# Managing Optics Using Open Standard Software

AKA: The OCP Networking OOM project

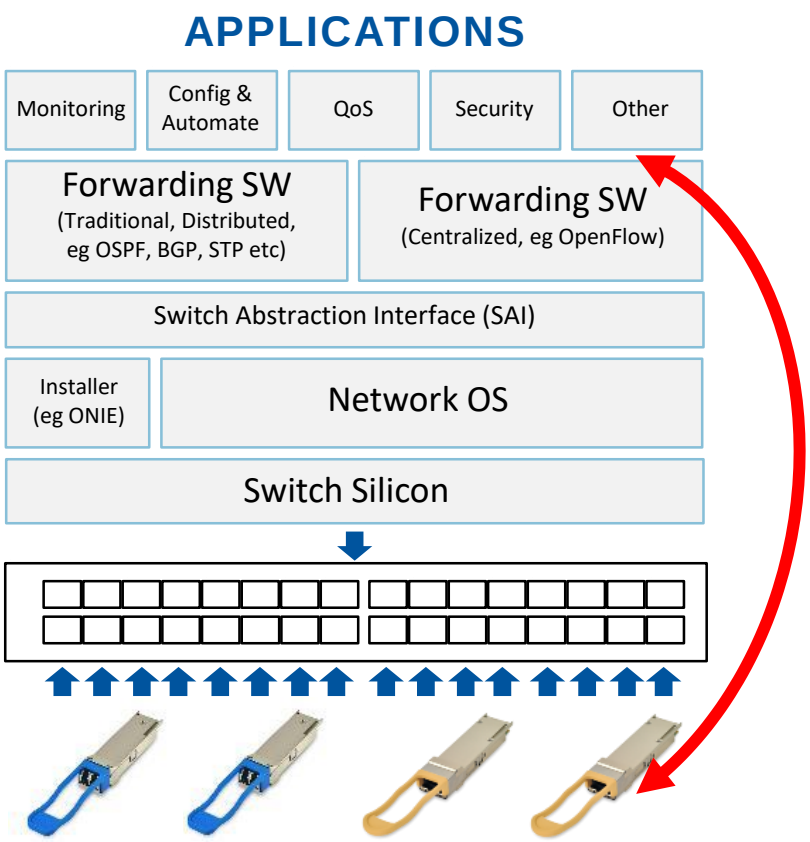

March 21, 2018

**Don Bollinger**

# Topics

- OOM is alive and well
- You need a better optical EEPROM driver
- optoe driver is shipping now

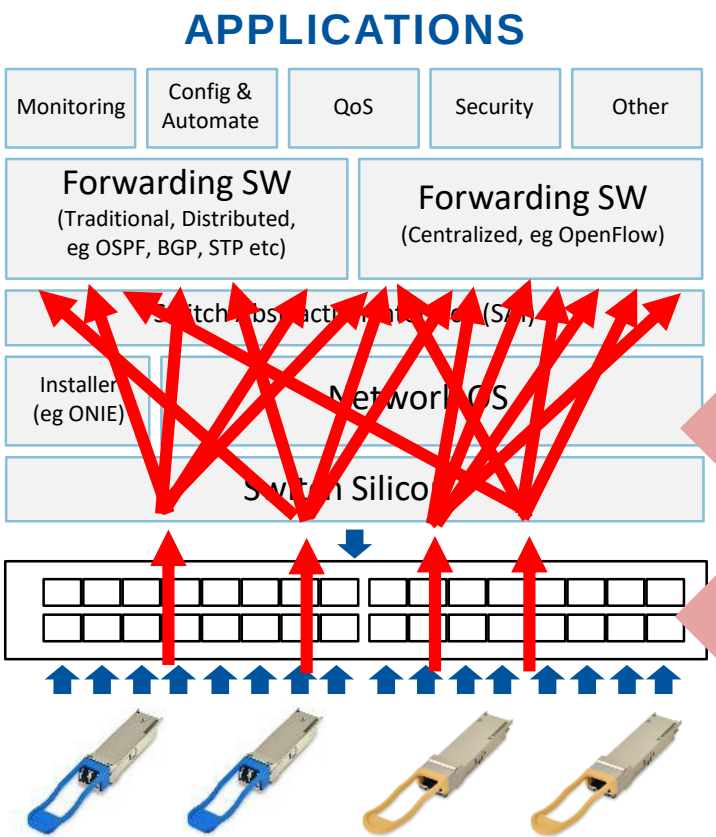
# Traditional Network Vendors Fully Exploit the Optical Data




Switch Vendors built all the plumbing to get optical data to the end user

| OEM Model (Vertically Integrated) |
| --- |
| Ethernet Switch Vendors (Cisco, Juniper, Arista...) |
| Optics vendors |

# Open Networking: The Data is getting lost in the “flexibility”




Management Apps have to be written to a specific switch and NOS combo

Each switch, each NOS has a different way to access optical data

Open Switch, Bare Metal Model

100s of 3rd Party Apps

Applications

Many 3rd Party & Open-source

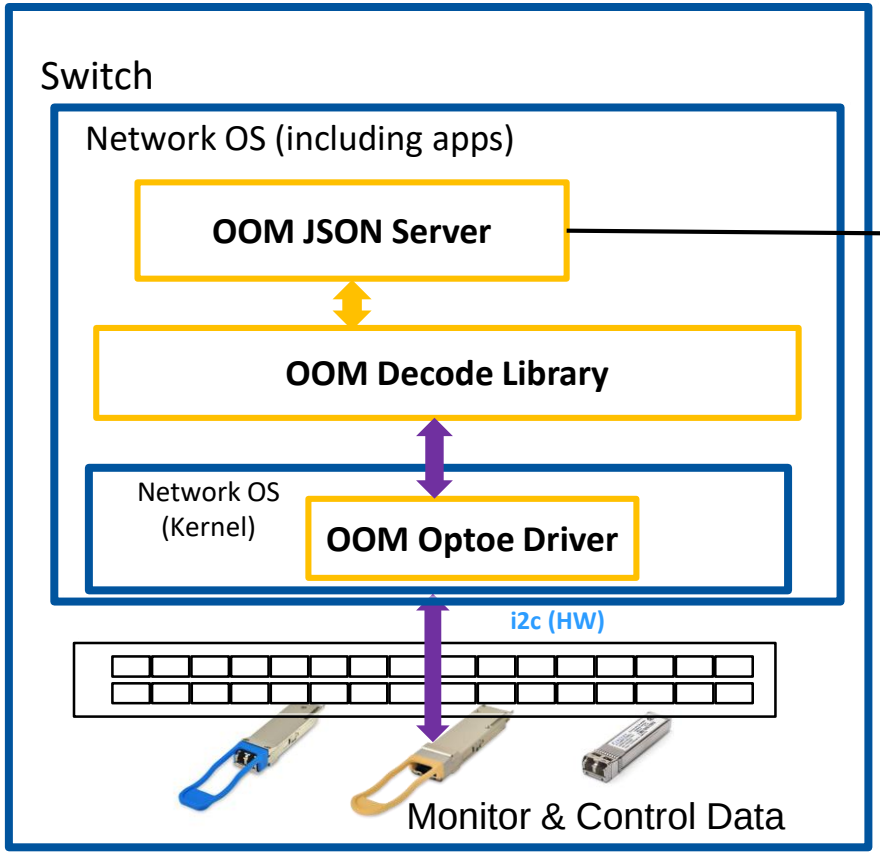
NOS Vendors

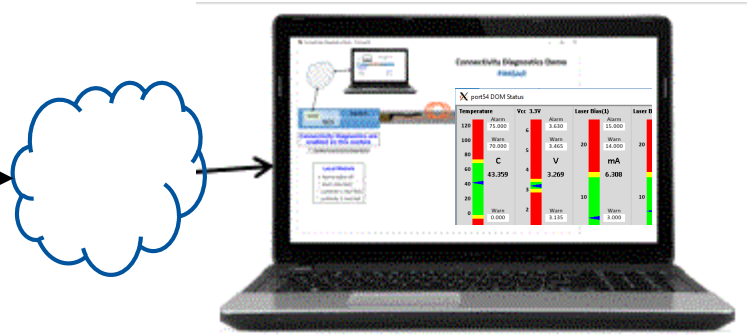
Switch Silicon Vendors

OEM/ODM White Box Vendors

Optics vendors

# Managing Optical Devices in Network Management Tools



***Your Management Tool Here!***

# What is the Open Optical Monitoring (OOM) decode library?

OOM is a Python package, providing a standard API to read/write optical transceiver modules.

- EEPROM data encoded/decoded in key/value pairs.

Same API: Any Linux-based NOS, any switch, any module vendor, any module type.

Open Source, easy to maintain, easy to extend.

```
from oom import *
for port in oom_get_portlist():              # enumerate the ports on the switch
    status = oom_get_memory(port, 'DOM')  # DOM = voltage, temp, {TX, Rx}Power, bias
    print port.port_name + str(status)

port0{'VCC': 3.30, 'TEMP': 23.55, 'TX_POWER': 0.57, 'RX_POWER': 0.56, 'TX_BIAS': 7.4}
port1{'VCC': 3.31, 'TEMP': 24.02, 'TX_POWER': 0.57, 'RX_POWER': 0.53, 'TX_BIAS': 7.3}
```

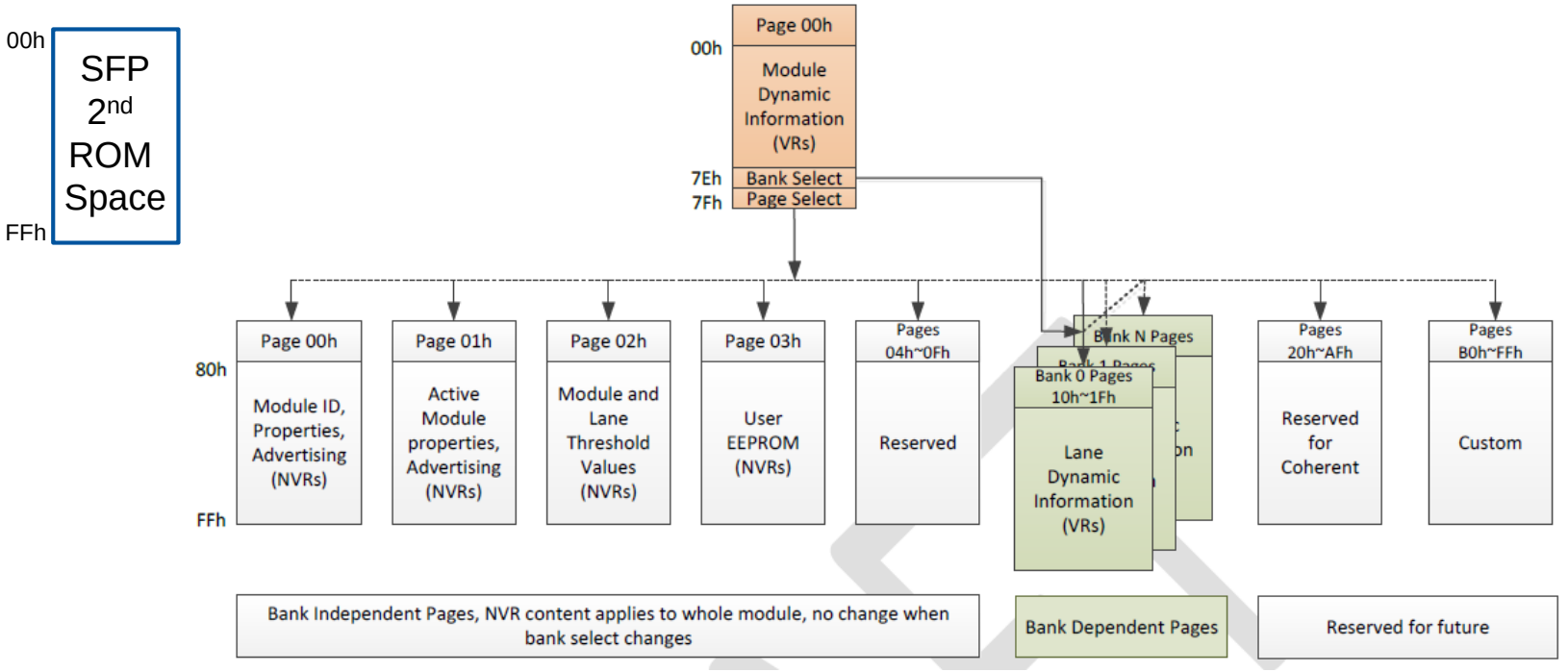
# OOM HAS A PROBLEM!!!

OOM works great…

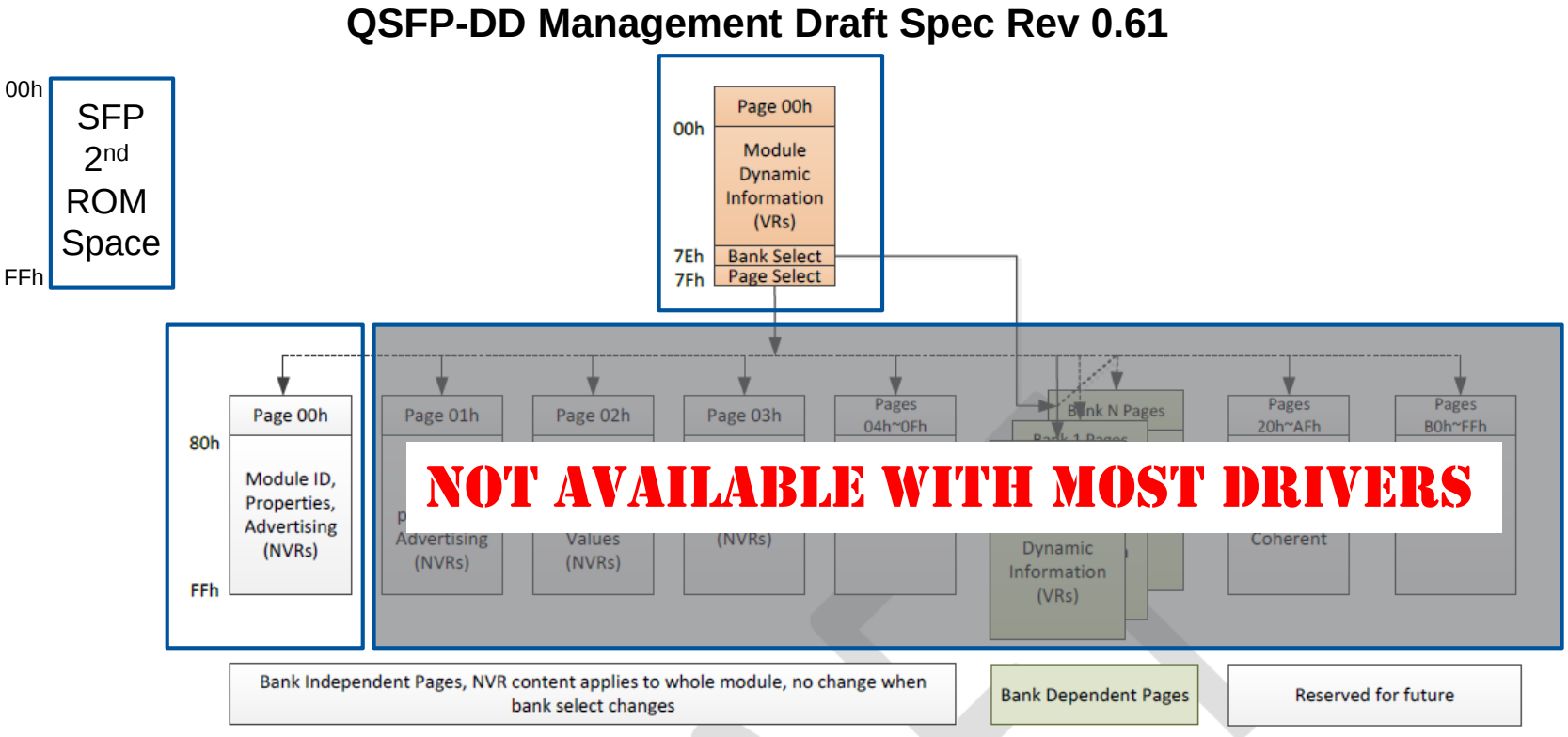
IF IT CAN READ AND WRITE THE EEPROM

# Transceiver EEPROM Layout

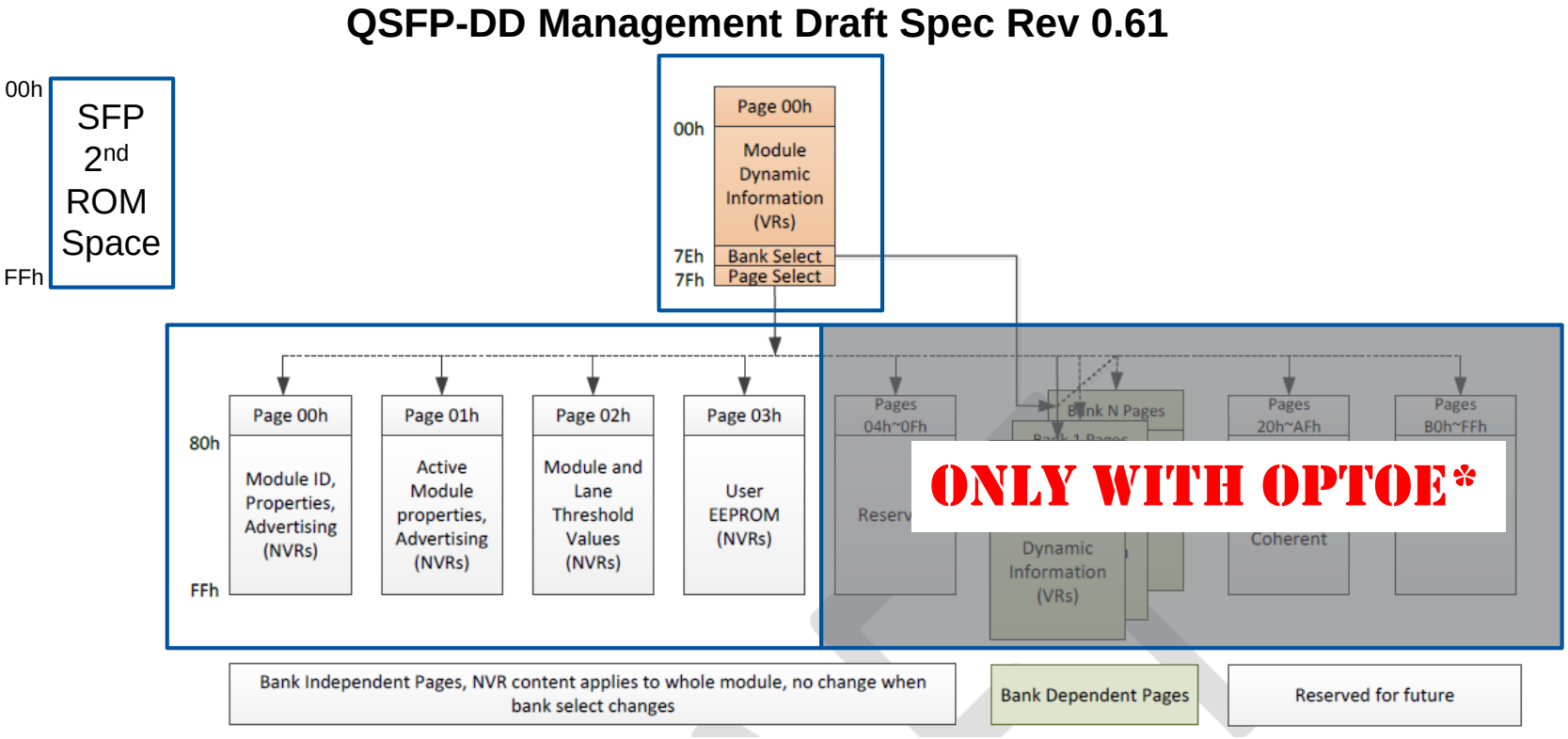
## QSFP-DD Management Draft Spec Rev 0.61



# at24.c – the Linux upstream EEPROM driver



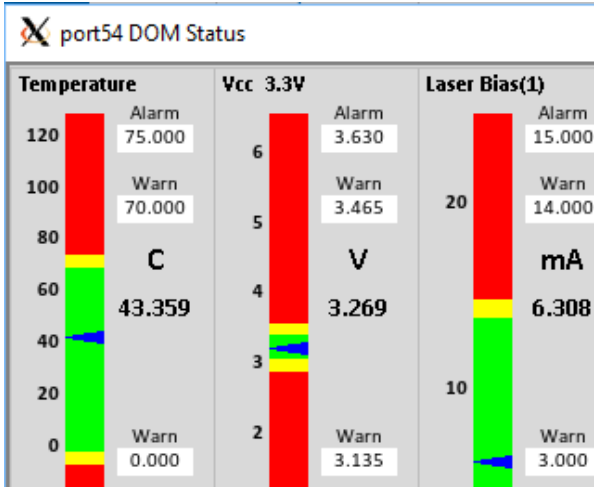
# at24.c – the Linux upstream EEPROM driver




* Bank pages coming when standardized

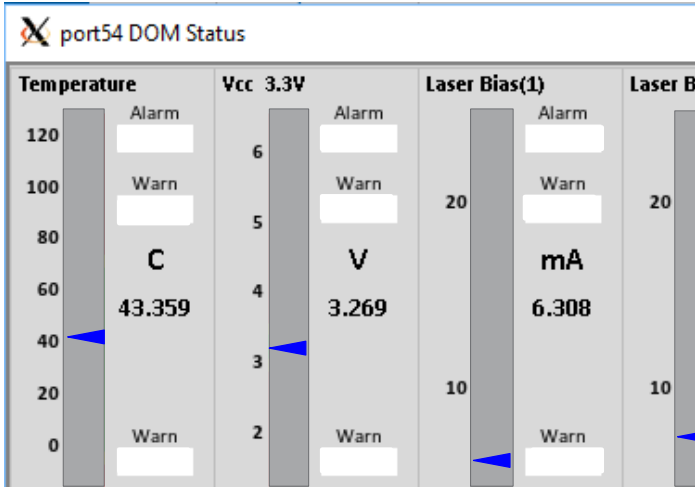
# What optics features require optoe driver?

- QSFP alarm/warning thresholds - page 3
- Some capabilities require write capability
  - Software TX_Disable
- Proprietary features use both
- Future features need to reach page 0x30
- ***QSFP-DD (8 channels) puts most per-channel data in pages 0x10-0x1F***



# Some observations

- at24 driver – no paging, no writes
  - Widely used for SFP access
- sff8436 supports paging
  - Max 256 (or 512) bytes on SFP devices
  - In use for SFP by some NOSs
  - Only 4 QSFP pages supported
- Proprietary drivers – no paging, no writes
  - Used for both SFP and QSFP
- How well does your driver work?
  - Send me a pointer, I'll check it



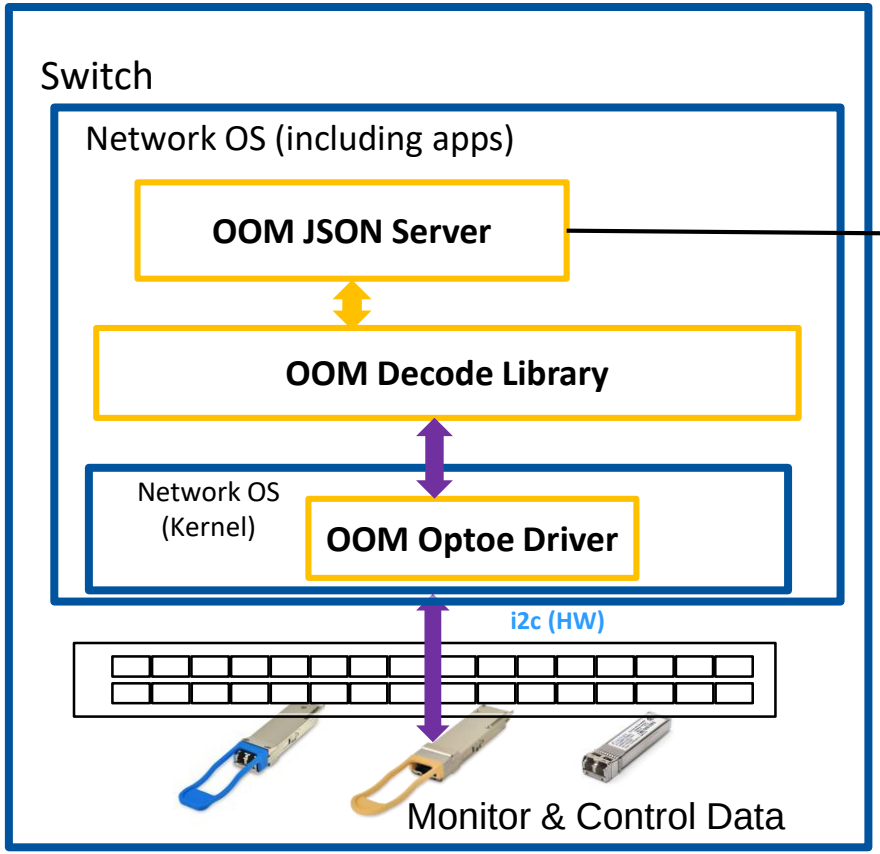
# optoe attributes

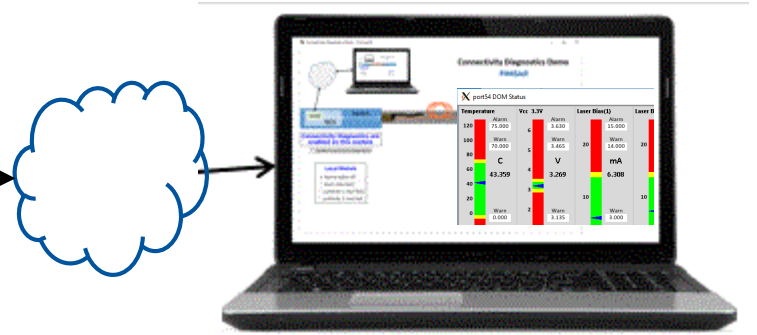
- Technical
  - Read & Write
  - Multiple I2C addresses (for SFP)
  - Multiple Pages, up to the architected limit (256 pages, >32KB)
  - No internal read/write buffer (256 pages: >32K of EEPROM data)

- Logistical
  - All flavors of SFP, QSFP, QSFP-DD/OSFP
  - Platform independent, NOS independent
    - **Available today on *ONL* and *SONIC* for *Accton*, *Inventec* and *Quanta* switches**
  - Next Step: Linux upstream

- Available at: https://github.com/opencomputeproject/oom

# Managing Optical Devices in Network Management Tools



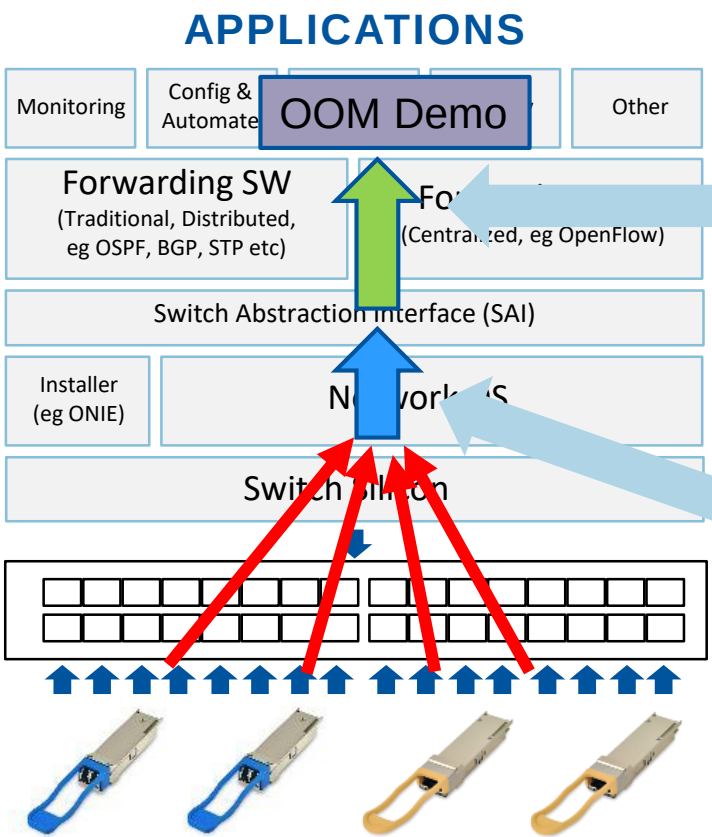



***Your Management Tool Here!***

# Backup

- Old slides, previous versions, noise…

# Finisar is Driving a Common Solution for Open Networking




One API (OOM) to access and decode optical data

One driver to standardize and upgrade access to optical data

**Open Switch, Bare Metal Model**

100s of 3rd Party Apps — Applications

Many 3rd Party & Open-source

- NOS Vendors
- Switch Silicon Vendors
- OEM/ODM White Box Vendors

Optics vendors

# SFF_8436 technical details (vs Sonic 4.9 candidate)

- Supports paging on QSFP, but only 4 pages (optoe: 256 pages)
- Does not page on SFP devices (512 bytes max) (optoe: 256 pages)
- Separate code for read/write and register read/write
  - Same i2c calls implemented twice, harder to maintain (optoe combines them)
- Separate code for SFP and QSFP
  - SFP paging logic should be same as QSFP (after dealing with 2nd I2C addr)
  - optoe combines i2c addr and paging, for SFP and QSFP, into one translate routine
- All reads/writes go through an internal buffer, sized to match the total addressable space of the EEPROM
  - 640 bytes for QSFP, 512 bytes for SFP, times 54 devices (no big deal)
  - More pages means a bigger buffer - >32K for the architected limit
  - 1.7 M of buffer space for 54 devices
  - memset(sff_8436->data, 0xff, ***SFF_8436_EEPROM_SIZE***) – on every read and write!
  - optoe reads/writes directly to the user buffer, faster and smaller

# (more) SFF_8436 technical details (vs Sonic 4.9 candidate)

- Sff_8436 doesn't support sfp_dwdm (type: 0xB), will treat it as QSFP, trying to use just one i2c addr, and page from that addr. (optoe does not query device for type, therefore supports everything)
- Sff_8436 bug: Reads that span i2c addresses (SFP) wrap back to the beginning of the address (optoe fixed this bug)
- Optoe supports QSFP-DD unchanged (needs testing)
- optoe builds on Linux 3.2, 4.1, 4.9, 4.15 (latest staging branch)
- Previous version of optoe submitted by 3 vendors to 2 NOSs
- optoe passes checkpatch.pl
- optoe is ready to submit to Linux upstream kernel
- sff_8436 uses 'sfp_compat', optoe uses 'dev_class', works the same

# Requirements for an Optical EEPROM Driver

- MUST
  - Read & Write
  - Multiple I2C addresses (for SFP)
  - Multiple Pages, up to the architected limit (256 pages, >32KB)

- High Priority
  - No internal read/write buffer (256 pages: >32K of EEPROM data)
  - One driver for all flavors of both SFP and QSFP
  - One driver for all platforms

*Note, we have a driver that meets these requirements*
https://github.com/opencomputeproject/oom

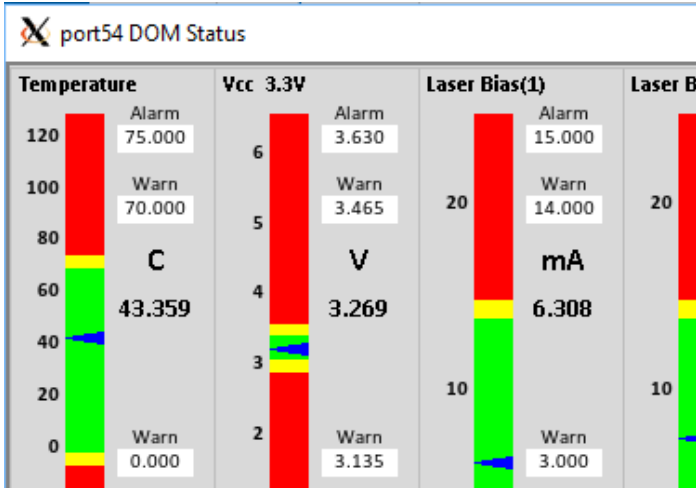
# Optoe details

- Exposes the EEPROM data in a bin_attribute file ‘eeprom’
  - Size is 32K+ bytes
- Maintains a port name for each device, in an attribute file ‘port_name’
  - Initially ‘unitialized’, write the desired name to set it
- Supports two device identifiers
  - *optoe1* for devices with one i2c address (QSFP). (*sff8436* also works)
  - *optoe2* for devices with two i2c addresses (SFP). (*24c04* also works)
- Build as a module, add via new_device, or via other i2c mechanisms

# optoe driver

- Supports both SFP and QSFP
- 256 pages supported on both
- Read and Write
- Available on OOM github site:

  https://github.com/opencomputeproject/oom
- Tested on Sonic, Cumulus, ONL
- Tested on Accton, Inventec switches



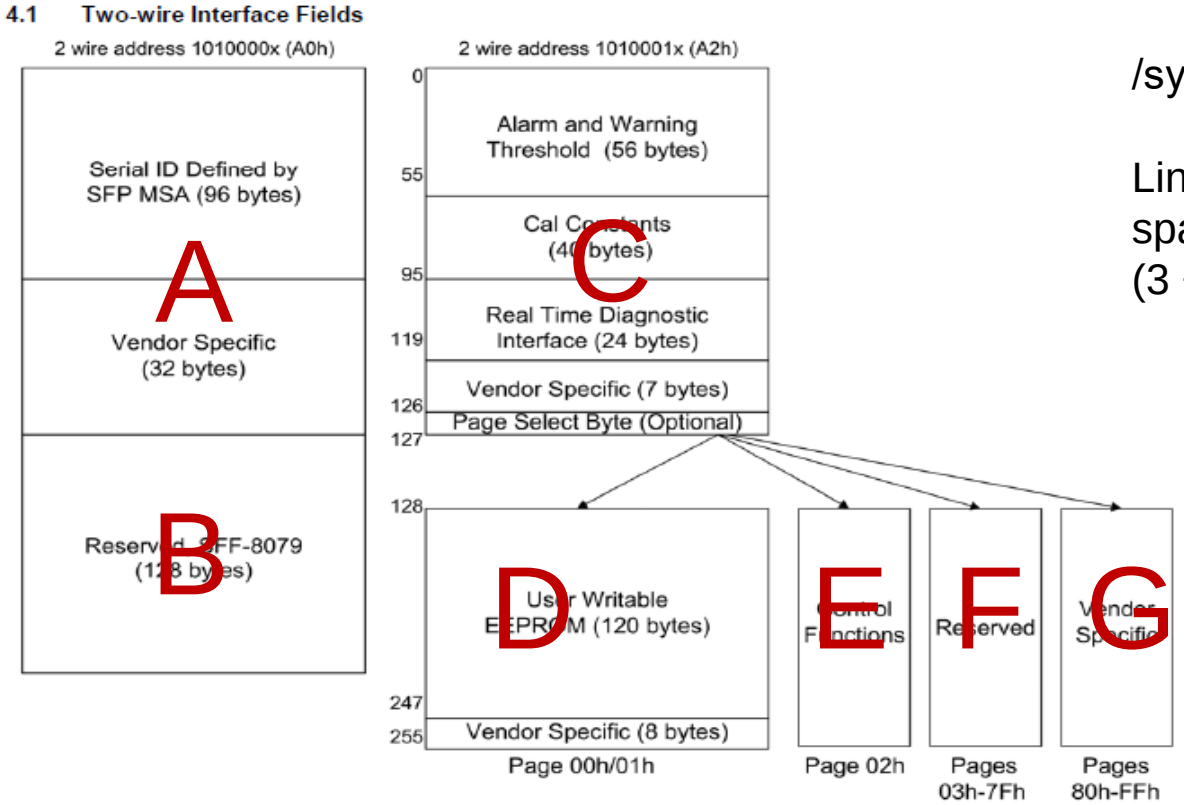
# Example setup

- QSFP+
  - echo optoe1 0x50 > /sys/bus/i2c/devices/i2c-64/new_device
  - echo port54 > /sys/bus/i2c/devices/i2c-64/port_name
  - more /sys/bus/i2c/devices/i2c-64/port_name
    - *port54*
  - od –c –j148 –N16 /sys/bus/i2c/devices/i2c-64/eeprom
    - *0000224 F I N I S A R C O R P*
- SFP
  - echo optoe2 0x50 > /sys/bus/i2c/devices/i2c-11/new_device
  - echo port1 > /sys/bus/i2c/devices/i2c-11/port_name
  - more /sys/bus/i2c/devices/i2c-64/port_name
    - *port1*
  - od –c –j20 –N16 /sys/bus/i2c/devices/i2c-11/eeprom
    - *0000224 F I N I S A R C O R P*

# Other design decisions taken

- sysfs
  - Only eeprom and port_name are supported
  - Other attributes are machine dependent, belong in CPLD driver
    - Present, TX_Fault, RX_LOS, TX_Disable…
- Interrupts on device remove/insert – not supported
- Presence not tested, just return error (ENXIO) if no device present
- ‘Page support’ register tested if accessing beyond page 0
- I2c accesses (read/write) are copied from at24, sff8436 drivers
- Currently available on OCP/OOM github site
  - Will propose for Linux mainstream when we have some adoption

# SFP Memory Layout mapped to /sys/…/eeprom




/sys/…/eeprom:

Linear address space from 0 to (3 + 128) * 128

A
B
C
D
E
F
G

# QSFP EEPROM mapped to /sys/…/eeprom

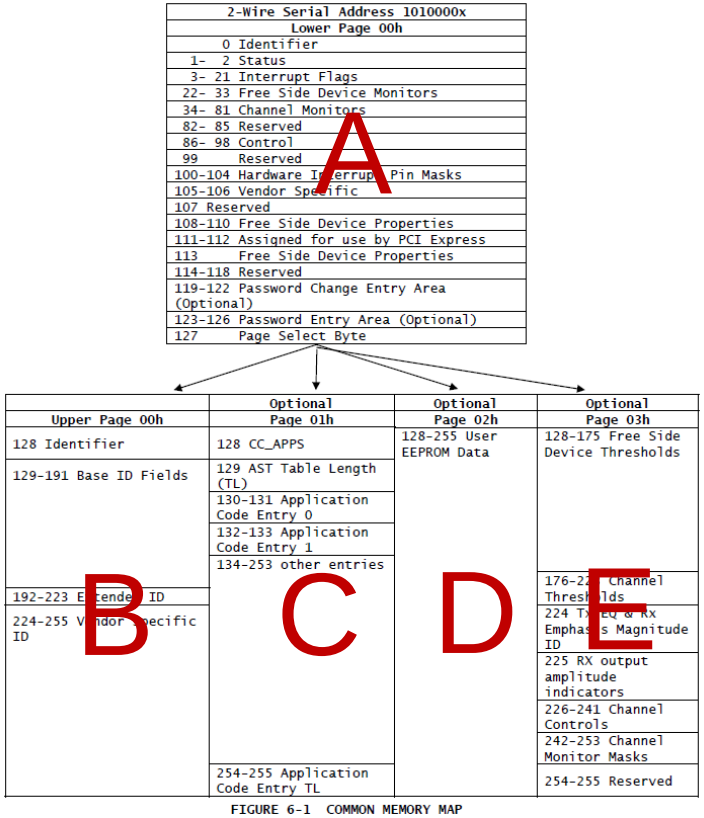



FIGURE 6-1 COMMON MEMORY MAP

F

‘F’ here is architected, but not part of the standard

/sys/…/eeprom:

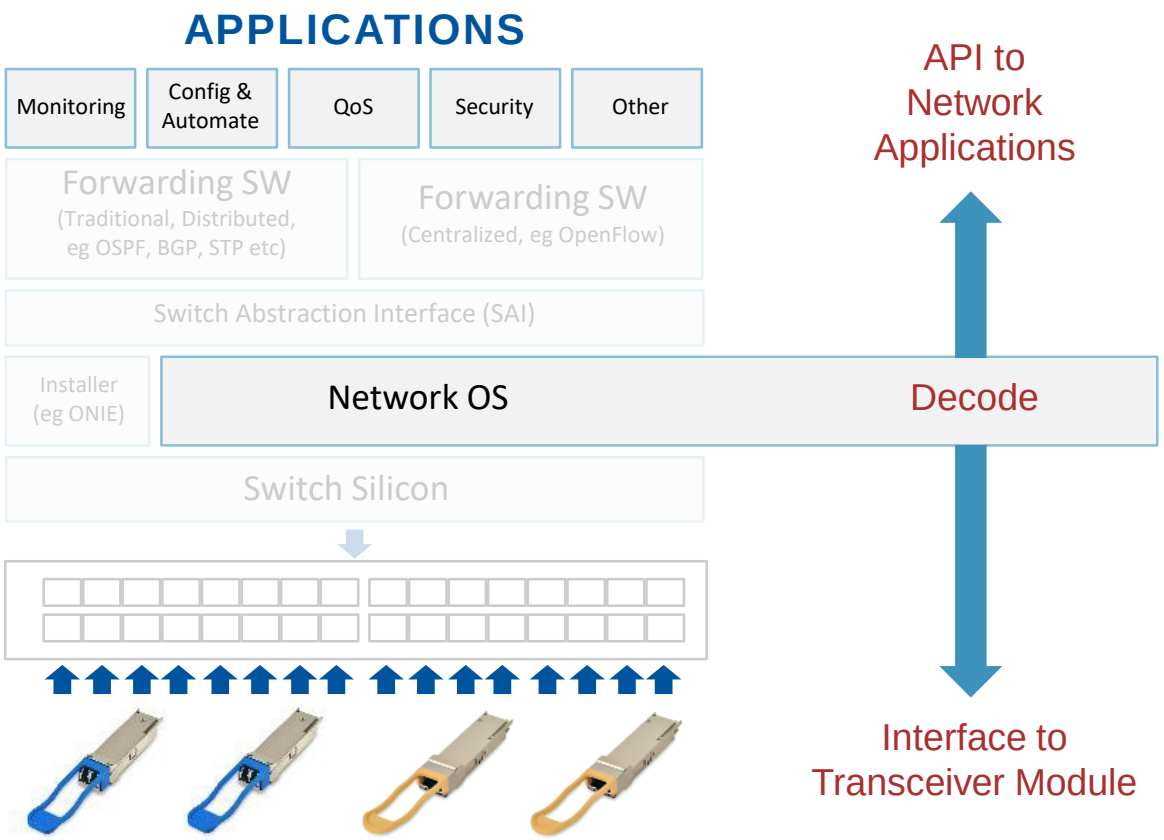
Linear address space from 0 to (1 + 128) * 128

A
B
C
D
E
F

# Feedback from August OCP Networking Workshop (todo list!)

- Explore getting OOM into Redfish, OOM Networks profile for Redfish?
- Some choices for protocols to expose the data
  - Collectd, grafana…
  - Redfish
  - GRPC
- Make a publicly available (github) demo – flashy, GUI, etc
- Dustin suggests put it into the Linux upstream
- Config is either the platform driver or an ACPI table, or the device tree (Dustin)
- ~10 folks would be willing to help out in bi-weekly conference calls to develop/deploy the driver
- Collect more use cases – what specific uses are being made of this optics data. Agenda item for the ongoing process.
  - Link budget - dbm sent, received, fiber loss, dust, … MUCH easier with T2DOC to ask just one side for the data from both
  - Inventory, including statistics of new device, to compare with current values
  - Energy consumption for the switch, turn off laser on unused links
  - Sysfs attribute – put the device in low power mode

# Open Optics Monitoring and Control (OOM)




**OOM** was kicked off by the OCP Networking group in October 2015… To address problems with consistent access to EEPROM information on optical transceivers during OCP Interop testing.

Sponsors:

**Accton/Edgecore**

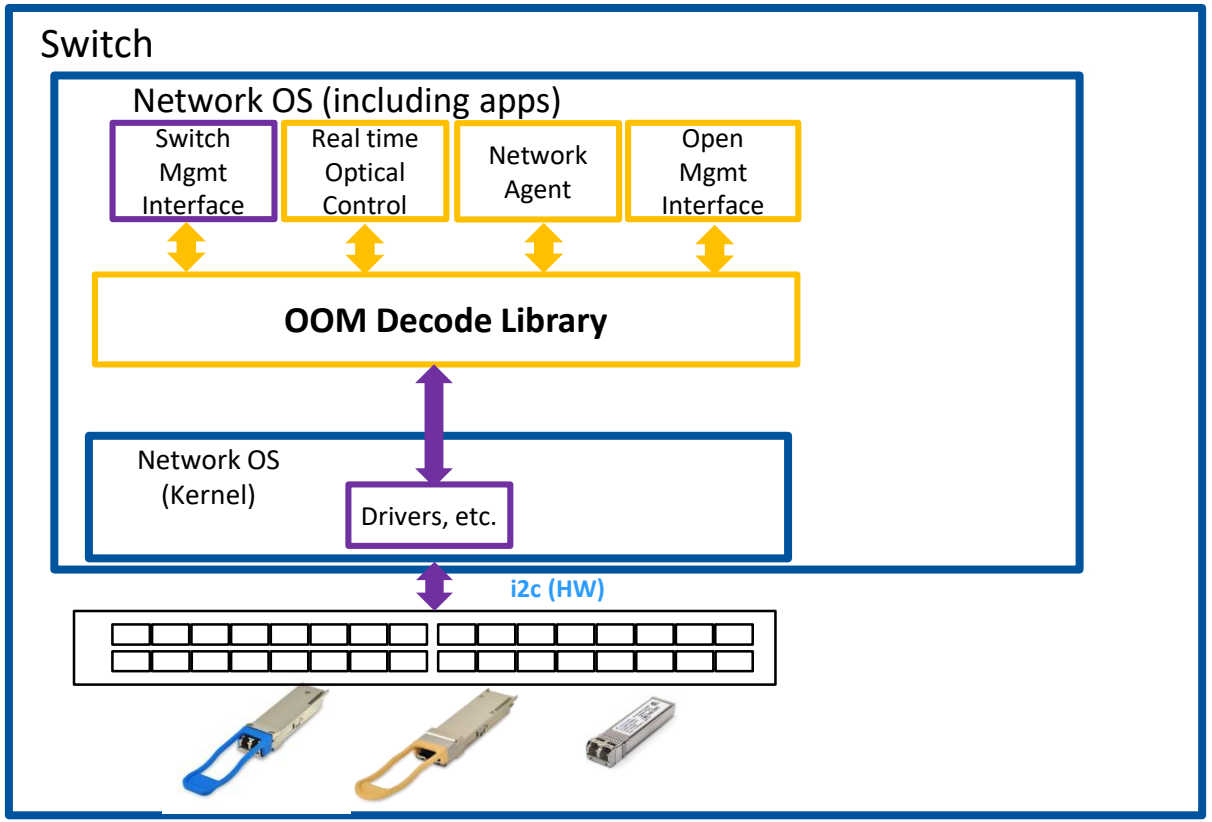
**Big Switch Networks**

**Broadcom**

**Cumulus Networks**

**Finisar**

# Simplified OOM Architecture




Open Source

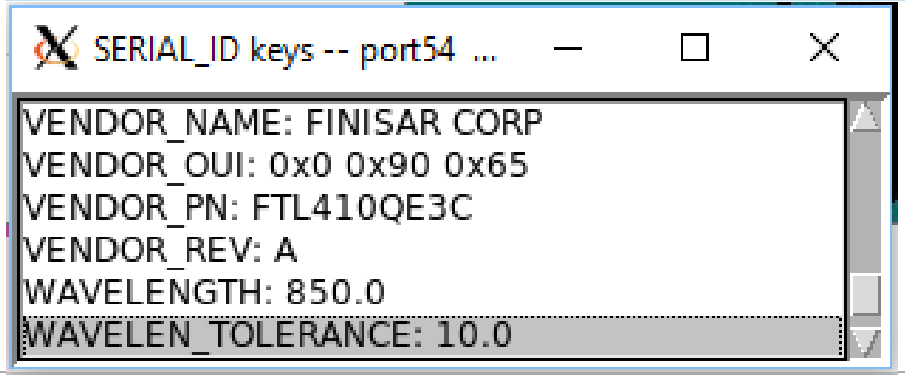
Vendor Specific

# Inventory with OOM

Record identifying info from all modules on the switch

Confirm intended vs actual parts

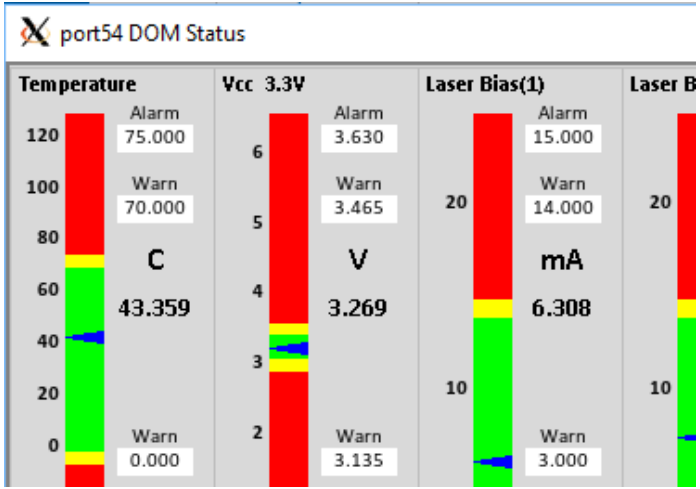
```
from oom import *
for port in oom_get_portlist():                    # enumerate the ports on the switch
    inventory  = oom_get_memory(port, 'SERIAL_ID')  # SERIAL_ID: 23 identifying keys
    add_record(port, inventory)                    # add this module to the database
    audit_record(port, inventory)                  # check for compliance
```




# Health Monitoring with OOM

## Monitor and display key health metrics

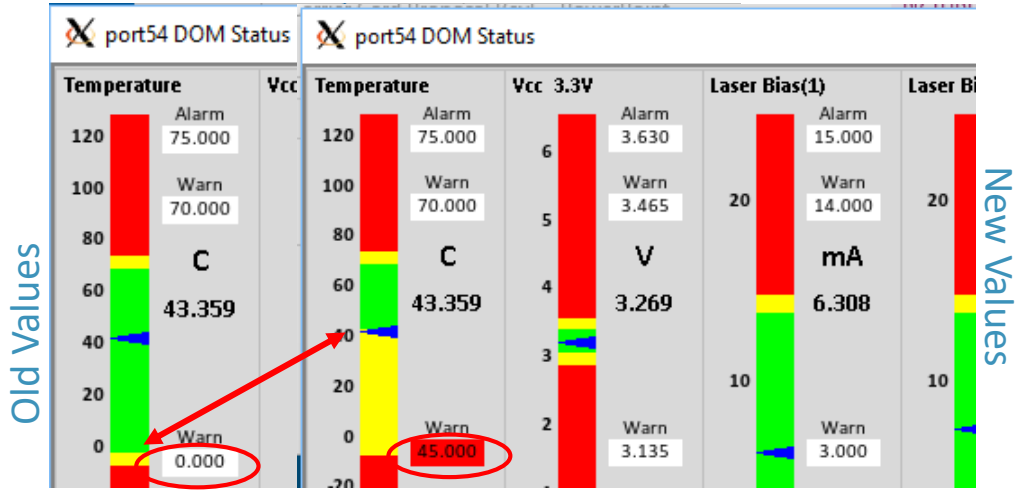
```
from oom import *
list = oom_get_portlist():                          # enumerate the ports on the switch
    health  = oom_get_memory(list[53], 'DOM')  # DOM: Digital Optical Monitoring
    show_port(list[53], health)                     # Display temp, voltage, laser, Rx/Tx power
```




# Diagnostics and Support with OOM

Vendor rep adjusts low_temp warning threshold to test alert handling

```
from oom import *
list = oom_get_portlist():                          # enumerate the ports on the switch
    oom_set_keyvalue(list[53], 'PASSWORD_ENTRY', secret)  # Vendor support password
    oom_set_keyvalue(list[53], 'TEMP_HIGH_ALARM', 45.0)  # Change threshold
```



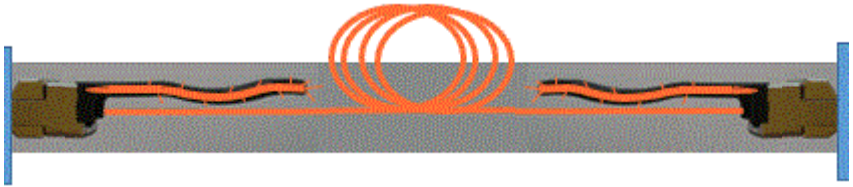

# Custom Uses - Vendor Value Added Content with OOM

Trigger flashing pull tab lights with OOM

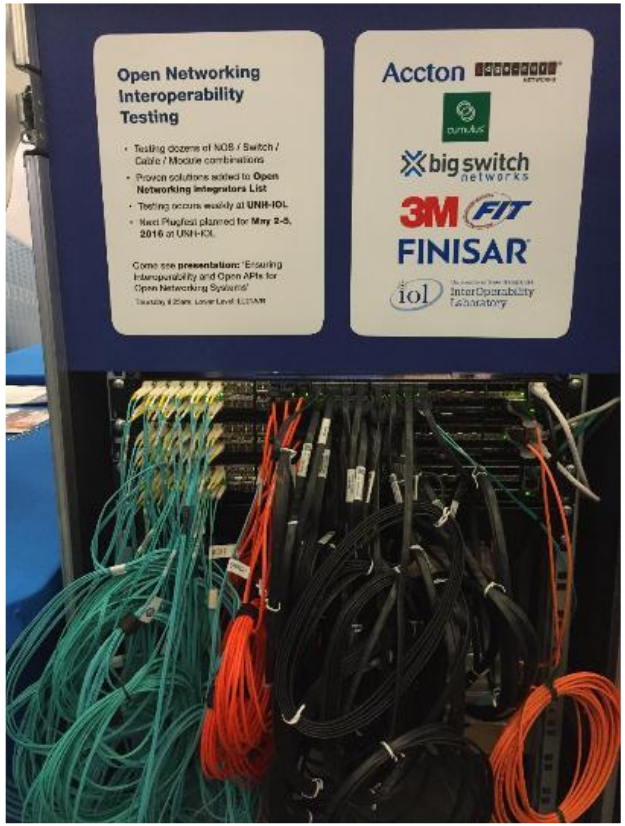
Easy documented process to add additional keys to OOM

In the switch, live, ***running normal production workloads***

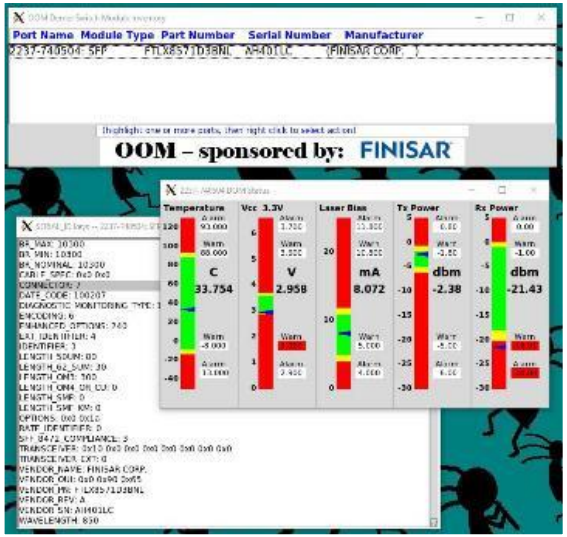
```
from oom import *
list = oom_get_portlist():                          # enumerate the ports on the switch
    oom_set_keyvalue(list[53], 'TAB_LIGHTS', flash)  # Make the lights flash
```



# OCP SUMMIT: Interoperability and Open APIs








**OOM demonstrated at**
**OCP Summit 2016, 2017, 2018**

# How can you Access & Participate in OOM?

- OOM is now an **OCP Accepted™** Project
- Download, use and improve!
- https://github.com/opencomputeproject/oom
- https://youtu.be/kkL2dk7zMOc
- Share your use-cases with us.
- Used in Interoperability testing at UNH IOL Plugfests.
- Demonstrated in numerous Linux-based NOSs, white box switches, evaluation boards and a module simulator.
- 200+ keys decoded for QSFP+, QSFP28, SFP+… CFPx limited keys available.

# Latest OOM News

- February 2017 – ‘Universal Python Shim’.
  - No longer need to compile C code to install OOM.
  - Extensible to support any (every) Linux-based NOS.
- July 2017 – Introduced CFP family support in OOM.
- October 2017 – ‘optoe’ driver released for transceiver EEPROM.
  - For any Linux-based NOS.
  - Accesses more transceiver EEPROM capabilities than existing drivers.
- January 2018 – ‘optoe’ driver is in Open Network Linux (ONL) for 5 Accton switches and one Quanta switch. We expect additional switches and NOS vendors soon.

# Latest OOM News

- May 2016 – OOM installs as a standard Python Package
- August 2016 – OOM Web Service using JSON available
- September 2016 – ‘Universal Shim’ developed (Cumulus + ONL)
- February 2017 – ‘Universal Python Shim’
  - No longer need to compile C code to install OOM
  - Extensible to support any (every) Linux-based NOS
- August 2017 – Reference Linux Kernel Driver available on github