HAL "multiKore":
making HAL safe & fast for multicore platforms

Michael Haberler
Mick Grant

the problem: HAL historic design assumptions <-> today's platforms

- assumes a single core running RT threads
- memory ordering: assumes a conservative memory model (x86)
- HAL object layout assumes a deprecated cache architecture causing false sharing
- (ab)uses non-preemptive scheduling as shared state synchronization method

examples where things go wrong

- (IO) pins and multiple threads updating - lacks atomics
- “handshaking” pins with implied sequencing
  - **iocontrol.tool-number** followed by **iocontrol.tool-change**
- hand-crafted queues (sampler, streamer, motion):
  - bump-index-after-store sequencing
- running **stepgen, pwmgen, encoder, sim_encoder** with thread functs on different cores

making HAL work safely with multiple cores

concern	addressed by
pin/signal value consistency	atomic load/store v2 pin/signal API
harden against load/store reordering	memory barriers: pin, signal, funct propagation on net/unlink
deprecate synchronization by scheduling policy	use lock-free data structures
sharing between threads: values < 64bits:	atomic load/store
sharing between threads: every value counts	ringbuffers
sharing between threads: last value counts	triple buffer
eliiminate false sharing	cache-pad signals by moving into HAL descriptors

impact on code base

~/machinekit/src$ git diff --stat origin/master..HEAD hal

…

302 files changed, 35867 insertions(+), 11043 deletions(-)

…dont panic:

API fully interoperable
all configs run & all regression tests pass

component status

- **major kudos to Mick Grant:**
  - adapted instcomp to generate v2 API calls
  - converted all .comps to v2 API
- **stepgen, pwmgen, encoder, sim_encoder**: disentangled
- most C API comps not converted yet (including motion)
  - largely mechanical process, no surprises

other improvements

- Internal API cleanup: single accessor function for HAL objects
- extensive timing analysis led to some red herrings coming up - rtapi_get_clocks() is largely irrelevant
- HAL name limit raised to 127
- “busy looping” a core on a job:
  - newthread busy 1000 posix nowait cpu=2
  - can use normal Linux API (eg eventfd(2)) for evented execution
- brought in primitives from concurrencykit.org
- found at https://github.com/mhaberler/machinekit/tree/disentangle

integration/merge status

- 64bit atomics required
    - need to drop ARMv6 for good (rpi1)
- core code is done
- API (mostly header) simplification needed
- need to decide on merge sequencing