IDF2011
INTEL DEVELOPER FORUM

Creating a Custom Embedded Linux* OS for Any Embedded Device using the Yocto Project*

Hands-on Lab

Tracey Erway - Product Marketing Engineer, Intel Corporation
Scott Garman - Embedded Linux Engineer, Intel Corporation
Ishu Verma - SW Technical Marketing Engineer, Intel Corporation

SFTL003

Sponsors of Tomorrow.™ intel

Agenda

- Introduction to the Yocto Project
- Key Concepts
- Recipes In-Depth
- Using Layers
- Building an Image
- Using the Emulator Environment
- Rebuilding for a New Target Device
- Tools for Application Development

The PDF for this Session presentation is available from our Technical Session Catalog at the end of the day at: intel.com/go/idfsessions

URL is on top of Session Agenda Pages in Pocket Guide

Welcome!

Getting to know you

- What brought you to this hands-on lab?
- Are you currently using Linux*?
- How are you building your Linux?
- What problems are you working on with embedded Linux?
- What topics are you most interested in?

Meet the Yocto Project*

The Yocto Project* is an open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of hardware architecture.

- Focused resources for system application developers who need to customize a Linux distribution for a device
- Validated and tested BSPs in a common format
- Automatically creates an application development SDK customized for each specific device
- Supported by embedded industry leaders across multiple architectures (IA, ARM, PowerPC, MIPS, etc)
- Is a great starting point for “roll your own” embedded developers and commercial distribution vendors.
- Enables easy transition from Proof of Concept (POC) to supported Commercial Linux with no loss of optimizations, code or design
- Proprietary code can be included in build structure within a separate layer, which can be kept private. (security)
- Project hosted by the Linux* Foundation

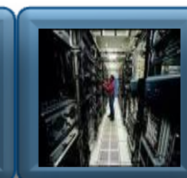
yocto PROJECT

It’s not an embedded Linux distribution – It creates a custom one for you.

Simple Electronics

M2M

Point of Sale



Networking & Storage

Industrial & Medical

www.yoctoproject.org

Participating Organizations

Silicon Vendors

OSVs

Embedded Tools, Consulting Services, Users…

Contact the Linux Foundation

If you are interested in becoming a participating organization.
(Take part in Governance, Advisory Board, Advocacy and Communications)
Project hosted by the Linux Foundation

Intel Roadmap – New BSPs at 1.1 Release

2Q 2011			3Q 2011			4Q 2011			1Q 2012			2Q 2012			3Q 2012		
APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP

Software Platform

- Yocto 1.0 (APR 2011)
- Yocto 1.0.1 (JUN 2011)
- Yocto 1.1 (OCT 2011)
- **Yocto 1.2** (APR 2012)

Hardware Platforms

- Yocto Project* BSP for Black Sand[1] 1-N450 Intel® Embedded Development Board[1] (Intel® Atom™ Processor N450)[2]
- Yocto Project BSP for Crown Bay Platform[1] (Intel® Atom™ Processor E6xx Series)[2]
- Yocto Project BSP for EMenlow Platform[1] (Intel® Atom™ Processor Z5xx Series)[2]
- Yocto Project BSP for Jasper Forest Platform[1] (Intel® Xeon® Processor 5500 and 3500 Series)[2]
- Yocto Project BSP for Sugar Bay Platform[1] (Intel® Core™ i3 Processor, Intel® Core™ i5 Processor, Intel® Core™ i7 Processor)[2]

Yocto Project v1.0 Feature Sampling

- Improved performance from v0.9
- Bitbake (build tool) parses metadata in parallel
- Improved reliability in Bitbake's fetcher
- Sandy Bridge BSP
- Supports most recent Linaro kernel and ARM* toolchain

Yocto Project v1.0.1 Features

- Maintenance and bugfix release v1.0

Yocto Project v1.1 Feature Sampling

- Create a compelling Image Creator interface.
- Complete multi-lib and OE-core configuration work.
- Documentation and/or tutorials to ease BSP creation.
- Improved Build Performance
- Upstream features to reduce the number of patches in Yocto Project

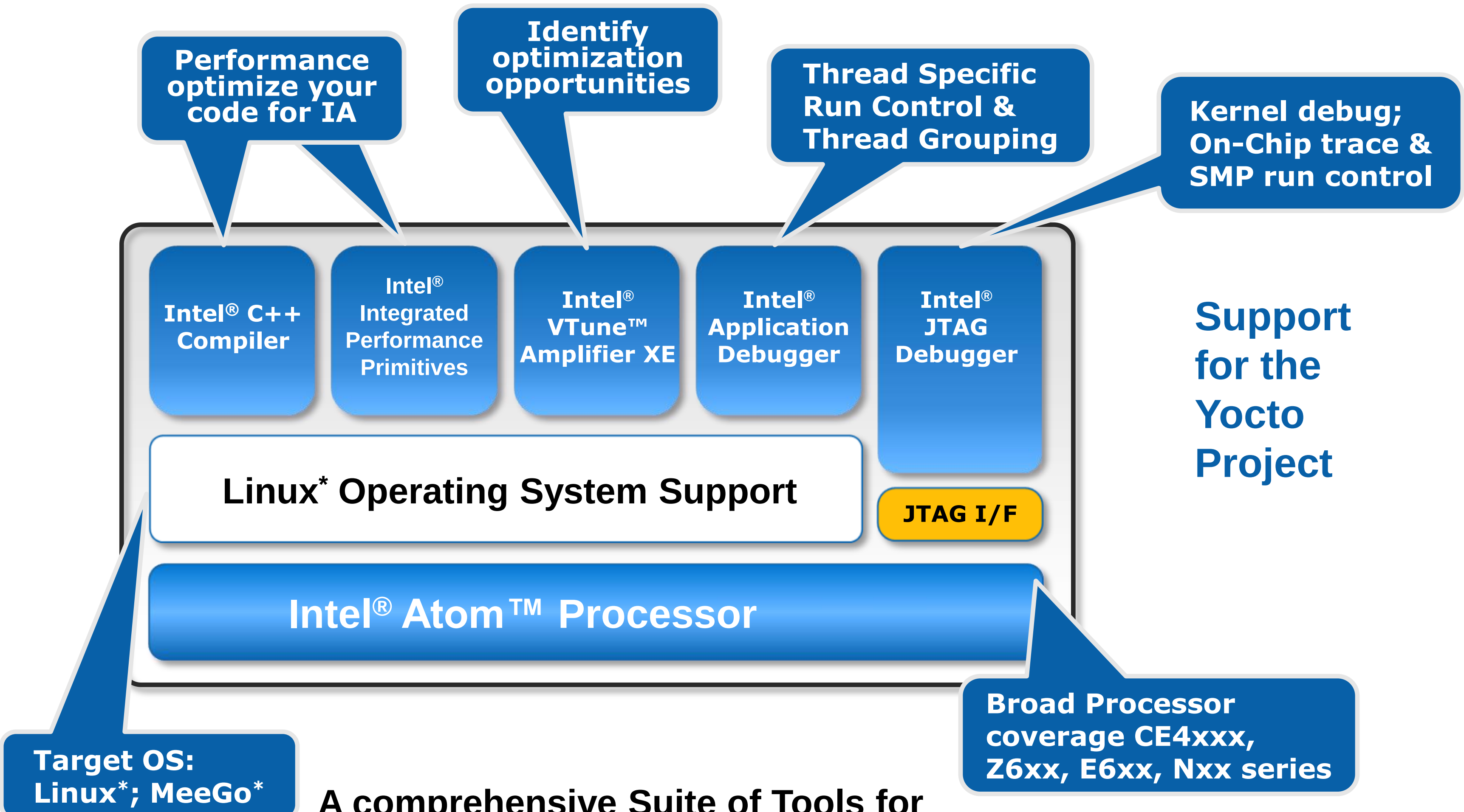
- Wind River has adopted parts of Yocto Project in WR Linux* 4.2 and 4.3
- "Plan of Vision" for the WR Linux 5 series

[1] Code name. [2] BSPs (Board Support Packages) are available to enable the Yocto open source project with no factory support implied or intended. * Other names and brands may be claimed as the property of others. Commercial supported Yocto project based OS distributions come from OSVs.

Where to Get Things

WHAT	WHO SUPPORTS	WHERE DISTRIBUTED	HOW TO GET
BSPs in common Yocto Project format	Community	YOCTO PROJECT WEBSITE	www.yoctoproject.org
Complete platform configuration, environment,	Community	YOCTO PROJECT WEBSITE	www.yoctoproject.org
Embedded Media and Graphics Driver - EMGD (Atom)	Yocto Project will test specific configurations - provided on website.	YOCTO PROJECT WEBSITE or ECG EDC WEBSITE	www.yoctoproject.org Integrated Image Or www.edc.intel.com Driver
Commercial OS Commercial Support	OSV	OSV	Thru OSV

Intel® Embedded Software Development Tool Suite for Intel® Atom™ Processor



Support for the Yocto Project

A comprehensive Suite of Tools for Embedded Development, Analysis and System Debugging

http://software.intel.com/en-us/articles/intel-tools-for-intel-atom-processors/

A Few Benefits of The Yocto Project*

- One common Linux* OS for all major architectures
- Just change one line in a config file and rebuild
- Easy transition to a commercial embedded Linux
- Build a complete Linux system in about an hour from pre-compiled sources (about 90 minutes with X) – quick start
- Start with a validated collection of packages
- Access to a great collection of app developer tools (performance, debug, power analysis, Eclipse*)
- Use Kernel development tools to manage patches
- Access to interaction with the Embedded Open Community

Meet Scott Garman

Yocto Project* Lab Prerequisites

To get the most out of this hands-on lab, you should be familiar with the following concepts and technologies:

- **Makefiles**
- **Autotools**
- **Package formats: RPM and/or DEB**
- **Root filesystem**

At least some experience building software within a Linux* environment is recommended

IDF2011
INTEL DEVELOPER FORUM

Key Concepts Agenda

- **Overview of the Yocto Project* Build System**
- **Yocto Project* Workflow**
- **Quick Start Guide in a Slide**
- **Exercise 1: Poky Directory Tree Layout**

Yocto Project* Build System Overview

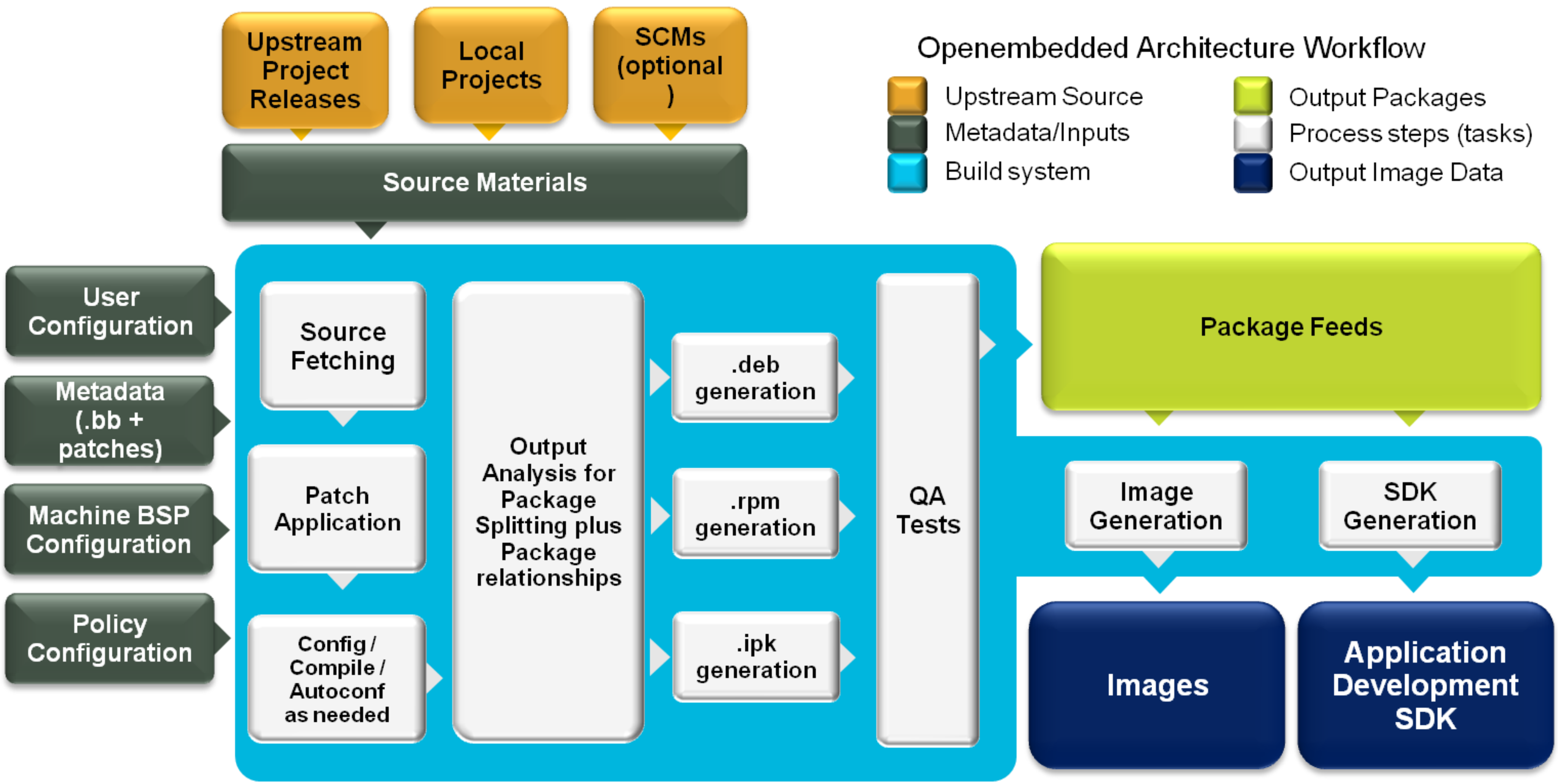
Poky = BitBake + metadata

- **Poky** – build system used by the Yocto Project*
- **BitBake** – a task executor and scheduler
- **Metadata** – task definitions
  - **Configuration (.conf)** – global definitions of variables
  - **Classes (.bbclass)** – encapsulation and inheritance of build logic, packaging, etc.
  - **Recipes (.bb)** – the logical units of software/images to build

Key Concepts

- The Yocto Project* provides tools and metadata for creating custom Linux* images
- These images are created from a repository of 'baked' recipes
- A **recipe** is a set of instructions for building packages, including:
  - Where to obtain the upstream sources and which patches to apply
  - Dependencies (on libraries or other recipes)
  - Configuration/compilation options
  - Define what files go into what output packages

Yocto Project* Workflow



IDF2011 INTEL DEVELOPER FORUM

Quick Start Guide in a Slide

Obtain our sources:

- Download poky-bernard-5.1.0-m3.tar.bz2
- tar xjf poky-bernard-5.1.0-m3.tar.bz2
- cd poky-bernard-5.1.0-m3

Build a Linux* image:

- source oe-init-build-env
- MACHINE=qemux86 bitbake core-image-minimal

some time passes

Run the image under emulation:

- runqemu qemux86

Exercise 1: Poky Directory Tree Layout

- **Objective:** Familiarize yourself with how the Poky metadata sources are organized

- Learn where you can find **conf files**, **BitBake class files**, and **recipe files**

Log into your lab computer:
Password: ***yoctoproject***

Poky Directory Tree Map

- **bitbake:** the BitBake utility itself
- **documentation:** documentation sources
- **scripts:** various support scripts (e.g, runqemu)
- **meta/conf:** important configuration files, bitbake.conf, reference distro config, machine configs for qemu architectures
- **meta/classes:** BitBake classes
- **meta/recipes-<xyz>:** recipes

Recipes In-Depth Agenda

- **Example Recipe: ethtool**
- **Standard Recipe Build Steps**
- **Exercise 2: Examining Recipes**

Example Recipe – ethtool_2.6.36.bb

SUMMARY = "Display or change ethernet card settings"

DESCRIPTION = "A small utility for examining and tuning the settings of your ethernet-based network interfaces."

HOMEPAGE = "http://sourceforge.net/projects/gkernel/"

LICENSE = "GPLv2+"

SRC_URI = "${SOURCEFORGE_MIRROR}/gkernel/ethtool-${PV}.tar.gz"

inherit autotools

Standard Recipe Build Steps

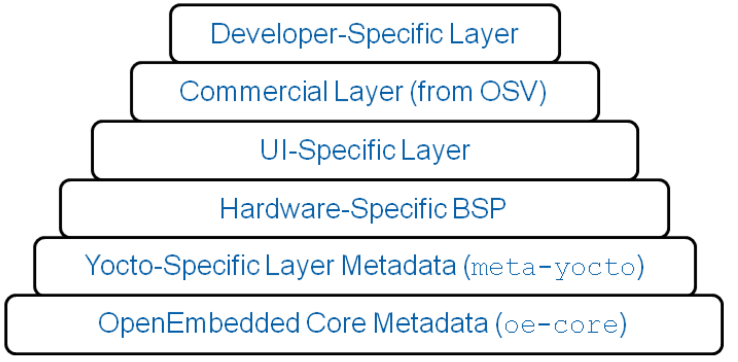
- Building recipes involves executing the following functions, which can be overridden when needed for customizations
  - **do_fetch**
  - **do_unpack**
  - **do_patch**
  - **do_configure**
  - **do_compile**
  - **do_install**
  - **do_package**

Exercise 2: Examining Recipes

- **meta/recipes-extended/bc/**
  - Uses LIC_FILES_CHKSUM and SRC_URI checksums
  - Note the DEPENDS declaration
- **meta/recipes-core/psplash/**
  - Uses SVN for sources
  - Sets up an init service
- **meta/recipes-multimedia/flac/**
  - Includes custom source patches
  - Customizes autoconf configure options
  - Breaks up output into multiple binary packages

Layers Agenda

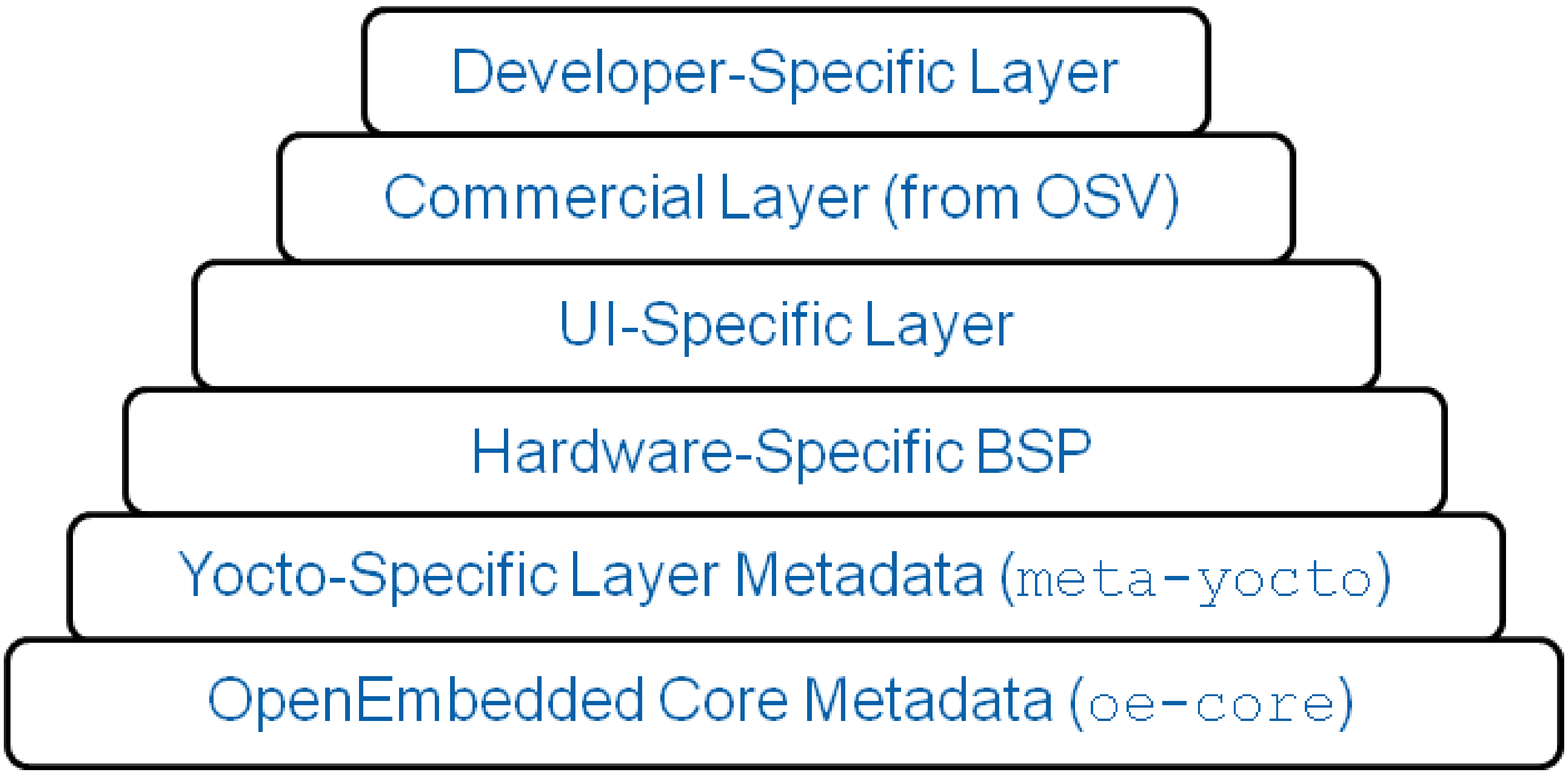
- **Introduction to Layers**
- **Stacking Customizations**
- **Adding Layers**
- **Board Support Packages**
- **Example machine configuration**
- **Kernel configuration**



Layers

- The Yocto Project* build system is composed of layers
- A **layer** is a logical collection of recipes representing the core, a Board Support Package (BSP), or an application stack
- All layers have a priority and can override policy and config settings of the layers beneath it

Stacking Customizations



Using Layers

- Layers are added to your build by editing the **build/conf/bblayers.conf** file:

```
BBLAYERS = " \
  /data/poky/meta \            # core system
  /data/poky/meta-yocto \      # yocto config and recipes
  /data/meta-skynet \          # my customization layer
  "
```

Board Support Packages

- BSPs are layers to enable support for specific hardware platforms
- Defines machine configuration for the "board"
- Adds machine-specific recipes and customizations
  - Kernel config
  - Graphics drivers (e.g, Xorg)
  - Additional recipes to support hardware features

Example Machine Configuration

```
TARGET_ARCH = "x86_64"

MACHINE_FEATURES = "kernel26 screen keyboard pci usbhost ext2 ext3 x86”
KERNEL_IMAGETYPE = "bzImage"

PREFERRED_PROVIDER_virtual/kernel = "linux-yocto"
PREFERRED_PROVIDER_linux-libc-headers ?= "linux-libc-headers-yocto"

PREFERRED_PROVIDER_virtual/libx11 ?= "libx
PREFERRED_PROVIDER_virtual/libgl  ?= "mesa
PREFERRED_PROVIDER_virtual/xserver ?= "xserver-xf86-dri-lite"
PREFERRED_PROVIDER_virtual/xserver-xf86 ?= "xserver-xf86-dri-lite"
XSERVER ?= "xserver-xf86-dri-lite \
        xf86-input-mouse \
        xf86-input-keyboard \
        xf86-video-intel"

MACHINE_EXTRA_RRECOMMENDS = "kernel-modules eee-acpi-scripts"
GUI_MACHINE_CLASS = "bigscreen"

IMAGE_ROOTFS_SIZE_ext3 = "2000000"
IMAGE_FSTYPES ?= "ext3 cpio.gz"

MACHINE_ESSENTIAL_EXTRA_RDEPENDS = "grub"
PREFERRED_VERSION_grub ?= "1.98"

SRCREV_machine_pn-linux-yocto_sugarbay ?= "41ec30ddc42912fec133a533b924e9c56ecda8f9"
SRCREV_meta_pn-linux-yocto_sugarbay ?= "5a32d7fe3b817868ebb697d2d883d743903685ae"
```

TARGET_ARCH = “x86_64”

PREFERRED_PROVIDER_virtual/kernel = “linux-yocto”

**XSERVER ?= “xserver-xf86-dri-lite \
xf86-input-mouse \
xf86-input-keyboard \
xf86-video-intel”**

IDF2011 INTEL DEVELOPER FORUM

Kernel Customization

- You can define a full kernel configuration set (defconfig) or use kernel configuration “fragments”
- Add a kernel configuration fragment (.cfg) to your layer
  - These include standard Linux* Kconfig values and are inserted into the generated defconfig
- Add a **linux-yocto.bbappend** recipe to your layer which includes your config file(s)

Adding E1000 Drivers

- meta-talk/recipes-kernel/linux-yocto/netdev.cfg:

CONFIG_NETDEV_1000=y
CONFIG_E1000E=y

- meta-talk/recipes-kernel/linux-yocto_git.bbappend:

SRC_URI_append = “file://netdev.cfg”

Images Agenda

- **Exercise 3: Building an Image**
- **Introduction to Images**
- **Example Image: my-nas-image.bb**
- **Booting an Image Under Emulation**
- **Exercise 4: Booting Your Image**

Exercise 3: Building an Image

- **cd ~/lab/poky**
- **source oe-init-build-env**
  - Sets up important environment variables
- **Set MACHINE="qemux86" in build/conf/local.conf**
  - Specifies that we're building an image for the qemux86 target
- **bitbake core-image-minimal**
  - Builds a minimal Linux image for the qemux86 target

Images

- Specify which packages to install
  - List individual package names and/or:
  - Set the **IMAGE_FEATURES** variable, which maps collections of packages (defined in task recipes) to named functionality, e.g, "apps-console-core package-management"
- Define commands to be run on the generated rootfs (e.g, installing configuration files into /etc)
- Built images are saved to **build/tmp/deploy/images/**

Example Image – my-nas-image.bb

```
IMAGE_FEATURES += "nfs-server apps-console-core package-management"

inherit poky-image

SRC_URI = "file://fstab \             # These files will be installed after the
           file://exports"            # rootfs is generated, see below

ROOTFS_POSTPROCESS_COMMAND += "setup_target_image ; "
setup_target_image() {
    # install configuration files
    install -m 0644 ${WORKDIR}/fstab ${IMAGE_ROOTFS}/etc/fstab
    install -m 0644 ${WORKDIR}/exports ${IMAGE_ROOTFS}/etc/exports
    # etc etc
}
```

Using Emulation

- Yocto uses QEMU, which supports all major architectures: x86(-64), arm, mips, ppc
- Simply set **MACHINE=qemux86** in local.conf and build your image
- **runqemu** script is used to boot the image with QEMU – it auto-detects as much as possible:

runqemu qemux86

Exercise 4: Booting Your Image

cd into your build/ directory, then run:

runqemu qemux86

Once the image has booted, log in as root (default password is empty, just hit Return)

IDF2011
INTEL DEVELOPER FORUM

Exercise 5: Changing Targets

- The Tunnel Creek boards use the “fri2” MACHINE type as defined in the meta-intel layer

- To build a core-image-minimal image which would boot on this board, simply edit your **build/conf/local.conf** file and set **MACHINE=“fri2”**

- Then rebuild: **bitbake core-image-minimal**

Embedded Software Development

- Embedded products are highly customized to provide special functions
- Quickly roll out new applications that utilize unique hardware features
- Embedded platforms needs
  - Run time supporting system
  - Application development
- Product-focused toolchain and development platform are essential for embedded software development

Yocto Project* ADT

Yocto Project* Application Development Toolkit

- Setup target system development environment on the host machine based on sysroot concept
  - GNU cross-development toolchain of build, packaging, and debug
  - Development headers and libraries
  - Sysroot represents target device root file system
- Optimized for use with Autotools
  - For autotool-enabled packages just pass host options to configure
  - For other projects should ensure the cross tools are used

Yocto Project* ADT (Cont.)

Yocto Application Development Toolkit

- Use hardware as development targets - Qemu with GL pass-through
- User mode NFS support
  - Allow emulator and host access the file system at same time
- Update packages on running systems and sysroot
- ADT installer, Eclipse plug-in and user space tool suite

Allow software and hardware development to happen in parallel

IDF2011
INTEL DEVELOPER FORUM

Yocto Project* 1.1 Upcoming Features

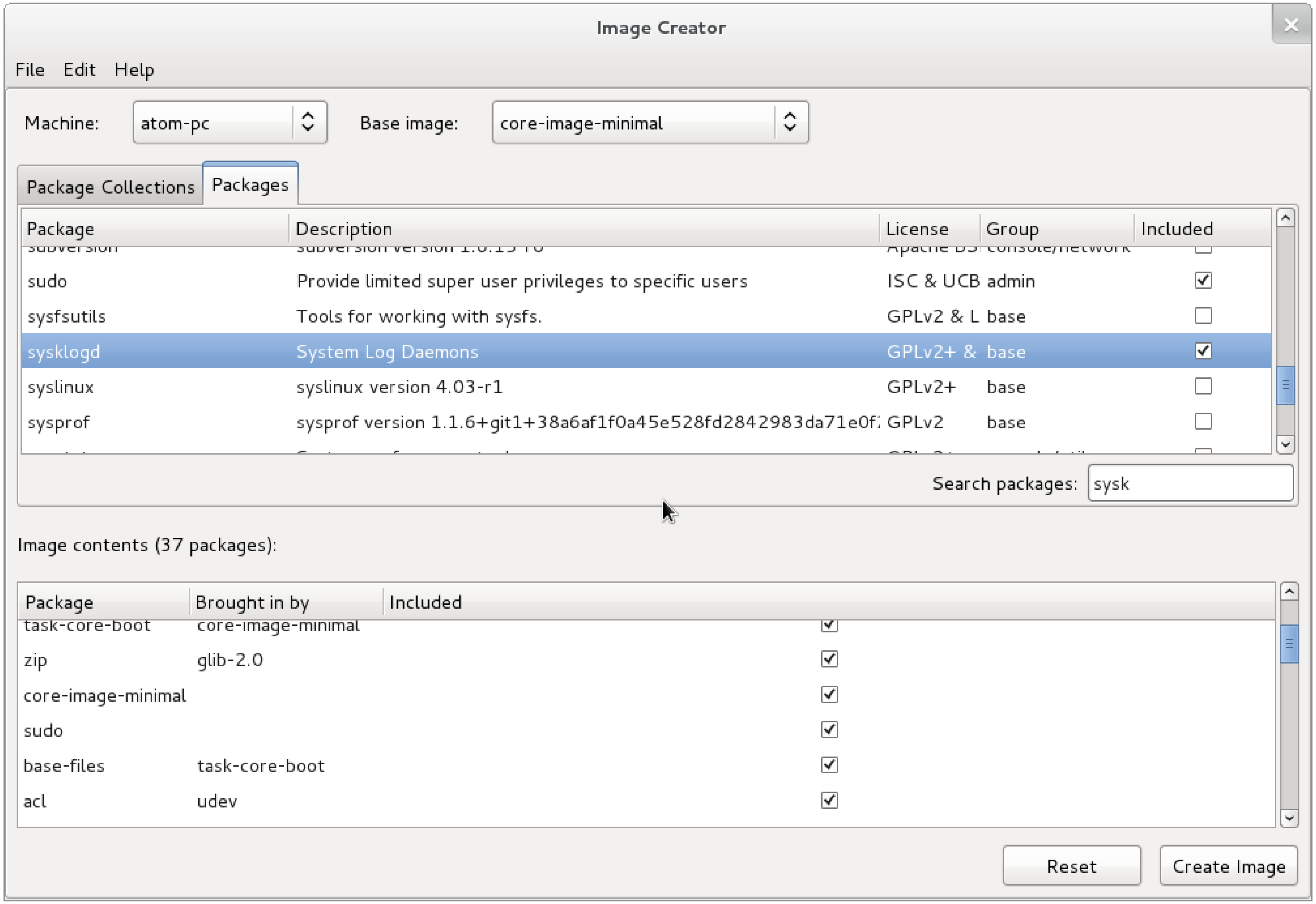
- **Multilib** – images which support 32-bit and 64-bit libraries installed at the same time
  - Use 64-bit support for specific applications, i.e. your actual product
- **x32 layer** – 32-bit memory address space using the CPU in 64-bit mode
  - Allows full use of 64-bit registers in the CPU with 32-bit pointers

IDF2011
INTEL DEVELOPER FORUM

Yocto Project* 1.1 Upcoming Features

- Enhanced layer tooling – to make layer creation and use easier and more robust
- Updated software – GCC v4.6, newer eglibc, etc.
- Image creator GUI – select the desired contents of the image, the target BSP and go. Easier to use than the command line and a text editor

Image Creator (Under Development)



Project Resources

- The Yocto Project* is an open source project, and aims to deliver an open standard for the embedded Linux* community and industry
- Development is done in the open through public mailing lists: openembedded-core@lists.openembedded.org, poky@yoctoproject.org and yocto@yoctoproject.org
- And public code repositories:
- http://git.yoctoproject.org and http://git.openembedded.net
- Bug reports and feature requests: http://bugzilla.yoctoproject.org

Please Fill out the Online Session Evaluation Form

Be entered to win fabulous prizes everyday!

Winners will be announced at 6pm (Day 1/2) and 3:30pm (Day 3)

You will receive an email prior to the end of this session.

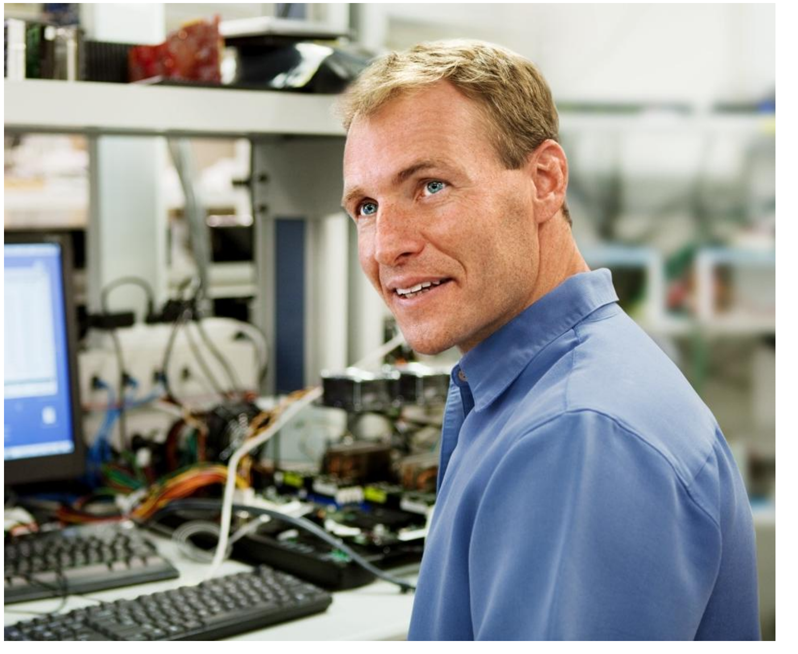
Scan Your Badge
Connect with Intel

- Speak with an Intel Representative:

  Have an Intel representative contact you by phone or email with information about Intelligent Connected Solutions!

- Get Free "Tech Notes":

  Receive electronic updates and newsletters that share product and technology highlights and keep you posted on upcoming events, seminars, webinars, and more!



Intel Privacy Notice: http://www.intel.com/privacy

Visit the Intelligent Connected Solutions Zone

The intelligence in embedded.

IDF2011

INTEL DEVELOPER FORUM

Q&A

Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.
- Intel may make changes to specifications and product descriptions at any time, without notice.
- All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
- Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- Crown Bay, Menlow, Jasper Forest, Sugar Bay and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user
- Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.
- Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. Go to: http://www.intel.com/products/processor_number
- Intel product plans in this presentation do not constitute Intel plan of record product roadmaps. Please contact your Intel representative to obtain Intel's current plan of record product roadmaps.
- Intel, Atom, Core, VTune, Sponsors of Tomorrow and the Intel logo are trademarks of Intel Corporation in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
-

Risk Factors

The above statements and any others in this document that refer to plans and expectations for the second quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Words such as “anticipates,” “expects,” “intends,” “plans,” “believes,” “seeks,” “estimates,” “may,” “will,” “should,” and their variations identify forward-looking statements. Statements that refer to or are based on projections, uncertain events or assumptions also identify forward-looking statements. Many factors could affect Intel’s actual results, and variances from Intel’s current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be the important factors that could cause actual results to differ materially from the company’s expectations. Demand could be different from Intel's expectations due to factors including changes in business and economic conditions, including supply constraints and other disruptions affecting customers; customer acceptance of Intel’s and competitors’ products; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Potential disruptions in the high technology supply chain resulting from the recent disaster in Japan could cause customer demand to be different from Intel’s expectations. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Revenue and the gross margin percentage are affected by the timing of Intel product introductions and the demand for and market acceptance of Intel's products; actions taken by Intel's competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel’s response to such actions; and Intel’s ability to respond quickly to technological developments and to incorporate new features into its products. The gross margin percentage could vary significantly from expectations based on capacity utilization; variations in inventory valuation, including variations related to the timing of qualifying products for sale; changes in revenue levels; product mix and pricing; the timing and execution of the manufacturing ramp and associated costs; start-up costs; excess or obsolete inventory; changes in unit costs; defects or disruptions in the supply of materials or resources; product manufacturing quality/yields; and impairments of long-lived assets, including manufacturing, assembly/test and intangible assets. Expenses, particularly certain marketing and compensation expenses, as well as restructuring and asset impairment charges, vary depending on the level of demand for Intel's products and the level of revenue and profits. The majority of Intel’s non-marketable equity investment portfolio balance is concentrated in companies in the flash memory market segment, and declines in this market segment or changes in management’s plans with respect to Intel’s investments in this market segment could result in significant impairment charges, impacting restructuring charges as well as gains/losses on equity investments and interest and other. Intel's results could be affected by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Intel’s results could be affected by the timing of closing of acquisitions and divestitures. Intel's results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust and other issues, such as the litigation and regulatory matters described in Intel's SEC reports. An unfavorable ruling could include monetary damages or an injunction prohibiting us from manufacturing or selling one or more products, precluding particular business practices, impacting Intel’s ability to design its products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other factors that could affect Intel’s results is included in Intel’s SEC filings, including the report on Form 10-Q for the quarter ended April 2, 2011.

Rev. 5/9/11

Backup Slides

MeeGo* and Yocto Project*

Requirements	MeeGo* Operating System	Yocto Project
Target segments	Segments: • IVI • Smart TV • Netbook • Tablets, Media Phones • Smart Phone	• Other embedded segments • Ideal for Machine to Machine (M2M), Industrial, Military-Aerospace-Govt (MAG), Networking
Application ecosystem • API compliance • At a core Linux level • Higher up in the stack, can have compliance at the segment level (such as an IVI compliance)	This is the core idea behind MeeGo. Compliance adherence guarantees application reuse across MeeGo devices in different segments.	Full customization (for designs which don't need an app ecosystem) save on footprint. Expect single app devices.
Multi architecture support	IA, ARM*	IA, ARM, PowerPC, MIPS
	MeeGo is an OS	Yocto Project is a set of build tools which create an OS