[RFC] Obtaining Management Buy-in for Mainline Development

Kevin Chalmers  
Director, Linux Core Product Development  
Texas Instruments

Chase Maupin  
Manager, Linux Systems Integration  
Texas Instruments
Agenda

• Introduction
• Customer Perspective
• Management Perspective
• Conclusion – “Selling” Mainline to Management
Purpose

• This presentation seeks to give engineers the tools and skills required to push for mainline development methodology inside of a business environment.

• During this presentation you will learn to:
  – Define today’s customers
  – Define mainline in terms management can understand
  – Show mainline benefits from the customer and management perspective
  – Define what you need from management to be successful

• The goal is that you can use this presentation (with some modification) to drive mainline development within your own company
The “Ideal” Customer

• Takes whatever code you provide them
• Adds their components
• Never comes back to you with issues
• Given a data sheet they will/can implement their required software without you
• They don’t ever update their code or expect new features to be delivered
• They do all the productization and quality control themselves
• Do not expect more than beta quality code from you

If you have this customer please give us their contact information
The Real Customer

• Expects high quality code to be delivered that enables the required features, not just a data sheet

• Expects to be able to get new features that are available upstream

• Expects continued and iterative development

• Expects support of the code delivered to them for some period of time

• Customers are becoming more SW savvy and are demanding to see plans for how you will support the above.
Customer Perspective
The ecosystem matters!

What’s most important when choosing a microprocessor?

- The chip itself: 30%
- The ecosystem surrounding the chip (software, tools, support, etc.): 61%
- The chip's supplier/vendor: 9%

Source: 2012 Embedded Market Survey, UBM / EE Times Group
Software & tools are most important

What are the most important factors in choosing a processor?

- **Software development tools available**: 71%
- **The chip’s performance**: 48%
- **The chip’s cost**: 40%
- **Available middleware, drivers, existing code**: 35%
- **HW development tools available**: 33%
- **The operating systems it supports**: 32%
- **The on-chip I/O or peripherals**: 29%
- **The chip’s power consumption**: 22%
- **Familiarity w/ architecture/chip family**: 14%
- **The supplier’s reputation**: 14%
- **Chip family’s future growth path**: 13%
- **The processor’s debug support**: 8%
- **Programmable logic on chip**: 4%

Source: 2012 Embedded Market Survey, UBM / EE Times Group
What is the Customer Thinking?

• What issues are all customer’s looking to address?
  – Ability to Innovate
  – Time to Market
Ability to Innovate

• When you buy a new car what features do you expect to just work?
  – Power windows, door locks, windshield wipers, the radio, the car to simply start
  – These are commodity requirements

• Today’s new features are tomorrow’s ‘commodity’ requirements
  – Would you buy a new car today without power steering?

• Customers have the same expectations for the Linux kernel
  – They simply want the commodity support to just work

They want to innovate creative applications & products
NOT develop and debug commodity support
Time To Market

1. Take the ‘Commodity’ Development and Debug off the table
2. Pull in Development and Debug Phases
3. Release to Market Sooner

Commodity Feature
New Feature

Concept | Market Release | Market Release
---|---|---
Proto | USB | UART | MMC/SD | Power | Feature 1 | Feature 2 | Feature 3 | Support | Feature 4 | Feature 3 | Feature 4 | Support

Development Phase | Debug Phase |
---|---|
Development Phase | Debug Phase
Management Perspective
The “Problem”

- Senior Management often does not understand mainline or upstream development
  - Your technical manager may understand but many times you need to talk to higher level managers for resources

- Managers like control, schedules, predictability
  - Mainline usually involves working with projects that you do not directly “control”

You have to show them how they can still control their product while investing in mainline
What is Management Thinking?

• How is mainline support going to address the following issues:
  – Operational Efficiency (ROI)
  – Increasing Quality & Robustness
  – Reducing Customer Support
  – Reducing Time to Market (TTM) – customer’s and ours
“Selling” Mainline to Management

7 Important topics you must address…

1. Where do we want to go?
2. Scope of Mainline
3. Definition of Mainline
4. Challenges without Mainline
5. Mainline Development
6. 12 months from now…
7. What do we need to succeed?

Determining a Successful Outcome:

“Why aren’t we doing this today?”
Where do we want to go?

• How do we plan on improving the quality & robustness of our Linux software this year?

• How do we plan to scale our Linux software to:
  – Support 10,000 customers,
  – 10s of devices,
  – New reference designs

• 12 months from now, what can our customers expect from our Linux software?
  – Are our Customers happy today? Field Application Engineers, Sales?
The Scope of Mainline

• Mainline applies to all 3 primary components of our Linux solution
  – Uboot  
  – Linux kernel  
  – Filesystem

• Each has its own *mainline* consisting of well documented development flows, release schedules, git trees & maintainers
Defining Mainline

**Mainline** - the ability to provide our customers a **stable release from the community**

“Can I go to kernel.org, pull the latest stable release and expect your device to support basic ‘commodity’ features?”

If this answer is “No”, then you are NOT mainline!
Challenges without Mainline
- Customer Support & New Device Development

Our Linux Solution

Development Window - reps 100% of our time

Time 0  Market Release
Challenges without Mainline
- Customer Support & New Device Development

Our Linux Solution

Device 1  Device 2  Device 3  Device 4  Device 5  New Device

New Device

New Feature 1
New Feature 2

Market Release

Time 0

3 - 4 months later

Market Release
Challenges without Mainline
- Customer Support & New Device Development

Our Linux Solution

Customers are still using these devices
BUT we are not improving the SW or resolving issues

Available Choices:
1. Push Market Release
2. Drop Support of Older Devices

Time 0
Market Release
Operational Efficiency with Mainline

Two Benefits of Mainline:

- **Time to Market** - Full entitlement of New Devices
- **Incremental New Device Development**

Development Window - represents 100% of our time

Customer Support for Existing Devices

Bandwidth

New Device Development

Customer Support

Bandwidth

New Device Development

Customer Support

Bandwidth

New Device

Time 0

Market Release
Bandwidth & Customer Support

Today

Device 1     Device 2     Device 3

Rebase patch set

Mainline Releases

R0   R1   R2   R3   R4   R5   R6
Bandwidth & Customer Support

Today

Device 1

Device 2

Device 3

Rebase patch set

Mainline Releases

R0  R1  R2  R3  R4  R5  R6

Tomorrow
Bandwidth & Customer Support

Today

Device 1
Device 2
Device 3

Rebase patch set

Mainline Releases

Tomorrow
Bandwidth & Customer Support

Today

- Device 1
- Device 2
- Device 3

Rebase patch set

Mainline Releases

- R0
- R1
- R2
- R3
- R4
- R5
- R6

Bandwidth

Tomorrow
Bandwidth & Customer Support

Today

Device 1

Device 2

Device 3

New Device

Rebase patch set

Mainline Releases

R0

R1

R2

R3

R4

R5

R6

Bandwidth

Tomorrow
Bandwidth & Customer Support

Today

Device 1

Device 2

Device 3

New Device

Rebase patch set

Mainline Releases

R0

R1

R2

R3

R4

R5

R6

Bandwidth

Tomorrow
Bandwidth & Customer Support

Today

Rebase patch set

Mainline Releases

Tomorrow
12 months from now...

- True mainline support for our entire Linux solution across platforms
- “Just works” Linux
- Continued support of Existing Devices
- New Device development is incremental
- Easy board port to customer’s platforms
- Bandwidth to focus on more middleware/applications
- Potential to focus on Reference Designs
What Do We Need to Succeed?

Commitment from Management Team

– Realize this is a long term commitment that requires dedicated time & resources

– Allow us the time and dedication required to establish credibility with the mainline community and to generate momentum

– As real progress is made and other key contributors take interest in our mainline effort, let’s make sure the door is open

– Promote our progress and commitment to mainline with customers. It becomes an incredible marketing tool!
Thank you
Introducing: The Next-Gen BeagleBone

An introduction worthy of a black tie affair.

• New color for Spring
• New and improved features
• Bold move to more performance for lower cost

Want a sneak peek and information on advanced ordering options? Make an impression. Register your interest today.
beagleboard.org/unzipped
BACK UP
Developer’s Perspective

- Infrastructure
- Integration Trees
- Testing
- Tracking Merge Windows
- Continuous Integration
Doesn’t Mainline enable our competition?

• Does the assembly or interchangeable parts enable competitive auto manufacturers?
  – Yes

• One of two cases:
  – Our competition is not driving mainline as aggressive as we would like to in which case we can take the lead and use this as a differentiation tool
  – Our competition is already driving mainline meaning they can use it against us and we are forced to adapt to what they are doing

• Mainline = Commodity support

• By supporting mainline we are removing the burden of commodity support from our customers

• This allows our customers to innovate solutions instead of worrying about commodity support
Increasing Quality & Robustness

3.a Dev finishes and 3.b Merge Window Opens

Maintainers push all their patches to Mainline Kernel. Things break.

3.a stable branch

2 Weeks

Linux Mainline Kernel Tree

3.a

3.b

3.c

3.a stable branch created (more on this later)

8-10 Weeks

3.b bug fixes are merged into mainline

The cycle starts again

Community Developers

Maintainer Trees (Representative)
Stable Kernel Releases

Linux Mainline Kernel Tree

3.a 3.b 3.c

Bug Fixes in Kernel Mainline are Ported to Stable Kernel

NOTE: Large changes are usually not ported as they are deemed too risky for a stable kernel

3.a.1 3.a.2 3.a.3 3.a.4 3.a.5

3.a stable branch

3.a.0 Release

2 Weeks

8-10 Weeks

.0 releases usually avoided