Porting Linux to a Baseboard Management Controller ASIC: Feedback and Perspectives

Luis Luciani, HPE, Distinguished Technologist
Jean-Marie Verdun, HPE, Distinguished Technologist
Where is the BMC and what does it do? Here!
The iLO firmware - a history based on RTOS(s)

- **2002**: Embedded Lights-Out Management Processor
- **2003**: LDAP integration
- **2006**: Integrated remote console
- **2007**: AES encryption
- **2008**: Dynamic power capping
- **2009**: Digital signed/secure update of iLO firmware
- **2010**: AES HW encryption engines
- **2011**: Kerberos authentication
- **2012**: Active health system
- **2013**: SNMP v3
- **2014**: iLO Federation
- **2015**: Full FIPS validation
- **2016**: Full FIPS validation
- **2017 / 2018**: Silicon Root of Trust
- **2018**: Secure Start
- **2018**: Runtime Firmware Verification
- **2018**: Secure Recovery
- **2018**: Server System Restore
- **2018**: Full FIPS Validation
- **2018**: Downgrade Prevention
- **2018**: Security Dashboard
- **2018**: 1-Button Secure Erase
- **2018**: Workload Performance Advisor

Ongoing security cipher and vulnerability updates
## Why enable Linux on GXP?

1. **Control Plane is integrated into a complex infrastructure**  
   Host is no longer a basic BareMetal infrastructure. Proper setup and monitoring in virtualized environment is a must-have feature.

2. **Linux is well known and understood environment**  
   Easy to access to developer resources, wide range of testing capability and drivers availability.

3. **Higher threats, different answers**  
   Security liability vary from end users to end users. Proprietary solution no longer covers the whole spectrum of expectations.

4. **Common software base between vendors**  
   IT infrastructure evolved from dozens of servers to thousands of server within enterprise world inducing multi vendors sourcing and compatibility challenges within the control plane.

5. **BMC hardware can now do it**  
   Faster CPU capabilities with better manufacturing process. Good ARM support.
Leveraging GXP security under Linux

Transfer of Ownership: Who can sign the firmware binary that runs on the hardware?

- HPE’s BMC ASIC (“GXP”) designed to run iLO 5 firmware
- Silicon Root of Trust designed to ensure HPE signed firmware
- No flexible “transfer of ownership” in the existing hardware

Proof of Concept Solution: Chain of Trust

- HPE GXPLOADER Binary- HPE signed and can run on ASIC
- Requires Customer Key Block
  - HPE Signed
  - ASIC locked to a unique ASIC ID value
  - Contains customer public key for U-BOOT validation

[Diagram showing the components of GXP security]

1. Micro-bootblock Silicon Root of Trust validation
2. ASIC ID
3. Custom key block
4. Write protected
5. HPE Signature
6. Customer Public Key
7. U-boot Customer Signature
8. OpenBMC u-boot loader code
9. GXPLOADER code
End user request comes to the PMO
After approval, end user sends HPE their public key

HPE sends back an image for a USB key
End user adds their:
  • iLO credentials
  • Signed OpenBMC image

End user inserts the USB key and powers on
Automatically:
1. Ownership is transferred
2. ROM is installed
3. OpenBMC is installed
4. Light turns blue
GXP BMC programmable interfaces
What are these buses used for?

**PECI (Platform Environment Control Interface)**
- Control Thermal management Reporting
- Single wire bus
- Digital Thermal sensing (ΔT)

**CIF (CPLD InterFace)**
- Proprietary Bus
- Self training, x1 to x8 lanes, up to 266Mhz clock
- Packet Protocol
- Focused on GPIO, FAN status and PWM
CPLD (Complex Programmable Logic Device)

**GPIO**
- x2 64 bits General purpose GPIO tunnelled through CIF
- Mainly Host driven (PGOOD etc…)
- Power control sequencing
- Host status

**Thermal management**
- x8 PWM for fans
  - Programmable duty cycles (256 steps)
  - Thermal protection watchdog timer
    - Fan Fault
# GXP Main address spaces

## Core Registers (GXP internal memory mapped register)

<table>
<thead>
<tr>
<th>Access 8-bit,16-bit, 32-bit ok</th>
<th>Address Range</th>
<th>Start</th>
<th>Ending</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>$000_0000</td>
<td>01F</td>
<td></td>
</tr>
<tr>
<td>Global Display</td>
<td>$000_0100</td>
<td>01F</td>
<td></td>
</tr>
<tr>
<td>SPI</td>
<td>$000_0200</td>
<td>03d</td>
<td></td>
</tr>
<tr>
<td>DVR</td>
<td>$000_0400</td>
<td>06f</td>
<td></td>
</tr>
<tr>
<td>Thumbnail Control</td>
<td>$000_0500</td>
<td>06f</td>
<td></td>
</tr>
<tr>
<td>Fan control</td>
<td>$000_0600</td>
<td>0df</td>
<td></td>
</tr>
<tr>
<td>QEI</td>
<td>$000_1000</td>
<td>11F</td>
<td></td>
</tr>
<tr>
<td>EC engine (x10 with $100 stepping)</td>
<td>$000_1000</td>
<td>3FF</td>
<td></td>
</tr>
<tr>
<td>Primary Ethernet</td>
<td>$000_4000</td>
<td>4FF</td>
<td></td>
</tr>
<tr>
<td>Secondary Ethernet</td>
<td>$000_5000</td>
<td>5FF</td>
<td></td>
</tr>
<tr>
<td>SPI ROM Ext Data #0</td>
<td>$000_6000</td>
<td>6FF</td>
<td></td>
</tr>
<tr>
<td>SPI ROM Ext Data #1</td>
<td>$000_8000</td>
<td>8FF</td>
<td></td>
</tr>
<tr>
<td>USB 3C UTMII PHY Registers</td>
<td>$003_1000</td>
<td>10F</td>
<td></td>
</tr>
<tr>
<td>USB 3C EHCI</td>
<td>$0e00_0000</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>USB 3C OHCI</td>
<td>$0e00_0100</td>
<td>02F</td>
<td></td>
</tr>
<tr>
<td>ARM VIC</td>
<td>$0eff_0000</td>
<td>0FF</td>
<td></td>
</tr>
</tbody>
</table>

## Host Registers

<table>
<thead>
<tr>
<th>Access 8-bit,16-bit, 32-bit ok</th>
<th>Address Range</th>
<th>Start</th>
<th>Ending</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysSupport Configuration</td>
<td>$8000_0000</td>
<td>07F</td>
<td></td>
</tr>
<tr>
<td>SysSupport Configuration</td>
<td>$8000_0000</td>
<td>07F</td>
<td></td>
</tr>
<tr>
<td>SMI Services</td>
<td>$800e_0000</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>MktSupport Configuration</td>
<td>$8027_0000</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>MktSupport</td>
<td>$8020_0000</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>vEHCI PCI Configuration</td>
<td>$804f_0000</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>vEHCI Host Controller Runtime Memory Mapped</td>
<td>$8040_0000</td>
<td>07F</td>
<td></td>
</tr>
<tr>
<td>vDevice Global Registers</td>
<td>$8040_0000</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>vEHCI Device Registers</td>
<td>$8040_1000</td>
<td>1FF</td>
<td></td>
</tr>
<tr>
<td>PTP PCI Configuration</td>
<td>$805f_0000</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>FNS UART Device</td>
<td>$8090_0000</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>SPI Configuration</td>
<td>$809c_0000</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>SPI Memory</td>
<td>$0200_0000</td>
<td>02FF</td>
<td></td>
</tr>
<tr>
<td>System UART A Device</td>
<td>$80fd_0200</td>
<td>02F</td>
<td></td>
</tr>
<tr>
<td>System UART B Device</td>
<td>$80fd_0300</td>
<td>03F</td>
<td></td>
</tr>
<tr>
<td>System UART C Device</td>
<td>$80fd_0800</td>
<td>0FF</td>
<td></td>
</tr>
<tr>
<td>Legacy KCS Device</td>
<td>$80fd_0400</td>
<td>04F</td>
<td></td>
</tr>
</tbody>
</table>

## Expansion device space (CPLD)

<table>
<thead>
<tr>
<th>Access 8-bit,16-bit, 32-bit ok</th>
<th>Address Range</th>
<th>Start</th>
<th>Ending</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIF X-Reg area</td>
<td>$5100_0000</td>
<td>42ff</td>
<td></td>
</tr>
<tr>
<td>CIF Address space</td>
<td>$5100_0000</td>
<td>42ff</td>
<td></td>
</tr>
</tbody>
</table>

---

THE LINUX FOUNDATION

# ossummit
Linux port where do we stand?

- PoC of all the drivers are available
- Ongoing upstreaming process
  - 5.20 window
    - Clock
    - DTS
    - UART
    - Watchdog
- U-boot upstreaming process started
- GXP-Loader is done and published
  - https://github.com/HewlettPackard/gxp-bootblock
From PoC to upstream

• Challenges to:
  — Understand from where to start
    • Yaml documentation of the DTS node
    • DTS styling
    • Initial drivers
    • What is a minimum basic acceptable state for a new SoC into the kernel?
  — Adopt relevant code styling
• But the community has proven to be extremely patient with us, we really appreciated this.
• ARM linux kernel support is good but complex coming from SoC fragmentation and lack for standard
  — Defaulting to a standard defconfig ended to integrate a new ARM Errata (unexpected on an aging Cortex A9)
• BMC are new beast with specifics which require standardization that can happen only through multiple stepping process
• Drivers need access to multiple memory regions which could lead to weird dts entry
What is next?

- **Infrastructure to support host start**
  - Network driver enablement
  - SPI setup
    - SPI Driver to read and copy initial ROM content within vROM
    - CPLD configuration for virtual ROM startup
  - GPIO setup for
    - Basic power button control
    - Power sequencing control
    - Events driven interrupt (Fan / hotplug)
  - Fan controller
  - Host vUART
  - Video encoder
  - Virtual USB hub (Keyboard/Mouse/VirtualNIC)
  - KCS driver
  - CHIF high speed tunneling for RAS logging
Want to help?

https://github.com/hewlettpackard/gxp-linux
https://github.com/hewlettpackard/gxp-uboot

Get in touch with the team:
  gxp-linux@hpe.com
  gxp-u-boot@hpe.com

Want to join our effort and develop?
https://osfci.tech

Any interest for meetups?