ELECTRONICS & DEFENSE

WR LEN

User Manual

1. Introduction

1.1. About White Rabbit technology

White Rabbit (WR) is an extension to Ethernet network with **accurate synchronization** and **Gigabit data transfer capability**. It has been conceived to fulfill the following goals:

- Time Precision: The WR technology provides a common clock for physical layer in the entire network, allowing synchronization at subnanosecond level with picoseconds precision.
- Scalability: The WR network is designed to be highly scalable with up to thousands of nodes. It also intends to be as modular as possible and compatible with non-WR devices.
- Distance Range: Taking into account the size and ranges of the majority industrial and scientific facilities, the WR network specifications have been chosen to support distance ranges up to several tens of kilometers between nodes using fiber cables.

1.2. About the WR-LEN

The **White Rabbit Lite Embedded Node** (WR-LEN) is a solution that brings subnanosecond accuracy to **WR daisy chains** where each WR-LEN board receives synchronization from a higher level of the hierarchy and provides it to a lower level. Within a time and phase transfer over a 1G optical fibers Ethernet, the WR-LEN provides dynamic calibration over distances up to tens of kilometers and its scalability goes beyond. Moreover, the WR-LEN versatility allows the user to transfer time and frequency in Master-Slave and Master-Master modes.

The following synchronization signals are available within the WR-LEN board:

- 1 PPS input/output signal, synchronized to the White Rabbit network.
- 10 MHz output signal, synchronized to the White Rabbit network.
- 10 MHz input signal. In this case the WR-LEN is configured as Grand Master and could be connected to an external master clock reference.

1.3. About this document

This document describes the essential information to enable the WR-LEN user to deploy a basic setup. The WR-LEN provides the same level of accuracy to other WR compliant devices. Section 2 describes the system set up including the main components of the WR-LENs, the material you will need for their interconnection and the measurement instruments. Section 3: System Configuration deals with the principal WR Core Shell. Please take your time to read this manual since it provides essential information on the usage of your WR-LEN.

2. System Setup

This section describes the WR-LEN setup to enable the user to quickly and easily start to use the board. It also includes the main hardware elements required and their connections.

2.1. What do you need?

In order to test the WR-LEN in the basic daisy-chain configuration you will need:

- Oscilloscope with at least 150 MHz bandwidth (500 MHz is recommended).
- Any PC/workstation that runs under Ubuntu OS.
- 2x WR-LEN boards.
- 2x 5V & 1.2A DC Power Supply.
- 2x SFPs LC.
    - 1x AXGE-1254-0531 (blue).
    - 1x AXGE-3454-0531 (purple).
- 1x LC-LC Optical Fiber (2 m).
- 2x Mini-USB (B) cables.
- 2x SMA-BNC cables.

2.2. The WR-LEN.

Safran offers two models of WR-LEN; one with a Double Micro D9 connector and another without. If you require this connector, inform your sales representative.

2.2.1. WR-LEN

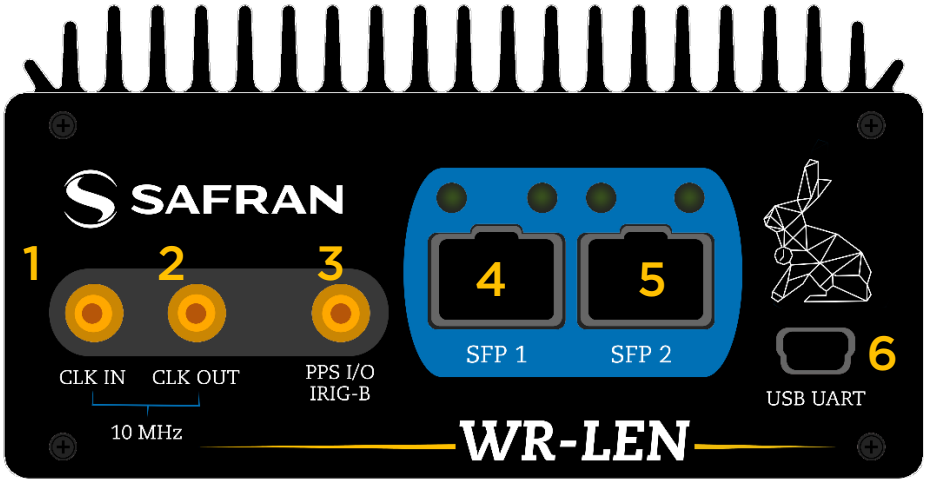


Figure 1: WR-LEN front view

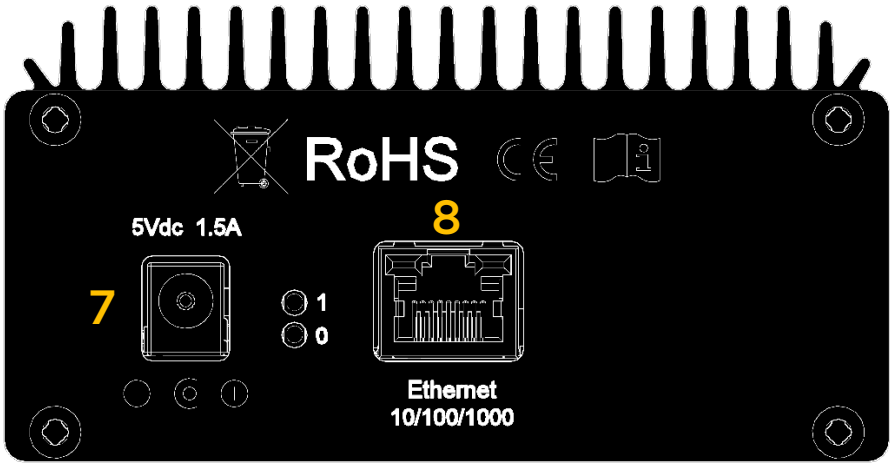


Figure 2: WR-LEN rear view

2.2.2. WR-LEN-2uDB9

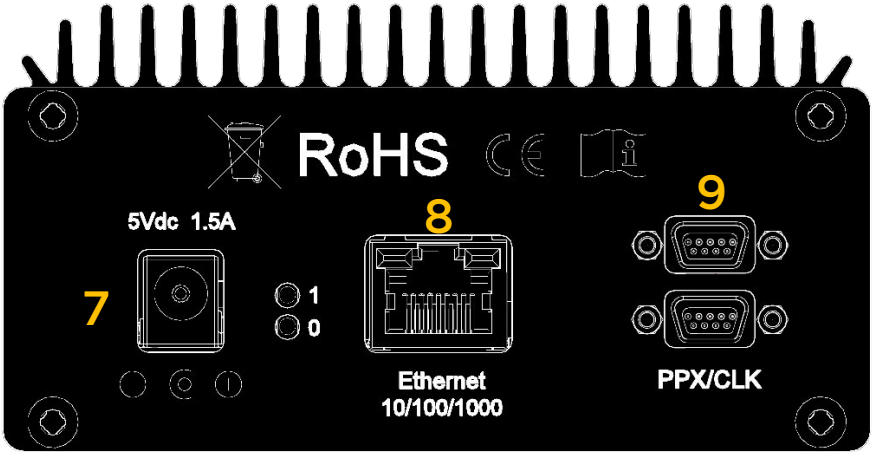


Figure 3: WR-LEN-2uDB9 rear view

Figures 1, 2, & 3 depict the main clock signals, connectors, and interfaces of the WR-LEN solution. A brief description of the listed components is provided below.

1. **SMA connector** to provide to the WR-LEN an external clock reference. Please notice that this clock input is essential to configure the WR-LEN as Grand Master.
2. **SMA connector** to provide 10 MHz output clock disciplined by the WR Core.
3. **SMA connector** has two functionalities depending on the board configuration.
    - **Grand Master:** The signal works as input; a Pulse Per Second (PPS) signal is received from an external reference as well as the 10 MHz at (1).
    - **Output signal:** The signal works as output. In this configuration a PPS signal synchronized with WR is displayed.
4. **SFP port 1**. Note that its default role is slave.
5. **SFP port 2**. Note that its default role is master.
6. **Mini-USB (B) connector** to establish a connection to the UART of the LM32 soft processor.
7. Board **power supply** (+5 V).
8. **10/100/100 Ethernet connector**.
9. **PPX/CLK connectors**, more information in section 5.1. Contact us to get this functionality.

Figure 4 shows the connection diagram between the OEM WR-LEN board and the connectors WR-LEN board.

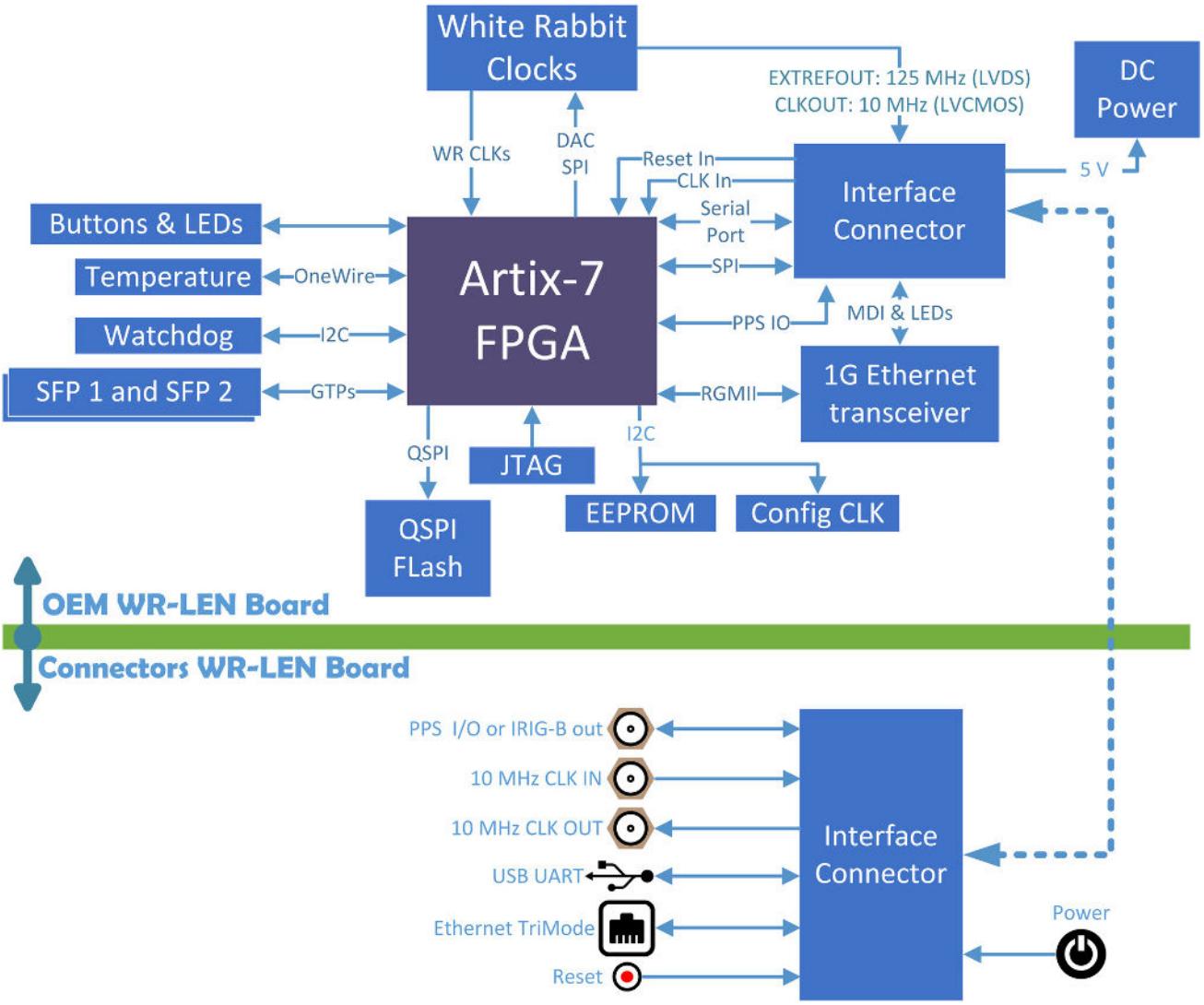


Figure 4: Connection diagram of the WR-LEN

2.3. Components interconnection

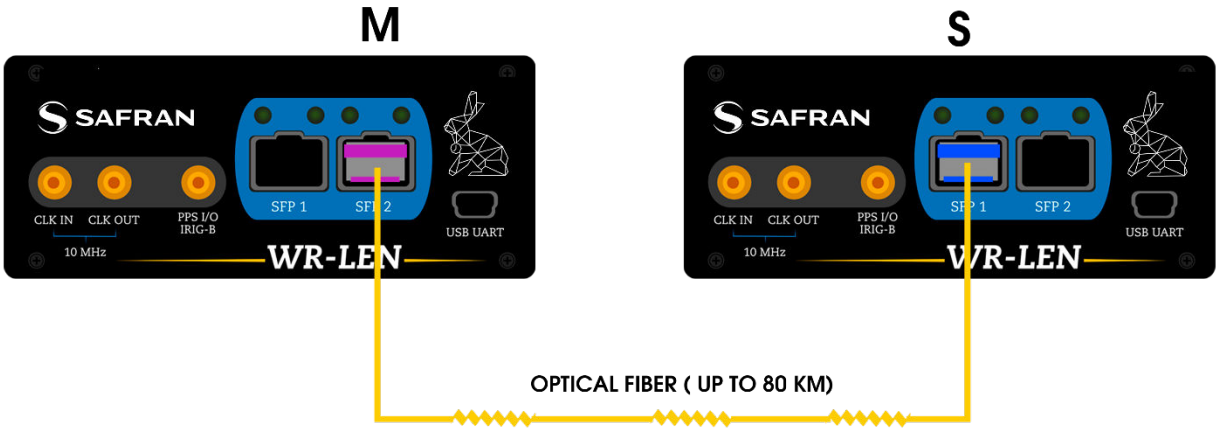


Figure 5: Basic daisy chain setup with two WR-LENs

Figure 5 shows an example of the basic setup using by the default WR-LEN configuration. This scheme will allow you to measure the level of synchronization via the 10 MHz signals. To synchronize the WR-LENs you have to link them with the above-mentioned SFP transceivers and fiber cables. It is really important to use the proper SFP at the right port, i.e., the blue SFP at slave port and the purple one at the master side. To set up the basic scheme, perform the following steps:

- Firstly, connect the SFP transceiver and fiber cable to synchronize the WR-LENs (for more info please check out sfps-info). Please notice that by default the WR-LEN is configured as slave at port 1 and as master at port 2 (Figure 2). However, the initial configuration can be modified though the GUI command. Once the connection is done you should observe a green LED ON above the SFP. This LED is called link LED, while the adjacent one is the activity LED.
- Feed the WR-LENs with the power supplies +5V connectors.
- Wait a minute (more or less) to the slave device reaches the synchronization.
- Connect the WR-LENs to your PC through the mini-USB cables.
- Plug the oscilloscope through the SMA-BNC cables to the WR-LEN's SMA connectors. You can compare the PPS outputs as well as the 10MHz output clocks.
- Check their synchronization levels via the coaxial cables on the oscilloscope. The signal should be subnano synchronized.

3. System Configuration

The previous section showed how to create the basic framework of the WR-LEN. The main hardware components and their connection were illustrated. However, as the WR-LEN is designed to let the WR-LEN user to test other schemes by changing the default configuration parameters, in this section the more advanced features of the WR-LEN will be cover.

3.1. Different ways to access to the WR-LEN nodes.

There are two different ways to access to the WR-LEN nodes to configure them:

- The WR PTP Core Dual Port (WRPC-2P) Shell Commands.
- The Virtual UART Shell (please refer to Appendix B).

3.1.1. Opening a connection with the WR-LEN USB UART

The WR-LEN contains many interfaces to provide access to the internal commands for controlling the WRPC. Using the frontal mini-USB connection is the easiest way to connect it. Thus, in order to enable it, connect the frontal mini-USB to an USB port of the PC. Then, you can use any program to open a serial communication, for example, **minicom**[1]:

```
##To connect the WR-LEN terminal
>$: sudo minicom -D /dev/ttyUSB0 -b 115200
```

Please take in account that the device port may change depending on the connected serial devices to your PC.

When multiple WR-LEN devices are employed in a network, this method may not be practical. To facilitate the management of multiple WR-LEN devices in a network, a virtual UART connection has been developed. The link between the software and the device is established inside the WR

[1] minicom can be installed in Ubuntu with `sudo apt-get install minicom`

network, so any WR-LEN connected to this network will be accessible through this interface. Also, you can connect the software directly to the WR-LEN using a cooper SFP.

To enable the connection to the WR-LEN, an IP must be previously assigned. This could be achieved using BOOTP or introducing the IP manually with the **ip**[2] command (through another interface, such as serial communication).

In order to run the virtual UART, a software tool is provided. This tool, that supports Ubuntu Linux, can be executed opening a new terminal and writing the following:

```
## To launch the virtual uart software
vuart-shell <ip>
```

In both cases, you must specify the IP of the device that you want to connect.

Regardless of the interface chosen to connect to the WR-LEN, the list of commands available are the same (described in Appendix A).

3.1.2. Testing the WRPC shell

The building and running process of the WRPC-2P are widely explained at the wrc and wrc-shell. Please, take a look to this manual for a deeper understanding.

You may want to find out more about your WR-LEN version. For this purpose, you can use the **ver** command.

```
##To print the version of the WR-LEN that is running.
wrc# ver
```

If everything is OK something like this will be displayed in the wrc terminal:

```
WR Core build: 5b232ae (unsupported developer build)
Built on Sep 9 2015, 16:43:58
Built for 103 kB RAM, stack is 2048 bytes
1-wire ROM ID (DS18B20U) : 0x1f000006
EEPROM FRU info is valid:
FRU VENDOR : 7S
FRU DEVICE : WRLEN
FRU SERIAL : 7SWRLENv1.0-S3_000
FRU PARTNUM : 7SWRLENv1.0-S3
FRU FID : 2015-07-17 13:22:53.837576
```

You may as well get information about the devices connected to the Wishbone bus inside the WRPC. To do so, the **sdb** command is available to print a SDB memory map

```
## To print the devices connected to the Wishbone bus inside WRPC
wrc# sdb
```

or, set the verbosity. Please see the PPSI manual about the meaning of the digits (a good try is verbose 1111).

```
verbose <digits>
```

Other commands are defined to get and set the WRPC IPv4 address, for example the **ip get** and **ip set** commands.

```
## To print the IPv4 address
wrc# ip get
```

[2] Please see *ip* command syntax in Appendix A for more information

```
## To set the IPv4 address of the WRPC-2P
wrc# ip set wr0/1
```

3.1.3. Writing EEPROM and Calibration

As you may have observed, WRPC-2P starts in WR Slave mode and its calibration values have been pre-established. However, the initial parameters can be modified (we encourage you to stop the daemon before making calibration adjustments).

```
## To stop the PTP daemon
wrc# ptp stop
```

To do this you should start creating an empty SFP database and then adding two Axcen transceivers with deltaTx, deltaRx and alpha parameters associated with them.

```
## To erase the SFP database
wrc# sfp erase

## To store the calibration parameters for SFP to the database in the internal
EEPROM.
wrc# sfp add <SFP_ID> wr0/1 <deltaTx> <deltaRx> <alpha>:
## For example
wrc# sfp add AXGE-1254-0531 wr0 46407 167843 73622176
wrc# sfp add AXGE-3454-0531 wr1 46407 167843 -73622176
```

Finally, the content of the SFP database can be verified executing the **sfp show shell** command.

```
## Checking the SFP database
wrc# sfp show
```

In order to detect the SFP transceiver plugged into the SPEC board, read its parameters from the newly created SFP database and force the calibration to be executed by typing the following commands:

```
## To stop WR PTP daemon.
wrc# ptp stop

## To print the ID of currently used sfp transceiver (plugged into the connector)
wrc# sfp detect

##  To get the calibration parameters from database for currently used SFP
      transceiver (sfp detect must be executed before match).
wrc# sfp match

## To execute the t24p calibration procedure and stores its result to EEPROM (in
      WR Slave mode).
wrc# calibration wr0/1 force
```

As the t2/t4 phase transition measures are stored into the EEPROM, the calibration procedure does not need to be repeated every time the Core starts; the calibration function should be executed just once for every new FPGA firmware.

Please take into account that the calibration force should executed only once per port wr-calibration.

3.2. The WRPC-2P operation mode.

The **mode** command will inform you about the WR node status. To set the channel mode type **mode** in the WRC terminal.

```
##To display the WR-LEN operation mode
wrc# mode <option>
```

where the following options apply:

- **master**: both of the ports are masters.
- **slave_wr0**: port 1 is slave and port 2 is master. This is the default mode.
- **slave_wr1**: port 1 is master and port 2 is slave.
- **gm**: Grand master mode, to use this mode the WR-LEN must be fed with the 10 MHz and 1 PPS external signals.

Please notice that you can check your WR-LENs operation mode just typing **mode** afterwards.

As you may have noticed, changing the mode stops the PTP daemon but it does not restart automatically. After performing a mode change, **ptp start** is required to start WR PTP daemon in the new mode.

```
## Start the daemon manually
wrc# ptp start
```

3.3. Configuring the WRPC-2P initial script.

To initialize the WRPC-2P software to the required state you can write your own init script to the EEPROM. This script will be executed by the WRPC-2P software each time it comes back from the reset state. It should read the detected SFP parameters and t2/t4 phase transition value from the EEPROM, configure the mode of operation to WR Slave, and start the PTP daemon.

```
## To clean the initialization script in the internal EEPROM.
wrc# init erase

## To add a shell command at the end of initialization script.
wrc# init add <cmd>

## To print all commands from the script stored in the internal EEPROM.
wrc# init show

## To launch the init script.
wrc# init boot
```

3.4. The PPS/10MHz

In order to setup the WR in Grand Master you first need to connect the **10MHz In** and **PPS In** SMA inputs to an external reference such as a GPS, Atomic Clock, etc. These signals must meet the following electrical requirements:

- PPS input: 50Ω, LVTTL or TTL levels
- 10 MHz input: 50Ω, TTL/LVTTL/sine ([1 -- 5]V RMS)
- t_{setup} (PPS-to-10MHz) > 20 ns (see Figure 6)

The 10 MHz is used to produce the internal frequency of WR (62.5MHz/125MHz) and the PPS is used only to ensure the alignment of the edges of both clocks at inter-second boundary.

The actual inter-second boundary is the 10 MHz rising edge after the rising edge of the PPS pulse (see Figure 6).

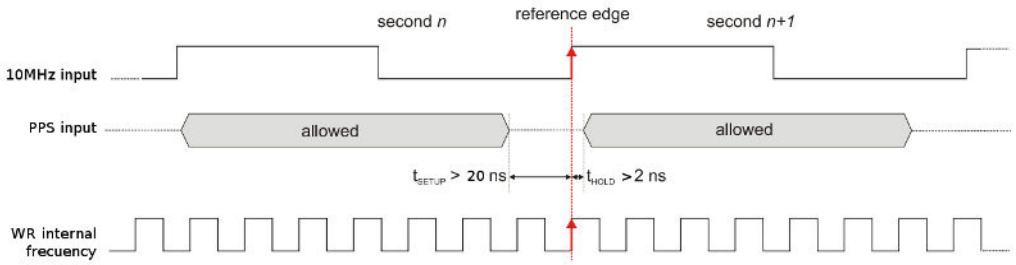


Figure 6: External reference requirements

Note: PPS input is sampled only once at boot (when the PLL is locking), then it will not be used at all.

Warning: The 10 MHz input must not "jump" (i.e. have temporary period different than 100 ns). Some GPS receivers were reported to produce such 10 MHz with such artifacts – these devices cannot be used with WR.

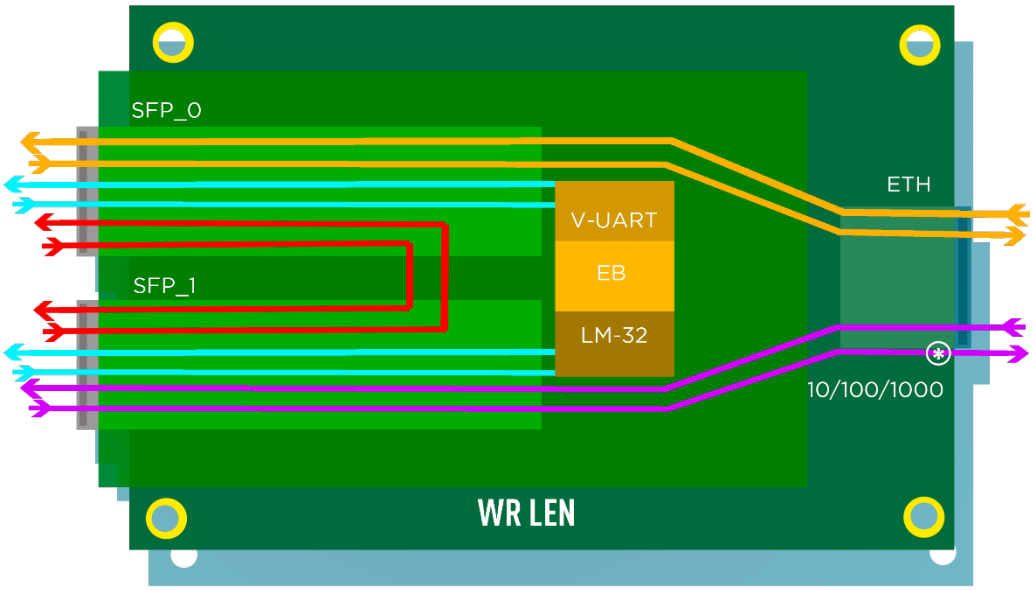
4. WR-LEN packet switching

Beyond the WR feature, the WR-LEN is able to forward regular ethernet packets between its three ports. Consequently, the WR-LEN can work as a standard L2 network device, though, there are a couple of constraints which must be obeyed in order to ensure the correct operation of the device. In this section we are going to break down these restrictions and also providing several user cases which illustrate the most common setups.

In the first place, it comes to mention the list of the WR-LEN ethernet ports; **Two front SF ports** and one b**ack RJ45 ethernet port**. Mainly, the differences between them are:

- Back Ethernet port:
  - Can autonegotiate three speeds (10/100/1000).
  - No WR.
  - Neither LM32 nor Etherbone connection.
  - Can handle only one host forwarding so we discourage you to connect it inside networks.
- Front SFP ports:
  - Just compatible with the optical standard 1000BASE-BX10.
  - There is a small routing table working on them. That means the ability to learn routes so, unlike the RJ45 port, the SFP ports can be plugged into networks.
  - WR support.
  - Connection to the LM32. That means ping reply ability.
  - Connection to Etherbone module (EB). That means remote flashing and v-uart connection.

Besides the feature provided by the ports, the switching functionality refers to the internal connection between the three ports and the WR-LEN ability of forwarding ethernet packets between them. Figure 7 depicts the connections:

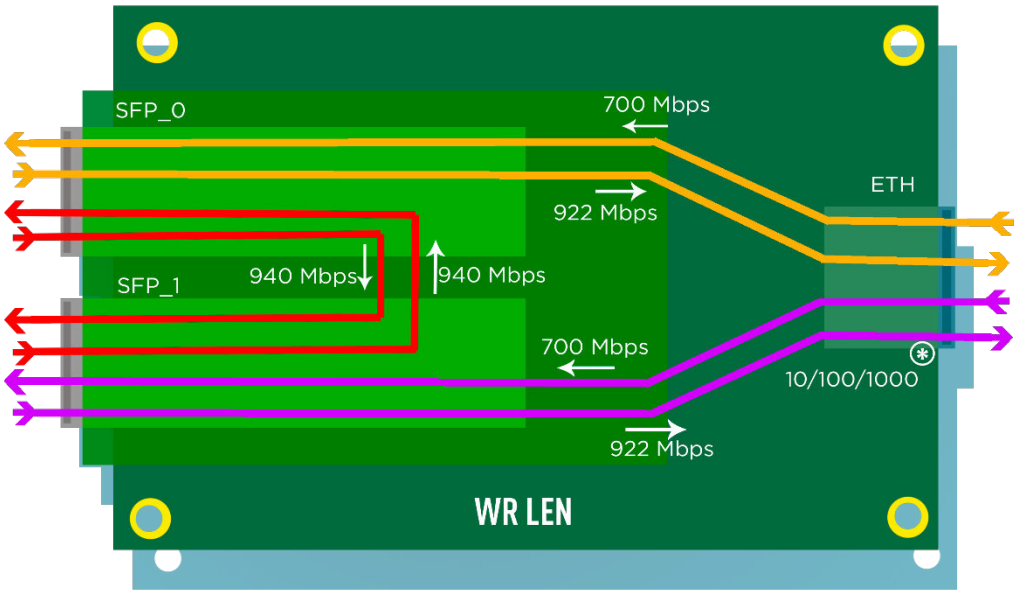


* No WR:

* The ETH port just allows to forward traffic from / to one host thus, we dicourage you to connect it into the networks.

Figure 7: WR-LEN internal switching connections

Some tests have been made to determinate the maximum switching speed allowed by the WR-LEN between its different ports.



* No WR:

* The ETH port just allows to forward traffic from / to one host thus, we dicourage you to connect it into the networks.

Figure 8: WR-LEN switching max. speed

4.1.User cases, recommendations & restrictions.

There are a couple of important remarks to mention in the last figure. Reading them carefully because the correct WR-LEN usage depends on their understanding.

1. **The EB (Etherbone) module is only reachable from the SFP ports**. The EB is in charge of flashing the device so if you want to flash the WR-LEN node, the connection must be done by one of the SFP ports. The flashing procedure is detailed in Appendix D.
2. **The V-UART module is only reachable from the SFP ports**. The remote connection by the v-uart was explained at subsection 3.1. The user must know that the v-uart connection must take place through the SFP ports. Figure 9: EB and V-UART connection shows how the devices must be connected in case that the user wants to open a v-uart connection or remotely flash the node.

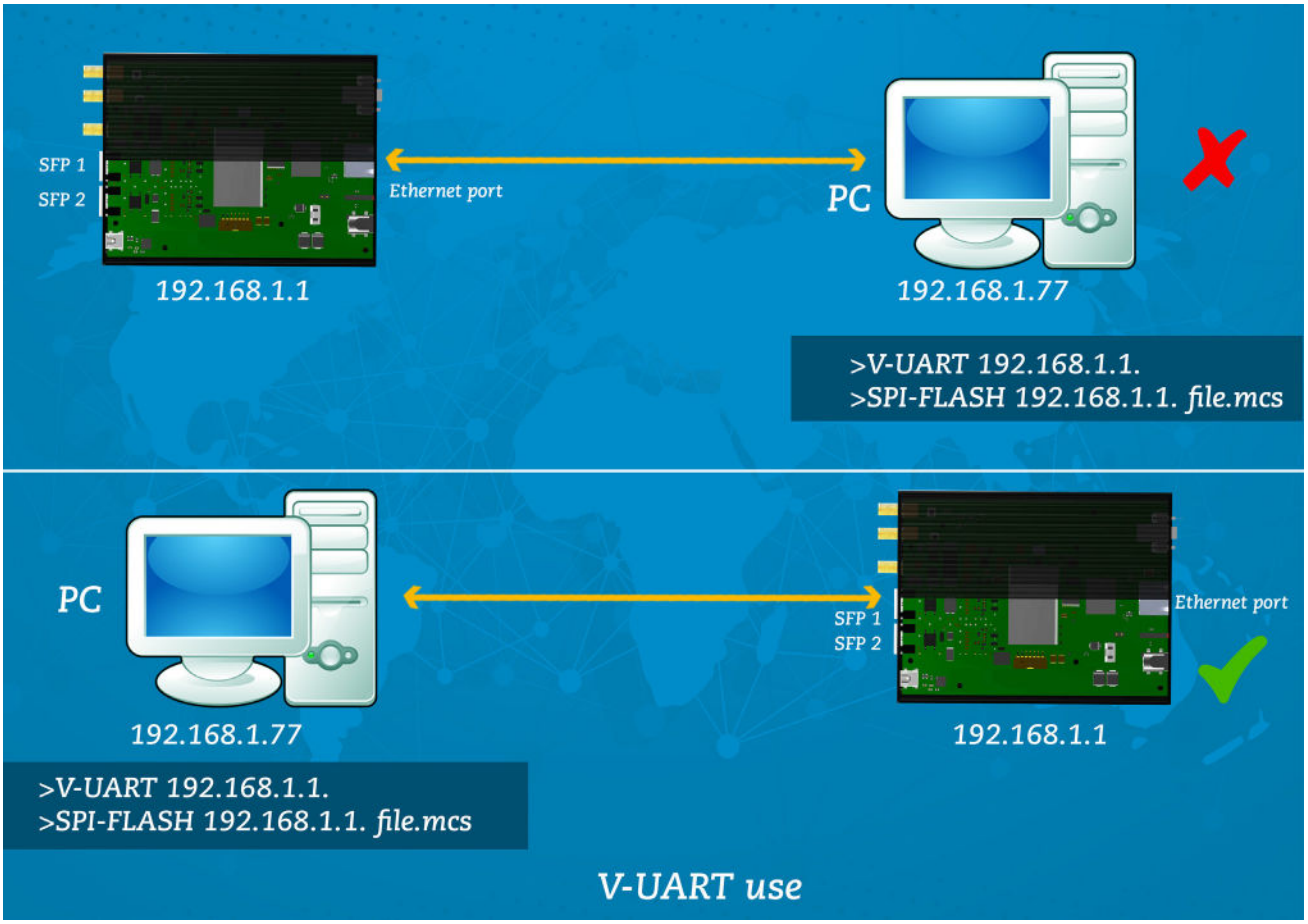


Figure 9: EB and V-UART connection

Additionally, if you have two WR-LENs and want to flash them, or open a V-UART connection, there is a possibility without using a **RJ45 to SFP** converter. Refer to the following image (Figure 10):

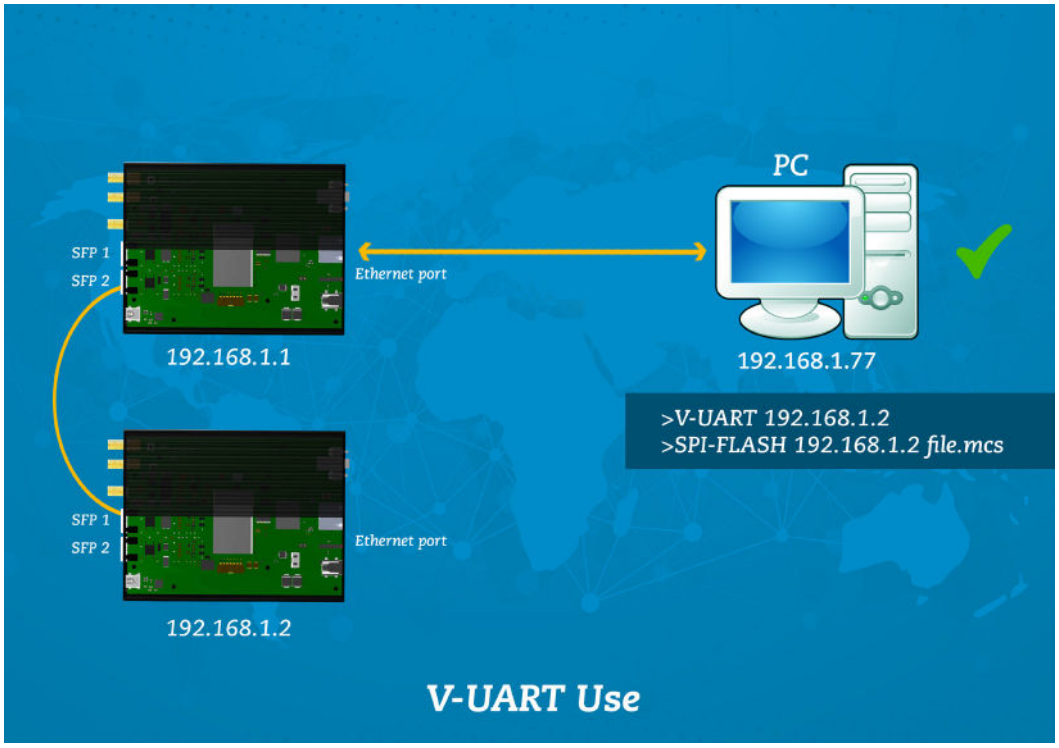


Figure 10: EB and V-UART connection recommendation

3. The WR-LEN LM32 processor is only reachable from the SFP ports. That means that the ping reply ability is just available by the connections made through the SFP ports.
4. The RJ45 port is not able to deal with more than one device. Connecting other switching devices as routers can cause wrong packet forwarding in the WR-LEN. Only final nodes, one ethernet port devices, are allowed to be connected in the WR-LEN RJ45 port. This restriction is detailed in Figure 11.

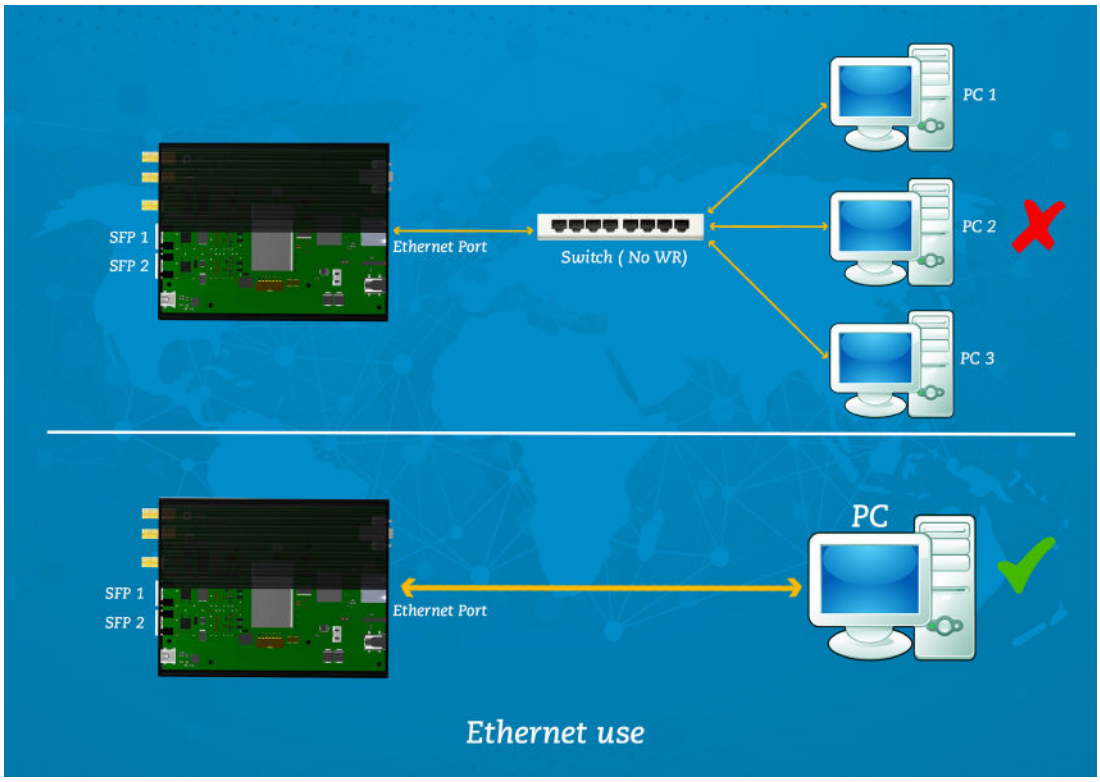


Figure 11: Back Ethernet use

5. Use the **WR network to connect different speed devices through the WR-LEN back ethernet port**. This point is rather a recommendation than restriction.

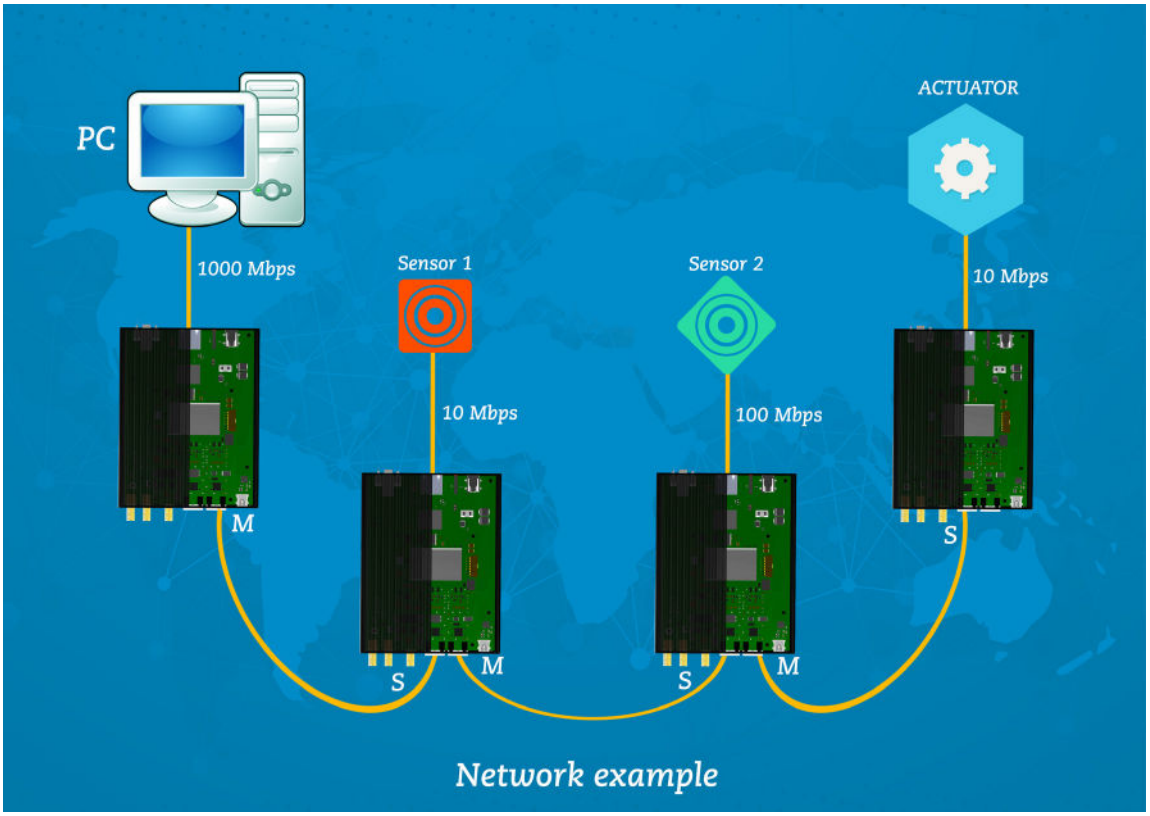


Figure 12: WR-LEN network example

4.2. WR-LEN switching management by WRC Shell.

To check the correct RJ45 working, there is a couple of wrc shell messages which are printed each time that is detected and also a command.

```
## When a new device is plugged into the back RJ45 port
wrc# ETH: Link up.1000Base-T

## When the device is removed from the back RJ45 port
wrc# ETH: Link down
```

Furthermore, when a packet is received through the back RJ45 port, the WR-LEN learns its origin MAC and how to reach it. That the reason why the RJ45 is not able to deal with more than one device. Just one entry is allowed through the back port.

```
## This is an example of back port MAC learning
wrc# Neighbor MAC discovered: c4:e9:84:4:78:4f
```

Unlike the back port and its unique MAC, the front SFP ports are able to learn five address per port. The WR-LEN uses these MACs to forwarding packets by the working of its Routing Table Unit (RTU). In case that the user wants to check the RTU content, there is a command to show it:

```
## the "eth" command dumps the content of the whole WR-LEN RTU.
wrc# eth
Neighbor MAC address: c4:e9:84:4:78:4f

EP0 entries: 1

0: MAC: 8:0:30:31:ab:e6 Age: 60 EP0

EP1 entries: 0
```

5. Running the Core

Restarting the WR PTP Core can be done by reprogramming the LM32 software or pressing the soft reset button.

The shell also contains the monitoring functions that you can use to check the WR synchronization status:

```
## To start GUI WRPC-2P monitor ([Esc] to exit back to shell).
wrc# gui
## To print one line log message with info about the WRC state.
wrc# stat
```

After running the WRC Shell in gui mode you may observe the following

```
TAI Time: Thu, Jan 1, 1970, 00:04:21

WR-LEN mode : WRC_SLAVE_PORT1

Link status:

wr0 : Link up (RX: 893, TX: 274), mode: WR Slave Locked Calibrated
IPv4: BOOTP running

wr1 : Link down

Servo state: TRACK_PHASE

Phase tracking: ON
Synchronization source: wr0

Timing parameters:

Round-trip time (mu): 919375 ps
Master-slave delay: 467098 ps
Master PHY delays: TX: 204147 ps, RX: 216898 ps
Slave PHY delays: TX: 204891 ps, RX: 232457 ps
Total link asymmetry: -14821 ps
Cable rtt delay: 60982 ps
Clock offset: 13 ps
Phase setpoint: 2082 ps
Skew: 13 ps
Manual phase adjustment: 0 ps
Update counter: 81
```

5.1. Adjusted delay Fan-out

Safran offers two models of LEN, one with a Double Micro D9 connector and another without. If you require this connector, contact with us and we will add the connector.

Starting with version 3.7, the WR-LEN includes the distribution of PPS and clock (10 MHz) signals from the stacked DB9 connector in the back side of the box. Both signals can be calibrated to adjust its phase respect to the signals in the front part of the box. Figure 13 depicts the output signals for the DB9 connector.

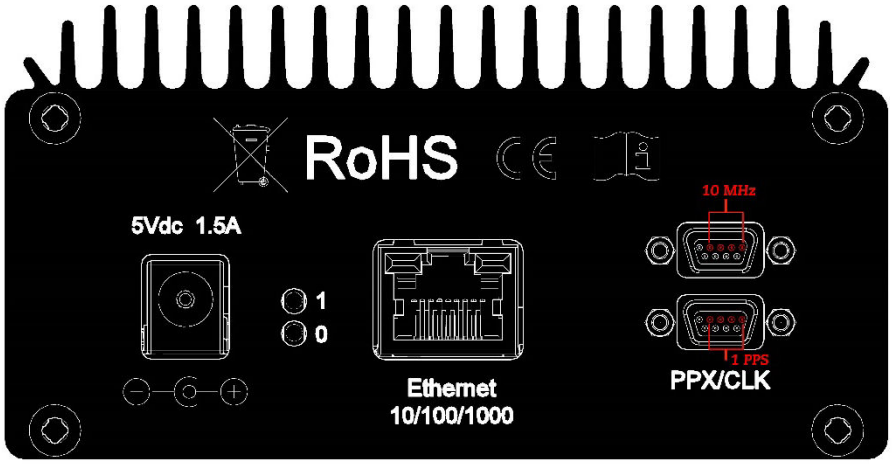


Figure 13: Stacked DB9 connector output signals

This functionality is configured using the **trig** command of the WRPC-2P shell. The syntax of this command is as follows:

```
trig pps_clk [on/off] [ -c -pps <coarse> <fine> -clk <coarse> <fine> [save] ]
```

The first argument is used to enable/disable the generation of the back signals. When using the command without -c option it will generate the signals with the saved delay offsets.

By default, the back signals (PPS & 10 MHz clock) are compensated respect to the frontal outputs to be synchronized sub-nanosecond. The user can modify this offsets using -c option.

Then, the user must provide delays for both signals. The first value indicates the coarse delay in steps of 16 ns. Any value introduced will be rounded below to the nearest multiple of 16. The second value is an hexadecimal integer indicating the fine delay. The operation range of that delay is 9,3 ns (approx.) in 35 ps steps.

To save the introduced delays in the internal memory, the key save can be used at the end of the command.

In addition to that functionality, the WR-LEN could be used to output trigger signals in the DB9 connector. To see more information about that topic see the Section 5.1.

6. References

- WR The White Rabbit Solution
- OHWR Link to the official White Rabbit site
- WR-LEN Info on our website about the WR-LEN
- sfps-info Info online about the White Rabbit-compliant SFPs
- wrc White Rabbit PTP Core User's Manual
- wrc-shell List of wrc shell commands.
- PPSI PPSi Manual.
- wr-calibration Info online about the WR calibration.
- wrpc.pdf White Rabbit PTP Core User Manual (v2.1).
- Safran Navigation & Timing Safran Navigation & Timing website.
- Vivado Official Vivado website.
- Platform-Cable-USB-II Xilinx Programming cable.
- JTAG-HS2-Programming-Cable Digilent Programming cable.
- Remote FPGA Flash Remote update method.

7. Appendix A: List of Commands

A list of the most commonly used commands is provided below:

```
## To stop the PTP daemon
wrc# ptp stop
## To set the WR-LEN mode
wrc# mode <option>
## To restart the PTP daemon again.
wrc# ptp start
## To start the WRC monitor. [Esc] to stop it.
wrc# gui
```

```
## To force the manually run of the spll init() function to initialize Soft PLL.
wrc# pll init <mode> <ref_channel> <align_pps>
## To check if SoftPLL is locked for the channel.
wrc# pll cl <channel>
## To set phase shift for the channel.
wrc# pll sps <channel> <picoseconds>
## To get current and target phase shift for the channel.
wrc# pll gps <channel>:
## To start SoftPLL for the channel.
wrc# pll start <channel>
## To stop SoftPLL for the channel.
wrc# pll stop <channel>
## To set the dac voltage value.
wrc# pll sdac <index> <val>
## To get the dac voltage value.
wrc# pll gdac <index>
```

```
## To clean the initialization script in FMC EEPROM
wrc# init erase
## To add shell command at the end of initialization script
wrc# init add <cmd>
## To print all commands from the script stored in EEPROM
wrc# init show
##  To  execute  the  script  stored  in  FMC  EEPROM  (the  same  action  is  done
automatically when
WRPC-2P starts after resetting LM32)
wrc# init boot
```

```
## To print which version of WRPC is running.
wrc# ver
## To set PPSi verbosity. See the PPSi manual about the meaning of the digits
      (hint: verbose 1111 is a good first bet too see how the ptp system is
      working).
wrc# verbose <digits>
```

To get and set the MAC addresses into the ports use the mac command:

```
## To print the MAC address
wrc# mac get
## To re-generate the MAC address from 1-wire digital thermometer or EEPROM
wrc# mac getp
## To set the MAC address
```

```
wrc# mac set <wr0/wr1> <MAC>
## To set the MAC address to the 1-wire EEPROM (if available)
wrc# mac get <wr0/wr1>
```

The **calibration** command tries to read t2/4 phase transition from EEPROM, if not found it will run the calibration procedure.

```
## Calibrate the WR-LEN
wrc# calibration
## Start calibration procedure and store the result to EEPROM
wrc# calibration <port> <force>
```

The **stat** command prints log messages and the bitslide values.

```
## To print one line log message with info about the WRC state
wrc# stat
## To print a log message for each second ([Esc] to exit back to shell).
wrc# stat cont
## To print log message for each time specified in nanoseconds param (Esc to exit
      back to shell).
wrc# stat cont <milliseconds>
## To print bitslide value at the wr0 port needed by calibration procedure.
wrc# stat bts
```

```
## To print the current time from WRPC-2P
wrc# time
## To print the current time in a raw format (seconds nanoseconds)
wrc# time raw
##  To set only the seconds part of WRPC-2P time (useful for setting time in
      GrandMaster mode, when nanoseconds counter is aligned to external 1-PPS and
      10MHz)
wrc# time setsec <sec>
## To set only the snanoseconds part of the WRPC-2P
wrc# time setnsec <nsec>
```

```
## To print the ID of currently used SFP transceiver
wrc# sfp detect
## To cleans the SFP database stored in FMC EEPROM
wrc# sfp erase
## To store the calibration parameters for the SFP to the database in FMC EEPROM
wrc# add <ID> <deltaTx> <deltaRx> <alpha>
## To print all SFP transceivers stored in database
wrc# sfp show
## To get calibration parameters from database for currently used SFP transceiver
      (it requires running _sfp detect_ first)
wrc# sfp match
## To print irigb state.
wrc# irigb
## To turn on/off irigb.
wrc# irigb on/off
## To print irigb time.
wrc# irigb print
```

```
## Prints a list of available commands.
wrc# help
## To print one line log message with info about the WRC state.
wrc# stat
## To print the devices connected to the Wishbone bus inside WRPC.
wrc# sdb
## Print the RTU (routing table unit) and shows the node connected to the
ethernet port
wrc# eth
## Shows the current IP adress.
wrc# ip get
## Sets the The IP adress which is valid for both ports.
wrc# ip set <ip>
```

```
## Enable/disable the generation of the back signals.
trig pps_clk [on/off] [ -c -pps <coarse> <fine> -clk <coarse> <fine> [save] ]
```

8. Appendix B: WR-LEN Virtual UART User Manual

8.1. Summary

This Appendix deals with the use of the Virtual UART Shell tool to connect with WR-LEN devices using the Etherbone protocol.

The WR-LEN's main control interface uses the serial communication via a USB cable to a PC. This is the most desirable way to monitor a WR-LEN, but in some cases is not possible to establish the serial connection. An example of this situation is a WR network with multiple nodes spread over multiple locations.

To solve this problem, a tool has been developed by Safran. It provides access to devices in a WR network to monitor devices over this network using the Etherbone protocol.

The WR-LEN devices from version 2.0 and forward support the Etherbone protocol. The following sections explain how to setup the system to run the shell interface for sending/receiving commands to the device and some issues that may occur.

8.1.1. System Setup

The version 1.0 of the Virtual UART shell tool only supports GNU/Linux systems. This manual covers the setup in Ubuntu systems.

8.1.2. Ubuntu OS Setup

The principal requirement is Python 2.7 with the usual modules which are included in a basic Ubuntu installation, with the following dependencies: curl, pip2, python2, get-pip.py, python-wxgtk3.0 & python-wxgtk3.0-dev. , and the libraries: ptsexcept.py and ptslogger.py. This makes the tool ready to be run on any Linux based OS. The official use of this tool is only supported for Ubuntu 20.04 LTS (32 bit and 64 bit versions).

8.1.3. Initial Device Setup

In order to establish a connection to a WR-LEN device using the Virtual UART shell, it must have an associated IP address.

By now, the only way to associate an IP address (without using the serial connection) is setting up a BOOTP server.

The setting up of a BOOTP server is not included in this manual. However, much information is available on the web. You can search for details about how to install and configuring the DNSmasq server on Ubuntu.

To configure the BOOTP server, you will need to know the MAC address of the WR-LEN device you want to assign an IP. To do this, please write in the configuration file */etc/dnsmasq.conf* to associate the IP 192.168.7.2 to a WR-LEN with the MAC 08:00:30:ee:9f:4a:

```
dhcp-host=08:00:30:ee:9f:4a,zen2p0,192.168.7.2,0m
```

In the example above, zen2p0 is an ethernet PCI Express card with the 192.168.7.1 IP address (sudo ifconfig zen2p0 192.168.7.1) connected with the LEN. The LEN uses a SFP RJ45 to allow this connection.

When the WR-LEN is powered up and connected to a WR network, it will send a BOOTP package over the network. If the BOOTP server is active, it will respond to the BOOTP petition with an IP for the device.

An example of configuration file for DNSmasq DHCP server is included at the end of this Appendix. Within the file it can be found the DHCP server configuration carried out for a network interface called *eno1* with an associated IP: 192.168.7.254. In addition, the BOOTP configuration is included in that example. You can check out how an IP (IP: 192.168.7.44) is associated to a WR-LEN's MAC address (08:00:30:ee:9f:4a).

8.1.4. Install the Virtual UART shell tool

The Virtual UART shell tool contains several Python libraries and tools developed by Safran.

To receive the necessary material, please contact our technical support.

Once the repository is in your local machine, open a terminal emulator and go to the directory just created. To start the tool type:

```
$ ./Install.sh
$ ./vuart.py -h
```

This will print the help information in the terminal.

8.2. Shell User Manual

This section discusses the use of the shell tool. The main purpose of the shell is to provide an easy access to the WR command interface over Ethernet. Thus, this tool mainly acts as a bridge to pass WRPC commands to the device and retrieve the results. In addition, it includes some extra functionalities to automatically monitor actions in the device.

The shell contains two operation modes. One is used to pass/receive commands to the WR-LEN while the other one allows the access to some extra functionalities of the tool. The first operation mode accepts the commands listed in Appendix A: List of Commands. Please notice that the commands starting with "_" are used to get into the special uses of the shell.

An interactive shell session can be started through the shell tool. Moreover, it also includes an unattended mode in which an input file containing WRPC commands. In this mode, the shell will execute each of these commands and close the connection when finished.

8.2.1. Interactive shell

To start an interactive session, you should specify the WR-LEN's IP as input parameter:

```
$ ./vuart.py <IP>
```

The **wrc#** prompt will appear as when you connect using the serial link with the WR-LEN. At this point, you could introduce any of the WRPC commands (Appendix A). A drawback with old firmware versions that disables some of the WRPC commands exists.

Please check the disabled commands by typing:

```
wrc# _ver
```

If your WR-LEN includes a firmware version which disables some of the functionalities that you need, search for a new firmware version and flash it. We encourage you to use firmware versions from October 29, 2015 onward.

There are two WRPC commands that should be carefully examined: **gui** and **stat cont**. These commands will refresh the output periodically to update the WR Core status. After using these commands press *ctrl-c* to exit and get back to the shell. We discourage you to use **gui** for a long time. This command consumes a lot of bandwidth and the same information can be obtained from stat command, which needs much less bandwidth.

8.2.2. Special shell's functionalities

The **_help** command allows you to know the special functions provided by the tool:

```
wrc# _help
Special commands:

· _help : Prints this help.
· _ver : Prints useful info about board version.
· _exit : Exits the shell.
· _load <filein> [<fileout>]: Loads an input file with WRPC commands.
· _refresh <value> : Change refresh interval. The value must be greater than 1.5
      secs.
Any other commands will be passed directly to the WR-LEN device.
```

You can then see a more detailed explanation:

- **_help** prints help information.
- **_ver** tells the user which WRPC commands are supported by the firmware version used in the WR-LEN.
- **_refresh** changes the refresh rate. It is expressed in seconds. Values lower than 1.5 are not allowed. We recommend to set a value greater than 5 seconds because, depending of the topology of your network, with high refresh rates some etherbone packets may be lost occasioning the down of the connection.
- **_load** [] allow the user to specify a text file with a series of WRPC commands that will be executed in batch mode. Also you could introduce a *fileout* name to save the output of the executed commands. After executing the last command of the input file, the control will be returned to the interactive shell. If you want to run a input file and close maybe it's best for you call the *vshell* with the flag *-i*. See the next section to run the tool in batch mode.
- **_exit** exits the shell.

Also, you can use the *-i* option to execute a script file. This file is composed by one WRPC command on each line. No special format or extension file name is needed. If you desire, you can save the output of the executed commands in a file using the *-o* option. Call the shell with *-h* option to show the help information.

8.2.3. Batch mode

Besides an interactive mode, sometimes the user wants to monitor the network devices for a long time. To allow the user making easy monitor scripts, a special batch mode is included in the tool. You can call the shell with -h option to print the allowed input options:

```
$ ./vuart.py -h
usage: vuart.py [-h] [--input INPUT] [--output OUTPUT] IP

VUART shell for WR-LEN

positional arguments:
 IP IP of the device

optional arguments:
 -h, --help show this help message and exit
 --input INPUT, -i INPUT
            Execute an input script of WRPC commands
 --output OUTPUT, -o OUTPUT
            Save the output of an input script to a file
```

Following the keyword *vshell* you must specify the IP. Next you can add more options for the batch mode:

- *-i* option specifies the path to a file that contains WRPC commands. The file's content is a sequence of commands, each one in a newline. The gui and stat cont commands will be omitted by this operation mode. An example of input file could be:
  sfp detect sfp show sfp match ptp start
- *-o* is used to create a file where the output from the executed commands will be saved.

You may include a call to the vshell tool inside a cron script or similar for save periodically the status of the network devices, using as input file one with a stat command inside of it.

8.3. Common issues

This section tries to solve some of the most common problems you might find when using the *vshell* tool.

8.3.1. Connection issues

If you obtain the following message when starting the tool:

> $ ./vuart.py The desired device cannot be connected, retrying... The desired device cannot be connected, retrying... The desired device cannot be connected, retrying... Error: Failed connection with the WR-LEN () See the manual for more details.

...this message indicates problems connecting with the WR-LEN via IP address. It may be caused by multiple scenarios:

1. Your WR-LEN has lost the associated IP address. This could occur if you have associated the IP of the WR-LEN by hand (using the **ip set** command). This method for associating IP addresses is not good enough if you need a robust network. We advise you to set up a BOOTP server that automatically associates IP addresses to the connected devices in the network.
2. The link is lost. This situation is indicated by the following messages:
   ```
   wrc# ip
   The connection seems to be lost. Retrying...
   Error:Connection with the WR-LEN is lost
   ```

```
See the manual for more details
```

To check if you can access to the WR-LEN by the IP try:

```
$ ping <IP>
```

If the sent packets by *ping* utility are lost, check the physical link with the WR-LEN. This may happen when the WR-LEN is suddenly disconnected from the network; for instance, a power failure or by scenario 1 above.

3. When running in some of the interactive commands (**gui** or **stat cont**), if you set a high refresh rate, some packets may be lost, occasioning the exit of the command. The connection delays of a network are unpredictable, so depending of the network load, the Etherbone packets could be lost. For a dependable monitor system we recommend not using **gui** command. Instead, use **stat** or **stat cont** with a refresh rate higher than 5 seconds.

8.3.2. Execution issues

Ethbone issues:

```
$ ./vuart.py <IP>
Exception AttributeError: "'EthBone' object has no attribute 'device'"...
```

This message occurs when the tool can't load the dynamic library it needs to handle the Etherbone protocol. Go to *lib* folder:

```
$ cd <path to vuart_shell>/vuart_shell/py7slib/lib/precompiled
```

Check that *libetherbone* for your architecture is there. If not, you need to copy it from the install package. You can also compile your own library by downloading the *etherbone-core* repo.

8.4. Dnsmasq.conf example

Here you can find an example of configuration file for the DNSmasq server. The configuration displayed could be appended to the bottom of the file. It includes the set up for a DHCP server and BOOTP.

```
# Configuration file for dnsmasq.
#

...

# Include another lot of configuration options.
#conf-file=/etc/dnsmasq.more.conf
#conf-dir=/etc/dnsmasq.d

#######################################################3
## specified interfaces.
interface=eno1
## force to listening only to specified if.
bind-interfaces
## Dhcp range. Pay attention to the IPs, this is only an example.
dhcp-range=192.168.7.2,192.168.7.99,12h
## Enable dnsmasq's built-in TFTP server.
enable-tftp
## Set the root directory for files available via FTP.
dhcp-boot=pxe.0,tfpservname,192.168.7.10

# Assign IP to WR-LEN
dhcp-host=08:00:30:ee:b3:e7,len0p0,192.168.7.44,0m
```

9. Appendix C: Advanced WR-LEN triggers

This functionality is only available under request. It needs an extra hardware soldered. Contact us at info.spain@nav-timing.safrangroup.com for further information.

The OEM-LEN can be programmed to trigger signals at specific programmed times. 5 channels are available for this effect accessible through Molex Searay connector (Figure 14). The channel 0 is driven to the pin d1, channel 1 to d2 until channel 4 that is driven to pin d5.

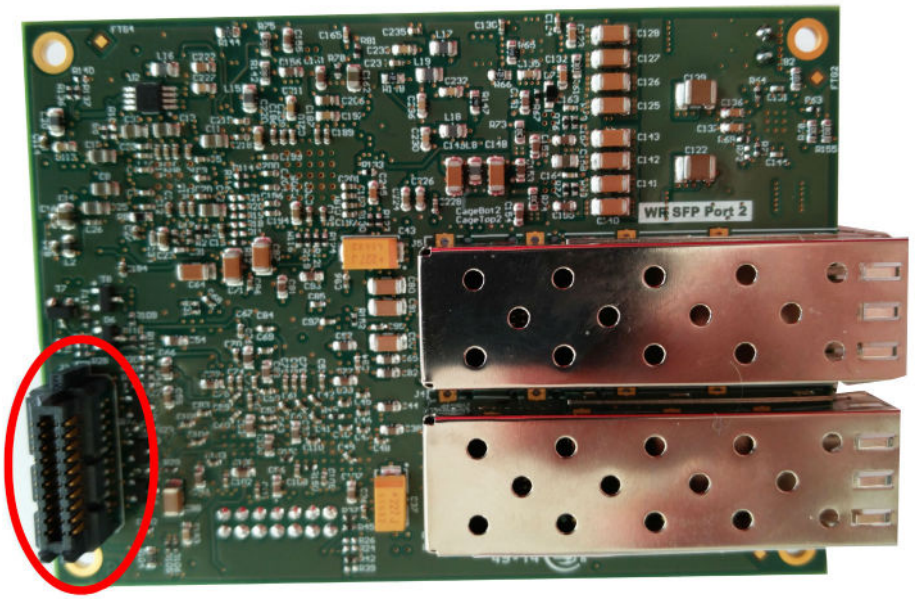


Figure 14: OEM-LEN with Molex Searay connector (red)

The channels can be programmed to trigger at specific time individually and also to repeat periodically. Each channel can be programmed to be triggered at a delay from another channel (but always maintaining the same period). The resolution of this triggers signal is 16ns (pulse width minimum is also 16ns).

The following commands should be entered through the UART interface.

```
## To trigger an output on a channel immediately
trig <ch> <len> now
```

Where `<ch>` is the channel to configure and `<len>` is the pulse width in nanoseconds.

```
##To program a trigger at specific time with the option to be repeated
     periodically
trig <ch> <len> -t <s> [ns] [-l <s> [ns]]
```

In this case, after the -t modifier we should introduce the POSIX time (seconds from January 1st, 1970 00:00:00) in and, optionally, its nanosecond value [ns]. If the periodic option needs to be activated, the -l modifier (loop) should be added and entered. Then, using the same syntax as before, the period with should be specified. Example:

```
wrc>trig 0 128 t 1447839978 16 l 2 64
```

This command will configure a trigger on channel 0, with a pulse width of 128 nanoseconds at time: 18 Nov 2015 09:46:18:000000016 GMT and repeat every 2 seconds and 64 nanoseconds. Next pulse will be 18 Nov 2015 09:46:20:000000080 GMT and so on. It is important to notice that at the moment we configure a new periodic trigger on any channel, the previous periodic trigger will stop. Only one channel can be periodically pulsing.

The last command to program delayed channels is

```
trig <ch_delayed> <len> -d <ch_master> <ns>
```

Considering that after the trig command you must enter the channel being configured, in this case, an example would look like:

```
wrc>trig 1 32 d 0 512
```

This command will configure the channel 1 to trigger always at a delay of 512 ns after the channel 0.

By default, the WR-LEN board is configured to generate a fan out of the PPS & 10 MHz clock in the back DB9 connector. In order to enable the trigger generation in the WR-LEN please contact with the technical support of Safran.

In the case that the configurable triggers have been enabled, the precise outputs through the 2 x uDB9 connectors are detailed in Figure 15.

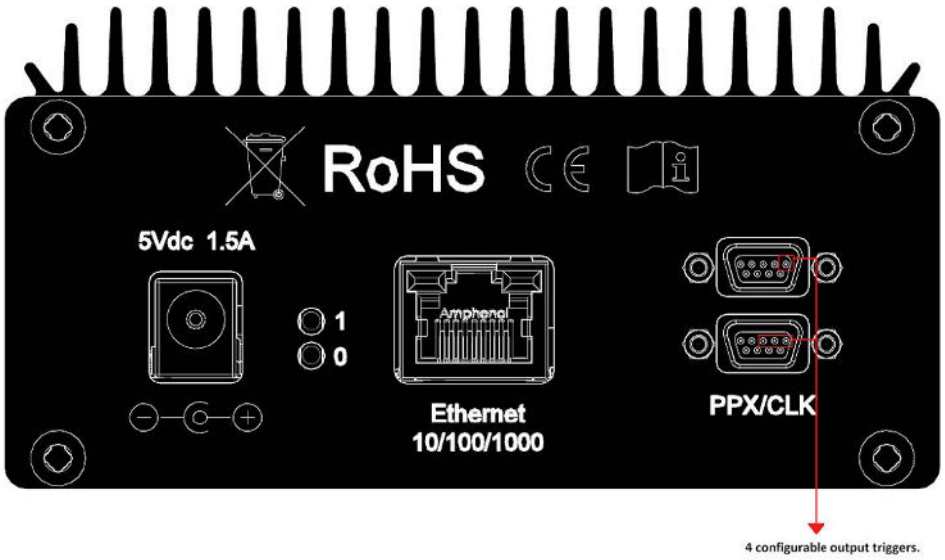


Figure 15: Stacked DB9 connector output triggers

10. Appendix D: Flashing the LEN

This method is detailed in the guides you can find on the LEN-specific webpage, including the procedures on how to flash the LEN remotely using the interface provided by Safran.

10.1. Remote Virtual UART Shell Flash

This method will explain how to flash the LEN remotely using Etherbone packets through one of the tools that you will find at py7slib .

The Wishbone SPI Flash Update allows to perform a remote update of the Image stored at the Flash memory, which will program the board's FPGA. This procedure is based on the QuickBoot Method for FPGA Design Remote Update method, and uses the Etherbone protocol to transmit the data and configure the device.

This method guarantees that a valid image will be loaded even if the Update Image is corrupted, since the Golden Image is never erased and acts as a backup if the flash programming has any errors.

If you are interested about how this method works, you can find more information at: Remote FPGA Flash.

The list of WR-LEN releases and betas are open in the bitbucket repository wr-len-releases. Choose the preferred version, download it and extract it. There is one folder and inside there are several files:

- **wrc_2p_len.mcs**: File to program the LEN nonvolatile flash memory. Use vivado and the JTAG connector. The LEN node has one of these flash memories depending on the SN of its HW:
  i. N25Q256
  ii. S25FL256SAGNFI003
  iii. S25FL256LAGNFV010
- **wrc_2p_len.bit**: Simple binary to program the LEN FPGA (XCA735T) through Vivado.
- **wrc_2p_len_initial.mcs**: File to program the LEN nonvolatile flash memory (n25q256-3.3v-spi-x1_x2_x4). This file should be used to write into the flash through the JTAG connector just once, the first time.
- **wrc_2p_len_update.mcs**: We use this file to program the flash just remotely.
- **wrc_2p_len.prm**: metadata from the file .mcs

To continue this process, you must have been able to set a static ip for your WR-LEN, and if you connect with it by the remote shell using:

```
$ ./vuart.py <IP>
```

Once you have configured this with the steps mentioned above, on the folder py7slib there are several tools and scripts where we will find *spiflash_update.py*. Execute to flash the LEN:

```
$ ./spiflash_update.py -l <IP> -b EB -m update -f wrc_2p_len_update.mcs
```

The process will take a few minutes, and once is finished you can connect with it and check that successful has been flashed to the new version by using the command *ver*

10.2. Flashing through the JTAG connector by the Vivado tool.

- Vivado Design Suite is a software suite produced by Xilinx for synthesis and analysis of HDL designs. It allows easy access to the WR-LEN flash memory.
- In case that you want to purchase the JTAG programming cable, the following have been successfully tested:
  - The Xilinx Platform-Cable-USB-II.
  - The Digilent JTAG-HS2-Programming-Cable.

10.2.1. Flashing Steps.

Before starting, keep in mind that arrived this point, is under your responsibility and Safran will not take any further action if there is any problem during this process except if the evaluator is instructed by Safran.

1. Open the WR-LEN box, fed it with the 5V power supply and connect the JTAG programming cable into its connector.

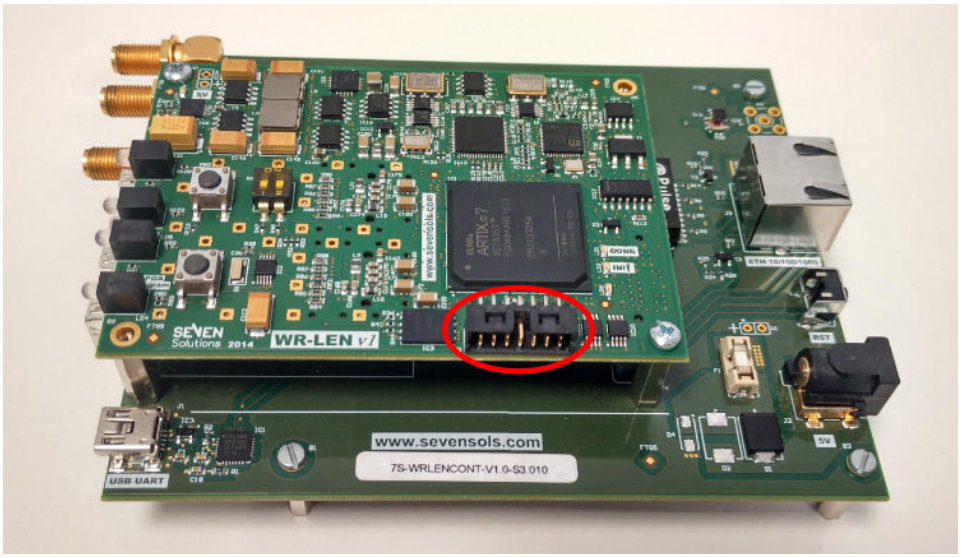


Figure 16: The WR-LEN JTAG connector

2. Launch the Vivado tool installed in your computer.

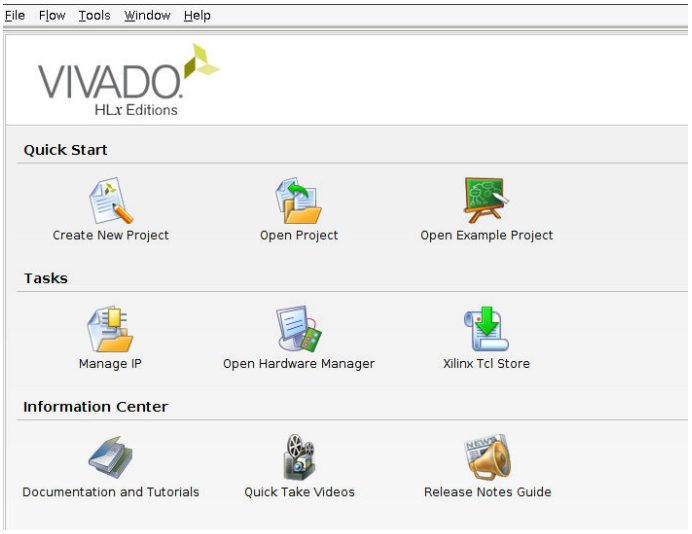


Figure 17: Vivado main screen

3. Open the Hardware manager.

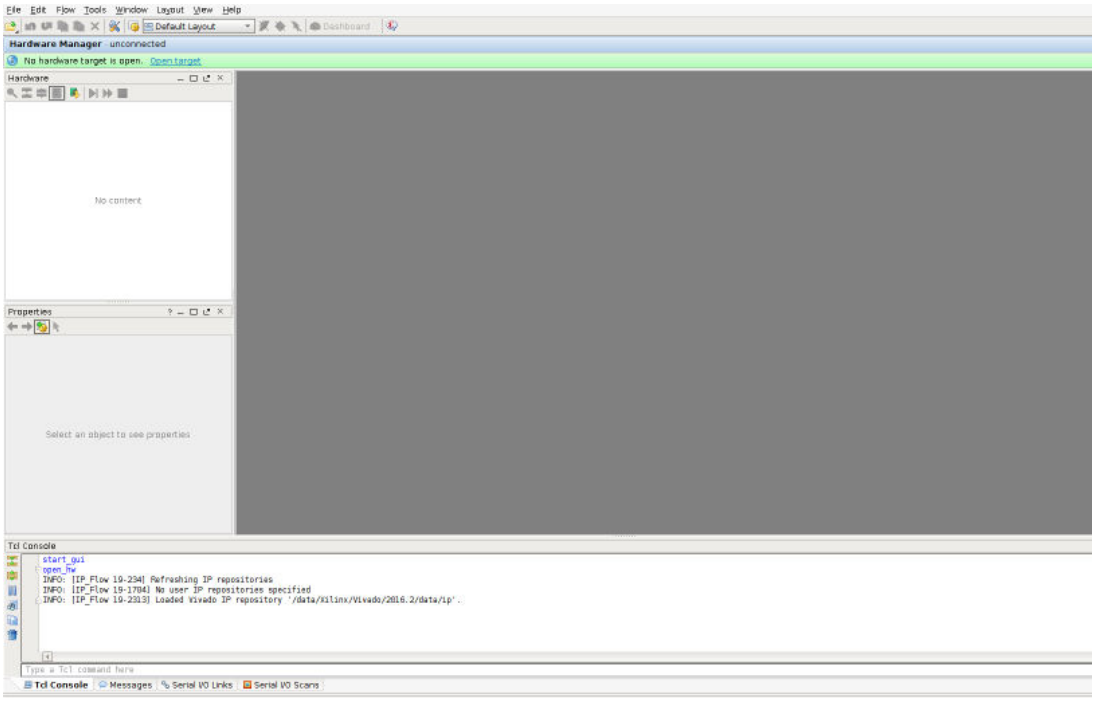


Figure 18: Vivado Hardware manager

4. Autodetect the device connected on the JTAG programming cable. If the WR-LEN has been properly detected, it must show `xc7a35t`, which is the WR-LEN FPGA model.

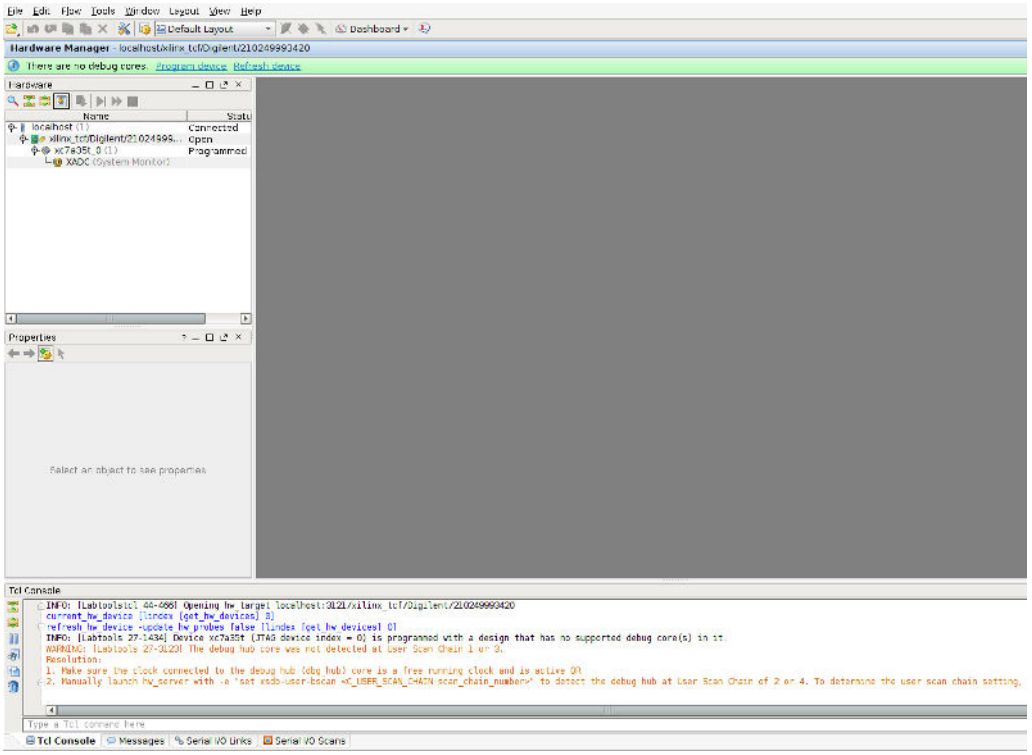


Figure 19: WR-LEN successful detection

5. Now you must select the flash memory used by the WR-LEN node. There are three options depending on the HW version.
   i. N25Q256
   ii. S25FL256SAGNFI003
   iii. S25FL256LAGNFV010

This cannot be automatically detected, so the customer must enter the flash memory manually, and an error message will appear if the wrong selection is made. This is a trial-and-error process; **you will need to make memory selections in this window until you do not receive an error message** (Density = 256; Type = SPI; Width = x1_x2_x4).

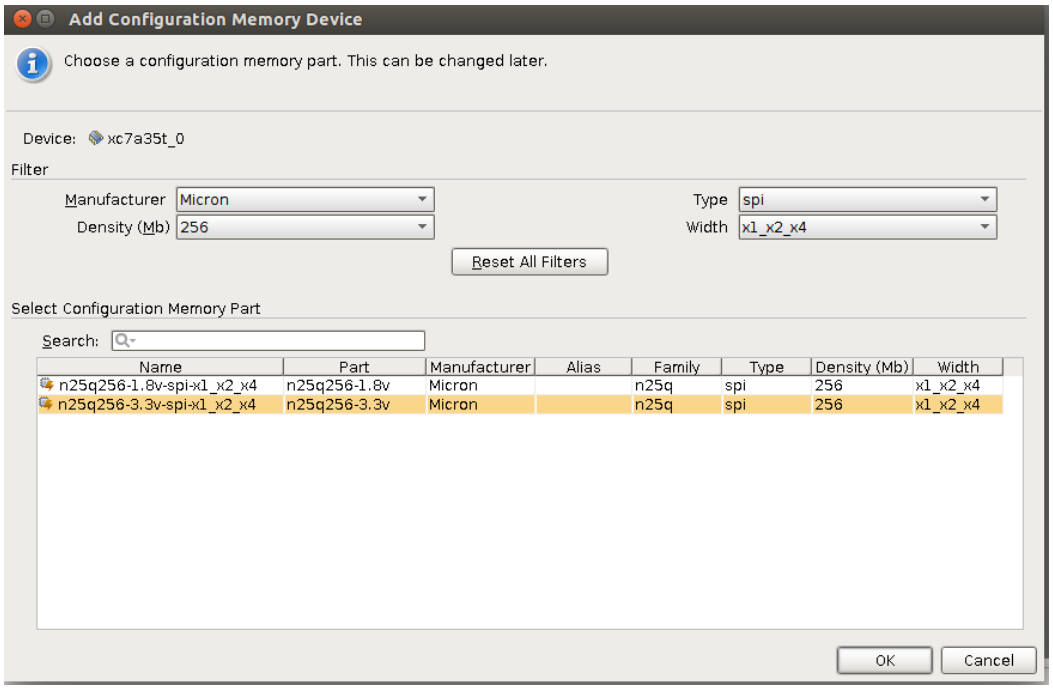


Figure 20: Example of n25q256 memory programming.

6. Finally, program the WR-LEN flash memory (Program Configuration Memory Device). Choose the wrc_2p_len_VERSION-DATE_initial.mcs from the wr-len-releases and select pull-up as `state of non-config mem I/O pins`.

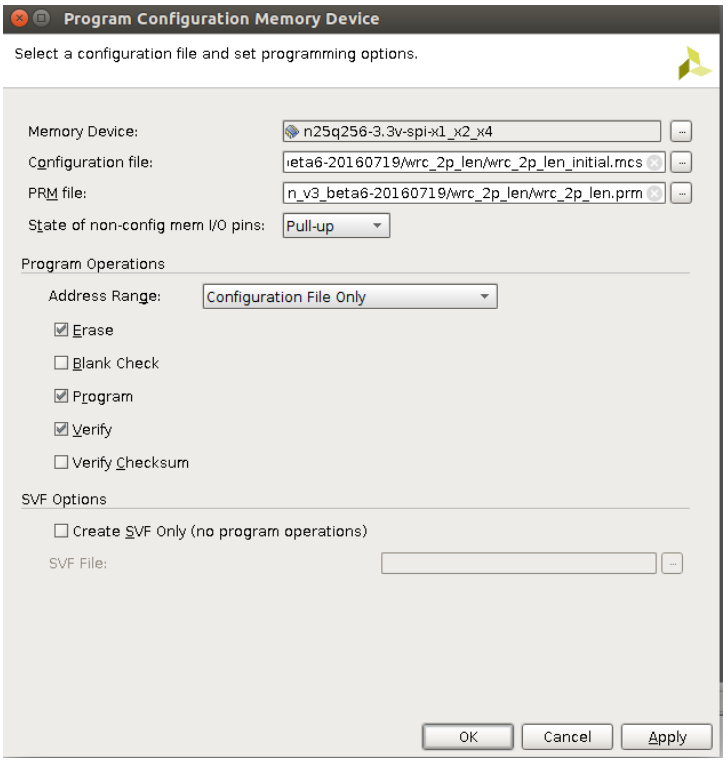


Figure 21: Program Configuration Memory Device

It should work after a couple of minutes and pressing the WR-LEN flashing button. The INIT and DONE LEDs must be on.

11. Safran Technical Support

For technical support, product specifications, and additional documentation, you can visit https://safran-navigation-timing.com/support-hub/white-rabbit/ to submit a support request.

More information on standard unit behavior or any other features or functions of the WR-LEN can be found on our website at https://safran-navigation-timing.com/product/white-rabbit-len/

Safran Electronics & Defense

safran-navigation-timing.com