Time is ready for the Civil Infrastructure Platform

Agustin Benito Bethencourt, Codethink
Noriaki Fukuyasu, The Linux Foundation
Our Civilization is Run by Linux
Transport
- Rail automation
- Automatic ticket gates
- Vehicle control

Energy
- Power Generation
- Turbine Control

Industry
- Industry automation
- Industrial communication
- CNC control

Others
- Healthcare
- Building automation
- Broadcasting
But there are issues to be solved…
A Power Plants Controller:
20-60 years products life-cycle
with very reluctant nature for product update and upgrade of hardware and base software platform
A Railway System:
25-50 years products life-cycle
with, again, very reluctant nature for product update and upgrade of hardware
and base software platform
We have a problem...
The Problems we face …

• The systems that support our modern civilization needs to survive for a VERY LONG TIME, and currently, the industrial grade super long term maintenance has been done by each individual companies.

• The systems not only have to survive for a long time, it has to be an “INDUSTRIAL GRADE”, that is robust, secure and reliable, and while doing so the industry will also require to catch up with the latest technology trends
The Solutions we need …

- **We need a Collaborative framework** to maintain one same open source based system for many, many, many years to keep it secure, robust and reliable.

- **AND** most importantly, we need to do this collaboratively in the upstream communities, not locally.
Civil infrastructure systems require a super long-term maintained industrial-grade embedded Linux platform for a smart digital future.
Requirements for the Civil infrastructure systems

- **Industrial Grade**
  - Reliability
  - Functional Safety
  - Security
  - Real-time capabilities

- **Sustainability**
  - Product life-cycles of 10 – 60 years

- **Conservative Upgrade/Update Strategy**
  - Firmware updates only if industrial grade is jeopardized
  - Minimize risk of regression
  - Keeping regression test and certification efforts low

---

**This has to be achieve with ...**

**Maintenance costs**
- Low maintenance costs for commonly uses software components
- Low commissioning and update costs

**Development costs**
- Don’t re-invent the wheel

**Development time**
- Shorter development times for more complex systems
3 – 5 years development time

2 – 4 years customer specific extensions

1 year initial safety certifications / authorization

3 – 6 months safety certifications / authorization for follow-up releases (depending on amount of changes)

25 – 50 years lifetime
3 – 5 years development time

0.5 – 4 years customer specific extensions

6 – 8 years supply time

15+ years hardware maintenance after latest shipment

20 – 60 years product lifetime
Things to be done: Creation of “Open Source Base Layer”

- Open source based reference implementation
- Minimal set for the controllers in the industrial grade systems

Non-CIP packages
Any Linux distribution (e.g. Yocto Project, Debian, openSUSE, etc.) may extend/include CIP packages.

CIP Reference Filesystem image with SDK

CIP Kernel

CIP Reference Hardware
What is the CIP initiative doing
CIP key actions

1. Establish an Open Source Linux based system that meets the Industrial Grade requirements.
2. Fill the gap between capabilities of the existing OSS and industrial requirements.
4. Trigger development of an emerging ecosystem including tools and domain specific extensions.
Establish a FOSS Linux based system that meets the Ind. Grade req.

1. Select the first CIP kernel and initial maintainer
   a. 4.4 as first CIP kernel. Maintenance expected for above 10 years (SLTS).
   b. Ben Hutchings as initial CIP-kernel maintainer.
   c. Kernel maintenance policies (WIP).

2. Define initial board platforms and provide support for it.
   a. Beaglebone Black and (RENESAS BOARD) as initial boards.
   b. BB upstream kernel support backported to CIP kernel.
Establish a FOSS Linux based system that meets the Ind. Grade req.

3. CIP kernel testing (WIP)
   a. Adapt kernelci.org project to CIP use case: board @ desk - single developer.
   b. kernelci VM to test kernels on a board connected to the dev. Machine.
   c. Shared tests and logs.
   d. CIP kernel tested on Beaglebone Black.

4. Add LAVA support to Fuego

5. Other
   a. KSPP patches backported to CIP-kernel.
   b. CIP whitepaper (WIP)
Next steps
Next steps by CIP

• Board @desk - Single dev
  • Release kernelci VM and test CIP kernel in the open within CIP group.
  • Increase test coverage.
  • Define milestone 2.

• Finish LAVA support to Fuego.
• Kernel maintenance: define next steps.
• Analysis: select additional software as part of CIP base system.
• Collaboration: kernelci.org, y2038, KSPP, RTL...
Please Join us!
Why join CIP?

• **Steer**: become a decision maker “by doing”.
• **Participate**: bring your use cases and ideas into the right forum.
• **Learn**: by working on daily basis in the open with others with common interest.
• **Collaborate**: share effort and knowledge. Stand on the shoulders of giants.
Contact Information and Resources

To get the latest information, please contact:

• Noriaki Fukuyasu: fukuyasu@linuxfoundation.org
• Urs Gleim: urs.gleim@siemens.com
• Yoshitake Kobayashi: yoshitake.kobayashi@toshiba.co.jp
• Hiroshi Mine: hiroshi.mine.vd@hitachi.com
• Agustín Benito Bethencourt: agustin.benito@codethink.co.uk

Other resources

• CIP Web site: https://www.cip-project.org
• CIP Mailing list: cip-dev@lists.cip-project.org
• CIP Wiki: https://wiki.linuxfoundation.org/civilinfrastructureplatform/
• Collaboration at CIP: http://www.gitlab.com/cip-project
• CIP kernel: git://git.kernel.org/pub/scm/linux/kernel/git/bwh/linux-cip.git
Call for new participants!

Provide a super long-term maintained industrial-grade embedded Linux platform.

Current members

Platinum Members

Silver Members