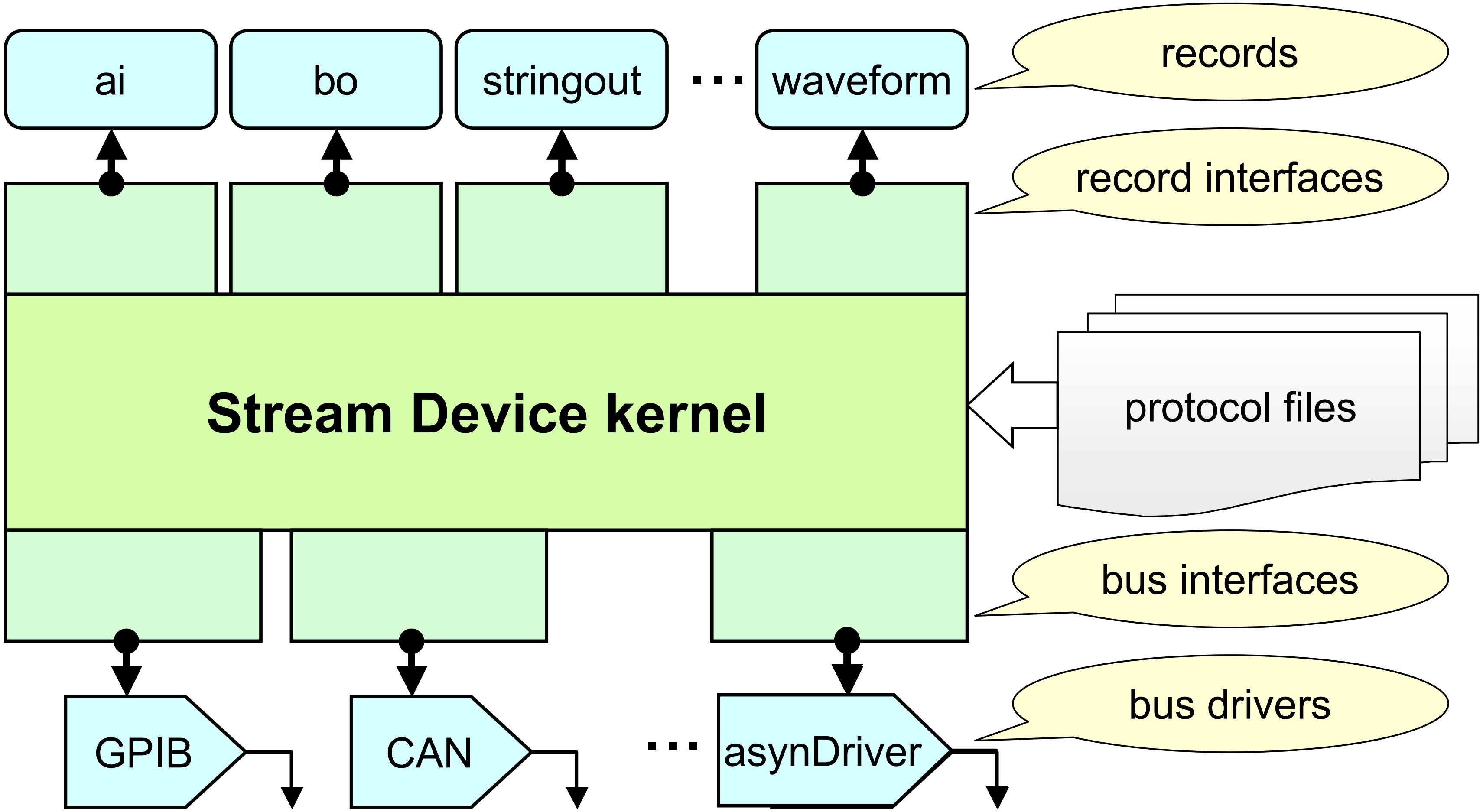
PAUL SCHERRER INSTITUT

Stream Device 2.x

A device driver for byte stream oriented I/O

Dirk Zimoch, EPICS Meeting 2004, Japan

What was Stream Device again?



Proven Features

- Plain text config file (protocol file)
  - No C coding necessary to support new devices
- Modular design
  - Support for all standard I/O record types
    - ai, ao, bi, bo, mbbi, mbbo, mbbiDirect, mbboDirect, longin, longout, stringin, stringout, waveform
    - other record types easy to implement
  - Independent of underlying I/O hardware

New Features

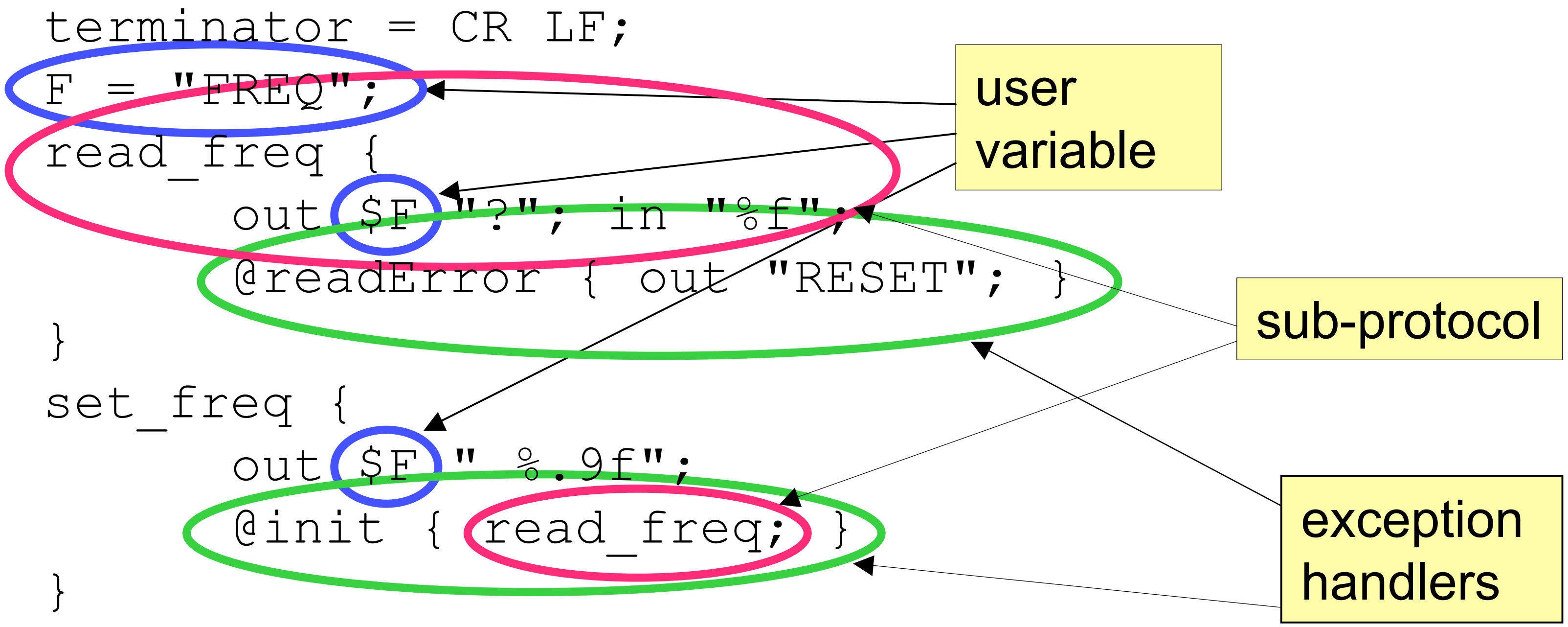
- EPICS R3.14.6 compatible
  - Runs on a PC as well as on vxWorks
- Support for asynDriver
  - Generic serial line, GPIB, and network driver included
  - Stream Device and other software can talk to the same device without interference.
- Lots of protocol extensions

New Protocol Features

- Exception handlers
  - E.g. init handler (used in `init_record`)
- Sub-protocols (use one protocol in an other one)
- More protocol variables
  - E.g. minimal and maximal input length
  - User defined variables
- Direct access to record fields

Examples

- Handlers, variables, sub-protocols



```
terminator = CR LF;
F = "FREQ";
read_freq {
      out $F "?"; in "%f";
      @readError { out "RESET"; }
}
set_freq {
      out $F " %.9f";
      @init { read_freq; }
}
```

user variable

sub-protocol

exception handlers

Format converter modules

- Standard formats (`%f %x %[a-z]` ...)
- Additional formats (`%b %D %{s0|s1|s2}` ...)
- New formats can be implemented easily
  - e.g. regular expression: `%/regexp/`
- User functions as pseudo-formats
  - e.g. checksum as `%@`
  - encode after format, decode before scan

Examples

- Formats, field access, user functions

```
read_strange_status_bits {
      out "stat?";
      in "%12B.!";
}
access_record_fields {
      out "%(A).3f, %(B)g, %(C)d, %(DESC)s";
      in "%f%(EGU)s";
}
apply_user_function {
      in "A=%d%@";
}
```

Some Gimmicks

- Reload protocols without reboot
- `dbior` shows all used protocols
- Support for calcout record
- Memory tracing to find leaks
- Dynamically allocated buffers
  - don't care about buffer sizes
  - implemented with new StreamBuffer class

StreamBuffer Class

- Used for all buffers in Stream Device
- Transparently uses static or allocated memory
- Implements appending, deleting, replacing of bytes or strings, and memory management in an efficient way
- Implements `printf` to append formatted values
- Can be used by other software as well

Where to get Stream Device 2.x

- Pre-release available from dirk.zimoch@psi.ch
- Might be instable (testers welcome)
- No complete documentation yet
  - but Stream Device 1.x documentation helps: www.delta.uni-dortmund.de/controls