WR PTP Core software: status and plans

WR Workshop, Amsterdam, 2016-03-15T09:00:00
Deadline: T0 + 1800

Alessandro Rubini
Independent consultant writing bits and bytes
FSFE, ILS, AGESCI, ALID, comPVter, lugMAN

Copyright © 2016 · Alessandro Rubini, CC-BY-SA 3.0 Unported
wrpc-sw -- what is it?

White Rabbit PTP Core

The WR "core" is a real HDL core, a logic block (LB core)
(Not an IP core: it's not an Integrated Peripheral)
It's an address space, made up of several other logic blocks
  A CPU running PTP and management code
  128k (usually) RAM
  A PPS generator
  The "endpoint" for the NIC peripheral
  A lot more...

http://www.ohwr.org/projects/wr-cores/wiki/Wrpc_core

White rabbit PC software, thus:

Must run PTP, just like the WR switch
May support IP (Internet Protocol): BOOTP, Ping, and more
Needs to access EEPROM for configuration
Has to be configurable, to support different users

http://www.ohwr.org/projects/wrpc-sw/wiki (outdated)
WRPC: structure from the wiki page
wrpc-sw: Kconfig screen (partial)

`config - WRPC vwrpc-v3.0-146-g68a7d84-dirty Configuration`

WRPC Configuration

Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes,
<M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, <>/
for Search. Legend: [*] built-in  [] excluded  <M> module  < >

^(-)

(128) Size for the temporary output string of pp_printf
(15) Poll interval, in seconds, for temperature sensors

[ ] Build low-level commands for development/testing
[*] Read init commands from flash storage
[ ] Print a stack trace if reset happens
[ ] Add a circular buffer for spll logging, used by tools/wrpc-
    Implementation of pp_printf (hex-and-int) --->
[ ] Build a binary that is the same every time
[*] Use hardware uart (and/or vuart if available)
[ ] Use software uart
[ ] Extra verbose messages for networking
[ ]Verbose messages in softpl1
[ ] Verbose messages in packet filter setup
[ ] More verbose messages in wr core
[ ] Offer an array of 3 fake temperatures, for testing
[*] Mini SNMP responder
[ ] Filter and rx/tx frames in a WLAN (as opposed to untagged)

<Select>  < Exit >  < Help >
Grzegorz dixit:

Bugfixes for synchronization under heavy traffic

Bugfixes to withstand heavy traffic

Lock helper PLL below reference frequency

Kintex-7 support

SDBFS and Flash support

Tx runt frames padding

Increased RAM size to 128kB
After version 3.0, initial minor things

**wrpc-dump**

Diagnostic tool, dumping info from a binary image

```
wrpc-dump <file> <offset> <name>
```

With a new wrpc-sw binary, just pass "name"

"name" is one of pl1, fifo, ppg, ppi, servo_state or ds for data-sets.

**build-cleanup**

Fix warnings in external packages

Rename/restructure Kconfig rules

Add dependency support

Quiet builds

**verbosity-cleanup**

No TRACE, no hard #define

We have CONFIG_NET_VERBOSE and friends

The code now uses "net_verbose(fmt, ...)"
Network Software Support

Network Code (by Wesley Terpstra) is refactored

Each user allocates its own "queue" circular buffer
An old proposal, from Feb 2013, but we couldn't apply it back then
This is a big saving in binary size

Simple UDP API
Simple rdate service ("time", port 37/udp)
Optional syslog support

```
wrpc_socket *ptpd_netif_create_socket(struct wrpc_socket *s,
                                        struct wr_sockaddr *bind_addr,
                                        int udp_or_raw, int udpport);

void fill_udp(uint8_t * buf, int len, struct wr_udp_addr *uaddr);
```

buffer code + arp + ip + udp + icmp + syslog = 8kB
(-: We evaluated picoTCP but we are 0.1 pico :-)
SNMP Support

wrpc-sw now includes a simple SNMP responder

Support for GET queries, not GETNEXT so far

Datacentric implementation

Kconfigurable out

Adam is keeping this uneven ball rolling

We support the following ones:

RFC1213 MIB::sysName.0 (string)          1.3.6.1.2.1.1.5.0
HOST-RESOURCES-MIB::hrSystemUptime.0 (ticks) 1.3.6.1.2.1.25.1.1.0
HOST-RESOURCES-MIB::hrSystemDate.0 (string) 1.3.6.1.2.1.25.1.2.0

laptopo% size lib/snmp.o

text  data  bss  dec  hex filename
   796   156  260  1212  4bc lib/snmp.o
Syslog Support

We have support for generating syslog messages

- Track phase: off and on events
- Temperature: over-temp, reminder, recovery
- Link: on events, with time-from-off
- Kconfigurable out

```
laptop% size lib/syslog.o
          text   data   bss   dec   hex filename
         993     16    88  1097  449 lib/syslog.o
```

Jan 1 00:00:12 192.168.16.242 (00: 01:02:03:03:03) Node up since 12 s
Feb 18 12:36:19 192.168.16.242 Tracking after 23.017 s
Feb 18 12:36:35 192.168.16.242 Lost track
Feb 18 12:36:57 192.168.16.242 2-th re-rtrack after 22.121 s
UDP Console

Work in progress, but why not?

1hr work so far

This may be useful for new developers

It's a very simple and.. ehm.. "intuitive" tool
Even if we have vuart and Etherbone

Kconfig-urable out

Yes, it's my toy, needless, but there (soon)
Packet Filter Rules

Preface: what is pfilt3r?

The WR node is a core, but may not be the only core
Most nodes include Etherbone
Some include other cores eager to receive frames (wr-nic by 7sol, or "streamer" by CERN)

Each frame is inspected and classified
Inspection is synchronous with frame arrival
Every data word (16bits) one instruction is executed
Rules cannot inspect data words in the future
At most 32 rules (some frames are 64 bytes)

New stuff in wrpc-sw

One pfilt3r rule-set for all use cases (28 rules)
We had 3 sets, chosen at config time, for no reason
The "third" core receives everything else
Everything tagged is dropped
Profiling Support

On oft-heard concernd with wrpc is CPU power

We have a *whole* PTP implementation running
And an interactive shell, and the softPLL, and the IP stack. and....

So we introduced the concept of task
A data structure with .name .init() and .job()
This allowed to introduce the "ps" command
Profiling Support

On oft-heard concern with wrpc is CPU power

We have a *whole* PTP implementation running
And an interactive shell, and the softPLL, and the IP stack. and....

So we introduced the concept of task
A data structure with .name .init() and .job()
This allowed to introduce the "ps" command

We discovered PTP takes 0.2% of CPU time

<table>
<thead>
<tr>
<th>iterations</th>
<th>seconds</th>
<th>micros</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>7306589</td>
<td>1000.59</td>
<td>1500</td>
<td>idle</td>
</tr>
<tr>
<td>1006</td>
<td>0.008298</td>
<td>uptime</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.000000</td>
<td>check-link</td>
<td></td>
</tr>
<tr>
<td>3652</td>
<td>0.616027</td>
<td>net-bh</td>
<td></td>
</tr>
<tr>
<td>6060</td>
<td>2.201437</td>
<td>ptp</td>
<td></td>
</tr>
<tr>
<td>172</td>
<td>2.778913</td>
<td>shell+gui</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.179206</td>
<td>stats</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.000000</td>
<td>spl1-bh</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.000000</td>
<td>ipv4</td>
<td></td>
</tr>
<tr>
<td>217</td>
<td>0.025623</td>
<td>arp</td>
<td></td>
</tr>
</tbody>
</table>
Temperature Framework (1/2)

Different implementations have different needs
CERN reads temperature from OneWire
GSI has another CPU reading temperature from OneWire
   This is in conflict with the above
Other groups read temperatures differently

We now have a temperature framework
Each group can define its own temperature agent
   The agent returns a set of temperatures
   Each temperature is name-tagged
Each agent can live in its own file (Kconfig)
   This uses ELF sections

The temperature framework is a task
   The polling interval is set through Kconfig
We have syslog reporting of temperature issues
   The threshold value is set through Kconfig
Temperature Framework (2/2)

We offer library support for common needs

Getting one temperature by name
Iterating over all temperatures
Getting all temperatures as a string

There is a faketemp agent for testing

wrc# temp
pcb:84.7500 roof:INVALID core:INVALID case:INVALID

wrc# faketemp 100 10 -4.5
wrc# temp

pcb:84.7500 roof:100.0000 core:10.0000 case:-4.4999

Unfortunately, OneWire code is CPU intensive

wrc# ps
iterations  seconds.micros  name
2693671    330.952034  idle
13110       12.208542  temperature
Host Builds

We added support to build wrpc for the host
It is an arch-wrpc build of ppsi, but for the host
It includes "minic" drivers, pps simulation etc
Not completely integrated yet (ppsi patch not merged)

It is mainly a tool of me and maybe other developers
I used it to implement UDP, syslog, snmp, temperature

laptop0% sudo ./wrc.elf
UART simulator on the host: use ctrl-C or ctrl-D to exit
Local MAC address: 18:67:b0:c9:b8:73
wrc# ver
WR Core build: wrpc-v3.0-146-g68a7d84-dirty (developer build)
Built: Mar 15 2016 07:44:47
Built for 128 kB RAM, stack is 2048 bytes
wrc# ps
 iterations    seconds.micros   name
 8836           0.013080       idle
 [...]
 8835           9.978917       relax
Peer Delay Mechanism

Cesar Prados implemented the Peer delay mechanism
It is a different way to get T1..T2
Each port must measure the link delay to its peer
It is 3 frames not 4 frames
(but in the end we need 5 frames and T1..T6)

(See handwaving as figure replacement)

The implementation is tested and reliable
But it has some shortcuts
We use the E2E mac address
ppsi needs two wrpc sockets for p2p, which is not there yet

The next step, when this is over, is transparent clocks
This is needed in some arcane implementations (7S)
José Louis Gutiérrez has an implementation beta
VLAN Support

VLAN support is as simple as possible, but no simpler

We have a vlan-supporting set of pfILTER rules

One VLAN gets to the CPU (run-time configurable)
Two VLANs get to Etherbone (build-time configurable)
One VLAN gets to the other cores

At run time, we choose whether to use or not vlans
If not, the old pfILTER rules are instantiated

Network support in wrpc deals with match/tag/untag
Ppsi ignores the VLAN awareness of wrpc-sw
(PPSI docs still state a different plan, though)
Next step

We plan to have UDP PTP in the WR node as an option

It is more interoperable with ignorants

This is not big work at this time

pfilter rules are already in place
wrpc-sw has UDP support
ppsi already is transport-independent
(it has UDP on arch-unix since inception)

Further steps are yet unknown, but I think we are done

There's 128k RAM, and no more than that....