Simulation Driver Driver Develpment

Michael Davidsaver
Osprey DCS

How, When, … Why?

- Test driven development is great!
- How to do this with hardware drivers?
- Cost/benefit?

Case studies

- HPI 6012/6016 radiation monitor
  - Tx only serial/ethernet
- FEED *(FPGA Embedded Ethernet Driver)*
  - UDP request/response
- CaenELS PICO8
  - uTCA
- MRF EVR/EVG
  - VME

HPI 6012/6016

- Counter for use with various detector chambers
- Sends one line of text at 1Hz
- Simple right?

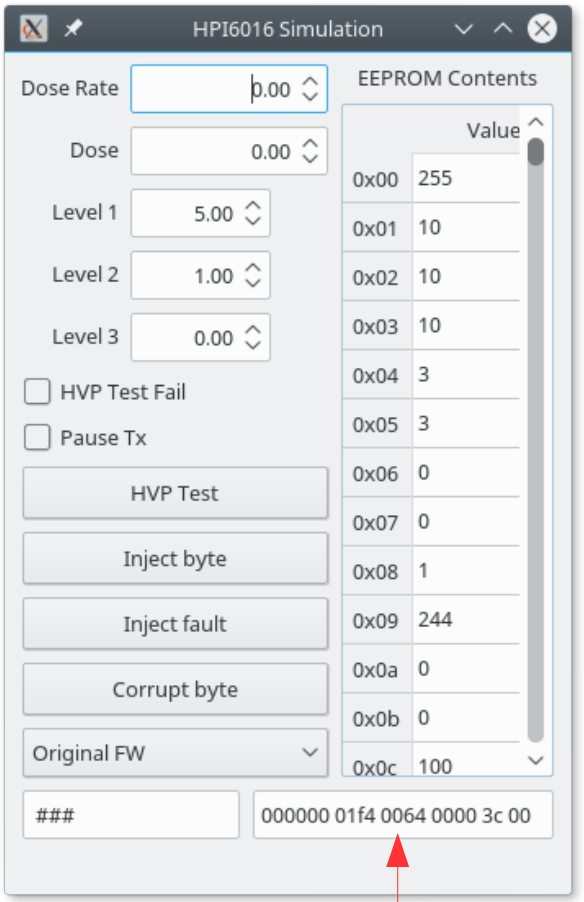
Osprey DCS

HPI 6012/6016 (2)

- Safety “related” device
- Closely scrutinized
  - NSLS2 beam mis-steering incident
- Develop simulator and driver concurrently
- Cross-test driver with simulator and real HW
  - Can test “impossible” faults
  - Part of (re)validation after driver changes

HPI 6012/6016 (3)

- Simulator
  - TCP server
  - GUI interface
- Simulate
  - Dose
  - Device alarms
  - Comm. faults



Output text

Osprey DCS

HPI 6012/6016 (4)

Testing process

==========

Initial testing uses the hpisim3.py software device simulation.

Repeat for both Original FW and Integrated Dose modes

...

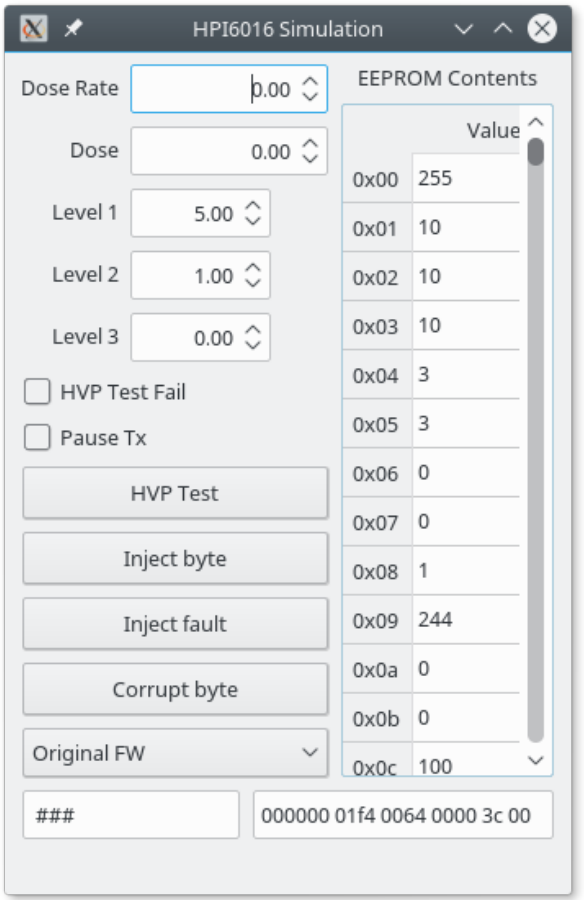
Return the dose rate to 0

Verify that alarms clear

Reduce simulation dose rate to -0.05

Verify that the fail holdoff counter begins counting down

Verify that $(P)Alrm:Fail-Sts becomes active when the counter reaches zero.



Osprey DCS

FEED *(FPGA Embedded Ethernet Driver)*

- UDP/IP request → response
- Register based
- Introspect device to get register name ↔ address mapping.
  - Compressed JSON encoded in ROM
- Develop simulator and driver concurrently
- Cross-test driver with simulator and real HW
  - Help identify FW (doc) issues
  - Test (re)connection and timeout behavior
  - Limited access to shared test stand

Osprey DCS

FEED Packet Format

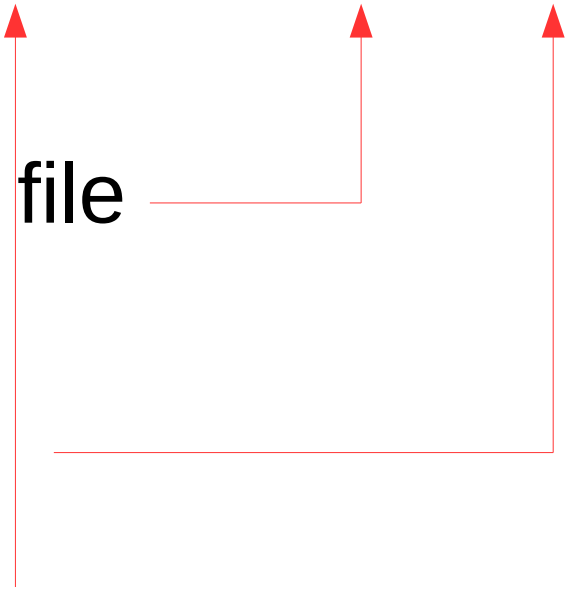
- 8 byte header echoed
- 26 bit address space
- Only 4 byte access
- Read and Write formats identical

	0x0	0x1	0x2	0x3
0x00	Header/Tag			
0x04				
0x08	Bits	Address 1		
0x0C	Data 1			
0x10	Bits	Address 2		
0x14	Data 2			
	...			

Osprey DCS

FEED simulator

```
./bin/linux-x86_64/feedsim -h
Usage: ./bin/linux-x86_64/feedsim [-hd] [-H <iface>[:<port>]] [-L none|rfs] [-S <sec>] <json_file> [initials_file]
```



- Load register description from file
- Optionally
  - Provide initial register values
  - Firmware specific logic

    eg. waveform arm/wait/readout sequence

Osprey DCS

CaenELS PICO8

- uTCA fast digitizer (pico ammeter)
  - PCIe w/ interrupts and DMA
- Develop simulator and driver(s) concurrently
  - Update Linux kernel module
  - Create EPICS driver
- FW bug in DMA handling
  - Developed driver (mostly) in advance of working HW

Osprey DCS

QEMU Device Models

- QEMU emulator provides callbacks on MMIO
  - static uint64_t sis_read(void *opaque, hwaddr addr, unsigned size) { ...
  - static void sis_write(void *opaque, hwaddr addr, uint64_t val, unsigned size) { ...
  - Lots of helpers for PCI device mechanics
- “Ultimate” bus analyzer

Everything except (most) timing!

```
$ qemu-system-x86_64 -device amc-pico-8,?
...
amc-pico-8.addr=int32 (Slot and optional function number, example: 06.0 or 06)
amc-pico-8.frib=bool
...
$ qemu-system-x86_64 -device amc-pico-8 ...
```

Models interrupts and Scatter DMA

Language	files	blank	comment	code
C	1	108	71	549

https://github.com/mdavidsaver/qemu/blob/vme/hw/misc/caen_pico8.c

MRF EVR/EVG

- Event timing cards
  - Various PCI/PCIe and VME
- Develop simulator with existing driver(s)
- Validate driver changes w/o access to HW

MVME3100 with QEMU

- Modeling MVME3100
  - e500v2 CPU already supported
  - Start from existing mpc8544ds
  - Add I^2C controller, eeprom, and RTC
  - Bootloader w/ RTEMS
- Modeling TSI148 PCI ↔ VME bridge
  - Really complicated device!
  - VME bus infrastructure

Osprey DCS

How, When, Why

- How?
  - Sockets (Serial) is easy
  - MMIO w/ emulator
- When?
  - Need to exercise handling of uncommon errors
  - Lack access to (working) hardware
  - Low-level tracing
  - More predictable, but longer, development time
  - $$$
- Why?
  -

How, When, Why

- How?
  - Sockets (Serial) is easy
  - MMIO w/ emulator
- When?
  - Need to exercise handling of uncommon errors
  - Lack access to (working) hardware
  - Low-level tracing
  - More predictable, but longer, development time
  - $$$
- Why?
  - Simulator is my bubble!

Osprey DCS