# Table of Contents

Introduction .................................................. 1

1 WRS Documentation ........................................... 1
   1.1 The Official Manuals .................................... 1
   1.2 Supported Hardware Versions ............................. 1

2 Upgrading WRS Software ................................. 2
   2.1 Upgrade from v5.0.1 to v6.0 ............................ 2
   2.2 Upgrade from pre-v5.0 to v5.0.1 or later ............... 5
   2.3 hwinfo for pre-v4.1 ................................... 5
   2.4 Upgrading from v4.0 and later .......................... 5
      2.4.1 Upgrading from v4.0 with the USB cable ........... 6
      2.4.2 Upgrading from v4.0 remotely .................... 6
   2.5 Upgrading from v3.x ................................... 6

3 Configuration of the Device ............................ 7
   3.1 The Configuration File ................................ 7
   3.2 Configuration Items that Apply at Build Time .......... 9
   3.3 Configuration Items that Apply at Run Time .......... 9
   3.4 Timing Configuration .................................. 21
      3.4.1 Common port configuration ......................... 22
      3.4.2 Per-port configuration ............................ 22
      3.4.3 SFP name matching ................................ 24
      3.4.4 Other Deployments ................................ 25
      3.4.5 Calibration ......................................... 25
   3.5 VLANs Configuration .................................. 25
      3.5.1 Example VLAN configuration ....................... 27
         3.5.1.1 Example VLAN configuration by dot-config .... 28
         3.5.1.2 Example VLAN configuration by tools .......... 29
   3.6 Front panel’s LEDs ................................... 30
      3.6.1 Status LEDs ....................................... 30
      3.6.2 Ports’ LEDs ....................................... 30

4 Booting with Barebox ..................................... 31
   4.1 Description of the menus ................................. 31
   4.2 Using wrboot .......................................... 32
   4.3 Creating an NFS-Root Environment for WRS ............ 33

5 WRS Command-Line Tools .............................. 33
   5.1 sdb-read ............................................. 35
   5.2 wrs_vlans ........................................... 36
   5.3 wrs_auxclk ........................................... 37
   5.4 wrs_pstats .......................................... 38

6 SNMP Support ............................................. 39
   6.1 The WRS MIB .......................................... 39
   6.2 show-pstats .......................................... 40
Introduction

The White Rabbit switch (or wrs) is a major component of the White Rabbit (wr) network. Like any modern managed switch, the wrs includes a CPU with its own operating system.

This manual is for people installing wrs devices, who need to configure them in their network.

1 WRS Documentation

Up to and including release v4.0 of wrs software this manual didn’t exist, and the “WRS Build Manual” included some information about configuration.

1.1 The Official Manuals

This is the current set of manuals that accompany the wrs:

- **White Rabbit Switch: Startup Guide**: hardware installation instructions. This manual is provided by the manufacturer: it describes handling measures, the external connectors, hardware features and the initial bring-up of the device.

- **White Rabbit Switch: User’s Manual**: documentation about configuring the wrs, at software level. This guide is maintained by software developers. The manual describes configuration in a deployed network, either as a standalone device or as network-booted equipment. The guide also describes how to upgrade the switch, because we’ll release new official firmware images over time, as new features are implemented.

- **White Rabbit Switch: Developer’s Manual**: it describes the build procedure and how internals work; use of scripts and wrs-specific programs etc. The manual is by developers and for developers. This is the document to check if you need to customize your wrs rebuild software from new repository commits that are not an official release point, or just install your wrs with custom configuration values.

- **White Rabbit Switch: Failures and Diagnostics**: describes various failure scenarios of a switch and ways how to recognize them. Additionally, it describes SNMP exports of a switch (WR-SWITCH-MIB).

The official PDF copy of these manuals at each release is published in the *files* tab of the software project in ohwr.org: (https://www.ohwr.org/project/wr-switch-sw/wikis/Documents#files). This doesn’t apply to release v4.0 and earlier.

The source form of all four manuals is maintained in wr-switch-sw/doc. Within the repository, three of them the *User’s Manual*, the *Developer’s Manual* and the *Failures and Diagnostics* are always tracking the software commits, while the *Startup Guide* may not be authoritative because it is bound to device shipping rather than software development.

1.2 Supported Hardware Versions

This document applies to versions 3.3 and 3.4 of the wrs device.

Very few specimens of wrs 3.0 though 3.2 were manufactured; if you are the owner of one of them, please refer to version 3.3 of the wrs-build document, that includes appendices about using older versions. As usual, it is in the *files* tab of https://www.ohwr.org/project/wr-switch-sw/wikis/Documents#files.

V1 and V2 were development items, never shipped.
2 Upgrading WRS Software

The WRS is shipped with a current version of its software image, which is sometimes called firmware. If your devices are running a previous version of the software you may want to upgrade, or you may want to replace the firmware images after rebuilding your own, as explained in the Developer’s Manual.

If you run version 4.1 or later please copy wrs-firmware.tar file into the /update partition via scp or web-interface and restart your switch.

When the running version during the update is at least v5.0, then update script performs the check of md5 sums of all files inside the wrs-firmware.tar. If at least one checksum is incorrect, the update is aborted and an error is reported via SNMP (object wrsFwUpdateStatus) until the next successful update. Additionally, the wrs-firmware.tar containing corrupted file is renamed to wrs-firmware.tar.checksum_error. This file is automatically removed during the next successful update.

When checksums in the wrs-firmware.tar are not available (for example during downgrading to version pre-6.0) appropriate warning message is printed to the console.

If this method of upgrading firmware works for you, you can ignore the rest of this chapter, which explains a transition between the initial way we passed MAC addresses and the safer approach we introduced in v4.1

2.1 Upgrade from v5.0.1 to v6.0

Apart from following the instructions in the previous section, there is no special action needed from the user when upgrading firmware from v5.0.1 to v6.0. Yet, special attention should be payed to a number of improvements in configuration and slight changes in the behavior of the WR Switch. Here are highlights:

- **IP address of the management port**
  WR switch running the defult configuration file included in the v6.0 release will try to get dynamic IP address for the management port at boot time. In case the DHCP server is not responding, WR switch will use default static IP assignment:

  - IP address: 192.168.1.254
  - Netmask: 255.255.255.0
  - Network: 192.168.1.0
  - Broadcast: 192.168.1.255
  - Gateway: 192.168.1.1

- **Low Phase Drift Calibration (LPDC)**
  Low Phase Drift Calibration is a new feature added in release v6.0 that improves phase stability between WR switch restarts to <10ps. Due to FPGA limitations this functionality is present only on ports 1-12. The LPDC requires an additional, automated calibration procedure to run for Tx and Rx path of each affected FPGA transceiver. The Tx calibration is performed once for all ports at startup of the WR switch. The Rx calibration is performed each time a link goes up on a port. Both Tx and Rx calibration is indicated by blinking of Link/WR Mode LED (left). The downside is that the calibration makes startup of the WR switch longer. Similarly, the time between connecting a fiber and the link going up (e.g. as observed in wr_mon) is noticeably longer.

- **Configuration of the WR switch** - while it was feasible to edit by hand the dot-config file in the pre-v6.0 firmware versions, it is highly discouraged now. Since the number of configuration options increased greatly, the dot-config file is very long and complex. Therefore, it is highly recommended to use the menuconfig to generate the dot-config (e.g. wrs_menuconfig tool on the WR switch). Highlights of the changes to configuration are provided in the following bullet points.
• Configuration of PTP/PPSi in menuconfig - up to firmware v5.0.1, the configuration of PPSi in menuconfig was limited to a global timing mode (e.g. GM, BC) and per-port proto (raw, udp), rx/tx delays, role (e.g. master, slave, auto) and fiber id. Any more complex configuration of PPSi had to be done via a custom ppsi.conf file and this was also limited. It was a known issue that the auto role that uses PTP’s Best Master Clock Algorithm was not implemented correctly. We have updated PPSi fixing all known issues related to compliance with IEEE1588-2008 (PTPv2) and we have already introduced a number of new features from the IEEE1588-2019 (PTPv2.1) edition. This resulted in a substantial increase and reorganization of configuration parameters for PPSi in menuconfig and dot-config. In short, the default behavior of Grand-Master, Free-Running Master and Boundary Clock are preserved, yet with different names for the configuration options which are compliant with the IEEE1588-2019. Here are the highlights of the changes

• Port Timing Configuration is organized in per-port sub-menus instead of a simple configuration strings. On each physical port, zero or one PTP Instance can configured (in future release we will allow more than one PTP Instance). If you do not want to use PTP on a given physical port, zero PTP Instances should be used. For each PTP Instance, a number of configuration parameters can be selected, including:
  • Network Protocol - it allows to set PTP Mapping to IEEE 802.3 or UPD/Ipv4, it used to be called proto
  • Delay Mechanism - it allows to set E2P or P2P mechanism, previously possibly only via custom ppsi.conf
  • SNMP monitoring - it can be disabled which was the case previously when role was set to non-wr
  • Profile - it allows to chose between WR and standard PTP, the choice made previously via the role setting
  • Desired State for BC or BMCA mode for GM - they replaces the role function for the Master/Slave/auto settings.
  • timestampCorrectionPortDS.egressLatency - it replaces the tx delay
  • timestampCorrectionPortDS.ingressLatency - it replaces the rx delay
  • externalPortConfigurationEnabled - External Port Configuration is a new concept (optional feature) introduced in the IEEE1588-2019 (PTPv2.1) that legalizes what had been done in WR switches since long: manual assignment of the Master/Slave role per port. This option is used when a WR switch is set to Boundary Clock Timing Mode. With this option enabled, the Best Master Clock Algorithm is disabled and each port must be assigned with the Desired State: Master/Slave/Passive. If External Port Configuration is disabled, the Best Master Clock Algorithm is used. Note that this option is global, i.e. it is not allowed to use it only on some ports.
  • MasterOnly - MasterOnly is a new concept (optional feature) introduced in the IEEE1588-2019 (PTPv2.1) that legalizes per-port assignment of Master role, equivalent of setting role to be Master in the pre-v6.0 firmware. This assignment can be done on per-port basis and it coexists with the Best Master Clock Algorithm. This option is used when a WR switch is set to Grand-Master, Free-Running Master, or Arbitrary Grand-Master.

• PTP options - it is a new sub-menu that allows to set a number of global PTP parameters which previously had to be modified by customizing ppsi.conf file. Thise include: domain, priority1, priority2, clockClass. While these parameters are pre-filled automatically depending on the Timing Mode, they can be overridden if need be.

• Timing Mode - setting timing mode provides default configuration for all the above-mentioned options such that the behavior of the WR switch is equivalent to the same
Timing Mode in pre-v6.0 firmware versions. There are two new modes available now: \textbf{Arbitrary Grand-Master} and \textbf{Custom}. The \textbf{Arbitrary Grand-Master} mode is similar to \textbf{Grand-Master} but has different clockClass and ARB timescale. The \textbf{Custom} allows full control over PTP settings, otherwise limited to options allowed for a given Timing Mode.

- \textbf{Configuration of VLANs in menuconfig} - pre-v6.0 firmware provided limited/simplified VLAN per-port configuration that did not allow setting some less-common configurations, otherwise allowed by CLI (\textit{wrsw_vlan}). As of v6.0 firmware, such more user-friendly setting is still provided by default, yet it can be extended by setting \textbf{Enable raw ports configuration}.

- \textbf{New configuration options in menuconfig}
  - \textbf{Source for a run-time replacement of leap seconds file} - the leap seconds file can be now fetch from a server and updated in runtime.
  - \textbf{PPS generation} - it is now possible to specify a number of aspects of the PPS generation, the default configuration might be slightly different from the pre-v6.0 firmware.
  - \textbf{LLDP options} - it provides configuration of Link Layer Discovery Protocol (LLDP) which is now supported. It is enabled by default. \textbf{Note that LLDP sends Ethernet frames of approximately 200 bytes. In the worst-case, transmission of such frames can increase traffic latency by 0.8us\textsuperscript{1}.} If you do not use LLDP and latency is of concern, you should disable LLDP option.
  - \textbf{Disable web interface} - web interface is now disabled by default and considered deprecated (no effort was put in making sure it works properly). \textbf{Users are strongly discouraged from using the web interface.}

- \textbf{DHCP forever configuration in menuconfig} - this behavior can be set for two parameters: Management port configuration and \textbf{Source for a run-time replacement of dot-config} (it is called \textit{Force to get the URL to a dot-config via DHCP} in the second case). In case the DHCP server was not available, in the pre-v6.0 firmware, the WR switch would start with local dot-config and no IP. As of v6.0, with these settings, the WR switch waits forever for the DHCP server to reply. This behavior was introduced to ensure that operational WR switches always boot with desired remote dot-config file. Even if, in case of a power cut, DHCP server takes longer to boot than a WR Switch. Unfortunately, the downside is that while waiting for the DHCP server, WR switch it is not accessible via the management port or management USB. The only way to stop the endless loop is to connect via the back USB (ARM Test) and follow the instructions, i.e. press Enter.

- \textbf{Updated look of wr\_mon} - this mostly commonly used Command Line Interface tool in the WR switch has been updated to provide more information, here are the highlights:
  - \textbf{Timing Mode} indicates the current timing mode of the Soft PLL, it is not the intended \textbf{Timing Mode} configured in the \textit{dot-config} file. So, a GM or BC switch will show the \textbf{Timing Mode} to be \textit{FR} (free running) until it locks to the source of time, i.e. input 10MHz & 1PPS or Slave port, respectively.
  - \textbf{PTP/EXT/PDETECT states} provides information about
    - \textbf{PTP} - indicates the PTP state of a given port, as established by Best Master Clock Algorithm, or configured via External Port Configuration. Note that ports which are down (not connected) are always placed in \textbf{DISABLED} state.
    - \textbf{EXT} - indicates the state of the currently used extension, for v6.0 firmware, it is the state of the White Rabbit state machine (more extensions in future will be

\textsuperscript{1} The maximum PTP frame size is 96 bytes which translates into 0.76us latency increase to the data traffic, in case the PTP frame is sent to a port at the same time as the data traffic frame is to be forwarded to this port. An additional 100 bytes of the LLDP frame, translates into additional 0.8us latency introduced by protocol frames.
supported). In the pre-v6.0 firmware, the state of PTP or extension was provided in the **PTP state** column.

- **PDETECT** - indicates the state of a state machine that governs the detection of extensions. When a WR link is being established, it shows **WA_MSG**, when it is successfully established, it shows **EXT_ON**

- **Synchronization status** - most of the parameter names were replaced with their equivalents used in the IEEE1588 standard:
  - **DelayMM** is the previously **Round-trip time (mu)**
  - **DelayMS** is the previously **Master-slave delay**
  - **Total link asymmetry** is the previously **delayAsymmetry**
  - **delayCoefficient** is the previously **alpha**
  - **ingressLatency** is the previously **Slave PHY delay RX** without the bitslide
  - **egressLatency** **Slave PHY delay TX**
  - **offsetFromMaster** is the previously **Clock offset**
  - **semistaticLatency** is a new parameter that indicates the bitslide value

### 2.2 Upgrade from pre-v5.0 to v5.0.1 or later

During the update from the pre-v5.0 firmware to v5.0.1 or later (or later) you might see the following errors on the console.

```
/wr/bin/sdb-read: can’t load library ’libm.so.1’
Creating SDB filesystem in /dev/mtd5
cp: can’t stat ’/wr/etc/sdb-for-dataflash’: No such file or directory
done
```

Please ignore this message, no real error occurred nor **hwinfo** partition (/dev/mtd5) was overwritten. The error is caused by an old firmware trying to run a binary (sdb-read to be precise) from the new firmware image. The problem became visible now, because between v4.2 and v5.0 we uplifted the buildroot, which changed the version of **libm** library from **libm.so.0** to **libm.so.1**.

### 2.3 hwinfo for pre-v4.1

Version 4.1 (October 2014) and later ones use a new way to pass hardware information to all levels of software, such information includes the MAC addresses for the management Ethernet and the SFP ports. Information is now stored in a Flash partition called **hwinfo**, using the SDB file format. SDB is defined in the **fpga-configuration-space** within ohwr.org. Before using SDB we used to edit the boot loader’s configuration at flash time.

The **hwinfo** structure is written to **dataflash** by the manufacturer. It is never changed even when performing a complete re-flash of the device, because the flashing scripts preserve the **hwinfo** memory area.

When upgrading from a pre-4.1 switch software, you need to create this **hwinfo** data structure. The procedure is mostly automatic, but you need to be aware of the steps involved, in case something goes wrong.

### 2.4 Upgrading from v4.0 and later

Version 4.0 and later of **wr-switch-sw** are able to upgrade themselves if you place the proper files in the **/update** directory of the wrs. However, in version 4.0 we forgot to provide for an upgrade of the boot loader and didn’t note that if the front USB cable is not plugged, the upgrade procedure blocks.
This latter problem happens because messages are written to the management USB port, to help people flashing from scratch, and the write is a blocking one by default: if nobody collects the USB data, the system waits for a recipient. With version 4.1.1 the problem was fixed using non-blocking operations (it is better to loose messages than to block the installation because nobody is reading).

Thus, there are two different ways to upgrade; which one you prefer we can’t tell. Both work, each with its own drawbacks. Each of them preserves the current MAC addresses.

2.4.1 Upgrading from v4.0 with the USB cable

This is the procedure if you are able to walk to your wrs and connect to the management USB port, even if the port is not actually used:

• Copy your own `wrs-firmware.tar` for at least v4.1 into the `/update` partition. This can be the official firmware or one you built yourself. Then reboot and wait for everything to settle (the system will reboot once more by itself).
• Copy `wrs-firmware.tar` again. And reboot again. The system will reboot once more by itself.
• Now you have a running updated version with your `hwinfo` in place and the old MAC addresses preserved.

We save you from the long description of what is happening in the various steps. If needed, it is in the `git` history of `wr-switch-sw`, at release point v4.1.

2.4.2 Upgrading from v4.0 remotely

If you can’t walk to the switch, the procedure is faster but more commands need to be typed on the root shell of the switch. We support a single upgrade provided you change the kernel and initial filesystem before rebooting.

• Copy your own `wrs-firmware.tar` for at least v4.1 into the `/update` partition. This can be the official firmware or one you built yourself.
• Create and mount `/boot` within the switch. This means running the following commands in `ssh`:
  ```
  mkdir /boot
  mount -t ubifs ubi0:boot /boot
  ```
• Copy `wrs-initramfs.gz` (which is to be found inside `wrs-firmware.tar`) to the `/boot` partition just mounted. This ensures the new upgrade procedure will run, the one that doesn’t block if the USB cable is unplugged.
• Copy `zImage` (again, to be found inside `wrs-firmware.tar`) to the `/boot` partition. This is need to be able to access the `hwinfo` partition at next boot.
• Reboot and wait for everything to settle (the system will reboot once more by itself after upgrading everything). The MAC addresses will be saved to `hwinfo` during the update procedure, thanks to the new kernel and new boot procedure you manually copied to `/boot`.

**Note:** if you forget to place the new kernel or `wrs-initramfs.gz` in `/boot`, no big damage will happen, but you’ll have lost your MAC address for the WR ports. You’ll find a randomly-chosen value, that will however be persistent over reboot (because it is saved to `hwinfo` after you boot with the new kernel.

2.5 Upgrading from v3.x

Upgrading from versions older than v4.0 (August 2014) requires physical access to the device and, unfortunately, requires some extra steps especially if you want to preserve your MAC addresses.
One possible path is flashing version 4.0 (please refer to v4.0 manuals) and then proceed as described in Section 2.4 [Upgrading from v4.0 and later], page 5. When flashing version 4.0 you’ll need to pass your MAC addresses on the command line of the flasher, so please take note of what they are.

Another option is flashing the latest firmware version and then build your own hwinfo structure by specifying your MAC addresses. wr-switch-sw includes specific tools for both steps. They are described in the Developer’s Manual, because they are expected to only be performed by the manufacturer, not the final user.

3 Configuration of the Device

The switch can boot embedded Linux using its internal NAND memory or as an NFS-Root host. In the latter case is especially useful for development, when binaries of various software daemons might need to be updated regularly. But this option implies some network traffic on your management network, as well as an NFS server able to host all of your switches.

The configuration of every wrs is described in a Kconfig-generated file. This configuration file can be found in the filesystem of wrs under /wr/etc/dot-config. The switch runs by default with a configuration provided with the stable firmware release. If you are running a small WR network, or just a single switch for lab tests, you can modify various settings directly from the shell or web interface. After logging to wrs (e.g. using ssh) you can call “wrs_menuconfig” command to modify the locally stored configuration file in a convenient way. However, this approach doesn’t scale well to large installations, because if a device needs to be replaced, its own configuration is lost.

For operational networks, it is recommended to setup a DHCP/TFTP server for central management of configuration files and to let wrs download its dot-config file at boot time. This also facilitates recovering a wr network in case of WRs hardware failure. It takes only a change in the DHCP database to boot new WRs without losing the desired configuration. In this case, each wrs device loads its own configuration file each time it is booted, and applies the choices before starting any service. The name of the configuration file can include the MAC, IP address or HOSTNAME of the device, to allow running several switches with different configurations in the same network. The location of the configuration file can be stored in the dot-config or be retrieved from DHCP server.

3.1 The Configuration File

The main configuration file for the wrs is /wr/etc/dot-config. You can create this file either by running “make menuconfig” within the local clone of wr-switch-sw repo or directly in the shell of your switch (e.g. through ssh), and making your choices. You can also edit the text file using external tools, or run other configurators: make config, make xconfig, make gconfig, make nconfig.

The configuration step creates .config, that you can copy to your wrs as /wr/etc/dot-config. After reboot, you’ll see your choices in effect.

The rest of this section describes various options that are provided through the configuration (dot-config) file.

CONFIG_DOTCONF_FW_VERSION
CONFIG_DOTCONF_HW_VERSION

Free-text items to describe switch’s firmware CONFIG_DOTCONF_FW_VERSION and hardware CONFIG_DOTCONF_HW_VERSION version. Additionally, the default value of CONFIG_DOTCONF_FW_VERSION can be used as a version of a Kconfig file. These fields do not affect any functionality of the switch.
CONFIG_DOTCONF_INFO

Free-text item to describe any additional information about dot-config. This field does not affect any functionality of the switch.

The next configuration item is a choice about source of the dot-config file (items starting with CONFIG_DOTCONF_SOURCE_). The following dot-config sources are implemented in current version:

CONFIG_DOTCONF_SOURCE_LOCAL

Use local dot-config file stored in /wr/etc/dot-config. In this case no network access is performed.

CONFIG_DOTCONF_SOURCE_REMOTE

Get a dot-config file from the URL provided in CONFIG_DOTCONF_URL.

CONFIG_DOTCONF_SOURCE_FORCE_DHCP

Get a network location of a dot-config file from a DHCP server. Server can be configured in a way to provide the entire URL to the dot-config in the “filename” configuration field of the DHCP server. In this case, provided URL has to be in the same form as CONFIG_DOTCONF_URL.

As an alternative, “filename” can be configured only as a path. It will be then interpreted as a path on a TFTP server, which IP address is taken from the configuration field “The BOOTP next server option” of a DHCP server.

If the DHCP service is not available at boot time, the switch will wait forever until it has obtained the DHCP lease from the server. If the DHCP server is reachable but switch fails to download dot-config file, it will report appropriate error over SNMP.

This choice is only available if CONFIG_ETH0_DHCP is chosen for network configuration.

CONFIG_DOTCONF_SOURCE_TRY_DHCP

The same as CONFIG_DOTCONF_SOURCE_FORCE_DHCP, but this option does not trigger errors in SNMP’s objects if switch fails to retrieve the URL to the dot-config via DHCP. Note that syntax and download errors of dot-config are notified in the same way as for other choices.

If the selected option triggers WRS to download a new dot-config file and it passes the validation process, the new dot-config will replace a local copy. In case there are errors or unknown configuration entries in the retrieved file, the old, local dot-config will be used.

The URL (stored in CONFIG_DOTCONF_URL or retrieved via DHCP) is of the form “protocol://host/pathname”. The special upper-case strings HOSTNAME, IPADDR and MACADDR are substituted with the current hostname, IP address or MAC address of the management port of the switch. By this, the same configuration string can be used to set up a batch of switches with different configurations.

The three parts of the URL are as follows:

protocol

We support http, ftp and tftp. Any other protocols result in an error, and the dot-config file is not replaced.

host

The host can be an IP address, or a name. In order to use a name you must specify a valid CONFIG_DNS_SERVER and optionally CONFIG_DNS_DOMAIN. Alternatively DNS configuration can be taken from the DHCP server. The values in the current dot-config are used to load the new file.
Chapter 3: Configuration of the Device

path

The pathname can include directory components and any of HOSTNAME, IPADDR, MACADDR.

For example this is a valid configuration for run-time update:

```plaintext
CONFIG_DOTCONF_SOURCE_REMOTE=y
CONFIG_DOTCONF_URL="tftp://morgana/wrs-config-IPADDR"
CONFIG_DNS_SERVER="192.168.16.1"
CONFIG_DNS_DOMAIN="i.gnudd.com"
```

It results in `wrs-config-192.168.16.9` being served to the WRS.

Please remember, that the new dot-config should include a valid `CONFIG_DOTCONF_SOURCE_*` setting, or you won’t be able to update the configuration at the next boot. In any case, you can always copy a configuration file using `ssh`, or use the web interface to change the configuration. Changes performed using the web interface are immediately active, because the web server takes proper action; the new file copied over with `ssh`, or any hand-edits, are only effective at next boot, unless overwritten by a remote configuration file. In case there are errors or unknown configuration entries in the retrieved file, the old one will be used.

### 3.2 Configuration Items that Apply at Build Time

The following items in dot-config are used at build time; changing them in the running switch has no effect:

- **CONFIG_BR2_CONFIGFILE**
  This string option lists a file to be used as Buildroot (BR2) configuration. A simple filename or relative pathname refers to the `configs/buildroot` directory; an absolute pathname is used unchanged.

- **CONFIG_KEEP_ROOTFS**
  A boolean option for developers: if set the build script does not delete the temporary copy of the generated filesystem and reports its pathname in the build messages.

### 3.3 Configuration Items that Apply at Run Time

The following items in dot-config are used at run time: at every boot the value (the old one or the just-downloaded one) is used in the appropriate way, before the respective service is started.

- **CONFIG_DOTCONF_SOURCE_LOCAL**
- **CONFIG_DOTCONF_SOURCE_REMOTE**
- **CONFIG_DOTCONF_SOURCE_FORCE_DHCP**
- **CONFIG_DOTCONF_SOURCE_TRY_DHCP**
- **CONFIG_DOTCONF_URL**
  The source and location of a config file to be used at a replacement the next time the system boots. See Chapter 3 [Configuration of the Device], page 7, and Section 3.1 [The Configuration File], page 7, for details.

- **CONFIG_LEAPSEC_SOURCE_LOCAL**
- **CONFIG_LEAPSEC_SOURCE_REMOTE_FORCE**
- **CONFIG_LEAPSEC_SOURCE_REMOTE_TRY**
- **CONFIG_LEAPSEC_URL**
  The `/etc/leap-seconds.list` file is used to get the current TAI offset. See [wr_date], page 34. The `CONFIG_LEAPSEC_SOURCE_LOCAL` choice forces the use the local leap seconds file that is stored locally on the switch (under `/etc/leap-seconds.list`). `CONFIG_LEAPSEC_SOURCE_REMOTE_FORCE` and `CONFIG_LEAPSEC_SOURCE_REMOTE_TRY` provide similar functionality. In both cases the `leap-seconds.list` is downloaded.
Chapter 3: Configuration of the Device

at boot time using the URL defined in CONFIG_LEAPSEC_URL. The address is defined in the same format as CONFIG_DOTCONF_URL. If the downloaded file is newer than the local copy stored in /etc/leap-seconds.list, it is used to update the local copy. In case the WRs is unable to fetch a leap seconds file from the provided location, using CONFIG_LEAPSEC_SOURCE_REMOTE_FORCE will result in an error generated through SNMP, while using CONFIG_LEAPSEC_SOURCE_REMOTE_TRY will suppress the alarm.

CONFIG_ETH0_DHCP
CONFIG_ETH0_DHCP_ONCE
CONFIG_ETH0_STATIC

Configuration of management port’s (eth0) IP. When CONFIG_ETH0_DHCP is used, then switch tries to obtain IP via DHCP forever. For option CONFIG_ETH0_DHCP_ONCE switch tries to get IP via DHCP once, if this try is unsuccessful then switch uses static IP. CONFIG_ETH0_STATIC forces switch to use provided static IP address.

CONFIG_ETH0_IP
CONFIG_ETH0_MASK
CONFIG_ETH0_NETWORK
CONFIG_ETH0_BROADCAST
CONFIG_ETH0_GATEWAY

Management port’s (eth0) static IP configuration when CONFIG_ETH0_DHCP_ONCE or CONFIG_ETH0_STATIC parameter is used.

CONFIG_HOSTNAME_DHCP
CONFIG_HOSTNAME_STATIC
CONFIG_HOSTNAME_STRING

These options describe how to set hostname of the switch. From DHCP (CONFIG_HOSTNAME_DHCP) or use a predefined value (CONFIG_HOSTNAME_STATIC) defined in option CONFIG_HOSTNAME_STRING.

CONFIG_ROOT_ACCESS_DISABLE

Disable root access via ssh. With this option enabled it is still possible to use sudo to get root privileges.

CONFIG_LDAP_ENABLE
CONFIG_LDAP_SERVER
CONFIG_LDAP_SEARCH_BASE
CONFIG_LDAP_FILTER_NONE
CONFIG_LDAP_FILTER_EGROUP
CONFIG_LDAP_FILTER_CUSTOM
CONFIG_LDAP_FILTER_EGROUP_STR
CONFIG_LDAP_FILTER_CUSTOM_STR

Set of options related to providing an authorization via LDAP for ssh. To be able to use LDAP please enable an option CONFIG_LDAP_ENABLE, provide LDAP server (CONFIG_LDAP_SERVER) and the search base (CONFIG_LDAP_SEARCH_BASE). It is possible to limit the access to a particular e-group used at CERN (CONFIG_LDAP_FILTER_EGROUP to enable and CONFIG_LDAP_FILTER_EGROUP_STR to provide the e-group’s name) or to provide the custom filtering string (CONFIG_LDAP_FILTER_CUSTOM to enable and CONFIG_LDAP_FILTER_CUSTOM_STR to provide the filter). For more information please refer to the Kconfig’s help.
Chapter 3: Configuration of the Device

CONFIG_AUTH_LDAP
CONFIG_AUTH_KRB5
CONFIG_AUTH_KRB5_SERVER

Choose the authentication method. CONFIG_AUTH_LDAP for LDAP authentication, CONFIG_AUTH_KRB5 for Kerberos authentication. For the later one it is obligatory to specify Kerberos Realm CONFIG_AUTH_KRB5_SERVER.

CONFIG_ROOT_PWD_IS_ENCRYPTED
CONFIG_ROOT_PWD_CLEAR
CONFIG_ROOT_PWD_CYPHER

This set of options allows setting the password for the “root” user (the administrator). The password is used to login to your switch using ssh (secure shell). If you choose CONFIG_ROOT_PWD_IS_ENCRYPTED, you will be prompted for a text version of a pre-encrypted password (CONFIG_ROOT_PWD_CYPHER). To encrypt your magic string, you must run “mkpasswd --method=md5 magic” on your Linux host (or switch). If you choose to configure an unencrypted password, then you are asked to specify it as CONFIG_ROOT_PWD_CLEAR. In this latter case encryption is performed at run-time to use the normal ssh authentication, but the clear-text password will remain in dot-config. By default the root password is an empty string, like in the initial wr-switch-sw releases.

CONFIG_NTP_SERVER

The NTP server used to prime White Rabbit time, at system boot. The option can be an IP address or a host name, if DNS is properly configured. The configuration value is stored in /etc/wr_date.conf. An empty string (default) disables NTP access at boot time.

CONFIG_DNS_SERVER
CONFIG_DNS_DOMAIN

The DNS server (as an IP address) and default domain. The values end up in /etc/resolv.conf of the runtime filesystem. By default the two strings are empty. If CONFIG_ETH0_DHCP or CONFIG_ETH0_DHCP_ONCE is used, /etc/resolv.conf file will be populated with DNS settings received from the DHCP server. If configuration items for static (CONFIG_DNS_*) and dynamic (CONFIG_ETH0_DHCP) DNS configuration are used simultaneously then information from both sources end up in the /etc/resolv.conf file. However, information from CONFIG_DNS_* is placed first.

CONFIG_REMOTE_SYSLOG_SERVER
CONFIG_REMOTE_SYSLOG_UDP
CONFIG_LOCAL_SYSLOG_FILE

Configuration for system log. The name (or IP address) of the server is stored in /etc/rsyslog.conf of the runtime filesystem. The UDP option, set by default, chooses UDP transmission; if unset it selects TCP communication. CONFIG_LOCAL_SYSLOG_FILE option indicates the file to which syslog messages will be stored in a local filesystem of the wrs. The file is rotated when reaching 1MB. The rotation is done through 10 indexed files syslog, syslog.1-9. By default these files are created in /tmp folder. If remote server is specified, the messages go to both, server and local file.
Logging options for the three main WRS processes and other programs. CONFIG_WRS_LOG_OTHER is currently used by:

- `wrs_watchdog` daemon
- `wrs_throttling` executed once at boot up
- `wrs_auxclk` executed once at boot up
- `wrs_custom_boot_script.sh` executed once at boot up
- Setting VLANs with `vlan.sh` at boot up

Each value can be a pathname, either to a file (e.g. `/dev/kmsg` is a possible “file” target) or a `facility.level` string, like `daemon.debug`, for syslog-based logging. An empty string suppresses all logging for a given daemon. Please note, that unknown facility names will generate a runtime error on the switch. All four strings default to “`daemon.info`”.

**Note:** all messages produced by these programs if syslog is configured will be passed to the syslog at the same configured `<facility>.<level>`, no matter the verbosity of a message. To change the verbosity of programs please use CONFIG_WRS_LOG_LEVEL_*.

Specify verbosity of programs as a string or number. The following levels are supported:

- `ALERT` or 1
- `ERROR` or 3
- `WARNING` or 4
- `INFO` or 6
- `DEBUG` or 7

Not supported levels are ceiled to the valid one.

By leaving this item empty, programs will use its default verbosity level (`INFO`).

**Note:** all messages produced by these programs if syslog is configured will be passed to the syslog at the same configured `<facility>.<level>`, no matter of verbosity of a message.

Specify verbosity of PPSi daemon as a string. This string will be passed to the PPSI after `-d` parameter. Please refer to the PPSI’s documentation for more details.

By leaving this item empty, PPSi daemon will use its default verbosity level.

**Note:** all messages produced by PPSi if syslog is configured will be passed to the syslog at the same configured `<facility>.<level>`, no matter of verbosity of a message.

Value can be a pathname, either to a file (e.g. `/dev/kmsg` is a possible “file” target) or a valid snmpd log option (without -L). Allowed strings are in the format “`S level facility`” (e.g. “`S 2 daemon`”). For example, “`s daemon`” will forward messages to syslog with daemon as facility. To set level (i.e. 5) use “`S 5 daemon`”. For details please check `man snmpcmd`. An empty string turns suppresses all logging. Please note that unknown facility names will generate a runtime error on the switch.
Chapter 3: Configuration of the Device

**Note:** It looks like `Notice` is not a default logging priority as written in `net-snmp` manual.

**CONFIG_WRS_LOG_MONIT**

The string can be a pathname (e.g. `/dev/kmsg`) or a `syslog` string. An empty string is used to represent no logging. If it is needed to select facility and level please leave an empty string here and change `/etc/monitrc` or `/usr/etc/monitrc` file directly. Please note that unknown facility names will generate a runtime error on the switch.

**CONFIG_PTP_OPT_EXT_PORT_CONFIG_ENABLED**

**CONFIG_PORTxx_zz**

These configuration items are used to set up timing parameters of all WR ports. Items are named according to the format `CONFIG_PORTxx_zz` where:
- `xx` – represents the port number (‘01’ to ’18’)
- `zz` – is the property name for the given port `xx`

The default configuration provided with the firmware release will most likely work for the majority of WR networks. In case some customization of these parameters is required, please see Section 3.4 [Timing Configuration], page 21, for details.

**CONFIG_N_SFP_ENTRIES**

**CONFIG_SFP00_PARAMS**

... **CONFIG_SFP17_PARAMS**

Configuration for SFP models.

**CONFIG_N_SFP_ENTRIES** indicates the number of SFP entries defined. Up to 18 SFPs can be defined.

**CONFIG_SFPxx_PARAMS** with index `xx` in range 00 to 17, contains SFP parameters. All SFP models their respective wavelengths you are using in your WR should be entered here.
- `vn` *(optional)* – Vendor Name of an SFP
- `pn` – Part Number of an SFP
- `vs` *(optional)* – Vendor Serial (serial number) of an SFP
- `tx` – TX delay of an SFP in picoseconds
- `rx` – RX delay of an SFP in picoseconds
- `wl_txrx` – Tx wavelength separated by "+" with Rx wavelength of an SFP; for example `wl_txrx=1490+1310` (for 1490nm Tx wavelength and 1310nm Rx wavelength)

See Section 3.4 [Timing Configuration], page 21, for details.

**CONFIG_N_FIBER_ENTRIES**

**CONFIG_FIBER00_PARAMS**

... **CONFIG_FIBER17_PARAMS**

These parameters specify the characteristics of fiber types used in your WR installation.

**CONFIG_N_FIBER_ENTRIES** indicates the number of fiber types defined. Up to 18 different fiber types can be defined.

**CONFIG_FIBERxx_PARAMS** with index `xx` in range 00 to 17, specifies the alpha value for each pair of used wavelengths. This parameter follows the format:
alpha_xxxx_yyyy=1.23e-04, alpha_aaaa_bbbb=4.56e-04, ...

Where:

- xxxx_yyyy and aaaa_bbbb are pairs of used wavelengths
- 1.23e-04 and 4.56e-04 are alpha values to be used for a particular combination of wavelengths.

The index (00 to 17) is then entered as CONFIG_PORTxx_FIBER port parameter to reference the connected fiber type. See Section 3.4 [Timing Configuration], page 21. You are expected to have no more than 18 fiber types installed in your WR network.

CONFIG_TIME_GM
CONFIG_TIME_ARB_GM
CONFIG_TIME_FM
CONFIG_TIME_BC
CONFIG_TIME_CUSTOM

The type of PTP clock this switch is. Only one of the five first items should be set (e.g. running "make menuconfig" offers them as an exclusive choice). The options select:

- CONFIG_TIME_GM a grand-master with external reference, e.g. GPS or Cesium.
- CONFIG_TIME_ARB_GM a arbitrary grand-master which designates a clock that is synchronized to an application-specific source of time.
- CONFIG_TIME_FM a free-running master (FM), used for isolated acquisition networks, without an external reference.
- CONFIG_TIME_BC a normal “boundary-clock” device that is slave on some ports and master on other ports.
- CONFIG_TIME_CUSTOM an option which leaves the possibility to define freely the PTP clock class.

CONFIG_PTP_OPT_DOMAIN_NUMBER

A domain consists of one or more PTP devices communicating with each other as defined by the PTP protocol. A domain defines the scope of PTP message communication, state, operations, data sets, and timescale. PTP devices may participate in multiple domains. For more details please refer to the IEEE 1588-2008 standard.

CONFIG_PTP_OPT_PRIORITY1

A user configurable designation that a clock belongs to an ordered set of PTP devices from which a PTP Master is selected. For more details please refer to the IEEE 1588-2008 standard.

CONFIG_PTP_OPT_PRIORITY2

A user configurable designation that provides finer grained ordering among otherwise equivalent PTP devices. For more details please refer to the IEEE 1588-2008 standard.

CONFIG_PTP_OPT_CLOCK_CLASS

An attribute defining the TAI traceability, synchronization state and expected performance of the time or frequency distributed by a Boundary Clock or Ordinary Clock. Its value can be set only if CONFIG_TIME_CUSTOM parameter is selected. The following table shows the default values used depending on the timing mode "CONFIG_TIME_xx" choice:

<table>
<thead>
<tr>
<th>Timing mode</th>
<th>CONFIG_PTP_OPT_CLOCK_CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_TIME_GM</td>
<td>6</td>
</tr>
</tbody>
</table>

The following table shows the default values used depending on the timing mode "CONFIG_TIME_xx" choice:
For more details please refer to the IEEE 1588-2008 standard.

**CONFIG_PTP_OPT_CLOCK_ACCURACY**
An attribute defining the accuracy of the Local Clock (e.g. local oscillator) of a Boundary Clock or Ordinary Clock. Its value is set automatically according to the timing mode "CONFIG_TIME_xx" choice. It can be also manually set when either **CONFIG_PTP_OPT_OVERWRITE_ATTRIBUTES** is set or the timing mode "CONFIG_TIME_CUSTOM" is selected. The following table gives the default values depending on the timing mode "CONFIG_TIME_xx" choice:

<table>
<thead>
<tr>
<th>Timing mode</th>
<th>CONFIG_PTP_OPT_CLOCK_ACCURACY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_TIME_GM</td>
<td>33</td>
</tr>
<tr>
<td>CONFIG_TIME_ARB_GM</td>
<td>33</td>
</tr>
<tr>
<td>CONFIG_TIME_FM</td>
<td>32</td>
</tr>
<tr>
<td>CONFIG_TIME_BC</td>
<td>254</td>
</tr>
</tbody>
</table>

For more details please refer to the IEEE 1588-2008 standard.

**CONFIG_PTP_OPT_CLOCK_ALLAN_VARIANCE**
An attribute defining the stability of the Local Clock of a Boundary Clock or Ordinary Clock. Its value is set automatically according to the timing mode "CONFIG_TIME_xx" choice. It can be also manually set when either **CONFIG_PTP_OPT_OVERWRITE_ATTRIBUTES** is set or the timing mode "CONFIG_TIME_CUSTOM" is selected. The following table gives the default values depending on the timing mode "CONFIG_TIME_xx" choice:

<table>
<thead>
<tr>
<th>Timing mode</th>
<th>CONFIG_PTP_OPT_CLOCK_ALLAN_VARIANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_TIME_GM</td>
<td>47360</td>
</tr>
<tr>
<td>CONFIG_TIME_ARB_GM</td>
<td>47360</td>
</tr>
<tr>
<td>CONFIG_TIME_FM</td>
<td>50973</td>
</tr>
<tr>
<td>CONFIG_TIME_BC</td>
<td>65535</td>
</tr>
</tbody>
</table>

For more details please refer to the IEEE 1588-2008 standard.

**CONFIG_PTP_OPT_TIME_SOURCE**
This information-only attribute indicates the source of time used by the grandmaster or free-running master. In case the timing mode is set to "CONFIG_TIME_BC" this configuration option is not used and thus hidden from options available e.g. through "make menuconfig".

The following table gives the default values depending on the timing mode "CONFIG_TIME_xx" choice:

<table>
<thead>
<tr>
<th>Timing mode</th>
<th>CONFIG_PTP_OPT_TIME_SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG_TIME_GM</td>
<td>32 (GNSS)</td>
</tr>
<tr>
<td>CONFIG_TIME_ARB_GM</td>
<td>32 (GNSS)</td>
</tr>
<tr>
<td>CONFIG_TIME_FM</td>
<td>160 (INTERNAL_OSCILLATOR)</td>
</tr>
<tr>
<td>CONFIG_TIME_BC</td>
<td>–</td>
</tr>
</tbody>
</table>
By default (CONFIG_PTP_PORT_PARAMS), PTP daemon (PPSi) configuration file is generated based on the values stored in CONFIG_PORTxx_zz parameters. If VLANs are configured, the items CONFIG_VLANS_PORTxx_VID are used as well. Any additional, global PPSi settings can be added by editing manually the /wr/etc/ppsi-pre.conf file, which is then used as the base for the final PPSi configuration file.

Alternatively, PPSi can use a fully custom, user-defined file for configuration (CONFIG_PTP_CUSTOM).

Finally, you can choose PTP_REMOTE_CONF to specify an URL whence the switch will download the ppsi.conf at boot time.

Please see the help provided within Kconfig for more details about the various supported options.

If you chose CONFIG_PTP_CUSTOM from the options described above, you can provide your own filename for the PPSi configuration file. The introduced filename is expected to be installed in the WRs filesystem.

If you choose CONFIG_PTP_REMOTE_CONF specify an URL (http://, ftp:// or tftp://) whence the switch will download the ppsi.conf at boot time. The filename in the URL can include HOSTNAME, IPADDR and/or MACADDR, so the same configuration string can be used to set up a batch of switches with different configurations (similar to the CONFIG_DOTCONF_URL, please refer to Section 3.1 [The Configuration File], page 7).

Additionally to the conditions listed above, if PTP Best Master Clock Algorithm (BMCA) is enabled some spurious PPS can be generated during the transitory phase e.g. when the whole network is initialized. To suppress those
Chapter 3: Configuration of the Device

**CONFIG_PPSGEN_GM_DELAY_TO_GEN_PPS_SEC** defines a delay time in seconds from the moment WRs became a PTP grandmaster to the moment PPS starts to be generated. By default, this parameter is set to 60s.

The **CONFIG_PPSGEN_PTP_THRESHOLD_MS** option is applied when WRs is synchronized to a regular (non-WR) PTP master. This threshold defines a maximum offset to master (in milliseconds) when the WRs is considered synchronized. Once the calculated offset falls below the configured threshold, PPS generation starts. To disable PPS, PTP offset must be greater than the **CONFIG_PPSGEN_PTP_THRESHOLD_MS + 20%**. Setting this parameter to 0 will block any PPS generation for those cases.

The **CONFIG_PPSGEN_PTP_FALLBACK** option, if activated, enables the PPS generation when a slave instance programmed to use an extension protocol (WR, LISync, ...) is falling back to regular PTP synchronization.

**CONFIG_SNMP_TRAPSINK_ADDRESS**
**CONFIG_SNMP_TRAP2SINK_ADDRESS**
**CONFIG_SNMP_RO_COMMUNITY**
**CONFIG_SNMP_RW_COMMUNITY**

Configuration for the SNMP agent. Addresses can be IP addresses or names (if DNS is configured and working), they are unset by default. Community values are strings and they default to public (RO_COMMUNITY) and private (RW_COMMUNITY).

**CONFIG_SNMP_TEMP_THOLD_FPGA**
**CONFIG_SNMP_TEMP_THOLD_PLL**
**CONFIG_SNMP_TEMP_THOLD_PSL**
**CONFIG_SNMP_TEMP_THOLD_PSR**

Threshold levels for FPGA, PLL, Power Supply Left (PSL) and Power Supply Right (PSR) temperature sensors. When any temperature exceeds threshold level, SNMP object WR-SWITCH-MIB::tempWarning will change accordingly.

**CONFIG_SNMP_SWCORESTATUS_DISABLE**

Force SNMP object wrsSwcoreStatus to be always OK. It can be used to ignore all Ethernet frames switching-related issues.

**CONFIG_SNMP_SYSTEM_CLOCK_MONITOR_ENABLED**
**CONFIG_SNMP_SYSTEM_CLOCK_DRIFT_THOLD**
**CONFIG_SNMP_SYSTEM_CLOCK_UNIT_DAYS**
**CONFIG_SNMP_SYSTEM_CLOCK_UNIT_HOURS**
**CONFIG_SNMP_SYSTEM_CLOCK_UNIT_MINUTES**
**CONFIG_SNMP_SYSTEM_CLOCK_CHECK_INTERVAL_DAYS**
**CONFIG_SNMP_SYSTEM_CLOCK_CHECK_INTERVAL_HOURS**
**CONFIG_SNMP_SYSTEM_CLOCK_CHECK_INTERVAL_MINUTES**

When **CONFIG_SNMP_SYSTEM_CLOCK_MONITOR_ENABLED** option is set, the local system time is compared to the time acquired from the external NTP server (according to **CONFIG_NTP_SERVER**). If the difference of time exceeds a given threshold (defined by **CONFIG_SNMP_SYSTEM_CLOCK_DRIFT_THOLD**) expressed in seconds, an error is reported through SNMP.

This comparison is done periodically at a rate expressed either in days (CONFIG_SNMP_SYSTEM_CLOCK_UNIT_DAYS), hours (CONFIG_SNMP_SYSTEM_CLOCK_UNIT_HOURS) or minutes (CONFIG_SNMP_SYSTEM_CLOCK_UNIT_MINUTES). According to the selected unit, the repetition rate will be stored in the appropriate location:
Chapter 3: Configuration of the Device

**Unit Storage**

```
...UNIT_DAYS ...CHECK_INTERVAL_DAYS
...UNIT_HOURS ...CHECK_INTERVAL_MINUTES
...UNIT_MINUTES ...CHECK_INTERVAL_MINUTES
```

The `CONFIG_SNMP_SYSTEM_CLOCK_MONITOR_ENABLED` option is available only if a NTP server has been defined (`CONFIG_NTP_SERVER`).

**Configuration parameters to generate WR-synchronized 10MHz clock on the clk2 output of the wrs front panel. All these parameters are directly passed to wrs_auxclk tool of wrs. In case you need to generate another clock frequency, please refer to Section 5.3 [wrs_auxclk], page 37.**

**CONFIG_WRSAUXCLK_FREQ**

**CONFIG_WRSAUXCLK_DUTY**

**CONFIG_WRSAUXCLK_CSHIFT**

**CONFIG_WRSAUXCLK_SIGDEL**

**CONFIG_WRSAUXCLK_PPSHIFT**

Limit the Rx bandwidth of the traffic that goes from WR ports to Linux. Throttling can be enabled to prevent Linux using 100% of the processing power to receive Ethernet frames coming from WR ports to the CPU. To enable the throttling set `CONFIG_NIC_THROTTLING_ENABLED`. `CONFIG_NIC_THROTTLING_VAL` contains maximum allowed bandwidth in KB/s.

**CONFIG_PPS_IN_TERM_50OHM**

Enable 50ohm termination for 1-PPS input.

**CONFIG_CUSTOM_BOOT_SCRIPT_ENABLED**

**CONFIG_CUSTOM_BOOT_SCRIPT_SOURCE_LOCAL**

**CONFIG_CUSTOM_BOOT_SCRIPT_SOURCE_REMOTE**

**CONFIG_CUSTOM_BOOT_SCRIPT_SOURCE_REMOTE_URL**

It is possible to run a custom script at boot time. In this case please set `CONFIG_CUSTOM_BOOT_SCRIPT_ENABLED`. To run a script `/wr/bin/custom_boot_script.sh` from the local filesystem please set `CONFIG_CUSTOM_BOOT_SCRIPT_SOURCE_LOCAL`.

As an alternative, you can choose `CONFIG_CUSTOM_BOOT_SCRIPT_SOURCE_REMOTE` and specify an URL (http://, ftp:// or tftp://) in `CONFIG_CUSTOM_BOOT_SCRIPT_SOURCE_REMOTE_URL` whence the switch will download the script to be executed at boot time. The filename in the URL can include `HOSTNAME`, `IPADDR` and/or `MACADDR`, so the same configuration string can be used to set up a batch of switches with different configurations (similar to the `CONFIG_DOTCONF_URL`, please refer to Section 3.1 [The Configuration File], page 7).

**CONFIG_LLDPD_DISABLE**

**CONFIG_LLDPD_TX_INTERVAL**

**CONFIG_LLDPD_MANAGEMENT_PORT_DISABLE**

**CONFIG_LLDPD_MINIMUM_FRAME_SIZE**

Set of parameters related to the LLDP daemon (lldpd) configuration. Starting from version 6.0, switch by default sends LLDP frames on all ports (including management). In some installations it may be necessary to disable LLDP traffic on the management port (option `CONFIG_LLDPD_MANAGEMENT_PORT_DISABLE`). Additionally, in some cases (e.g. low latency networks) it may be necessary to disable LLDP at all (`CONFIG_LLDPD_DISABLE`).
The transmission frequency of LLDP frames can be changed using option `CONFIG_LLDPD_TX_INTERVAL`.

Networks that would benefit from the LLDP, but have low latency constraints can use option `CONFIG_LLDPD_MINIMUM_FRAME_SIZE`. With this option LLPD daemon includes only minimal set of information into LLPD frames.

The table below contains comparison of LLDP frame fields for standard and minimal frame size (with `CONFIG_LLDPD_MINIMUM_FRAME_SIZE` option enabled). Size of some of those fields (like System name or System Description) depends on the network configuration.

<table>
<thead>
<tr>
<th>Standard frame</th>
<th>Minimum frame</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>14</td>
<td>ETH header</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>Chasis ID (with MAC)</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>Port ID (with MAC)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Time To Live</td>
</tr>
<tr>
<td>2+len(System name)</td>
<td>2+len(System name)</td>
<td>System name</td>
</tr>
<tr>
<td>2+len(System desc.)</td>
<td>2+len(WR-SWITCH)</td>
<td>System description</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>Capabilities</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>Management Address</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>Port description</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>End of LLDP frame</td>
</tr>
<tr>
<td>69</td>
<td>58</td>
<td>Total length</td>
</tr>
</tbody>
</table>

`CONFIG_HTTPD_DISABLE`

Disable web interface on a switch.

`CONFIG_MONIT_DISABLE`

Disable monitoring of running processes by Monit. Monit by default re-spawns processes that have died. This option should be used only during development.

`CONFIG_FAN_HYSTERESIS`

`CONFIG_FAN_HYSTERESIS_T_DISABLE`

`CONFIG_FAN_HYSTERESIS_T_ENABLE`

`CONFIG_FAN_HYSTERESIS_PWM_VAL`

Use hysteresis to control fans. Enable fans with PWM value `CONFIG_FAN_HYSTERESIS_PWM_VAL` when PLL’s temperature exceeds `CONFIG_FAN_HYSTERESIS_T_ENABLE`. Disable fans when temperature drops below `CONFIG_FAN_HYSTERESIS_T_DISABLE`. These options are intended to be used during development to reduce noise generated by a switch. Don’t use in production as this may affect the synchronization performance.

`CONFIG_READ_SFP_DIAG_ENABLE`

Enable readout of additional monitoring information (DOM) like temperature, Tx/Rx power, from SFP transceivers. Please note that not all Gigabit Ethernet SFP transceivers provide the DOM structure.

`CONFIG_OPTIMIZATION_DEBUGGING`

`CONFIG_OPTIMIZATION_NONE_DEBUGGING`

`CONFIG_OPTIMIZATION_SIZE_SPEED`

`CONFIG_OPTIMIZATION_SPEED`

Compilation options. Chose one of these four choices to control the compilation flags.
CONFIG_OPTIMIZATION_DEBUGGING: Should be the optimization level choice for the standard edit-compile-debug cycle.

CONFIG_OPTIMIZATION_NONE_DEBUGGING: Compile without optimization and with debug information

CONFIG_OPTIMIZATION_SIZE_SPEED: Optimize for size. Enables all -O2 optimizations except those that often increase the code size.

CONFIG_OPTIMIZATION_SPEED: GCC performs nearly all supported optimizations that do not involve a space-speed trade-off.

CONFIG_RTU_HP_MASK_ENABLE
CONFIG_RTU_HP_MASK_VAL
Set the mask which VLAN priorities are considered High Priority traffic (this only concerns the traffic which is fast-forwarded). To enable a custom mask please set CONFIG_RTU_HP_MASK_ENABLE. CONFIG_RTU_HP_MASK_VAL shall contain the mask to be used.

CONFIG_VLANS_ENABLE
Enable VLANs configuration. All below VLAN config options (CONFIG_VLANS_* require this field to be set.

CONFIG_VLANS_PORT_xx_MODE_ACCESS
CONFIG_VLANS_PORT_xx_MODE_TRUNK
CONFIG_VLANS_PORT_xx_MODE_DISABLED
CONFIG_VLANS_PORT_xx_MODE_UNQUALIFIED
VLANs port mode configuration for ports 1..18. It can be one of: Access, Trunk, Disabled or Unqualified. For details please refer to the Section 3.5 [VLANs Configuration], page 25

CONFIG_VLANS_PORT_xx_UNTAG_ALL
CONFIG_VLANS_PORT_xx_UNTAG_NONE
Define whether to remove a VLAN tag from egress frames on port 1..18.

CONFIG_VLANS_PORT_xx_PRIO
Priority value used when tagging incoming frames or to locally override the priority (in Unqualified and Disabled modes). -1 disables the priority overwrite. Valid values are from -1 to 7.

CONFIG_VLANS_PORT_xx_VID
Define the VID tagging incoming frames and notify PPSi which VLAN shall be used for synchronization; only one VLAN number shall be used. This parameter is available for MODE_ACCESS mode. The range of a valid VID is from 0 to 4094. For details please refer to the Section 3.5 [VLANs Configuration], page 25

CONFIG_VLANS_PORT_xx_PTP_VID
Notify PPSi which VLAN(s) shall it use for synchronization; semicolon separated list is allowed. This parameter is available for MODE_TRUNK, MODE_DISABLED and MODE_UNQUALIFIED modes. The range of a valid VID is from 0 to 4094. For details please refer to the Section 3.5 [VLANs Configuration], page 25

CONFIG_VLANS_RAW_PORT_CONFIG
Expert mode. Allow to control all VLAN parameters (CONFIG_VLANS_PORT_xx_PTP_VID, CONFIG_VLANS_PORT_xx_VID, CONFIG_VLANS_PORT_xx_MODE_yy and CONFIG_VLANS_PORT_xx_UNTAG_xx ) without any restrictions. The user must be aware of what it is doing.
CONFIG_VLANS_VLAN

Provide a configuration for VLAN from the range 0000-4094. This option is a comma separated string in the following format:

fid=<0..4094>,prio=<-1|0..7>,drop=<y|n>,ports=<1;2;...;...-...;18>

Where:
- **fid** is a associated Filtering ID (FID) number. This parameter may be useful for complex VLAN configurations. In simple cases it can be omitted. One FID can be associated with many VIDs. RTU learning is performed per-FID. Associating many VIDs with a single FID allowed shared-VLANs learning.
- **prio** is a priority of a VLAN; can take values between -1 and 7; -1 disables priority override, any other valid number takes precedence over the port priority.
- If **drop** is set to **y**, yes or 1, all frames belonging to this VID are dropped (note that frame can belong to a VID as a consequence of per-port Endpoint configuration); can take values **y**, yes, 1, **n**, no, 0.
- **ports** is a list of ports separated with a semicolon sign (";"); ports ranges are supported (with a dash character).

Example:

```
fid=4,prio=2,drop=n,ports=1;3-5;15
```

It sets FID as 4, priority 2, don’t drop frames belonging to this VLAN, assign ports 1, 3-5 and 15 to this VLAN.

For more details about VLANs configuration please refer to the Section 3.5 [VLANs Configuration], page 25.

### 3.4 Timing Configuration

This section describes how timing configuration works in the switch. Please note that, comparing to previous firmware releases, the timing configuration has considerably evolved in version 6.0.

Timing configuration now depends on four sets of **dot-config** variables: common port information, per-port information, sfp information and fiber description.

This is, for explanation’s sake, an example of such items:

```bash
# common port information
PTP_OPT_EXT_PORT_CONFIG_ENABLED=yes

# per-port information
CONFIG_PORT09_IFACE="wri9"
CONFIG_PORT09_FIBER=2
CONFIG_PORT09_INSTANCE_COUNT_1=yes
CONFIG_PORT09_INST01_PROTOCOL_RAW=yes
CONFIG_PORT09_INST01_PROFILE_WR=yes
CONFIG_PORT09_INST01_DESIRADE_STATE_SLAVE=yes
CONFIG_PORT09_INST01_EGRESS_LATENCY=226214
CONFIG_PORT09_INST01_INGRESS_LATENCY=226758

# sfp information
CONFIG_SFP00_PARAMS="pn=AXGE-1254-0531,tx=0,rx=0,wl_txrx=1310+1490"

# fiber information
CONFIG_FIBER02_PARAMS="alpha_1310_1490=2.6787e-04"
```

When making timing calculation for port **wri9**, assuming the auto-detected SFP model is “AXGE-1254-0531”, the software uses configuration in the following way:

- The port has known **tx** (CONFIG_PORT09_INST01_EGRESS_LATENCY) and **rx** (CONFIG_PORT09_INST01_INGRESS_LATENCY) delays (around 226ns); the values depend on
trace length and other hardware-specific details and are determined by the WR calibration procedure. These values are used by the WR PTP daemon for accurate estimation of the transmission link asymmetry.

- The port is configured as WR slave (CONFIG_PORT09_INST01_DESIRADE_STATE_SLAVE) using raw Ethernet (CONFIG_PORT09_INST01_PROTOCOL_RAW) to synchronize using the White-Rabbit protocol (CONFIG_PORT09_INST01_PROFILE_WR) and is deployed using fiber type 2 (CONFIG_PORT09_FIBER) – this number is just a local enumeration of fiber types corresponding to an entry in the fiber information section of the configuration. Most typical WR installations use a single type of fiber, indexed as "0" for every port.

- The SFP transceiver used in this example emits 1310nm light for tx and receives 1490nm light (this is part of the specification of the transceiver, and cannot be auto-detected). Moreover, it has 0 constant delay in both directions (tx/rx), determined by the WR calibration procedure.

- The relative asymmetry of the fiber type being used (type 2 here), when driven with 1310nm and 1490nm wavelengths, is characterized with an alpha parameter of 0.00026787 (i.e. a ratio of 1.00026787). This value depends on both, the fiber type and the wavelength used, and is determined, again, by the WR calibration procedure.

Please note that only one alpha value has to be provided for each fiber type. The alpha in the opposite direction (alpha_1490_1310) is calculated by the software.

### 3.4.1 Common port configuration

The following table lists the configuration items that are common to all port instances of the WR switch.

- **CONFIG_PTP_OPT_EXT_PORT_CONFIG_ENABLED**
  This option enables forcing the PTP state for each port instance using CONFIG_PORTxx_INSTyy_DESIRADE_STATE_* parameters. When set, each port instance can be forces to always stay either in the slave, master or passive state. The PTP Best Master Clock Algorithm (BMCA) is then disabled. This option is only available for a boundary clock (CONFIG_TIME_BC). For more details please refer to the IEEE 1588-2020 (clause 17.6.2)

- **CONFIG_PTP_SLAVE_ONLY**
  A slaveOnly Ordinary Clock utilizes the slaveOnly state machine which does not enable transition to MASTER state. This option is only available if PTP_OPT_EXT_PORT_CONFIG_ENABLED is not used. For more details please refer to the IEEE 1588-2020 (clause 9.2.2.1)

### 3.4.2 Per-port configuration

- **CONFIG_PORTxx_INSTANCE_COUNT_1**
- **CONFIG_PORTxx_INSTANCE_COUNT_0**

Each physical WRs port can be configured to run several PPSi instances. However, only one instance per-port can be active at any given moment. This functionality is required to support several PTP extensions (e.g. wr, HA - High Accuracy profile). The active instance will dependent then of the profile used by the peer on the other side of the link.

WR switch firmware version 6.0 implements only a single, WR extension, therefore the possibility of running several instances on a single port is disabled. The number of port instances must be set to either 1 (CONFIG_PORTxx_INSTANCE_COUNT_1=yes) or 0 (CONFIG_PORTxx_INSTANCE_COUNT_0=yes) if the port is not used. **xx** represents the port number in the range 01 to 18.
Common instance configuration

A set of parameters common to all instances running on the same physical port. The xx value in the parameter name represents the port number.

**CONFIG_PORTxx_IFACE**

Used to define the physical port interface name: "wri[1-18]"

**CONFIG_PORTxx_FIBER**

Identify the fiber type. This is a number (zz) referring to the corresponding `CONFIG_FIBERzz_PARAMS`.

**CONFIG_PORTxx_CONSTANTASYMMETRY**

Defines an additional constant asymmetry (e.g. of a fiber amplifier used in longhaul WR networks) that should be compensated by WR PTP calculations.

Instance information description

All port instance parameters shared the following naming scheme: `CONFIG_PORTxx_INSTyy_pp`, where xx represents the port number, yy the instance number (01 to 02) and pp a key describing the parameter itself.

**CONFIG_PORTxx_INSTyy_PROTOCOL_RAW**

**CONFIG_PORTxx_INSTyy_PROTOCOL_UDP_IPV4**

Network transport layer selection, either raw Ethernet or UDP over IPv4 can be used.

**CONFIG_PORTxx_INSTyy_PROFILE_WR**

**CONFIG_PORTxx_INSTyy_PROFILE_PTP**

Profile selection. `CONFIG_PORTxx_INSTyy_PROFILE_WR` must be set to use the White Rabbit profile and `CONFIG_PORTxx_INSTyy_PROFILE_PTP` for pure PTP profile.

**CONFIG_PORTxx_INSTyy_MECHANISM_E2E**

**CONFIG_PORTxx_INSTyy_MECHANISM_P2P**

PTP delay mechanism, either ‘end-to-end’ (E2E) or ‘peer-to-peer’ (P2P) can be used.

**CONFIG_PORTxx_INSTyy_DESIRADE_STATE_MASTER**

**CONFIG_PORTxx_INSTyy_DESIRADE_STATE_SLAVE**

**CONFIG_PORTxx_INSTyy_DESIRADE_STATE_PASSIVE**

If `CONFIG_PTP_OPT_EXT_PORT_CONFIG_ENABLED` is enabled then the desired PTP state of the instance must be defined as follows:

<table>
<thead>
<tr>
<th>State</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASTER</td>
<td><code>CONFIG_PORTxx_INSTyy_DESIRADE_STATE_MASTER</code></td>
</tr>
<tr>
<td>SLAVE</td>
<td><code>CONFIG_PORTxx_INSTyy_DESIRADE_STATE_SLAVE</code></td>
</tr>
<tr>
<td>PASSIVE</td>
<td><code>CONFIG_PORTxx_INSTyy_DESIRADE_STATE_PASSIVE</code></td>
</tr>
</tbody>
</table>

**CONFIG_PORTxx_INSTyyASYMMETRYCORRECTION_ENABLE**

This option is only accessible when the PTP profile is selected otherwise this option is enabled by default. It is used to force the servo to integrate on its calculation the computation of the delay asymmetry.

**CONFIG_PORTxx_INSTyy_BMODE_AUTO**

**CONFIG_PORTxx_INSTyy_BMODE_MASTER_ONLY**

Indicates the BMCA mode to be used. The choice is available only when `CONFIG_PTP_OPT_EXT_PORT_CONFIG_ENABLED` is disabled. Either the regular PTP BMCA (`BMODE_AUTO`) or PTP masterOnly feature (`BMODE_MASTER_ONLY`) can be used. For more details please refer to the IEEE 1588-2020 (clause 9.2.2.2)
Chapter 3: Configuration of the Device

CONFIG_PORTxx_INSTyy_EGRESS_LATENCY
CONFIG_PORTxx_INSTyy_INGRESS_LATENCY
Defines the reception (CONFIG_PORTxx_INSTyy_INGRESS_LATENCY) and transmission (CONFIG_PORTxx_INSTyy_EGRESS_LATENCY) constant delays expressed in picoseconds.

CONFIG_PORTxx_INSTyy_ANNOUNCE_INTERVAL
The mean time interval between transmissions of successive PTP Announce messages. The value expressed as base 2 logarithm.

CONFIG_PORTxx_INSTyy_ANNOUNCE_RECEIPT_TIMEOUT
The number of announceInterval timeout that must pass without receipt of an Announce message before the occurrence of the PTP event ANNOUNCE_RECEIPT_TIMEOUT_EXPIRES. The value expressed as base 2 logarithm. For more details please refer to IEEE 1588-2020 standard.

CONFIG_PORTxx_INSTyy_SYNC_INTERVAL
The mean time interval between transmission of successive PTP Sync messages, i.e., the sync-interval, when transmitted as multicast messages. The value expressed as base 2 logarithm.

CONFIG_PORTxx_INSTyy_MIN_DELAY_REQ_INTERVAL
The minDelayRequestInterval specifies the minimum permitted mean time interval between successive Delay Req messages. The value expressed as base 2 logarithm. This option is available when 'end-to-end' delay mechanism is selected (CONFIG_PORTxx_INSTyy_MECHANISM_E2E).

CONFIG_PORTxx_INSTyy_MIN_PDELAY_REQ_INTERVAL
The minPDelayRequestInterval specifies the minimum permitted mean time interval between successive Pdelay Req messages. The value expressed as base 2 logarithm. This option is only available when 'peer-to-peer' delay mechanism is selected (CONFIG_PORTxx_INSTyy_MECHANISM_P2P).

CONFIG_PORTxx_INSTyy_MONITOR
Option to decide whether timing-related errors for a given port will be reported through SNMP.

3.4.3 SFP name matching
Each time you plug an SFP transceiver into any of the wrs ports, it has to be matched against dot-config entries specifying timing parameters for the supported transceivers. The matching algorithm reads from the SFP its vendor name (vn), part number (pn) and vendor serial (vs). Then, SFP parameters are compared with values stored in CONFIG_SFPxx_PARAMS dot-config entries:

• The first try is to match a complete set of SFP identifiers (vn, pn and vs).
• If a corresponding entry cannot be found, the match is limited to vn and pn and compared only with those dot-config entries that do not specify any vendor serial.
• If the match still cannot be found, it is limited again, to pn only.

To understand better the operation of SFP matching algorithm, please see below some examples:

• CONFIG_SFP00_PARAMS="vn=Axcen Photonics,pn=AXGE-3454-0531,vs=AX12390009629, tx=0,rx=0,..."
  This entry can be matched only to one SFP transceiver as it specifies full set of parameters, including the unique vendor serial number (vs).
• CONFIG_SFP01_PARAMS="vn=Axcen Photonics,pn=AXGE-3454-0531,tx=0,rx=0,
  wl_txr=1310+1490"
This entry may be matched only to all SFPs with vendor name "Axcen Photonics" and part number "AXGE-3454-0531", with exception to the SFP that was already matched to the previous entry CONFIG_SFP00_PARAMS (with vendor serial defined).

- CONFIG_SFP02_PARAMS="pn=AXGE-3454-0531,tx=0,rx=0,\(\text{wl}_\text{txrx}=1310+1490\)"

This entry will be matched to all SFPs with part number "AXGE-3454-0531", that were not matched by any of the entries listed earlier.

### 3.4.4 Other Deployments

All examples from previous sections match typical WR networks we setup at CERN with a single mono-modal fiber and 1310/1490 nm light wavelengths.

If you are using dual-fiber transceivers, which is acceptable for short links, you use the same wavelength in both directions, over two fibers of the same length. In this case you may omit the \(\text{wl}_\text{txrx}\) parameter in SFP dot-config configuration and the alpha_\text{xx}_\text{xx} parameter in fiber configuration. The missing parameters will cause warning messages to log destination, but are not fatal, and a default alpha of 0 is used.

If you are using a pair of transceivers with different wavelengths, and long fibers, you should provide an appropriate value of alpha, according to laboratory measures of your fiber type. The CONFIG_FIBER\text{xx}_\text{PARAMS} items are parsed as a list of comma-separated assignments, so you can specify multiple wavelength pairs. The accuracy of your value depends on the length of the fiber link. For a 10km fiber (100us round-trip) you need to know alpha up to 1e-7 if you want the related uncertainty to be less than 10ps.

### 3.4.5 Calibration

Calibration of per-port and per-SFP delays as well as alpha is described in the White Rabbit Calibration procedure: [http://www.ohwr.org/projects/white-rabbit/wiki/Calibration](http://www.ohwr.org/projects/white-rabbit/wiki/Calibration).

The delays used in the examples come from values listed in the calibration wiki page, and you should not be surprised by the fact that the transceiver (SFP) specifies the delays as zero. By performing the calibration procedure using this very transceiver type, the whole delay is assigned to the port. Other transceiver types can be calibrated to either positive or negative values, to cope with the difference between them and the default AXGE devices.

### 3.5 VLANs Configuration

VLANs are handled at two levels:

- Per-port configuration of the Endpoint. It allows to:
  - tag ingress frames with VLAN-tag and specific priority
  - specify behaviour depending on whether the frames are tagged or untagged (\text{pmode})
  - override priority before it is mapped into Traffic Class, i.e. translate into Traffic Class different priority than the one received in the VLAN-tag
  - remove VLAN-tag of given VID(s) from egress frames
  - override VID in the VLAN-tag of a priority-tagged frame (VID=0x0)
- Per-VID by configuring the RTU. It allows to:
  - limit the ports to which frames with a given VID are forwarded
  - override priority (Traffic Class) for a given VID
  - drop frames with a given VID

In terms of VLAN-tag, there are four types of VLAN-tags that can extend the Ethernet Frame header:

- \text{none} – tag is not included in the Ethernet Frame
• **priority** – tag that has VID=0x0
• **VLAN** – tag that has VID in the range 0x001 to 0xFFE (1 to 4094)
• **null** – tag that has VID=0xFFF (4095)

The behaviour of each pmode that can be set per-port depends on the type of VLAN-tag in the received frame.

- **Mode ACCESS** (0x0), frames with:
  - *no VLAN-tag*: are admitted, tagged with the values of VID and priority that are configured in pvid and pprio respectively
  - *priority tag*: are admitted, the value of VID in their VLAN-tag is overridden with the value configured in n pvid. This new value of VID is provided to the RTU. If pprio is not -1, the value of priority provided to RTU is overridden with the configured pprio, the value of priority in the VLAN-tag is unchanged.
  - *VLAN tag*: are discarded
  - *null tag*: are discarded

- **Mode TRUNK** (0x1), frames with:
  - *no VLAN-tag*: are discarded
  - *priority tag*: are discarded
  - *VLAN tag*: are admitted; if pprio is not -1, the value of priority provided to RTU is overridden with the configured pprio, the value of priority in the VLAN-tag is unchanged.
  - *null tag*: are discarded

- **Mode DISABLED** (0x2), frames with:
  - *no VLAN-tag*: are admitted. No other configuration is used even if set.
  - *priority tag*: are admitted; if pprio is not -1, the value of priority provided to RTU is overridden with the configured pprio
  - *VLAN tag*: are admitted; if pprio is not -1, the value of priority provided to RTU is overridden with the configured pprio
  - *null tag*: are admitted; if pprio is not -1, the value of priority provided to RTU is overridden with the configured pprio

- **Mode UNQUALIFIED** (0x3), frames with:
  - *no VLAN-tag*: are admitted. No other configuration is used even if set.
  - *priority tag*: are admitted. The value of VID in their VLAN-tag is overridden with the value configured in n pvid. This new value of VID is provided to the RTU. If pprio is not -1, the value of priority provided to RTU is overridden with the configured pprio, the value of priority in the VLAN-tag is unchanged. **Note:** From version v6.0, providing a VID for this mode is supported in the dot-config, "Raw ports configuration" needs to be enabled.
  - *VLAN tag*: are admitted; if pprio is not -1, the value of priority provided to RTU is overridden with the configured pprio
  - *null tag*: discarded.

Modes and their behaviour are summarized in the table below:
Chapter 3: Configuration of the Device 27

The best and recommended way to configure VLANs is to use the `wrs_menuconfig` tool for generating the `dot-config` file. The `dot-config` file is used at startup to set:

- **per-port and per-VLAN configuration** - the `dot-config` is read by the `wrs_vlans` tool at startup
- **PPSi PTP VLAN configuration** - the `ppsi.conf` file is generated at startup from the `dot-config` file, it is then read by PPSi.

For an example configuration using `wrs_menuconfig` and `dot-config`, please see Section 3.5.1.1 [Example VLAN configuration by `dot-config`], page 28.

An alternative for VLAN configuration, especially when experimenting with VLANs, is to use the `wrs_vlans` tool directly and to provide a custom `ppsi.conf`. Beware that this method is more error-prone.

To have synchronization working with VLANs, a proper PTP VID needs to be provided for ports in TRUNK, DISABLED and UNQUALIFIED mode. In the `dot-config` file, it is the `CONFIG_VLANS_PORTxx_PTP_VID` configuration option. For ports in ACCESS mode, the PTP VID is derived from the VID (`CONFIG_VLANS_PORTxx_VID` in the `dot-config` file). As an alternative you can write a custom PPSi configuration file with VLANs specified per-port. You can simply copy the file generated in the WRS filesystem (`/etc/ppsi.conf`) to a central TFTP/HTTP/FTP server where `dot-config` files for your switches are stored and fetched on boot time or permanently store it in the flash (for details, please check the configuration options `CONFIG_PTP_*` in the Section 3.3 [Configuration Items that Apply at Run Time], page 9).

In the PPSi config file, for every VLAN-enabled port you should add the following line:

```
vlan <VID1>,<VID2>,...,<VIDn>
```

where `VID` is a VLAN ID configured on the port.

For an example configuration please see Section 3.5.1.2 [Example VLAN configuration by tools], page 29.

From the firmware v5.0, configuration of VLANs via the `dot-config` file was possible with some limitations/simplifications which made the life of the user easier but prevented some exotic VLAN configurations. From the firmware v6.0, all possible configuration can be set via `dot-config`. By default, the more user-friendly configuration is used (similar to the one in v5.0). To have full control over VLAN configuration, "raw ports configuration" must be enabled.

Another alternative working on pre-v5.0 to set VLANs is to use the web interface. However, as it is in v5.0, the web-interface is not capable to store VLANs configuration into a `dot-config`.

### 3.5.1 Example VLAN configuration

This section describes how to configure VLANs on a switch using the `dot-config` and available command line tools. The description assumes that switch has only these 3 ports.
In this configuration, port 1 is synchronised to an upstream WR device. This device does not need to have any VLAN configuration. Port 1 is in **ACCESS** mode, thus it tags the ingress Ethernet frames. VLAN-tags with VID 1 and priority 4 are added so that frames received at this port belong to VLAN 1. Port 1 also untags the egress frames. In this configuration, only port 1 belongs to VLAN 1, which means that none of the traffic received on port 1 is forwarded to other ports. The only traffic received on port 1 that is not dropped are the PTP messages which are forwarded to the PTP daemon (**PPSi**). Such an arrangement can be useful if the synchronisation is to be propagated through WR network, i.e. between the upstream and this switch, but the data needs to be separated.

The data is forwarded between ports 2 and 3. These ports belong to VLAN 2 (VID=2). Port 3 is in **ACCESS** mode and it tags the ingress frames with VID 2 and priority 7. This could be a port connected to a WR node that is a source of critical traffic. This WR node does not need to support VLANs, however its traffic needs to have the highest precedence in the network.

The traffic from port 3 is forwarded only to port 2. This port is in **TRUNK** mode. It does not untag egress frames which means that the device connected to it (a switch or a node) must be VLAN-aware. Port 2 accepts only frames that are already tagged with the VLAN-tag. Out of the frames received at port 2, only these with VID=2 are forwarded, all the other frames are dropped. The frames with VID=2 are forwarded to PTP daemon and to port 3.

### 3.5.1.1 Example VLAN configuration by dot-config

To configure the switch in the way described in the Section 3.5.1 [Example VLAN configuration], page 27, the *wrs_menuconfig* tool should be used to generate the **dot-config** file. In the *wrs_menuconfig* tool, VLANs needs to be enabled in the VLANs submenu. Then, the Ports and VLANs configurations need to be filled in properly, as can be seen in the figure below.

![Example VLAN configuration](image)

Such generated **dot-config** file will contain the following config options (and much more, of course):

```plaintext
PTP_OPT_EXT_PORT_CONFIG_ENABLED=yes
# Port 1 configuration
CONFIG_PORT01_IFACE="wri1"
CONFIG_PORT01_FIBER=0
CONFIG_PORT01_INSTANCE_COUNT_1=yes
CONFIG_PORT01_INST01_PROTOCOL_RAW=yes
```
Chapter 3: Configuration of the Device

3.5.1.2 Example VLAN configuration by tools

To configure the switch in the way described in the Section 3.5.1 [Example VLAN configuration], page 27, using the wrs_vlans command line tool and custom ppsi.conf file, please perform the following actions:

Clear the current configuration:

```
# wrs_vlans --clear
# wrs_vlans --rvid 0 --del
```

Set ports’ configuration by using the wrs_vlans tool:

```
# wrs_vlans 
   --port 1 --pmode 0 --pprio 4 --pvid 1 --puntag 1 
   --port 2 --pmode 1 --pprio -1 
   --port 3 --pmode 0 --pprio 7 --pvid 2 --puntag 1
```

Set VID configuration by using the wrs_vlans tool:

```
# wrs_vlans 
   --rvid 1 --rpprio 4 --rdrop 0 --rmask 0x0001 
```
For details about `wrs_vlans` please refer to Section 5.2 `wrs_vlans`, page 36.

PPSi configuration that should be placed into `ppsi.conf`:

```plaintext
port wri1-raw
  proto raw
  iface wri1
  desiredState slave
  profile wr
  vlan 1

port wri2-raw
  proto raw
  iface wri2
  desiredState master
  profile wr
  vlan 2

port wri3-raw
  proto raw
  iface wri3
  desiredState master
  profile wr
  vlan 2
```

NOTE: The `ppsi.conf` in `/etc/` folder is automatically generated from the `dot-config` file on startup, thus you will lose your changes if you update directly `ppsi.conf` in `/etc/`. To use custom `ppsi.conf`, you need to specify its path in `dot-config` using the parameter: `CONFIG_PTP_CUSTOM`.

### 3.6 Front panel’s LEDs

There are two LEDs on the front panel describing switch status and two LEDs for each WR port. Each LED can be off, green or orange color, or the combination of both giving yellow. For more details please refer to following sections.

#### 3.6.1 Status LEDs

The status LED is placed together with power indicator LED on the left side of the front panel. The status LED is the right one.

During barebox/kernel boot the status LED is off. When `startup-mb.sh` starts the LED is set to yellow. If the HAL starts successfully then the LED is set to green. If the HAL caught a SIGNAL (error) sent by other process, HAL sets the status LED to orange. When the regular reboot is performed status LED is turned off. In case of a kernel crash the LED remains unchanged.

#### 3.6.2 Ports’ LEDs

Under each WR port there are two LEDs, left and right. The left LED has two functions. First: it blinks `orange` when a particular port is populated with an SFP and the Low Phase Drift Calibration is ongoing; this happens after inserting a fiber and after the WR switch start/reboots. Second: it is on (permanently, not blinking) when a particular port is populated with an SFP and the link is up. Its color depends on the state of PTP instance running on the port:

- **green**: WR slave
- **yellow**: WR master
- **orange**: all other

If there are multiple instances on a particular port the slave state (green) takes precedence over master (orange). Master takes precedence over other states (yellow).
The right LED is green when a particular port is synchronized to the master. When a packet is transmitted or received on a port the right LED blinks orange. If a port is also synced then it blinks yellow instead.

4 Booting with Barebox

After the initial installation, the boot loader offers an interactive menu, where the first entry is selected by default. The menu is a simple ASCII interface on the serial port, and looks like the following:

```
Welcome on WRSv3 Boot Sequence
1: boot from nand (default)
2: boot from TFTP script
3: edit config
4: exit to shell
5: reboot
```

If flashing of the whole system was successful, the first entry will simply work, booting the switch without any network access. Although a DHCP client runs by default after boot, everything will work even if you leave the Ethernet port unconnected or you have no DHCP server when the switch is operational.

If booting from NAND memory fails (for example because you erased the kernel or incurred in other mishaps during development) the menu is re-entered automatically.

The other entries are provided to help developers.

4.1 Description of the menus

The individual menu items perform the following actions:

1: boot from nand (default)
   This entry is selected by default after 5 seconds of inactivity on the serial port. It boots the system from its own NAND memory. This “just works”.

2: boot from TFTP script
   This entry tries to download a barebox script from your TFTP server; if successful it then executes it. Developers are expected to customize the script to support any kind of environment, from customized kernel command-line to NFS-Root environments. See Section 4.2 [Using wrboot], page 32, for details.

3: edit config
   This fires the editor on the configuration file, and saves it to flash when the user is done. This may be useful to change the MAC or IP address of the ARM network port, change the autoboot timeout or change the autoselect choice. Please note that saving save the whole /env file tree, so you can also change the init scripts interactively and have them stored persistently on the flash. Users are not expected to change any configuration, though, as further updates may fail.

4: exit to shell
   By choosing this entry, the user can access the shell-like interface of barebox. The entry is aimed at developers who know what they are going to type.

5: reboot
   This entry is useful to see and log the exact boot messages. Since the serial-USB converter is switch-powered and not USB-powered, you won’t be able to hook at the serial port soon enough after power-on. Actually, the menu startup time is a few seconds long for the very same reason.
4.2 Using wrboot

If you use the `wrboot` script option, you can for example run an NFS-Root system or do whatever customization and testing you want.

**Note:** with the Linux kernel 2.6.39 we suggested use of `root=nfs`, but this convention is no more supported in Linux, please use `root=/dev/nfs`.

The complete filesystem after a successful build is called `images/wrs-image.tar.gz`, and is not included in the release firmware file, because an installed switch runs an initramfs system with a separate `/usr` partition in flash memory.

The `boot from TFTP script` menu entry looks for the script using three different names, from most specific to most generic; the first found is be used. When using the boot script, the WRS first performs a DHCP query, and then uses that IP address to retrieve the script using the following names (the `eth0.ethaddr` is stored by the manufacturer in static storage and retrieved by the boot loader; the `eth0.ipaddr` comes from dhcp):

```
wrboot-eth0.ethaddr $eth0.ipaddr/wrboot
wrboot
```

As an example, the following excerpt shows what I see in my logs when only providing `wrboot`. The last message uses a different IP address because my script forces a static address into the kernel, whereas the initial one was assigned to the boot loader using DHCP.

```
dhcpd: DHCPOFFER on 192.168.16.224 to 02:0b:ad:c0:ff:ee via eth0
atftpd[5623]: Serving wrboot-02:0B:AD:C0:FF:EE to 192.168.16.224:1029
atftpd[5623]: Serving 192.168.16.224/wrboot to 192.168.16.224:1030
atftpd[5623]: Serving wrboot to 192.168.16.224:1031
mountd[21014]: NFS mount of /tftpboot/192.168.16.9 attempted from 192.168.16.9
```

We chose to place the IP-address-based name in a subdirectory because this is the default place where the NFS-Root filesystem is mounted from, as shown in the log excerpt above. So you'll have your `wrboot` in the same place.

**Note:** recent barebox versions require scripts to include a leading `#!/bin/sh`. Examples in `wr-switch-sw` did not include the line until April 2014 included.

The `binaries` subdirectory of `wr-switch-sw` includes a number of known-working wrboot scripts as examples;

- **wrboot-static-ip**
  The script forces a static IP address, server and gateway, and a custom mount point for an NFS-root system.

- **wrboot-dhcp**
  The script preserves the DHCP-assigned address, and runs a custom NFS-root system.

- **wrboot-install**
  This performs an installation, by loading everything to RAM and forcing install mode. Please check comments in the script.

- **wrboot-nand**
  This script is a copy of the default boot script executed by standalone switches. Booting from a script allows changing the kernel command line or anything else it may be useful to developers.
4.3 Creating an NFS-Root Environment for WRS

In order to create an NFS root directory, you should uncompress `wrs-image.tar.gz` that is created at build time in a newly-created empty directory:

```
tar xzf $WRS_OUTPUT_DIR/images/wrs-image.tar.gz
```

If you use a released `wrs-firmware.tar`, however, you’ll have no overall filesystem for the switch, and you should rebuild it from two parts. This is how to create your NFS filesystem from a released `wrs-firmware` file (please adapt for your local pathnames):

```
FW=/tftpboot/wrs-firmware.tar
DIR=/opt/root/wrs-3

mkdir -p $DIR

tar xOf $FW wrs-initramfs.gz | zcat | \n  (cd $DIR && sudo cpio --make-directories --extract)
tar xOf $FW wrs-usr.tar.gz | sudo tar xzf - -C $DIR/usr
```

The above commands extract to `stdout` the two parts of the `wrs` filesystem, to then uncompress them to the proper directories. The first `tar` pipe is less friendly because the `initramfs` is a compressed `cpio` archive, and `cpio` as a command lacks automatic decompression and the `-C` (change directory) option.

5 WRS Command-Line Tools

Tools are build from source files in `userspace/tools` while the scripts are copied directly from `userspace/rootfs_override/wr/bin`.

The following tools and scripts are provided:

- **lm32-vuart**: The tool allows connecting to the virtual UART of the LM32 soft-core CPU. It can be used to observe log messages from the SoftPLL (the same as `FPGA Test` physical UART port available in the back of a WR switch). Use Ctrl+C escape combination to go back to WR Switch Linux shell.

- **load-virtex**
- **load-lm32**

  They load into the FPGA the gateware and the LM32 application. They are used by the init scripts of the Linux system. The LM32 loader can also change variables in the loaded binary, and read or write variables without stopping the running CPU. This is limited to 32-bit integer variables, though. See the commit message for details.

- **mapper**
- **wmapper**

  The former reads from a file using `mmap` (usually you run it on `/dev/mem`) and writes to `stdout`. The latter reads from `stdin` and writes using `mmap`. They are classic tools distributed in the `Linux Device Drivers` examples since 1998.

- **com**

  The program is a simple program for talking with serial ports.

- **wr_phytool**

  A tool to read and write PHY registers in the switch.

- **wr_mon**

  A simple monitor of White Rabbit status. It prints to `stdout` using the standard escape sequences for color output. Please check `wr_mon`’s help (`--help` parameter) for further information. (This is probably the most important diagnostic tool on the switch.)
wrs_pps_control
A tool to manually enable/disable/read status of PPS output. It can also enable/disable/read status of the 50ohm termination for 1-PPS input. Usage:

“wrs_pps_control pps on” enables the PPS output,
“wrs_pps_control pps off” disables the PPS output,
“wrs_pps_control pps read” checks whether PPS output is enabled or disabled.
Switching the output on/off is independent of the PPSi process, but PPSi switches the PPS output back on when a link restart is detected and PPSi comes into ‘TRACK_PHASE’ state. To on/off/read the 50ohm termination for 1-PPS input use \texttt{wrs_pps_control} with a parameter \texttt{50ohm-term-in} followed by \texttt{on}, \texttt{off} or \texttt{read}.

wr_date
The program can read or set the White Rabbit date. Usage:

“\texttt{wr_date set value}” sets an arbitrary date and time as the system time,
“\texttt{wr_date set host}” passes the host time to White Rabbit.
If the file \texttt{/etc/leap-seconds.list} exists, it is used to pass the TAI offset to the kernel, and to consider it in setting White Rabbit time to the current TAI value. The program is meant to prime the White Rabbit counter at boot time, and is run by \texttt{/etc/init.d/wr_date} – this script uses NTP to set host time as a first step, if \texttt{/etc/wr_date.conf} exists and includes a line of the form \texttt{ntpserver 192.168.16.1}. The file \texttt{/etc/wr_date.conf} is created at boot time by the script \texttt{apply_dot-config} if a NTP server is defined (\texttt{CONFIG_NTP_SERVER}).
With “\texttt{wr_date get}” you can read White Rabbit time, and by using \texttt{wr_date get tohost} you can set host time from White Rabbit time. This can be useful in slave switches that are not synced to NTP at boot.

wrs_version
Print information about the software, gateware, hardware version of the WRS. Please check the help message.

shw_ver
A symbolic link to \texttt{wrs_version}, to be compatible with older versions that used this tool name. The name is inconsistent with anything else in the switch, so it was replaced.

wrs_vlans
The tool allows to configure VLAN settings for each port and for the RTU daemon. The \texttt{--help} option lists all configuration items of the tool. For details please refer to the Section 5.2 \texttt{wrs vlans}, page 36.

apply_dot-config
The script is used to apply \texttt{dot-config} settings to the current configuration files. It is run at boot time before any service is started and by the web interface to apply changes in the local \texttt{dot-config}. The \texttt{dot-config} mechanism is documented in Chapter 3 \texttt{Configuration of the Device}, page 7.

assembly_ppsi_conf.sh
The script is used to assemble ppsi configuration file based on information stored in \texttt{dot-config}. This script is called by \texttt{apply_dot-config}.

change_dot-config
This script changes the current \texttt{dot-config} file. It is designed to be the back-end of the web interface, when changing configuration items. The script does nothing to \texttt{apply} the changes, and it only performs editing. It is the responsibility of the caller to ensure the proper service is restarted with the new configuration.
Chapter 5: WRS Command-Line Tools

sdb-read  The tool, copied from the fpga-config-space project, is documented in the next section.

wrs_auxclk  The tool allows to setup the parameters of a clock generated on the clk2 SMC output on the front panel.

wrs_pstats  The tool is used to read various per-port statistics counters from the console. Please note that the same values are also provided through SNMP objects.

wrs_sfp_dump  Dump the content of SFPs internal memory. This tool can read SFP info from HAL's shmem or directly from SFPs via i2c bus. Please note that reading directly via i2c can cause race condition on i2c bus. The tool is not recommended to be used in production. The race condition can cause errors while reading SFPs memory, wrong notification of LEDs, the false insert/remove SFP notifications.

This tool can also write SFPs internal memory. This functionality can be used to fix SFPs reporting i.e. wrong checksum. Use this option with care.
For more details please refer to the tool’s help.

wrs_throttling  The tool is used to control Rx bandwidth throttling of the traffic that goes from WR ports to Linux. It configures the FPGA module with a maximum allowed bandwidth in KB/s. Throttling can be enabled to prevent Linux using 100% of the processing power to receive Ethernet frames coming from WR ports to the CPU. This tool is executed by default at boot time with parameters from the dot-config. For more information, please refer to the Section 3.3 [Configuration Items that Apply at Run Time], page 9.

5.1 sdb-read
[Note: this documentation section comes from the ohwr project called fpga-config-space.]
The sdb-read program can be used to access an sdbfs image stored in a disk file or an FPGA area in physical memory. It works both as ls (to list the files included in the image) and as cat (to print to its own stdout one of the files that live in the binary image).
The program can be used in three ways:
sdb-read [options] <image-file>
  This invocation lists the contents of the image. With -l the listing is long, including more information than the file name.

sdb-read [options] <image-file> <filename>
  When called with two arguments, the program prints to stdout the content of the named file, extracted from the image. Please note that if the file has been over-sized at creation time, the whole allocated data area is printed to standard output.

sdb-read [options] <image-file> <hex-vendor>:<hex-device>
  If the second argument is built as two hex numbers separated by a colon, then the program uses them as vendor-id and device-id to find the file. If more than one file have the same identifiers, the first of them is printed.

The following option flags are supported:
-1
  For listing, use long format. A verbose format will be added later.
Chapter 5: WRS Command-Line Tools

- `e <entrypoint>`
  Specify the offset of the magic number in the image file.

- `m <size>@<addr>`
- `m <addr>+<size>`
  Either form is used to specify a memory range. This is the preferred way to read from a memory-mapped area, like an FPGA memory space. Please note that in general you should not read a “file” in FPGA space, because this would mean read all device registers. The form “<image-file> <filename>” is thus discouraged for in-memory SDB trees (i.e. where <image-file> is /dev/mem).

- `r`
  Register the device with a read method instead of the data pointer. In this way the tool can be used to test the library with either access method. If mmap fails on the file (e.g., it is a non-mappable device), read is used automatically, irrespective of -r.

5.2 wrs_vlans

The `wrs_vlans` shell tool can be used to setup VLANs in the switch. The configuration can be read from the dot-config file pointed by -f or --file parameter (for dot-config details please check Section 3.2 [Configuration Items that Apply at Build Time], page 9). Additionally, the configuration can be specified using parameters below. The details of VLANs configuration are discussed in Section 3.5 [VLANs Configuration], page 25. The `wrs_vlans` configuration is divided into two parts:

```
wrs_vlans --port <port number or range> [options]
  Per-port Endpoint VLAN configuration. Used to set VID and priority for ingress frames tagging, egress untagging and port mode. For port modes please refer to the Section 3.5 [VLANs Configuration], page 25.

wrs_vlans --rvid <vid> [options]
  Per-VLAN configuration of the Routing Table Unit, used to configure port mask describing which ports belong to a given VLAN. RTU uses this information to be able to forward incoming frames only to ports inside the VLAN.
```

Both per-port Endpoint and per-VLAN RTU configuration has to be performed in order to have a full VLAN setup on a WR Switch.

For per-port configuration, multiple ways of specifying ports are supported:

```
wrs_vlans --port 1 [options]
  Selected configuration will be applied only to port 1.

wrs_vlans --port 1,3,4 [options]
  Selected configuration will be applied to ports 1,3 and 4.

wrs_vlans --port 5-8 [options]
  Selected configuration will be applied to port range 5 to 8.

wrs_vlans --port 5-8,15 [options]
  Selected configuration will be applied to port range 5 to 8 and port 15.
```

To configure each Endpoint the following options may be used:

```
--pmode <0..3>
  Sets VLAN mode for a port (0 – ACCESS, 1 – TRUNK, 2 – DISABLED, 3 – UNQUALIFIED)

--pvid <0..4094>
  Sets VLAN id for tagging ingress frames.
```
--pprio <-1|0..7>
Sets priority for tagging ingress frames. -1 disables priority overwrite.

--puntag <0|1>
Disables or enables egress untagging. By default, if you configure ingress tagging, all VIDs are untagged on egress.

To configure VLANs in RTU the tool has to be used with parameter specifying VLAN id to be set up and then the list of configuration options:

wrs_vlans --rvid <vid> [options]
Possible RTU VLAN configuration options:

--rmask <0x0..0x3ffff>
Mask defines which physical ports of the WRS belong to a configured VLAN.

--rvid <0..4094>
Assigns filtering ID fid to the configured VLAN. Multiple VLANs can be configured to have the same fid. This way they create a group where learning a new MAC address in one VLAN implies learning this MAC in the rest of VLANs in the group as well.

--rprio <-1|0..7>
Forces 802.1q priority override for VLAN. Setting prio to -1, cancels the override.

--rdrop <1/0>
Forces (if set to 1) or disables (if set to 0) frames drop for the configured VLAN.

--del
Deletes selected VLAN from the RTU configuration.

In addition to that wrs_vlans can be also used to display and clear current VLAN configuration of the switch:

wrs_vlans --plist
Current Port VLAN configuration

wrs_vlans --list
Current RTU VLAN configuration.

wrs_vlans --clear
Clear configuration. Add a default rule to pass all traffic between ports.

wrs_vlans tool can be called multiple times to make a set of per-port and per-VLAN configurations. However, it is also possible to configure multiple ports/VLANs in one go. For example to configure ports 1,2,3,6 to VLAN 5 and ports 4,5 to VLAN 6 with tagging ingress frames one could call wrs_vlans with these parameters:

wrs_vlans --port 1-3,6 --pmode 0 --pvid 5 --port 4,5 --pmode 0 --pvid 6 \
   --rvid 5 --rmask 0x27 --rvid 6 --rmask 0x18

5.3 wrs_auxclk

The wrs_auxclk shell tool can be used to configure parameters of a clock signal generated on the clk2 SMC connector on the front panel.

Note: you need to have WRS hardware at least in version 3.4 to have clk2 output.

By default wrs_auxclk is called by init scripts to generate 10MHz clock signal with 50% duty cycle. This configuration can be modified by using various options:

--freq <f>
Desired frequency of the generated clock signal in MHz. Available range from 4kHz to 250MHz.
--duty <frac>
Desired duty cycle given as a fraction (e.g. 0.5, 0.4).

--cshift <csh>
Coarse shift (granularity 2ns) of the generated clock signal. This parameter can be used to get desired delay relation between generated 1-PPS and clk2. The delay between 1-PPS and clk2 is constant for a given bitstream but may be different for various hardware versions and re-synthesized gateware. Therefore it should be measured and adjusted only once for given hardware and gateware version.

--sigdel <steps>
Clock signal generated from the FPGA is cleaned by a discrete flip-flop. It may happen that generated aux clock is in phase with the flip-flop clock. In that case it is visible on the oscilloscope that clk2 clock is jittering by 4ns. The --sigdel parameter allows to add a precise delay to the FPGA-generated clock to avoid such jitter. This delay is specified in steps, where each step is around 150ps. This value, same as the --cshift parameter, is constant for a given bitstream so should be verified only once.

--ppshift <steps>
If one needs to precisely align 1-PPS output with the clk2 aux clock using --cshift parameter is not enough as it has 4ns granularity. In that case --ppshift lets you shift 1-PPS output by a configured number of 150ps steps. However, please have in mind that 1-PPS output is used as a reference for WR calibration procedure. Therefore, once this parameter is modified, the device should be re-calibrated. Otherwise, 1-PPS output will be shifted from the WR timescale by <steps>*150ps.

5.4 wrs_pstats
The wrs_pstats shell tool can be used to read per-port statistics counters from FPGA. When it is executed without any parameters all displayed values are counted from the moment the tool was started. In case you’re interested in the values gathered from the start of WR switch, you can use -s option. The following counters for each port are reported:

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:Tu-run</td>
<td>Number of TX underrun errors</td>
</tr>
<tr>
<td>1:Ro-run</td>
<td>Number of RX overrun errors</td>
</tr>
<tr>
<td>2:Riv-cd</td>
<td>Number of invalid 8B10B characters received</td>
</tr>
<tr>
<td>3:Rsyn-l</td>
<td>Number of RX link synchronization lost events</td>
</tr>
<tr>
<td>4:Rpause</td>
<td>Number of received pause frames</td>
</tr>
<tr>
<td>5:Rpf-dp</td>
<td>Number of received frames dropped by the Packet Filter</td>
</tr>
<tr>
<td>6:Rpcs-e</td>
<td>Number of PCS errors during frame reception</td>
</tr>
<tr>
<td>7:Rgiant</td>
<td>Number of received giant frames</td>
</tr>
<tr>
<td>8:Rrunt</td>
<td>Number of received runt frames (smaller than 64 bytes)</td>
</tr>
<tr>
<td>9:Rcrc_e</td>
<td>Number of CRC errors in received frames</td>
</tr>
<tr>
<td>10-17:Rpcl_0-7</td>
<td>Number of received frames qualified by Packet Filter to classes 0 to 7</td>
</tr>
<tr>
<td>18:Tframe</td>
<td>Number of transmitted frames</td>
</tr>
<tr>
<td>19:Rframe</td>
<td>Number of received frames</td>
</tr>
<tr>
<td>20:Rrtu_f</td>
<td>Number of RX frames dropped due to RTU full</td>
</tr>
<tr>
<td>21-28:Rpri_0-7</td>
<td>Number of received 802.1Q frames with priorities 0 to 7</td>
</tr>
<tr>
<td>29:RTUreq</td>
<td>Number of RTU requests</td>
</tr>
<tr>
<td>30:RTUresp</td>
<td>Number of RTU responses</td>
</tr>
<tr>
<td>31:RTUdrop</td>
<td>Number of frames dropped by the RTU</td>
</tr>
<tr>
<td>32:RTUhp</td>
<td>Number of high priority frames routed by RTU</td>
</tr>
</tbody>
</table>
33:RTUf\-f  Number of forwarded frames matched by RTU fast match engine
34:RTUn\-f  Number of not forwarded frames matched by RTU fast match engine
35:RTUfst  Number of RTU fast match decisions
36:RTUful  Number of RTU full match decisions
37:RTUfwd  Total number of frames forwarded by RTU
39:NIC_Tx  Number of frames sent from WR Switch ARM to that port

6 SNMP Support

The White Rabbit Switch supports SNMP. The default read-only “community” name is private, but you can change it from the Kconfig interface before building. The default read-write community is private.

The switch supports all the standard information through the net-snmp installation. The additional, switch-specific information are in the “enterprise.96.100” subtree, where 96 is CERN and 100 is White Rabbit. The associated MIB is in the directory userspace/snmpd, where related source files live as well.

There is currently no support for traps.

6.1 The WRS MIB

This section contain a summary of the WR-SWITCH-MIB, for details please refer to the document White Rabbit Switch: Failures and Diagnostics. Objects from 96.100.2 to 96.100.5 are obsolete, they were used during early implementation of the WRS snmp.

96.100.1

This is a simple scalar as a test. It is an integer value that is incremented each time you access it. It can be used to test basic functionality.

96.100.6

wrsStatus – MIB’s branch with collective statuses of the entire switch.

96.100.7

wrsExpertStatus – Branch with detailed statuses of switch subsystems.

The easiest way to retrieve the values is using snmpwalk, telling it to access our MIB file in order to use symbolic names. Assuming wrs is the DNS name for your White Rabbit Switch and WR_SWITCH_SW is an environment variable pointing to this package:

```
snmpwalk -c public -v 2c wrs \n-m +${WR_SWITCH_SW}/userspace/snmpd/WR-SWITCH-MIB.txt \n1.3.6.1.4.1.96.100
```

Using SNMP version 1 instead of 2c is fine as well, but you won’t receive the 64-bit values for slave/tracking information.

The output you will get, is something like the following:

```
WR-SWITCH-MIB::wrsScalar.0 = INTEGER: 1
WR-SWITCH-MIB::wrsMainSystemStatus.0 = INTEGER: ok(1)
WR-SWITCH-MIB::wrsOSStatus.0 = INTEGER: ok(1)
WR-SWITCH-MIB::wrsTimingStatus.0 = INTEGER: ok(1)
[...]
WR-SWITCH-MIB::wrsConfigSource.0 = INTEGER: remote(4)
WR-SWITCH-MIB::wrsConfigSourceUrl.0 = STRING: tftp://192.168.1.1/config-192.168.1.10
WR-SWITCH-MIB::wrsBootInfoReadout.0 = INTEGER: ok(1)
WR-SWITCH-MIB::wrsBootLoadFPGA.0 = INTEGER: ok(1)
WR-SWITCH-MIB::wrsBootLoadLMS2.0 = INTEGER: ok(1)
```
Another example is to print all objects exported by the switch.

```
snmpwalk -c public -v 2c wrs -m all \
-M ${WRS_OUTPUT_DIR}/build/buildroot-2016.02/output/build/netsnmp-5.7.3/mibs/\n:$\{(WRS_SWITCH_SW)}/userspace/snmpd/ \
1
```

### 6.2 show-pstats

To visualize all port statistics in a single window, this package includes the simple tool `userspace/snmpd/show-pstats`. It is a Tk script, so you need to install tk8.5 or any other version.

The script receives one or more host names (or IP addresses) on the command line. They must refer to a switch (or switches) or the program fails like this:

```
laptopo% ./show-pstats morgana
Error in snmpwalk for host morgana
No log handling enabled - using stderr logging
.1.3.6.1.4.1.96.100.2.1.: Unknown Object Identifier (Sub-id not found: enterprises -> )
```

If everything goes well, you’ll get a window like the following one:
Command `snmptable` can also be used to get similar results:

```
snmptable -Cw 80 -c public -v 2c 192.168.1.10 -m all \n-M $WRS_OUTPUT_DIR/build/buildroot-2016.02/output/build/netsnmp-5.7.3/mibs/\n:userspace/snmpd/ WR-SWITCH-MIB::wrsPstatsHCTable
```

Output is in text form and looks like:

```
SNMP table: WR-SWITCH-MIB::wrsPstatsHCTable

<table>
<thead>
<tr>
<th>wrsPstatsHCPortName</th>
<th>wrsPstatsHCTXUnderrun</th>
<th>wrsPstatsHCRXOverrun</th>
</tr>
</thead>
<tbody>
<tr>
<td>wri1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri17</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wri18</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

SNMP table WR-SWITCH-MIB::wrsPstatsHCTable, part 2

<table>
<thead>
<tr>
<th>wrsPstatsHCRXInvalidCode</th>
<th>wrsPstatsHCRXSyncLost</th>
<th>wrsPstatsHCRXPauseFrames</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

(...)

Unfortunately output due to the number of counters is very wide. Number of characters per line can be limited by switch Cw, in example was set to 80.

**Appendix A Bugs and Troubleshooting**

Even if the package is already released and used in production, some details can be suboptimal, while some procedures may be tricky and need more explanation. We are collecting all those issues in our project pages. Please visit:

- Issues for WR Switch SW project: [http://www.ohwr.org/project/wr-switch-sw/issues](http://www.ohwr.org/project/wr-switch-sw/issues)
- Issues for WR Switch HDL project: [http://www.ohwr.org/project/wr-switch-hdl/issues](http://www.ohwr.org/project/wr-switch-hdl/issues)

If you have any problem with this firmware and you don’t find help in the above links, feel free to reach us on the *white-rabbit-dev* mailing list.