Use Case of CIP for Power Plant Systems
Self-introduction

My Background & experience

• Embedded software engineer (RTOS)
• Linux controller development lead
• Development Group manager
• Development chief specialist
  - Embedded software, Hardware, FPGA, Web-based software

Introduction of my company & department

• Toshiba Energy Systems & Solutions
  - provides energy-related systems
  - renewable energy, power distribution and VPP
  - hydrogen, nuclear and thermal

• Power Platform Development Department
  - Product development and supply to support power infrastructure worldwide
Purpose of This Presentation

For whom
  → People who are using and will use Linux for high reliable embedded products

How to be
  → Join and be part of CIP

Approach
  → What cases we have experienced and CIP effectiveness
Product Introduction

Four major product groups based on target and cycle

DCS controller

100 ms & more control. It performs boiler control, monitoring throughout plant.

Control module for turbine

3 to 10 ms control. It performs valve control, frequency detection and speed detection exclusively.

Control module for generator

0.5 to 3 ms control. It performs I/O of generator signals and conducts high-speed control.

I/O control module

Module for DI/DO/AI/AO and to control them. It is generally used for monitoring & control in power plant.

RTOS or μITRON

RTOS or FPGA
Product Introduction – Features

Products are equipped with rigorous features for stable supply of electricity.

1. Realization of high availability with redundancy
   - Machine & transmission path are all redundant.
   - Triplex redundancy for important turbine control
   - Continuous control realized by real-time switching

2. Long-term supply & maintenance
   - Over 10 years of monitoring & control with the same product
   - Continuous supply of hardware & software
   - Response to revised/abolished hardware

3. Prompt response to abnormality & accountability
   - Log function to overcome two conflicting problems; analysis is allowed without affecting real-time control.
   - Differentiation between single failure and common cause failure
   - Evidence-based report

4. Secured & safe device
   - Evaluation and measures for vulnerability
   - Patch application with maintaining reliability
   - Application of the latest features such as whitelist
   - Differentiation between failure and attack

Despite these constraints, we decided to apply Linux for more enhanced functionality and speedy developments.
**Product Introduction - history of our controllers**

**Before**

Products based on real-time OS & self-made μITRON

**After**

Linux-based controllers

- **Self-made OS**
- **Self-made RTOS μITRON**
- **Linux rt**
- **CIP rt Linux**

Closed self-reliance management:
Software, drivers and hardware are all internally prepared.

Departure from closed self-reliance management
OSS utilization for other than core portions

- **1990〜**
- **2000〜**
- **2010〜**
- **2020〜**
Why Linux?

- Various open protocol
- Easy to use hardware
- Utilization of stack from 3rd vendors
- Similar operation on PC

Overview of systems we have developed

- Development efficiency and speed improved by utilizing various OSS and 3rd party vendor software; benefits and advantages provided

<table>
<thead>
<tr>
<th>Main Control Application</th>
<th>Communication Control Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Land (Ex. Debian)</td>
<td>OPC-UA IEC 61850 Modbus TCP</td>
</tr>
<tr>
<td></td>
<td>Foundation Fieldbus</td>
</tr>
<tr>
<td></td>
<td>Profibus DP Device Net</td>
</tr>
</tbody>
</table>

However, some issues have emerged
Issues after Linux Application

Kernel correction cases

1. Problems in kernel/driver (all low-frequency)
   a. Kernel panic with e1000 (network) driver
   b. WDT error occurrence and revision of timer handler (as yet unproven*)
   c. PCI I/F hang-up and addition of retry function (as yet unproven*)

* Measures were taken based on assumptions on the problems but their effectiveness has not been confirmed as the problems only occurred on site.

2. Addition of new functions to old kernel
   a. Addition of Precision Time Protocol (PTP) function
   b. Peripheral function update to implement whitelist function
   c. Addition of DMA function to improve UART performance

3. Others
   a. Application of security patches as necessary
   b. Addition of logging & dump to analyze less frequent problems
Issues after Linux Application

Case 1

Oops & kernel panic with e1000 driver

What happened
• Sudden reboot with watch dog timer error

General description
• Kernel panic occurred during continuous energization on site.
• e1000 driver was identified from a back-trace.
• The incident occurred after 3 years of product shipment.
Issues after Linux Application

Driver check
• Easily found a driver patch for the problem

Too much time elapsed after product shipment
• Is this patch simply applicable to the kernel being used for products without problem...?

Patch contents check
• The scale is smaller than expected and there is no problem!

This issue was fixed but what about other patches?

Source:
https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1009545
Issues after Linux Application

Checked the update history of hardware drivers being used

Target hardware

- LAN driver : Approx. 450
- Super I/O* : Approx. 70
- I2C : Approx. 50
  * WDT, LED and UART

Check method

- Step 1: Get all the change log of kernel (upstream).
- Step 2: Get all commit log from changelog related to drivers.
- Step 3: Filter BUG FIX from list of commit logs.
- Evaluation with the help of internal kernel experts

Fortunately there happened to be no item that pose problems but the checking required great effort.
Issues after Linux Application

Case 2

Addition of new functions to existing products

What happened

• Development conducted to add the following functions
  • Precision Time Protocol (PTP) function
  • Whitelist function for security
  • DMA function to improve UART performance
• Hard to support these functions in our existing products

General description

• Backports were no more necessary because development of applying to CIP had just started.
• However this kind of additions might be happened in the future.
• On such an occasion, we are willing to utilize CIP and also contribute to CIP.

CIP will motivate us to perform backporting for useful functions to be added in the future.
CIP Utilization and Contribution to CIP

CIP utilization
From Case 1
• Driver bug fix patch to be utilized
• Kernel processing patch to be utilized
From Case 2
• Motivation to backport new function leading to long-term maintenance
• Cooperation and consultation for backports of new functions possibly

Contribution to CIP
• Backport functions to be shared with CIP
• Information of requests and issues in using products to be shared with CIP
• Merging kernels and drivers that we have corrected into the mainline
Today’s presentation

- Embedded-Linux-applied our products
- Issues after Linux application that we experienced
- Utilization of and contribution to CIP

Hope CIP members be increased & long-term maintenance be shared by everyone in CIP
Thank you