- A terminal is a text-based interface to Linux O/S
- Commands are interpreted by a shell program (such as BASH)
- A terminal may connect via RS-232, Ethernet, etc.
- xterm is a terminal which connects via the X window server, i.e. receives input via x11 HID and outputs to x11 display
- Most applications which can be launched from the Desktop (i.e. a drop-down menu) can be equivalently launched by typing a command into a Linux terminal
Ubuntu, used in our lab exercise, does not contain a copy of Gnome.
Thus, desktop (e.g., Gnome) is actually generated by the AM3517 in our example – PC acts only as a ‘dumb’ display.
Both client and server are available whenever X is started.

As with many Linux packages, x11 was designed from beginning with networking in mind (client/server).
x11 may export Human Interface Device (HID) I/O in addition to Display.
Exporting a desktop is handled by x11 (Desktop managers are “just another client” to x11).
Exporting x11 Display

Most applications launched from a terminal will query terminal's DISPLAY environment variable

```
# startx &
# export DISPLAY=192.168.1.10:0.0
# gedit myfile.txt
```

Processes, such as the Gnome display manager, which exist outside the context of the terminal may take a display argument

```
# startx &
# gnome-session --display=192.168.1.10:0.0
```

Lab Exercises

Lab 1: Connect and boot the AM3517 EVM
Boot the AM3517 from an SD/MMC card.

Lab 2: Start Gnome (Linux GUI) desktop running on the AM3517 EVM
There are four parts to this lab which correspond to the four different ways that you can invoke Gnome:

a. Automatically start GNOME on the AM3517 EVM's LCD
b. Manually start GNOME on the AM3517 EVM's LCD
c. Display the GNOME GUI remotely on the host PC via secure SSH connection
d. Export the GNOME GUI onto the host PC
Lab Exercises

Lab 1: Connect and boot the AM3517 EVM
Boot the AM3517 from an SD/MMC card.

Lab 2: Start Gnome (Linux GUI) desktop running on the AM3517 EVM
There are four parts to this lab which correspond to the four different ways that you can invoke Gnome:
   a. Automatically start GNOME on the AM3517 EVM's LCD
   b. Manually start GNOME on the AM3517 EVM's LCD
   c. Display the GNOME GUI remotely on the host PC via secure SSH connection
   d. Export the GNOME GUI onto the host PC

Lab 3: Running video (and audio) on the AM3517
   a. Play a .wmv video movie trailer
   b. (Take Home Lab) View video from a USB Webcam
   c. (Take Home Lab) Listen to audio via USB speakers

Lab 4: Investigating Linux Networking with the AM3517
   a. Configure Linux Ethernet connection, both temporarily and permanently
   b. Configure a Samba Server
   c. Configure an HTML Server

Lab Setup (H/W)

Copyright © 2010 Texas Instruments. All rights reserved.
Lab Setup (S/W)

- AM3517 Zoom board
- Gnome
- x11
- Linux

- Windows XP
- Terra Term (COM4)
- VMware (ubuntu_x11_client)
- Gnome (exported)
- x11
- Linux