A Domain-Specific Language for Device Drivers

Christopher Conway

16 December 2002

Project Goals

A language for device drivers that is:

- Robust
- Simple
- Platform-independent
- Useful

Hardware interface

Defined using *ports*, *registers* and *variables*.

```
device NS8390 {
  port registers = 0..0x0f ;
  port dataport = 0x10..0x17, littleendian ;
  port reset = 0x18..0x1f ;

  register CommandReg = registers(0),
          read mask '......**' : bit[8];

  variable registerPage = CommandReg[7..6] : int{0..2};

  ...
}
```

Device operations

In C, register writes are a jumble of bit operations:

```
outb(E8390_NODMA+E8390_PAGE0+E8390_START,
        nic_base+ NE_CMD);
outb(count & 0xff, nic_base + EN0_RCNTLO);
outb(count >> 8, nic_base + EN0_RCNTHI);
```

The equivalent NDL code:

```
command = { nicState=START, remoteDmaState=DISABLED } ;
remoteDmaByteCount = count ;
```

Operating System Interface

Device operations are grouped into device functions. Functions expose an external interface defined by a use protocol:

```
protocol {
  NetworkDevice :
      init ( start DevFunc* stop )*
      ;
  DevFunc :
      set_multicast_list
      | start_transmit
        ...
      | interrupt
      ;
}
```

Synchronization

Three levels of protection:

```
critical {
  /* simple mutual exclusion */
}

critical(irq) {
  /* mutex + disables the device's IRQ */
}

critical(ALL_IRQ) {
  /* mutex + disables all processor IRQs */
}
```

Interrupt Handlers

Interrupt handling routines are tagged with the conditions under which the should run. A compiler-generated top-level interrupt function evaluates the conditions and dispatches control.

```
critical function receive()
@( interruptStatus.packetRxIrq
   || interruptStatus.rxErrorIrq ) {
    ...
    interruptStatus = { packetRxIrq=ACKNOWLEDGED,
                        rxErrorIrq=ACKNOWLEDGED } ;
}
```

Device Identification

The operating system needs a way of associating a physical device with a device driver.

```
identification {
  REALTEK { name="RealTek RTL-8029",
            id=0x802910ec },
  HOLTEK32 { name="Holtek HT80232",
             id=0x005812c3,
             ioBits=16 },
  HOLTEK29 { name="Holtek HT80229",
             id=0x559812c3,
             ioBits=32 }
  ...
}
```

Conclusion

NDL demonstrates an advance in clarity and expressiveness. A lines-of-code comparison between C and NDL:

	C	NDL
8390	1000	684
NE2000	507	142
Total	1507	826

Future Work

- Build a compiler.
- Incorporate static verification.
- Test semantics on a broader class of drivers.