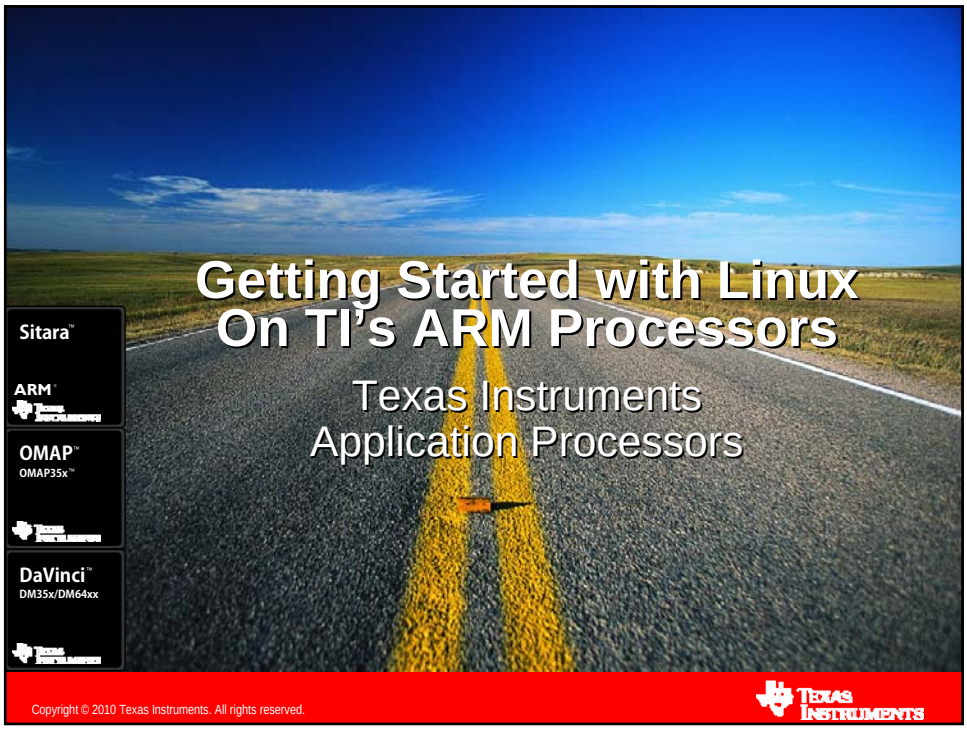


Chapter Outline

- **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**

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

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 To Go For More Info**

Where can I get additional skills? (from TI)

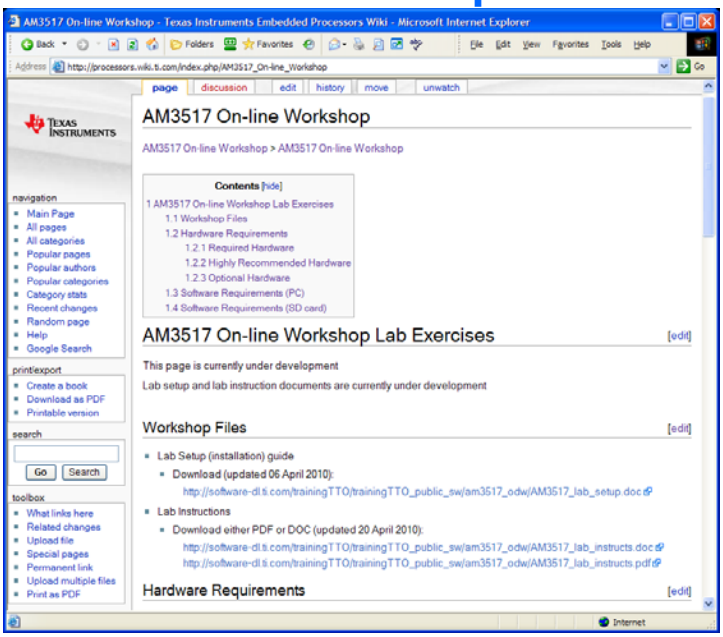
Texas Instruments Curriculum

Building Linux based Systems (ARM or ARM+DSP processors)	DaVinci / OMAP / Sitara System Integration Workshop using Linux (4-days) www.ti.com/training
Building BIOS based Systems (DSP processors)	System Integration Workshop using DSP/BIOS (4-days) www.ti.com/training
Developing Algo's for C6x DSP's (Are you writing/optimizing algorithms for latest C64x+ or C674x DSP's CPU's)	C6000 Optimization Workshop (4-days) www.ti.com/training

Online Resources:

- OMAP / Sitara / DaVinci Wiki http://processors.wiki.ti.com
- TI E2E Community (videos, forums, blogs) http://e2e.ti.com
- This workshop presentation & exercises http://processors.wiki.ti.com/index.php /AM3517_On-line_Workshop
- DaVinci Open-Source Linux Mail List http://linux.davincidsp.com/mailman/listinfo/davinci-linux-open-source
- Gstreamer and other projects http://linux.davincidsp.com or https://gforge.ti.com/gf/
- TI Software http://www.ti.com/dvevmupdates, http://www.ti.com/dms http://www.ti.com/myregisteredsoftware

Find These Workshop Materials



http://processors.wiki.ti.com/index.php/AM3517_On-line_Workshop

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

TI Embedded Processors Portfolio

Microcontrollers			ARM-Based		DSP
16-bit	32-bit Real-time	32-bit ARM	ARM+	ARM + DSP	DSP
MSP430	**C2000™**	**ARM**	**ARM9 Cortex A-8**	**C64x+ plus ARM9/Cortex A-8**	**C647x, C64x+, C674x, C55x**
Ultra-Low Power	Fixed & Floating Point	Industry Std Low Power	Industry-Std Core, High-Perf GPP	Industry-Std Core + DSP for Signal Proc.	Leadership DSP Performance
Up to 25 MHz	Up to 300 MHz	<100 MHz	Accelerators	4800 MMACs/ 1.07 DMIPS/MHz	24,000 MMACS
Flash 1 KB to 256 KB	Flash 32 KB to 512 KB	Flash 64 KB to 1 MB	MMU	MMU, Cache	Up to 3 MB L2 Cache
Analog I/O, ADC LCD, USB, RF	PWM, ADC, CAN, SPI, I²C	USB, ENET, ADC, PWM, SPI	USB, LCD, MMC, EMAC	VPSS, USB, EMAC, MMC	1G EMAC, SRIO, DDR2, PCI-66
Measurement, Sensing, General Purpose	Motor Control, Digital Power, Lighting, Sensing	Host Control	Linux/WinCE User Apps	Linux/Win + Video, Imaging, Multimedia	Comm, WiMAX, Industrial/ Medical Imaging
$0.49 to $9.00	$1.50 to $20.00	$2.00 to $8.00	$8.00 to $35.00	$12.00 to $65.00	$4.00 to $99.00+

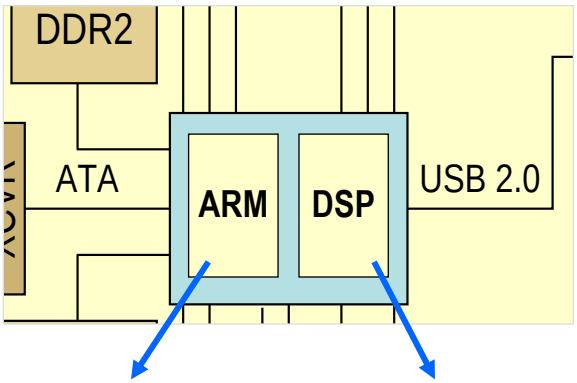
Texas Instruments

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?



Linux

- **User Controls, GUI, OSD**
- **Peripheral Drivers**
- **Ethernet** (other system comm)

DSP/BIOS™

- **Video processing** decoding, encoding, etc.
- **Audio processing** decoding, encoding, etc.

Key System Blocks

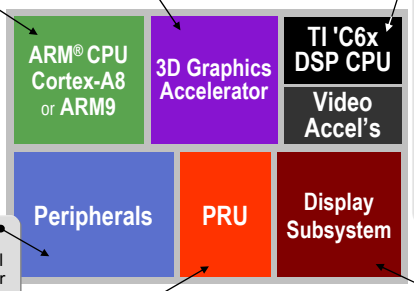
An integrated solution that reduces System complexity, Power consumption, and Support costs

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

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 req'd

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



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

Prog. Real-time Unit (PRU)
- Use this configurable processor block to extend peripheral count or I/F's
- Tailor for a proprietary interface or build a customized system control unit

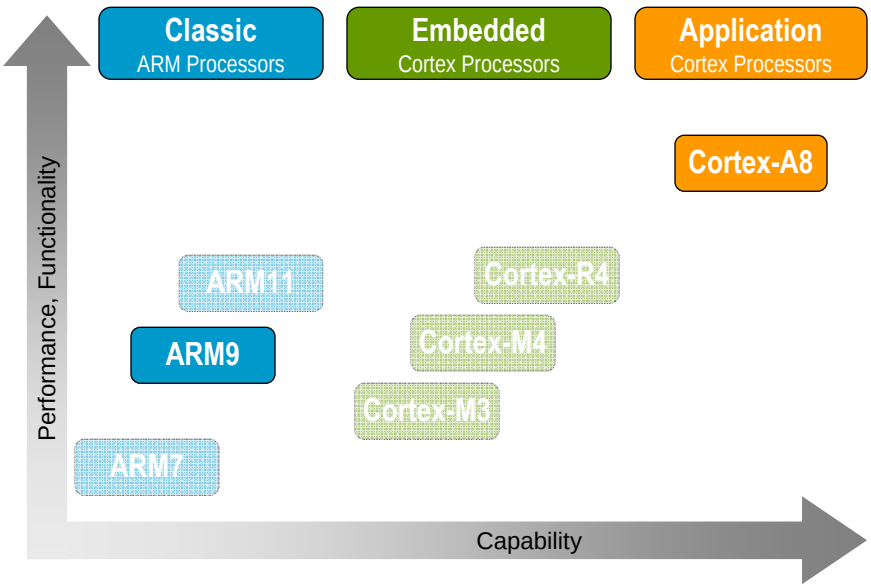
NOTE
Features not available on all devices

Texas Instruments

Outline

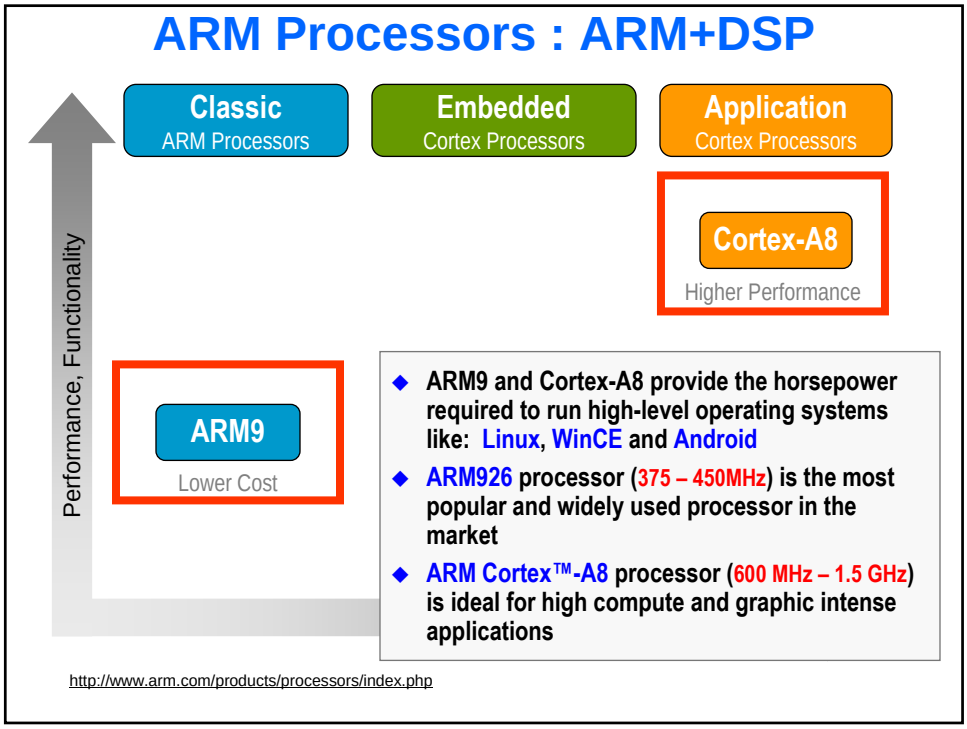
- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
  - ARM Core
  - DSP Core
  - Accelerators
- Peripherals
- Device Families : Sitara, DaVinci, C6L/C6A8
- Choosing a Device : Web Tool
- Tools Overview

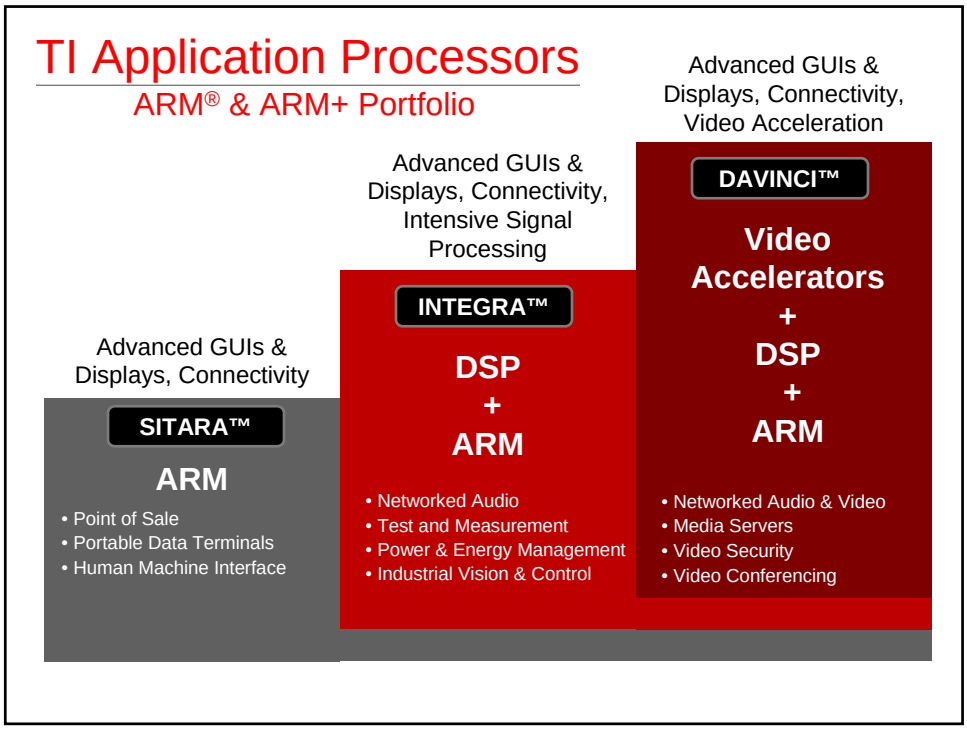
TI ARM CPU Processor Cores



http://www.arm.com/products/processors/index.php

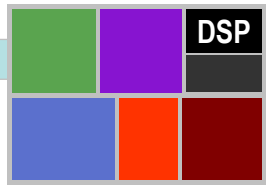
TI's ARM core's supporting Linux...


ARM Processors : ARM+DSP
Classic
ARM Processors
Embedded
Cortex Processors
Application
Cortex Processors
Cortex-A8
Higher Performance
Performance, Functionality
ARM9
Lower Cost
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
http://www.arm.com/products/processors/index.php


TI Application Processors
ARM® & ARM+ Portfolio
Advanced GUIs & Displays, Connectivity, Video Acceleration
DAVINCI™
Video Accelerators + DSP + ARM
• Networked Audio & Video
• Media Servers
• Video Security
• Video Conferencing
Advanced GUIs & Displays, Connectivity, Intensive Signal Processing
INTEGRA™
DSP + ARM
• Networked Audio
• Test and Measurement
• Power & Energy Management
• Industrial Vision & Control
Advanced GUIs & Displays, Connectivity
SITARA™
ARM
• Point of Sale
• Portable Data Terminals
• Human Machine Interface

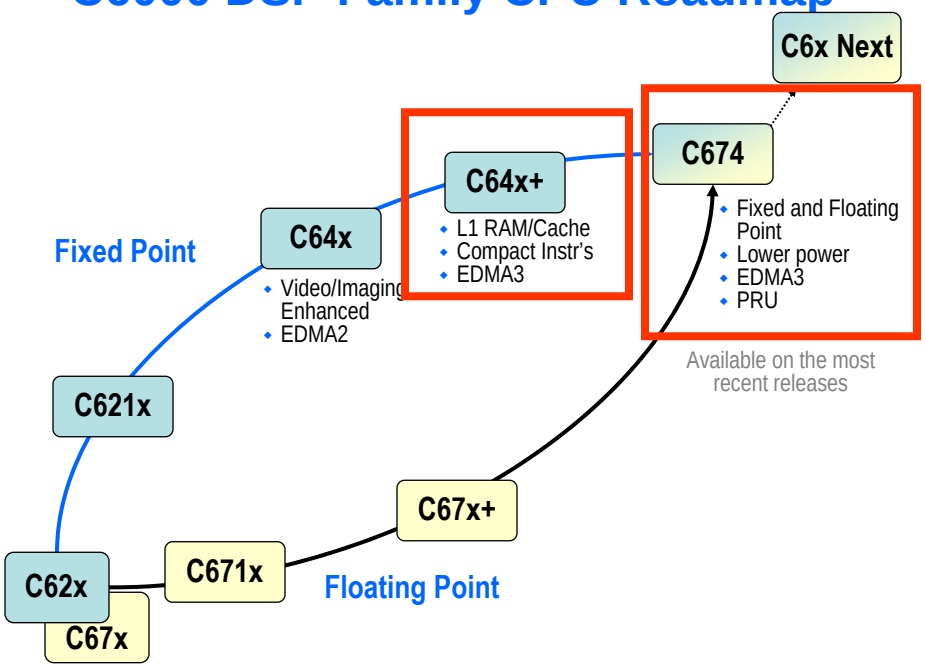
Outline

- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
  - ARM Core
  - DSP Core
  - Accelerators



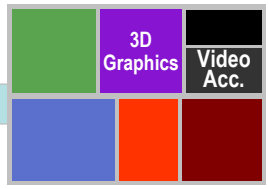
- Peripherals
- Device Families : Sitara, DaVinci, C6L/C6A8
- Choosing a Device : Web Tool
- Tools Overview

C6000 DSP Family CPU Roadmap



Outline

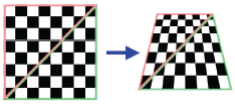
- **TI Embedded Processors Portfolio**
- **What Processing Do You Need? (CPU, Accel)**
  - **ARM Core**
  - **DSP Core**
  - **Accelerators**



- **Peripherals**
- **Device Families : Sitara, DaVinci, C6L/C6A8**
- **Choosing a Device : Web Tool**
- **Tools Overview**

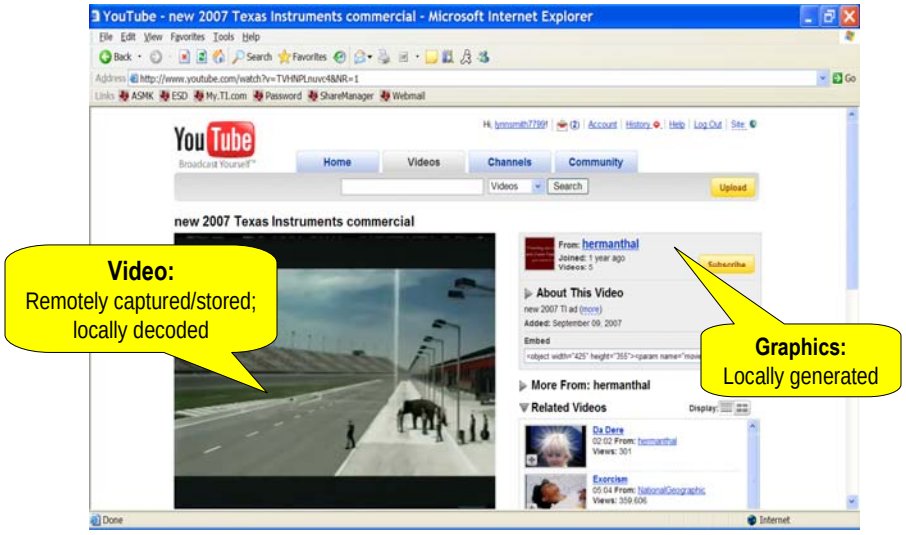
Video ≠ Graphics

Video	Graphics
◆ **Captured**	◆ **Generated**
◆ **Motion**	◆ **Animation**
▪ Integrated image	▪ Multi-layered image
▪ Base image + motion	▪ New delta images
◆ **YUV/YCbCr format**	◆ **RGB format**
▪ Convert to RGB for display	▪ Ready for display
◆ **Video codec** : block-based	◆ **Graphics :** pixel-based
▪ MPEG, WMV, JPG, etc.	▪ BMP, GIF, etc.
◆ **Natural attributes**	◆ **Calculated attributes**
▪ Digitized analog image	▪ Synthesized digital image



- 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 accelleration is required; higher resolution – as well as demand for smooth 3D rendering – drives need for graphics accelleration

Simple Graphics/Video Example



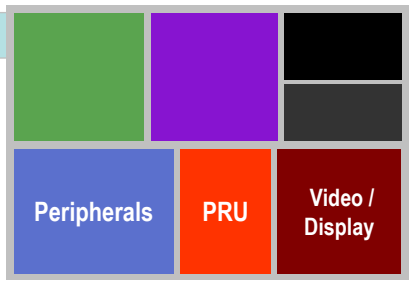
Hardware Accelerator Summary

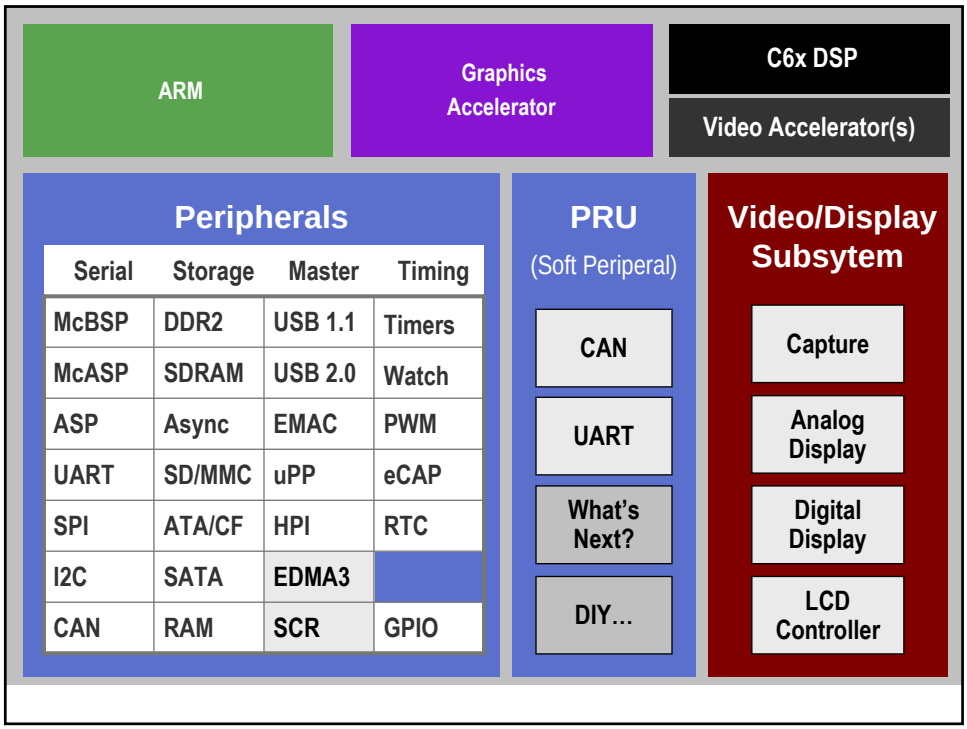
Devices	3D Graphics	Video/Imaging/Audio	Video Port
DM355		MJCP: JPEG, MPEG4	OSD,Resizer,Previewer,H3A
DM365/368		MJCP & HCP: JPEG, MPEG4, H264, VC1, Facial; MP3,AAC,WMA,AEC	OSD,Resizer,Previewer,H3A
AM3517	PowerVR SGX		OSD,Resizer,Color Key
AM3715	PowerVR SGX		OSD,Resizer,Color Key
DM6446		VICP: VidEnc,Image,Other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

Note: "Hardware accellerated" 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

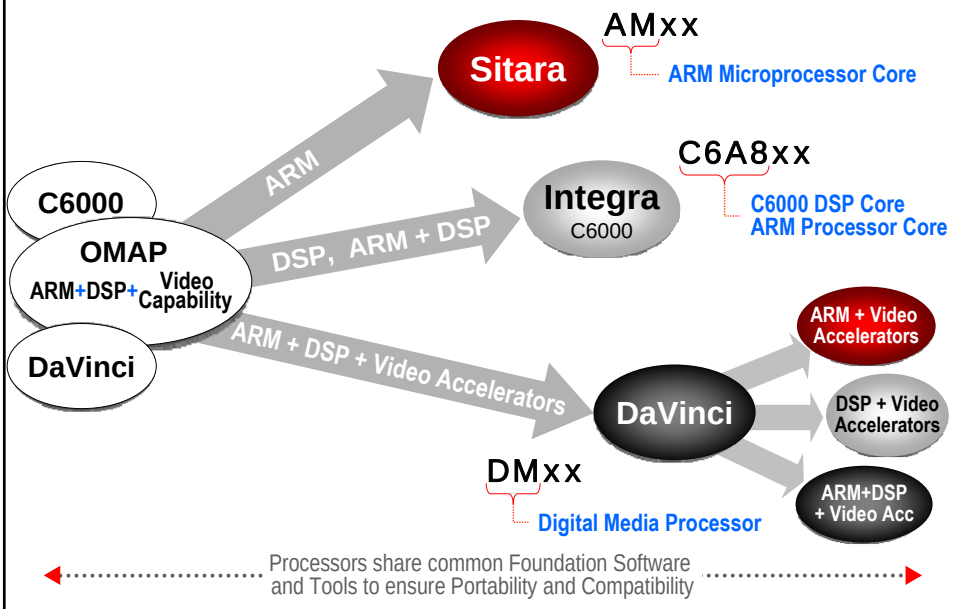




Outline

- 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



ARM and ARM+DSP/Video

	General Purpose ARM Only	General Purpose ARM+DSP	Video Oriented ARM / ARM+DSP
ARM926	AM1705 AM1707 AM1806 AM1808	OMAP-L137 OMAP-L138 (also C6L13x)	DM355 DM365/DM368 DM644x DM6467
Cortex A8	OMAP3503 OMAP3515		OMAP3525 OMAP3530
	AM3505 AM3515		
	AM3703 AM3715		DM3725 DM3730
	AM3982 AM3984	C6A8167 C6A8168	
	Sitara (AM)	"Integra" (C6L, C6A8)	DaVinci (DM)

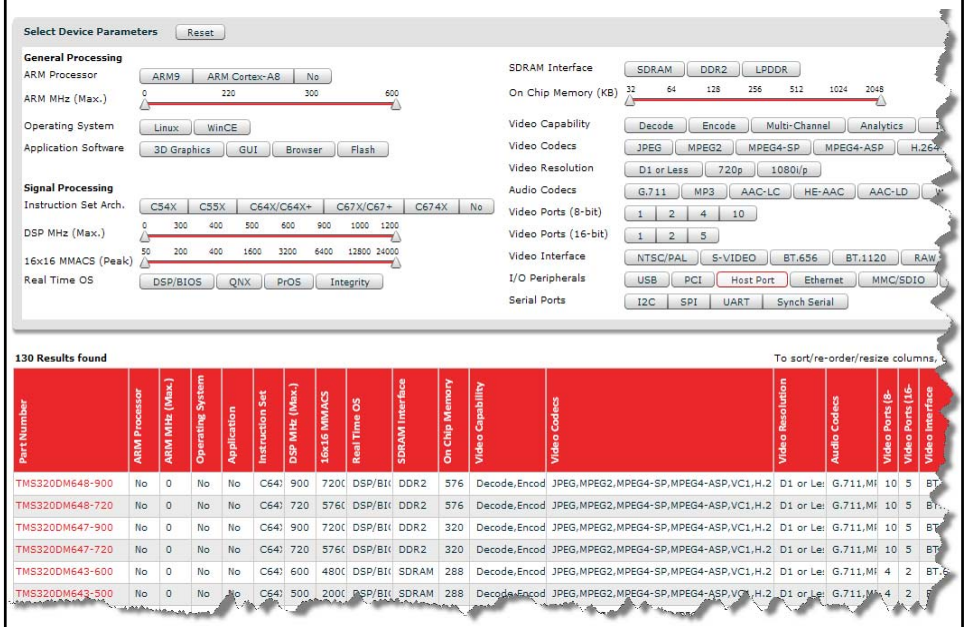
PRU

Pin-for-Pin 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

DSP & ARM MPU Selection Tool



http://focus.ti.com/en/multimedia/flash/selection_tools/dsp/dsp.html

Outline

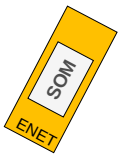
- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
- Peripherals
- Device Families : Sitara, DaVinci, C6L/C6A8
- Choosing a Device : Web Tool
- Tools Overview
  - Development Kits
  - Software Development Kits (DVSDK, SDK)

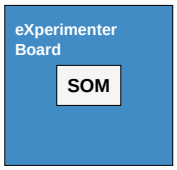
Outline

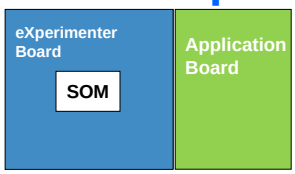
- TI Embedded Processors Portfolio
- What Processing Do You Need? (CPU, Accel)
- Peripherals
- Device Families : Sitara, DaVinci, C6L/C6A8
- Choosing a Device : Web Tool
- Tools Overview
  - Development Kits
    - Software Development Kits (DVSDK, SDK)

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
- NOT supported by TI

eXperimenter Kit

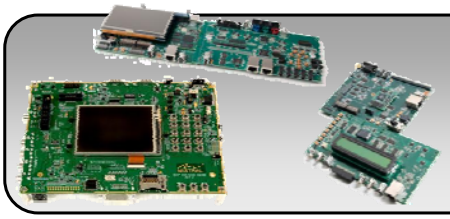
Use Case
- Evaluation of processor functionality
- Application development with limited peripheral access

Evaluation Module

Use Case
- Touch-screen application development with full peripheral access
- Application specific development

Texas Instruments

Evaluation and Development Kits

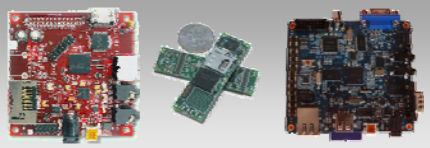


Development Kit Contents:

- Evaluation board and Documentation
- Software Development Kits
- Development Tools

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

Community Boards & Modules



To Access:

Contact TI partners for more information or click link to buy now

	Tool	Part Number	Price *	Availability
Low Cost Kits	Beagle Board (OMAP35x)	Beagle	$149	Community
	Hawkboard (OMAP-L138)	ISSPLHawk	$89	Community
SOM	OMAP35x System on Module	OMAP35x SOM-LV	$99	Logic
	Overo OMAP35x Computer on Module	Overo	$149-$219	Gumstix
	KBOC OMAP35x System on Module	KBOC	$139	KwikByte

* Prices subject to change

Texas Instruments

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

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

- *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*

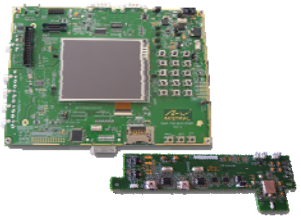
Wireless SDK : Getting started with WL1271 on OMAP35x EVM

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

Documentation

- User Guides
- Complete API reference
- Application Notes
- Demo applications and sample code



- **Evaluate 802.11b/g/n and *Bluetooth*® capability, and begin SW development**
- **Included in EVM box: Mar 2010**
- **Standalone Connectivity Card upgrade available from Mistral** http://www.mistralsolutions.com/WL1271
- **WL1271 module available from LS Research and its distributors** http://www.lsr.com/products/radio_modules/802.11_BGN_BT/tiwi.shtml

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)

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

Texas Instruments

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/sgpp/datasheet?target_arch=ARM&target_os=GNU%2FLinux
- **Check SDK/DVSDK or Linux PSP release notes for tool versions**

	Lite	Personal	Professional
30-day Installation Support	No	Yes	Yes
Unlimited Support	No	No	Yes
Priority Defect Correction	No	No	Yes
Access to Updates, Knowledge Base	No	Yes	Yes
Big Endian, Neon support	Yes	Yes	Yes
GNU C/C++ Compiler	Yes	Yes	Yes
GNU Debugger (GDB)	Yes	Yes	Yes
Eclipse IDE	No	Yes	Yes
Annual Subscription Price (per Host)	Free	$399	$2799

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

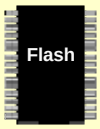
Outline

- **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

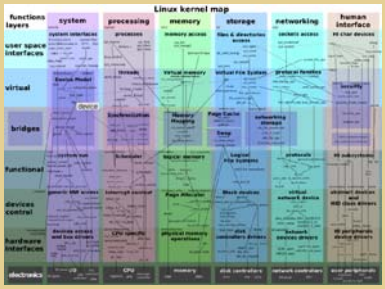
❶ Bootloader

- Provides rudimentary h/w init
- Calls Linux kernel and passes boot arguments



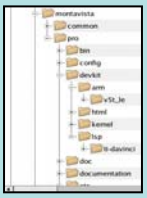
❷ Kernel

- Initializes the system (and device)
- Manages system resources
- Provides services for user programs



❸ 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

Power On

Boot Loader

U-Boot

- ARM assembly code
- Passes args to Linux (bootargs)

Linux Kernel

Linux Kernel

Initialize hardware

File System

Init Process

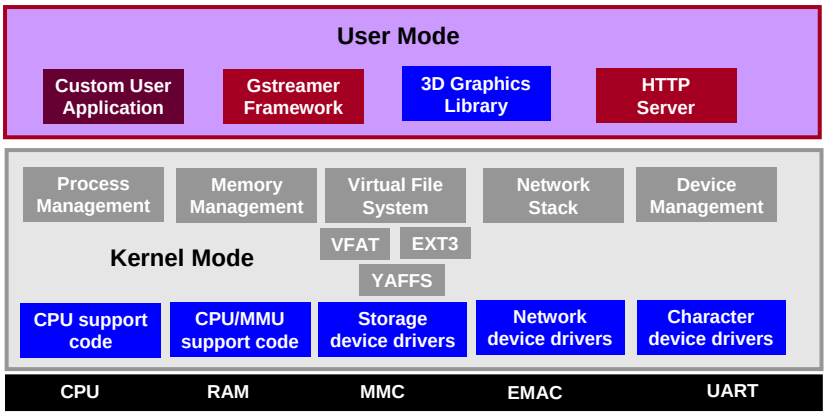
/sbin/init – 1st process exe by kernel

Login Prompt

- Login console
- Usually one of first prog's to run

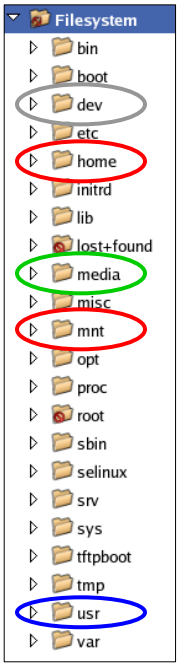
Looking more closely at the Kernel ...

What's in the Linux Kernel



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
- DVSDK GSG directory for TI tools, examples, working directory
- "root" user is different, that user's folder is at /root

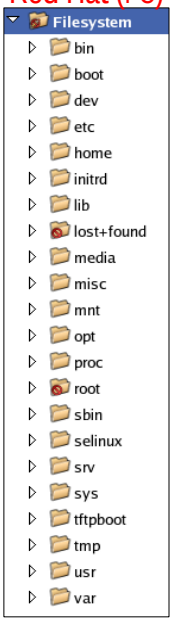
/media – Usually find CDROM drive(s) mounted here

/mnt – 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

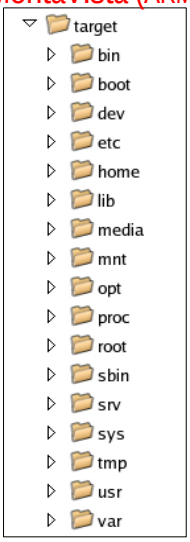
/usr – Storage for user binaries
- X86 Compiler for Ubuntu programs (gcc) is stored in here

Filesystems: Red Hat vs. Montavista

Red Hat (PC)



MontaVista (ARM)



- Tools/Host filesystem: location our dev'l 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.

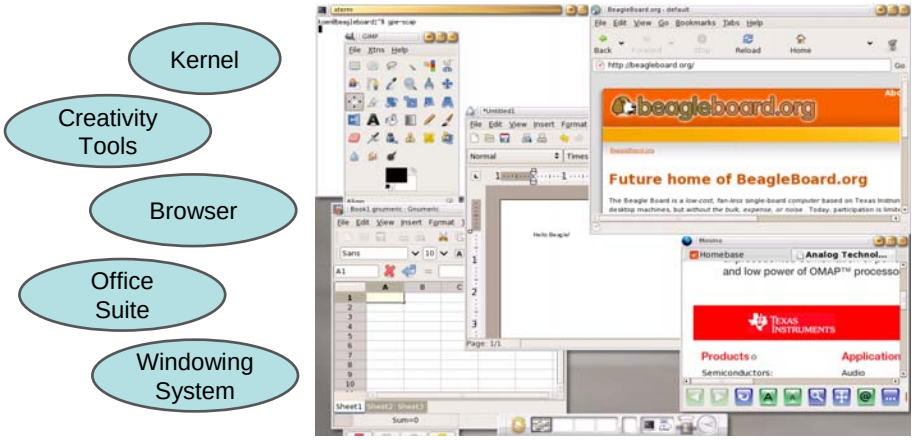
- This may be a bit of an understatement – even experienced users usually use a distribution
- Creating a distribution takes a lot of effort
- Maintaining a distribution … takes even more effort
- In fact, using a distribution even takes quite a bit of effort

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.

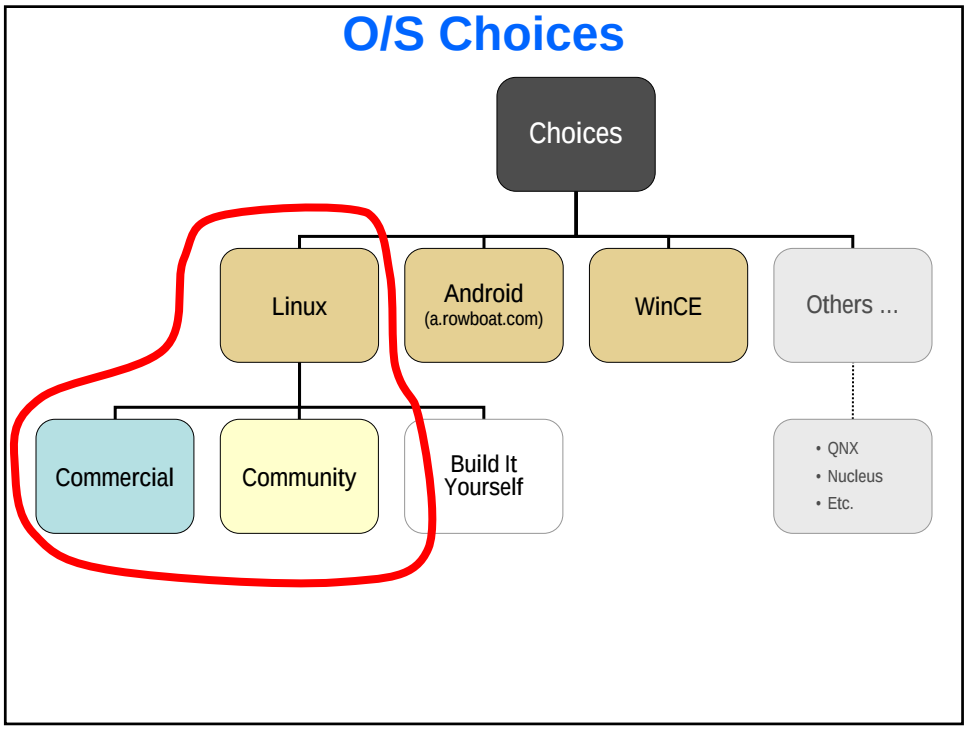
Linux Distributions

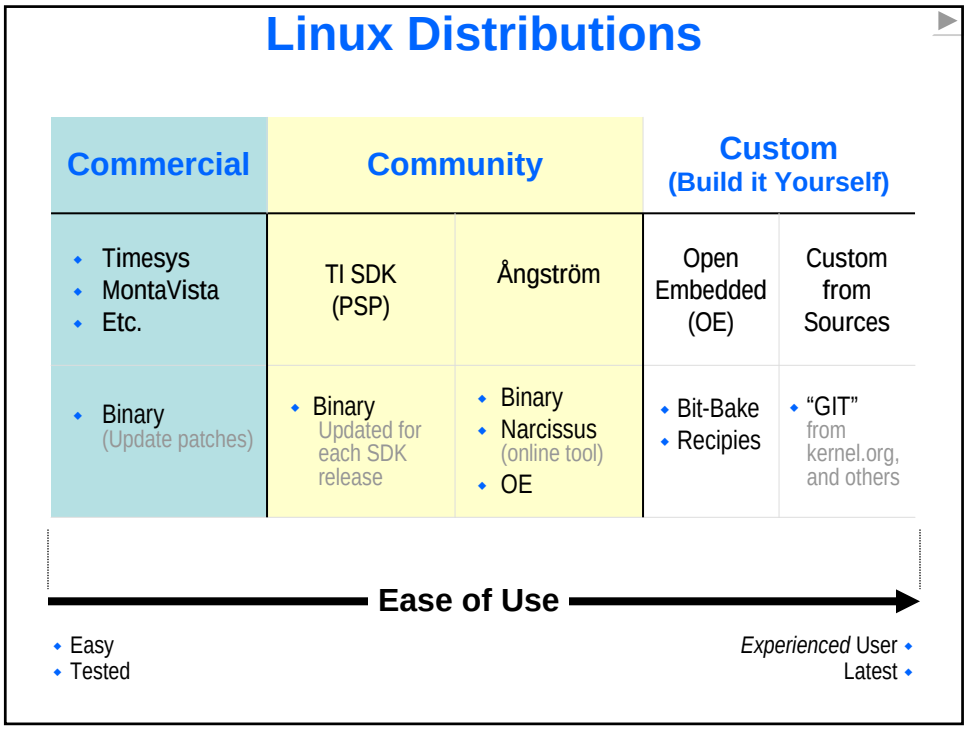


- Linux isn't complete without a distribution
- MontaVista and Timesys, for example, provide commercial (i.e. production) distribution for TI's DaVinci / OMAP processors
- A few distributions supporting the open-source BeagleBoard (OMAP35x-based) include: OpenEmbedded, Ubuntu, Fedora, Android, Gentoo, 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
Choices
Linux
Android
(a.rowboat.com)
WinCE
Others ...
Commercial
Community
Build It
Yourself
• QNX
• Nucleus
• Etc.


Linux Distributions
Commercial
Community
Custom
(Build it Yourself)
Timesys
MontaVista
Etc.
TI SDK
(PSP)
Ångström
Open
Embedded
(OE)
Custom
from
Sources
Binary
(Update patches)
Binary
Updated for
each SDK
release
Binary
Narcissus
(online tool)
OE
Bit-Bake
Recipies
"GIT"
from
kernel.org,
and others
Ease of Use
Easy
Tested
Experienced User
Latest

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

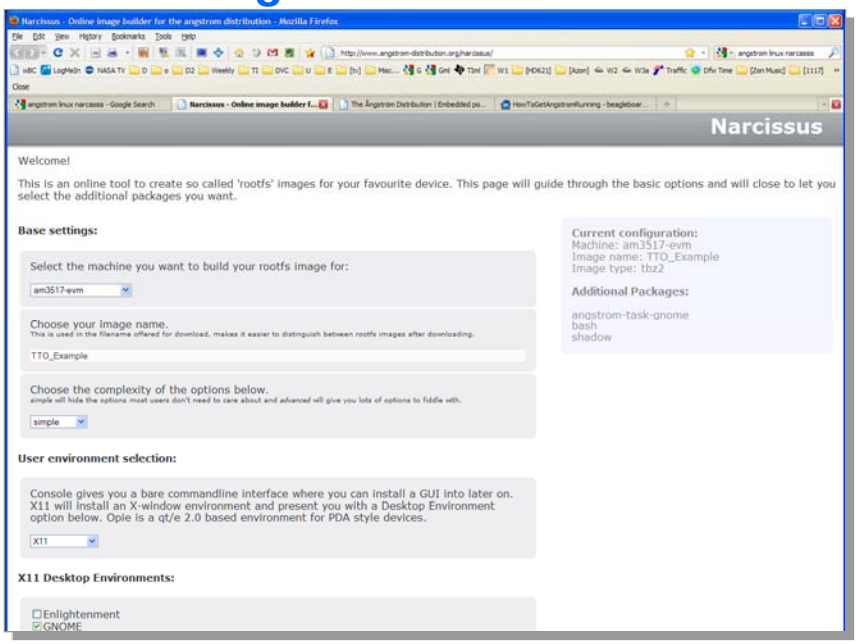
- **TI Linux SDK (PSP)**
  - Pre-built snapshot of Linux tested against specific version of TI Software Development Kits
  - Updated at each new SDK/DVSDK release
  - PSP = Platform Support Package (name of TI team)
  - Currently, a “BusyBox-based” bare-bones distro (“lean/mean”)
  - Arago open-source OE project
    - Advantage of OE – recipies 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 – recipies 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 recipies)

Ångström Narcissus

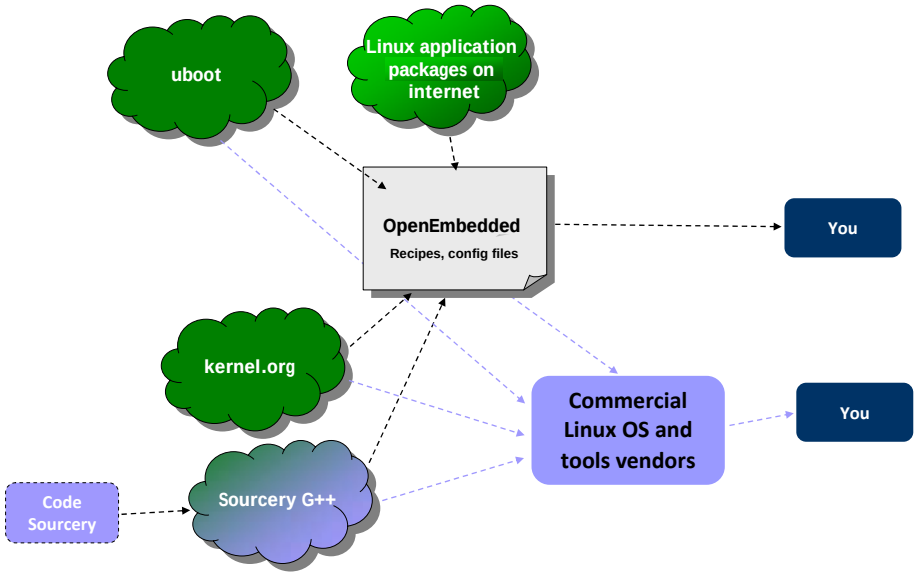


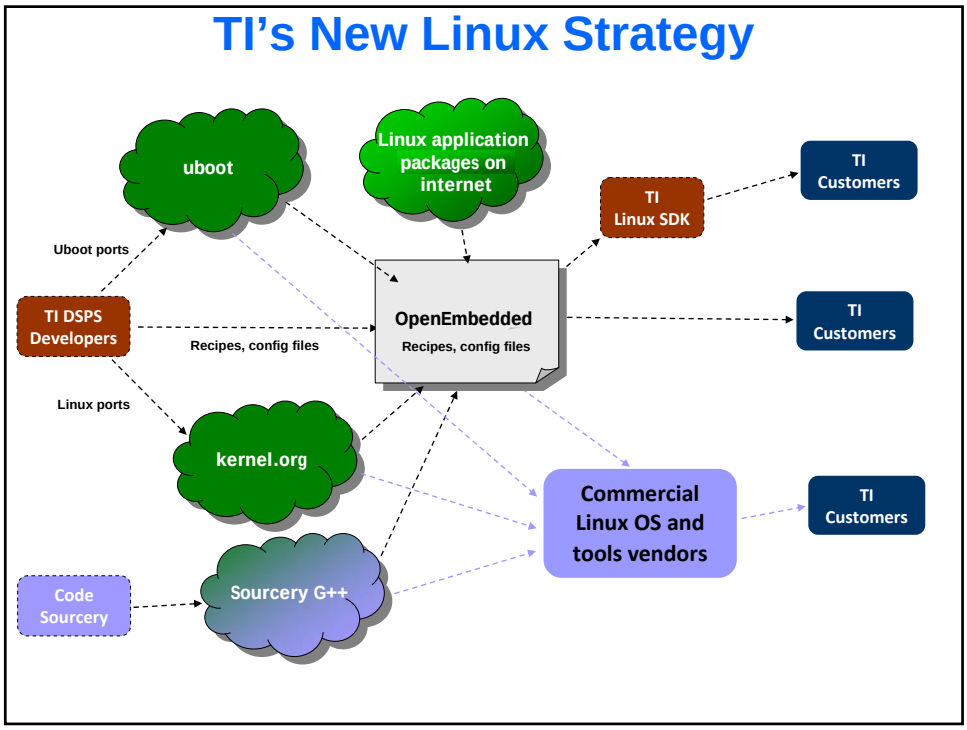
(http://www.angstrom-distribution.org/narcissus)

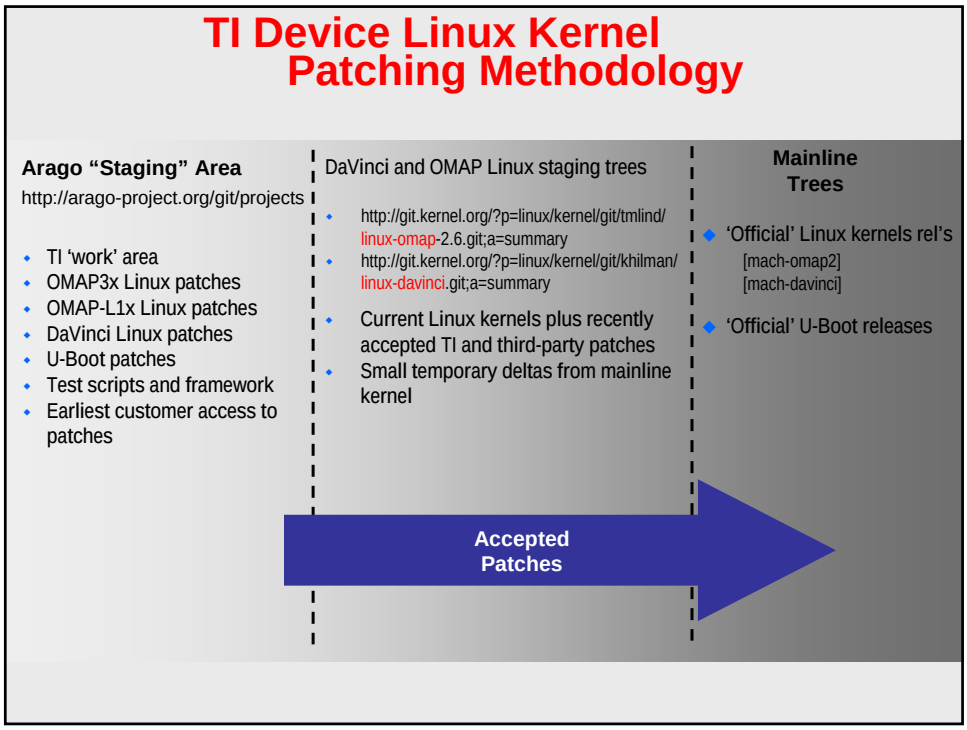
DIY Options

- **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 it's 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.*

DIY Linux : Open Embedded (OE)




TI's New Linux Strategy
uboot
Linux application packages on internet
TI Linux SDK
TI Customers
Uboot ports
TI DSPS Developers
Recipes, config files
OpenEmbedded
Recipes, config files
TI Customers
Linux ports
kernel.org
Commercial Linux OS and tools vendors
TI Customers
Code Sourcery
Sourcery G++


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
http://git.kernel.org/?p=linux/kernel/git/tmlind/linux-omap-2.6.git;a=summary
http://git.kernel.org/?p=linux/kernel/git/khilman/linux-davinci.git;a=summary
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

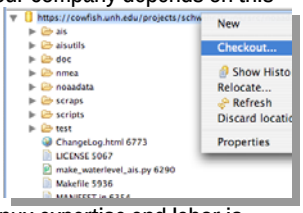
- Commercial: provide support, off-the-shelf Linux distributions or GNU tools
- Consultants: provide training, general embedded Linux development expertise, or specific expertise for developing drivers or particular embedded applications
- http://www.tiexpressdsp.com/index.php/Linux_Consultants_and_Commercial_Linux_Providers

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



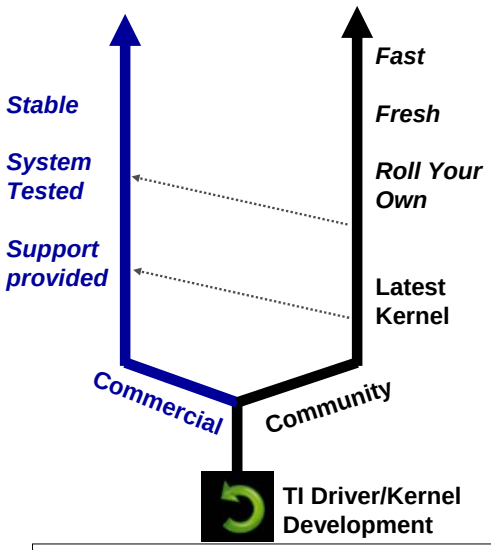
Looking at MVL vs GIT example…

Example Comparison : MVL Pro 5.0 vs GIT

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

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

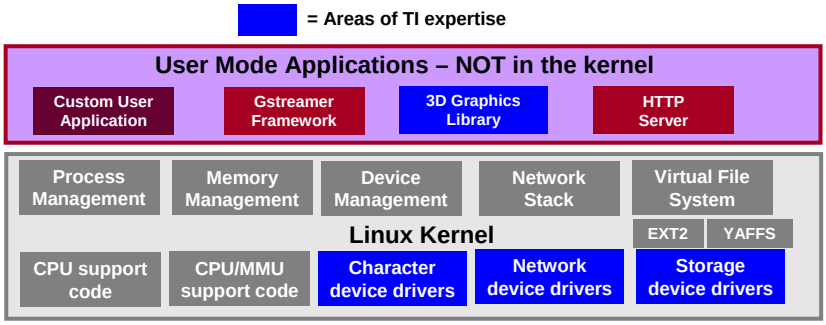
Texas Instruments 38

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

	TI	Community Forums
U-Boot	Only supports parts specific to TI devices	u-boot@lists.denx.de
Linux Kernel	Supports drivers specific to TI devices, not generic stack, scheduler, … http://e2e.ti.com http://www.tiexpressdsp.com/index.php/Category:Linux	http://www.kernel.org/pub/linux/docs/lkml/ for generic kernel & many specific forums such as http://www.linux-usb.org/mailing.html and https://www.redhat.com/mailman/listinfo/video4linux-list
GNU tools	Does not support	http://gcc.gnu.org/lists.html covers numerous gcc- and gdb-related lists
OpenEmbedded	Support limited to specific questions about TI devices or packages	http://wiki.openembedded.net/index.php/Mailing_Lists
User-mode applications	TI does NOT support Linux application code, except for applications provided in the DVSDK	Applications have their own mailing lists: http://gstreamer.freedesktop.org/lists/, http://httpd.apache.org/lists.html

Texas Instruments

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.)
- …

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

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

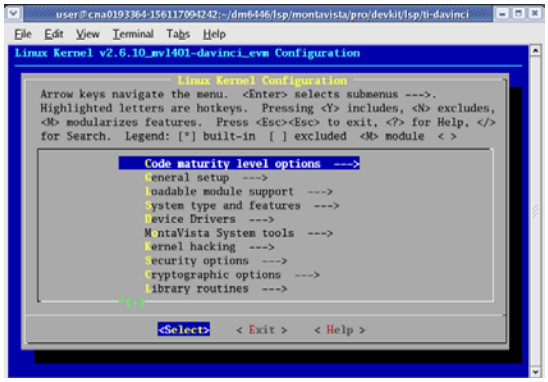
```
> make ARCH=arm CROSS_COMPILE=arm_v5t_le- davinci_dm644x_defconfig
```

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



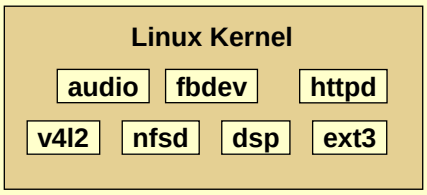
Outline

- 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)

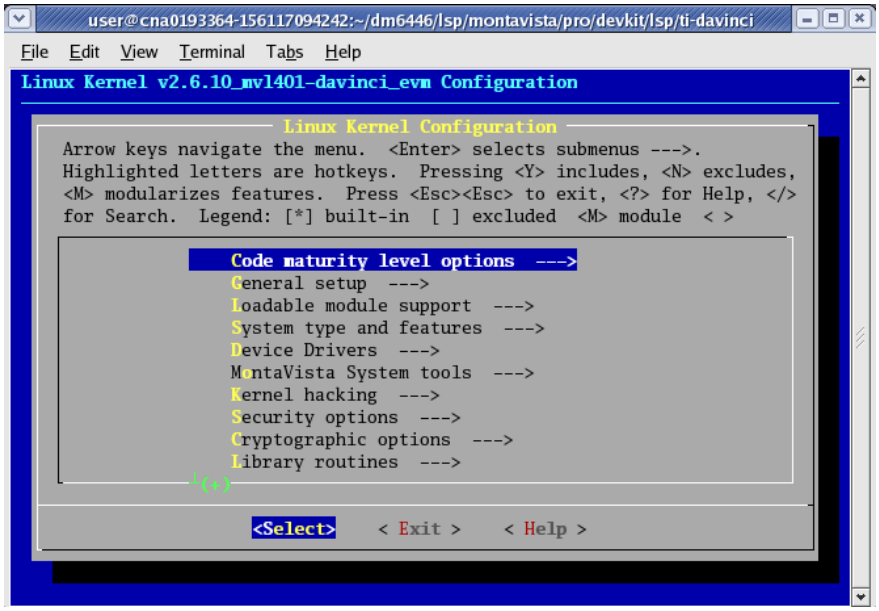


Kernel Module Examples:

fbdev	frame buffer dev
v4l2	video for linux 2
nfsd	network file server dev
dsp	oss digital sound proc.
audio	alsa audio driver

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

Static Linux Kernel Configuration




```
#> 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**

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

- 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

.ko = kernel object

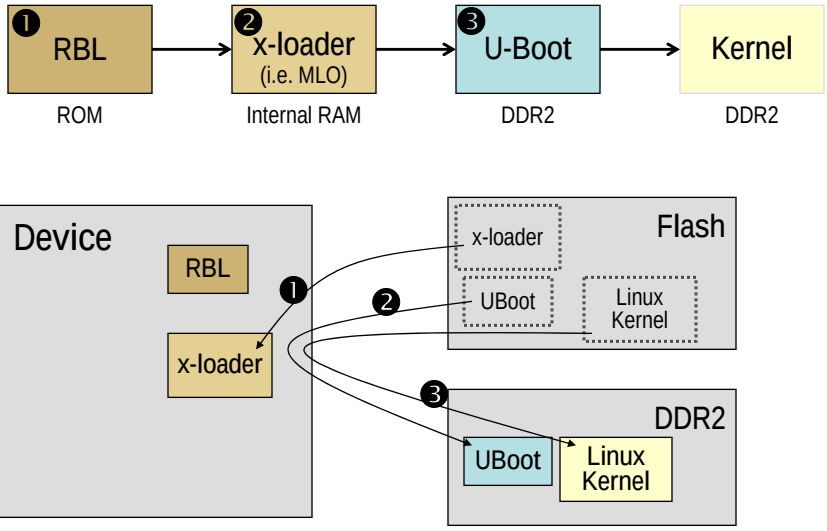
Getting Started with Linux Workshop

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



Bootloader Components

Boot stage	Operations	User Config'd	DaVinci	Sitara
First-level	This is ROM'd code for detecting desired boot type (NAND, UART, ...) and loading executable code of second-level bootloader from selected peripheral/interface	No	RBL	RBL
Second-level	The primary function of this boot loader is to initialize external memory and system clocks so that a larger, more advanced boot loader (in this case U-boot) can be loaded.	Board Designer	UBL	XLDR
Linux boot	"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 uImage format and is also responsible for passing arguments to the kernel	Yes	U-boot	U-Boot

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

Outline

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

To Boot Linux, You Need…

1. Bootloader (U-Boot)
2. Linux Kernel
3. Filesystem

- At reset, U-Boot bootloader is executed
- 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

Where are these 3 located?

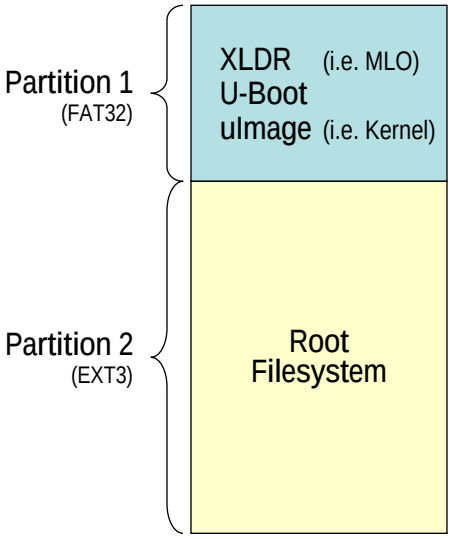
Where Do You Find ...

Where located:	DM6446 EVM Default	AM3517 1-day Wkshp	
1a. UBL or Xloader/MLO	Flash	MMC	
1b. Bootloader (U-Boot)	Flash	MMC	
2. Linux Kernel	Flash	MMC	
3. Filesystem	Hard Drive	MMC	
	"HDD boot"	"MMC 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 Flash. Also, the SDK provides an MMC image you can burn to a card.

The MMC card would look like ...

SD / MMC Boot



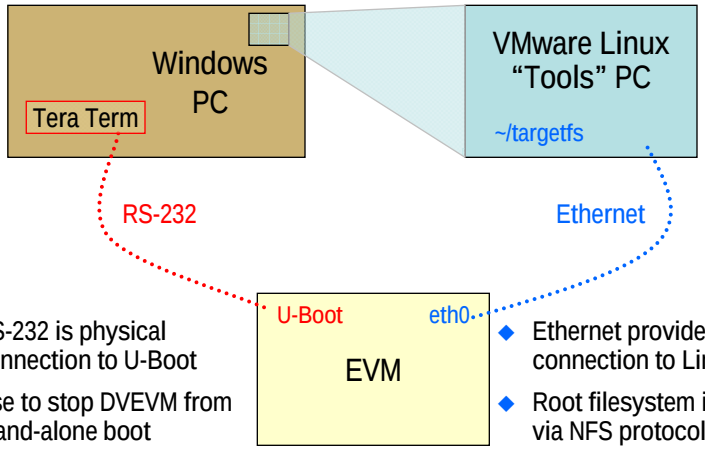
A 3rd boot option is ...

Where Do You Find …

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

DSP TEXAS INSTRUMENTS TECHNOLOGY

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
  - bootcmd - 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

Mode	IP	Linux Kernel	Root Filesystem
1.	dhcp	Flash	HDD
2.	dhcp	Flash	NFS
3.	dhcp	TFTP	HDD
4.	dhcp	TFTP	NFS
5.	dhcp	MMC	MMC

U-Boot's bootcmd variable specifies the root filesystem

Flash	`setenv bootcmd bootm 0x2050000`
MMC	`setenv bootcmd "mmc init; fatload mmc 0 ${loadaddr} uImage; run mmcargs; bootm ${loadaddr}"`
TFTP	`setenv bootcmd 'dhcp;bootm'`

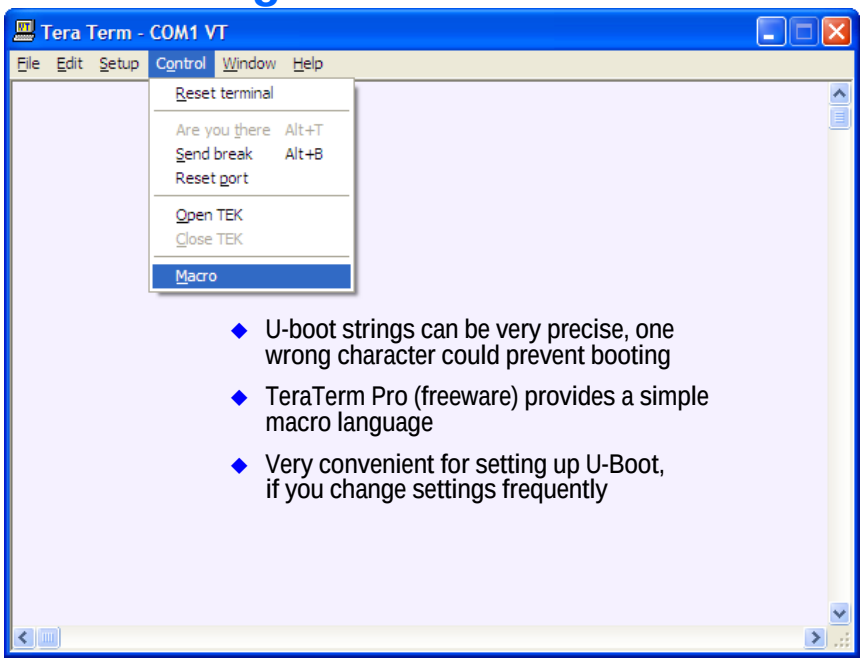
Boot Variations (filesystem)

Mode	IP	Linux Kernel	Root Filesystem
1.	dhcp	Flash	HDD
2.	dhcp	Flash	NFS
3.	dhcp	TFTP	HDD
4.	dhcp	TFTP	NFS
5.	dhcp	MMC	MMC

U-Boot's bootargs variable specifies the root filesystem

HDD	`setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/hda1, nolock mem=120M`
MMC	`setenv bootargs console=ttyS0,115200n8 noinitrd root=/dev/mmcblk0p2 rootfstype=ext3 rootwait nolock mem=120M`
NFS	`setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs nfsroot=$(serverip):$(nfspath),nolock mem=120M`

Using Tera Term Macros

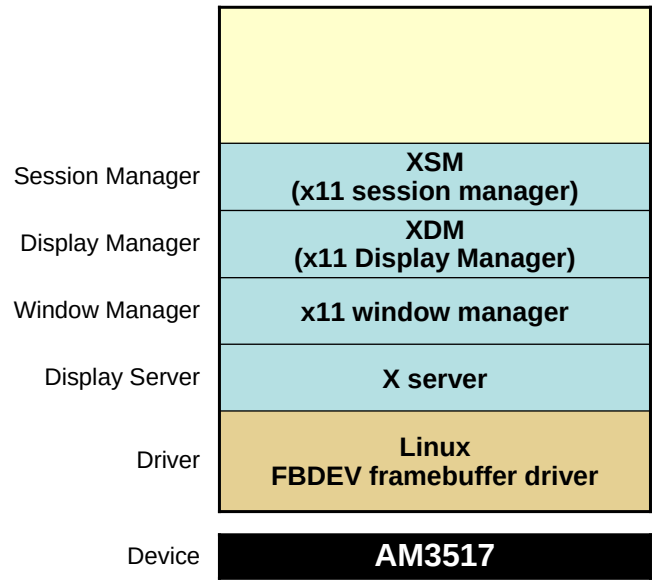


- U-boot strings can be very precise, one wrong character could prevent booting
- TeraTerm Pro (freeware) provides a simple macro language
- Very convenient for setting up U-Boot, if you change settings frequently

Getting Started with Linux Workshop

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

x11 Without Desktop

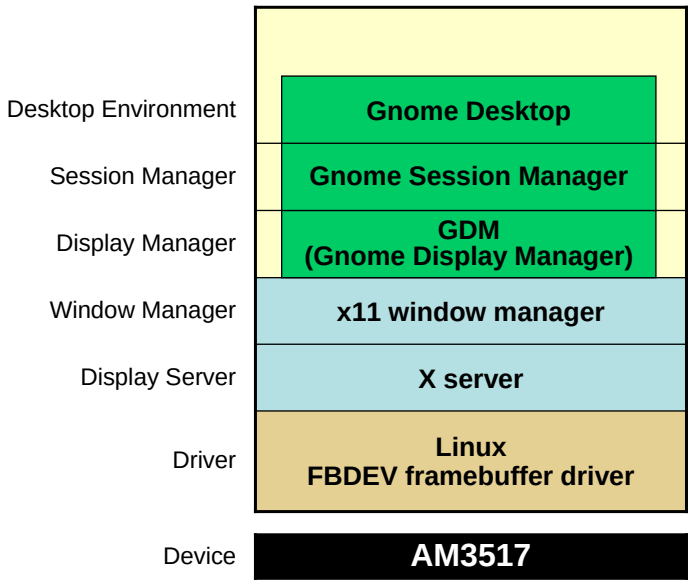


Gnome Desktop Environment

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

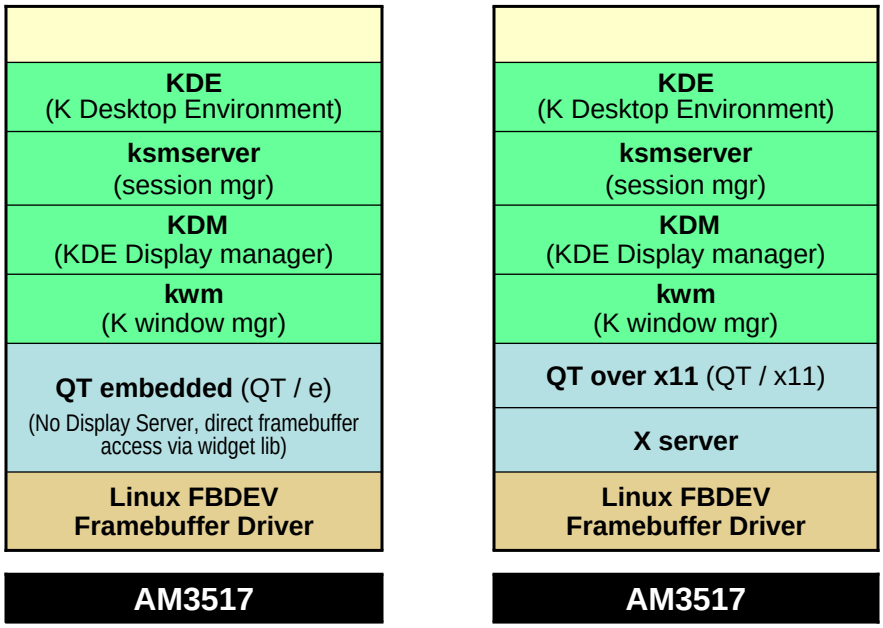
Looking at Gnome graphically ...

Gnome Desktop Environment

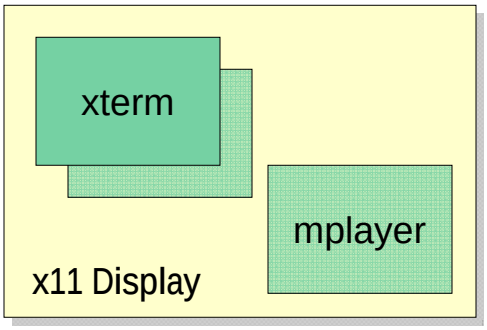


KDE ... as a quick comparison

K Desktop Environment



x11 (Window Manager)

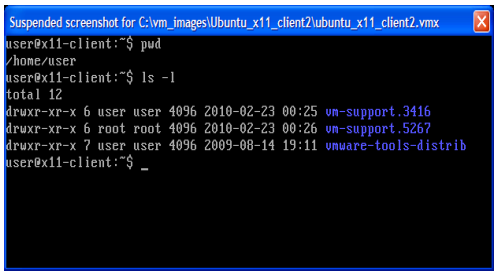


- **Also known as X, or X Window**
- **x11 is the most common Linux display manager**
- **x11 also manages Human Interface Devices (HIDs)**
- **Various applications send graphical data to x11, which displays each in an independent, resizable window**

Desktop Managers

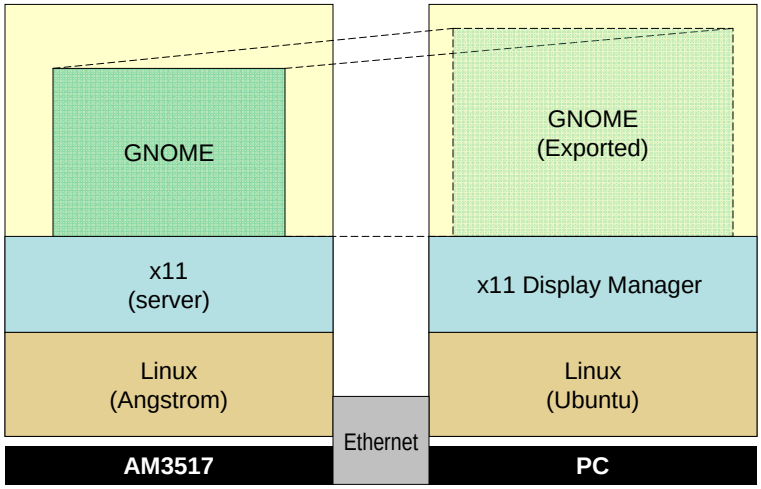
- Gnome is one of many Linux display managers
- Desktop managers provide a “Desktop” or similar environment (i.e. for handheld device)
- Applications (such as mplayer or xterm) typically interface directly to x11 and are therefore independent of the Display Manager
- Gnome display manager is “just another 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



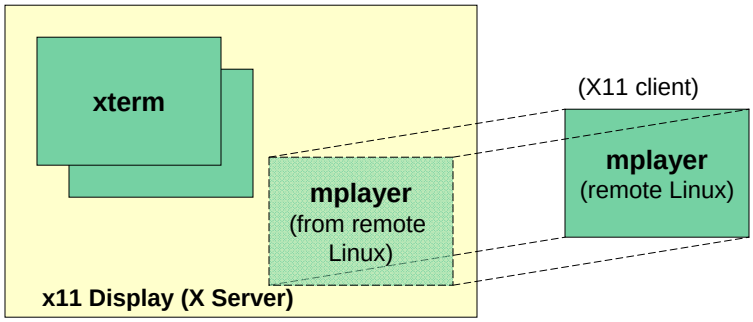
- A terminal is a text-based interface to Linux O/S
- Commands are interpreted by a shell program (such as BASH)
- A terminal may connect via RS-232, Ethernet, etc.
- xterm is a terminal which connects via the X window server, i.e. receives input via x11 HID and outputs to x11 display
- Most applications 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

Exported Desktop



- 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

Remote x11



- 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 (Destop 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

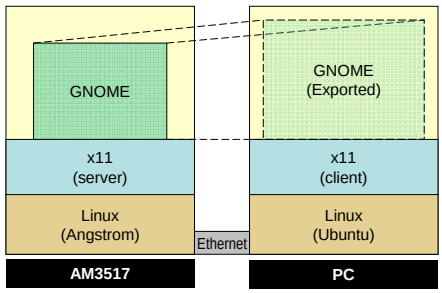
Lab 1: Connect and boot the AM3517 EVM

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

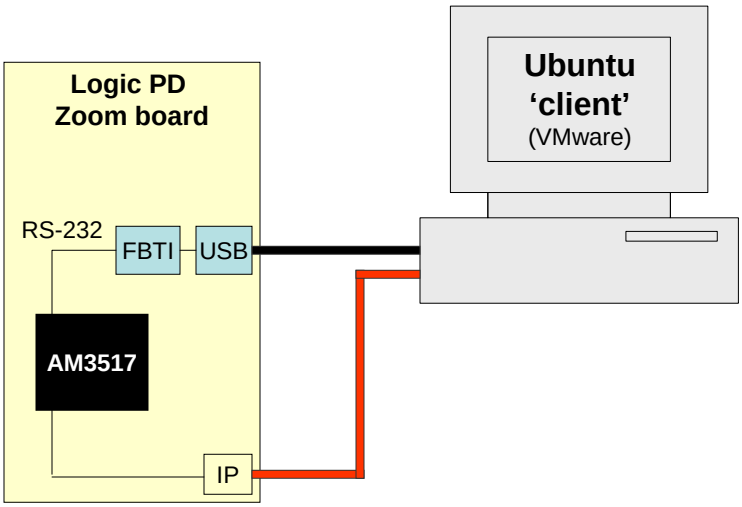
Lab 3: Running video (and audio) on the AM3517

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

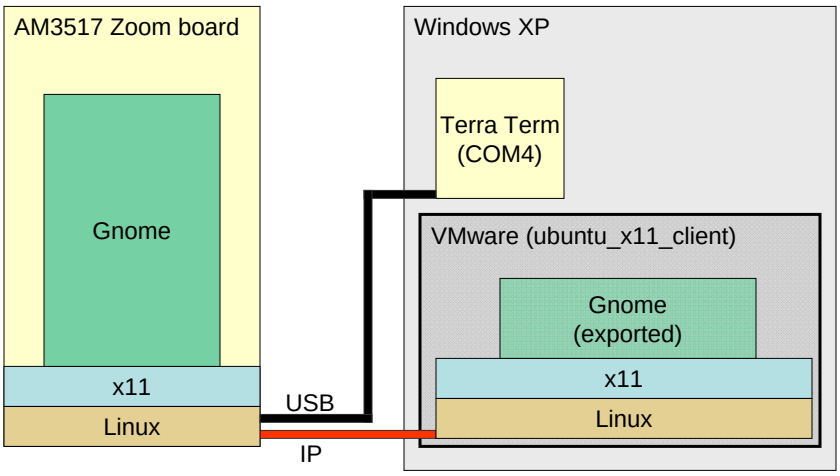
Lab 4: Investigating Linux Networking with the AM3517

a. Configure Linux Ethernet connection, both temporarily and permanently
b. Configure a Samba Server
c. Configure an HTML Server

Lab Setup (H/W)



Lab Setup (S/W)


AM3517 Zoom board
Gnome
x11
Linux
USB
IP
Windows XP
Terra Term
(COM4)
VMware (ubuntu_x11_client)
Gnome
(exported)
x11
Linux