Hyrax-1000 Series

MODULAR COMPUTING PLATFORM FOR RAILWAY APPLICATIONS

CONFIGURATION MANUAL

Version: 1.0 for firmware V24.02.00 | Date: 03.07.2025

Contents

1 General Information 1
1.1 DISCLAIMER 1
1.1.1 Copyright 1
1.1.2 SAFETY INFORMATION 1
1.1.3 RECYCLING 2
1.1.4 GPL Statement for Hyrax-1000 Software 2
1.1.4.1 Disclaimer of Warranty 2
1.1.4.2 Limitation of Liability 3
1.1.4.3 Obtaining License Texts and Sources 3
1.1.5 Regulatory Limits for Changes in Country and Transmit Power Settings 3
2 About this Document 3
3 Hardware Overview 3
3.1 Base CPU system 4
3.1.1 Board management controller 5
4 BIOS description 5
4.1 Entering BIOS setup 5
4.2 Setup USB boot 7
4.3 Redirect BIOS output to serial port 1 8
5 Software Description 9
5.1 Software Overview 9
5.1.1 Operating System 9
5.1.1.1 Open Source Licenses 10
5.1.2 Boot Concept 10
5.1.3 System Maintenance 11
5.1.3.1 Update the system software 11
5.1.3.2 Save the current configuration 11
5.1.3.3 Restore a configuration 12
5.1.3.4 Factory reset 12
5.2 Westermo Eltec Packages 12
5.2.1 Kernel Modifications 12
5.2.1.1 Watchdog handling 12
5.2.2 Status LED handling 13
5.2.2.1 Status LED DBUS interface 14
5.2.3 Activity LEDs for WiFi and Broadband Modules 15
5.2.4 Reset button handling 15
5.2.5 Serial Port Configuration 16
5.2.5.1 Cabling hints and hardware details 17
5.2.5.1.1 RS422 with Hyrax-1000 17

5.2.5.1.2 RS485 with Hyrax-1000 17
5.2.5.1.2.1 Cabling 17
5.2.5.1.2.2 Termination 18
5.2.6 Revision EEProm 18
5.2.7 MVB Bus Support 19
5.2.8 SIM card switching and Cellular Network Setup 19
5.2.8.1 Script `pcmcx-wwan-setup` 19
5.2.8.1.1 Parameter “sim_info” 20
5.2.8.1.2 Parameter “sim_switch” 20
5.2.8.1.3 Parameter “restore_sim_gpio” 21
5.2.8.1.4 Parameter “wwan_switch” 21
5.2.8.1.4.1 Configuration file `/etc/pcmcx/pcmcx-wwan-setup.yaml` 22
5.3 Standard Linux functionality 22
5.3.1 Audio Support 22
5.3.2 GNSS Support 22
5.3.3 Temperature Sensors 23
5.3.4 WiFi Support 24
5.3.5 Cellular Radio Support 26
5.3.6 CAN Bus Support 27
5.4 Build Software from Source 28
5.4.1 Shipment Details 28
5.4.2 Build the Binaries from Source 29
5.4.2.1 Short version of installation hints 29
5.4.2.2 Longer version of installation hints 29
5.4.2.2.1 Contents of the source archive 29
5.4.2.2.1.1 Subdirectory “tools” 30
5.4.2.2.1.2 Subdirectory “linux” 30
5.4.2.2.1.3 Subdirectory “filesystem” 31
5.4.2.2.1.4 Subdirectory “scripts” 32
5.4.2.2.2 Filesystem build details 33
5.4.3 Add own applications 33
6 Appendix 34
6.1 Open Issues 34
7 Frequently asked Questions 34
7.1 Which is the system time, if the power CAP supply is discharged? 34
7.2 What are the default values set after factory reset? 34
7.3 How can the hardware revision be detected? 34
7.4 How can the systems software revision be detected? 34
7.5 How can the default keyboard layout be changed? 34
7.6 How can we use serial port 1 for console redirection? 35

7.7 Can the wireless country code be changed? 35

Index 37

1 General Information

1.1 DISCLAIMER

1.1.1 Copyright

1.1.2 SAFETY INFORMATION

Electrical safety

- To prevent electrical shock hazard, disconnect the power cable from the electrical outlet before reloading the system.
- When adding or removing devices to or from the system, ensure that the power cables for the devices are unplugged before the signal cables are connected. If possible, disconnect all power cables from the existing system before you add device.
- Before connecting or removing signals cables from motherboard, ensure that all power cables are unplugged.
- Make sure that your power supply is set to the correct voltage in your area. If you are not sure about the voltage of the electrical outlet you are using, contact your local power company.
- If the power supply is broken, do not try to fix it by yourself. Contact a qualified service technician or your retailer.

Operation safety

- Allow only appropriate trained personal to handle the devices. Observe the ESD protective measurements.
- Before installing the motherboard and adding devices on it, carefully read the manuals that came with the package.
- Before using the product, make sure all cables are correctly connected and the power cables are not damaged. If you detect any damage, contact your dealer immediately.
- To avoid short circuits, keep paper clips, screws, and staples away from connectors, slots sockets and circuitry.
- Avoid dust, humidity, and temperature extremes. Do not place the product in any area where it may become wet.
- Place the product on a stable surface.
- If you encounter technical problems with the product, contact a qualified service technician or your retailer.

1.1.3 RECYCLING

Please recycle packaging environmentally friendly:

Packaging materials are recyclable. Please do not dispose packaging into domestic waste but recycle it.

Please recycle old or redundant devices environmentally friendly:

Old devices contain valuable recyclable materials that should be reutilized. Therefore please dispose old devices at collection points which are suitable.

1.1.4 GPL Statement for Hyrax-1000 Software

This software product contains software covered by the GNU GPL (see below in this document), it may in addition contain other parts covered by other licenses (such as LGPL). A list of all modules and their licenses (“FOSS” list) is available on request (see link below). The source code of all GPL-covered modules can also be requested by owners of the Hyrax-1000 (see link below).

For the GPL-covered parts this license is valid:

```
Copyright (c) 2022-2025, Westermo Eltec GmbH

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program. If not, see
<https://www.gnu.org/licenses/>.
```

FOSS and sources are not included in the binary distribution in the products and in the product documentation due to space limitations.

1.1.4.1 Disclaimer of Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

1.1.4.2 Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

1.1.4.3 Obtaining License Texts and Sources

This software product contains software covered by the GNU GPL and other licenses.

The licenses of all software can be found on the running system in the files “/usr/share/doc/PACKAGE/copyright”, where “PACKAGE” is the package name.

Additionally a text file containing the license texts is part of the binary shipment.

The sources of all packages can be obtained on request (handling fees for sources may apply).

Ask your sales worker or send a letter to the address below, containing you product name, hardware and software version number:

```
Westermo Eltec GmbH
Galileo-Galilei-Str. 11
55129 Mainz
Germany
https://www.eltec.com
support.eltec@westermo.com
```

1.1.5 Regulatory Limits for Changes in Country and Transmit Power Settings

Make sure that only persons with proper knowledge also in regulatory matters have access to the access point’s configuration settings. They must be aware of the consequences of an improper setting of country and transmit power (there may be additional settings). To do so, the standard configuration password must be changed before the access point is deployed. And this new password must be given to knowledgeable and responsible persons only.

One example of a regulation affecting country selection is that in Germany, as of October 2016, the frequencies in the range 5150 MHz - 5350 MHz must be used in closed rooms and similar environments only. For more information please see www.bundesnetzagentur.de.

2 About this Document

This user manual is intended for system developers and integrators. It is not intended for end users.

The document describes the hardware and software components of the Hyrax-1000.

Information about mechanical and electrical installation of Hyrax-1000 can be found in the product-specific installation manual.

The internal version of this document is 880c02d.

3 Hardware Overview

The Hyrax-1000 is a modular computing system with several data interfaces, designed for e.g. data logging applications.

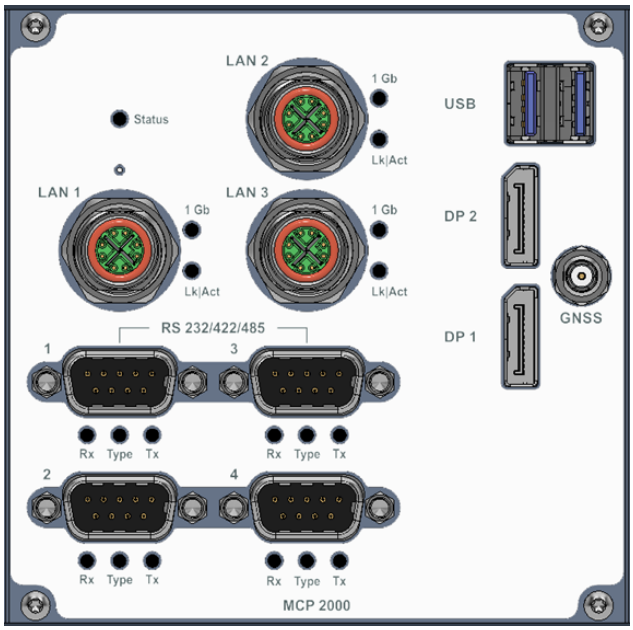
Depending on the modules connected, interfaces are among others RS232, RS422, RS485, CAN, MVB, GPS.

Three 1GBit LAN interfaces and optionally WiFi and LTE can be used to send the data acquired to other systems.

The hardware is certified according to the EN 50155 (IEC 60571) standard, and can be used in rough environments from -40°C to +70°C.

The following section describes the hardware and the several modules.

3.1 Base CPU system



CPU module

The following table gives an overview of the system.

Depending on the order number of the Hyrax-1000, the components used may be different.

CPU Module Overview

CPU	Intel Atom x5/x7 E39S (Apollo Lake-I)
RAM/eMMC	x5-E3930 2x 1.3/1.8 GHz, 2 GB RAM, 8 GB eMMC
	x5-E3940 4x 1.6/1.8 GHz, 8 GB RAM, 32 GB eMMC
onBoard Storage	eMMC or SSD for Firmware and Add-Ons
Internal Storage	up to 980 GB M.2 SSD via SATA 3.0 (120 GB Standard)
Ethernet Interfaces	3x 1 Gbit @ M12 x-coded connectors
USB Interfaces	2x USB-A 3.0 as service ports
Graphic Interfaces	2x DP 1.2 Intel HD Graphics 500
Serial Interfaces	1x RS232/RS422 @ D-Sub 9-Pin
	3x RS232/RS422/RS485 @ D-Sub 9-Pin
Audio Interfaces	1x Line-In @ Jack-Plug 3,5mm
	1x Line-Out @ Jack-Plug 3,5 mm

	1x Mic-In @ Jack-Plug 3,5 mm
GNSS Interface	GPS, GLONASS, GALILEO, BEIDOU

For a description of hardware details and connectors see the *Installation Manual*.

3.1.1 Board management controller

The Board Management Controller (BMC) performs several crucial functions, including watchdog supervision, monitoring of temperature via an LM75 sensor, and power management.

The watchdog feature initiates a board reset should it remain untriggered beyond a specified time interval.

In the event that the LM75 sensor records a temperature of 105°C or higher, the system undergoes a temporary shutdown lasting 5 minutes to prevent potential damage.

The BMC initiates system startup only when the temperature registers below 82°C; otherwise, it waits until the temperature drops below this threshold.

Furthermore, a watchdog driver is in place to retrieve the last reset cause following system boot-up. This driver also grants user-mode applications the capability to activate the watchdog hardware when needed.

See 5.2.1.1 Watchdog handling for description of the watchdog driver.

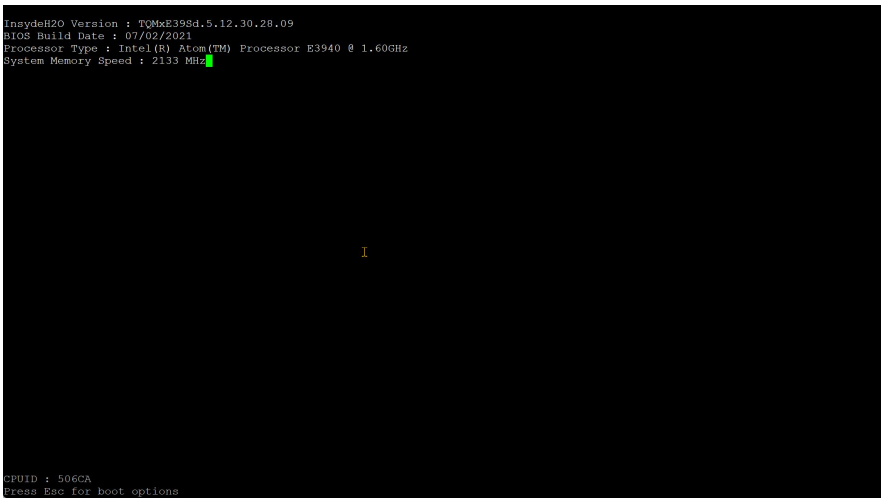
4 BIOS description

This chapter describes the procedure to boot from USB drive and redirect BIOS console output.

> **Hint**
>
> Any modifications to the BIOS settings must be completed within a 2-minute timeframe, because the watchdog is not triggered while the system is operating within the BIOS setup. Consequently, if the 2-minute limit is exceeded, the system will initiate an automatic reboot.

4.1 Entering BIOS setup

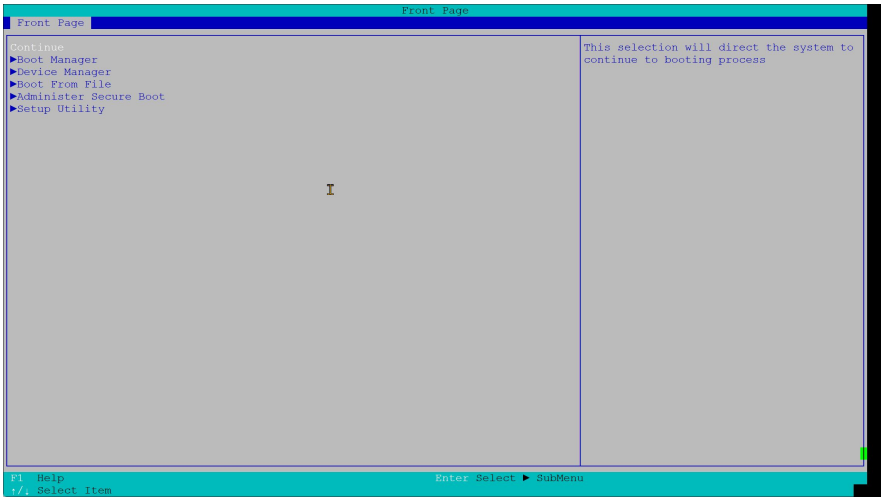
To enter the BIOS setup press *ESC* during system startup.



Startup screen

ESC pressed during startup

After a short time the BIOS setup is visible.



BIOS front page

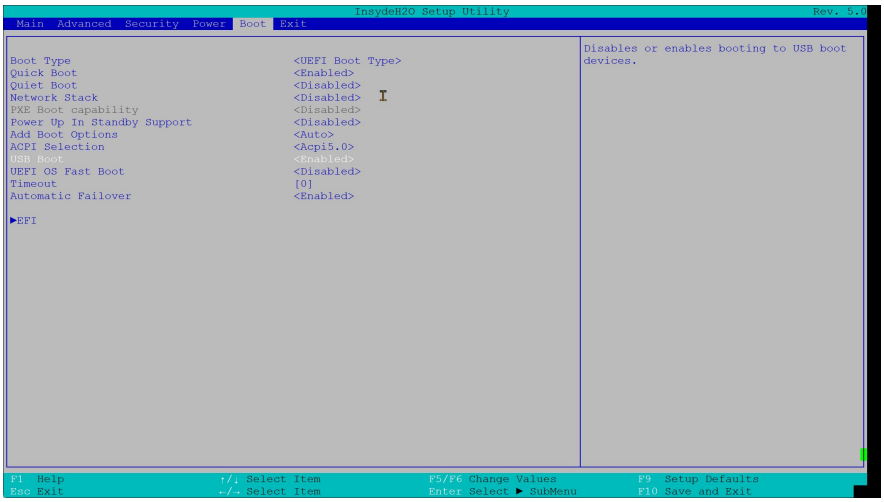
4.2 Setup USB boot

To boot from a USB drive with an installation image, USB boot must be enabled.

Additionally the boot type must be *UEFI Boot Type*, as the Hyrax-1000 installation USB drive uses EFI boot.

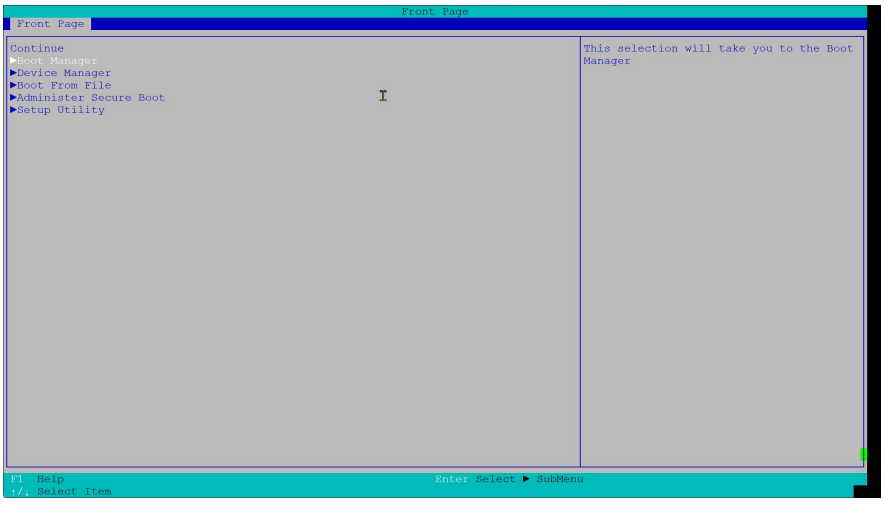
Review and adjust these settings within the *Setup Utility* menu.

Ensure that USB boot is enabled.



USB is enabled

If USB boot is enabled choose the *Boot Manager* from the front page.

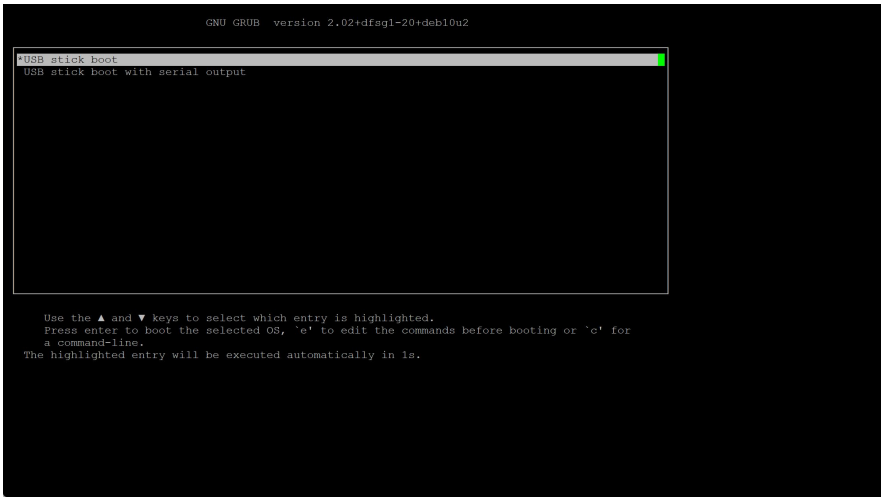


BIOS front page

Choose the USB media as boot source.

Boot from USB

Within a few seconds, the system should automatically execute the selected USB image from the boot menu of the USB drive.



USB drive boot menu

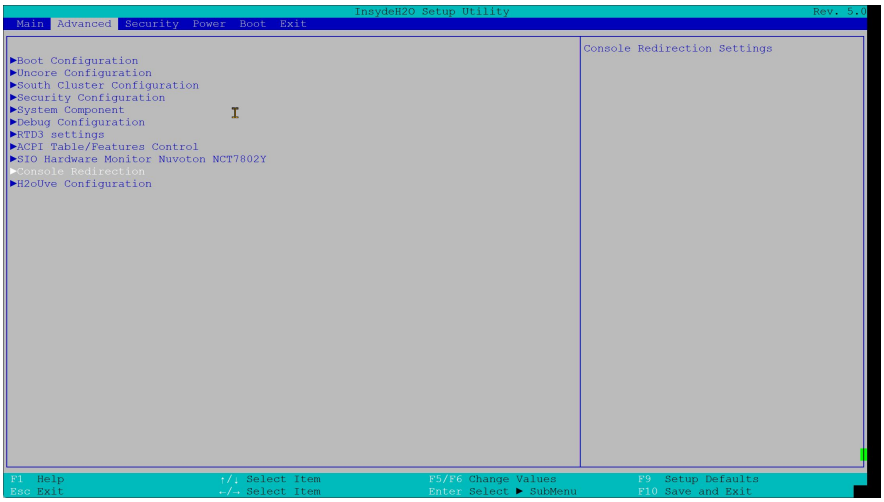
4.3 Redirect BIOS output to serial port 1

The BIOS output can be redirected to serial port 1.

You can leverage terminal emulation software such as Putty or MobaXterm, available for Microsoft Windows, to utilize this feature. This allows you to directly access and view the console output on a Windows-based system.

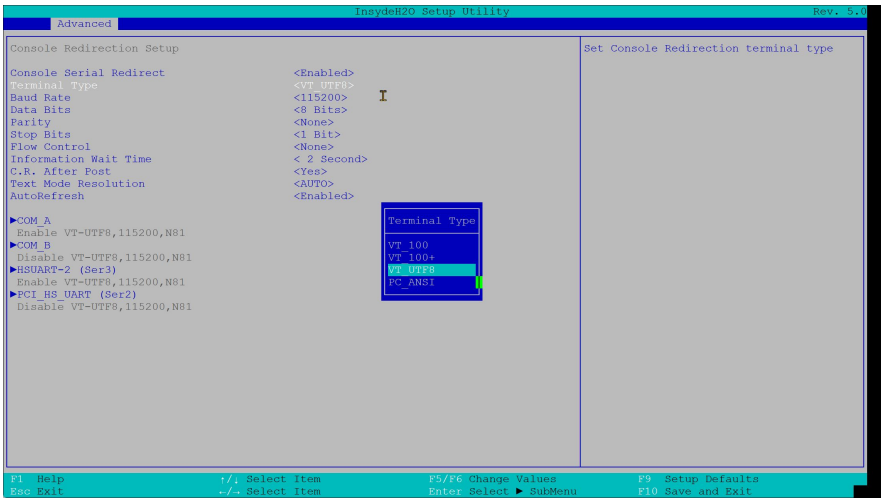
To enable the redirection feature navigate to the *Advanced* menu and enter *Console Redirection* setup.

Ensure that the *Console Serial Redirect* feature is enabled.



BIOS Advanced Console Redirection

For best results set the “Terminal Type” to “VT_UTF8”.



BIOS console terminal type

5 Software Description

The following chapters describe the operating system and packages installed on the system.

The first part gives an overview description.

The second part describes the different packages not coming with the base operating system.

The third part of this chapter describes how to build the system from source and add new applications.

5.1 Software Overview

5.1.1 Operating System

The Hyrax-1000 operating system is based on Debian Linux 11, leveraging the Debian Live project.

The entire filesystem is encapsulated within a single SquashFS file, simplifying updates and enabling the restoration of the initial system state. It’s important to note that this SquashFS file is read-only.

To facilitate system changes, such as altering IP addresses, a writable space is essential. This is accomplished through an *overlay* filesystem, which stores modifications to the underlying SquashFS filesystem in a writable storage location. The user sees only the the combination of both filesystems. The original state can be restored by clearing the writable overlay portion.

Stored within the eMMC storage are two versions of the operating system: the *standard system* and a *rescue system*. The rescue system is started if the standard system fails to start on three consecutive attempts. This could occur due to boot issues or watchdog-initiated reboots.

> **Note**
>
> The standard operating system is installed on the eMMC. The optional SSD storage can be used for customer data or an alternative operating system.

5.1.1.1 Open Source Licenses

The licenses of the open source applications used to build the operating system are located on the booted system. See the files */usr/doc/the-package-name/copyright* for the license of the package *the-package-name*.

A combined file with all package licenses is included with each delivery.

5.1.2 Boot Concept

During production of the Hyrax-1000 the eMMC is prepared for the system to boot.

On a booted system, `lsblk` shows the following partitions.

eMMC partitions

Partition Name	Purpose
efi_boot	Contains all files needed for EFI boot. Legacy boot is not supported.
grub_boot	Contains all files needed for Grub - the boot loader - to boot.
rootfs_a	Contains the SquashFS file with the standard file system.
rootfs_b	Contains the SquashFS file with the rescue file system.
applfs_a	This partition contains the overlay for the standard file system.
applfs_b	This partition is empty, as the rescue image does have an overlay.
interim	This partition should be empty. It is used for system update purposes and should usually not be used.

During the system startup, the boot loader *Grub* checks the watchdog counter in the Grub environment file. (see: 5.2.1.1 Watchdog handling).

- If the watchdog counter is less than three, the standard image from *rootfs_a* is started. This image mounts the overlay filesystem from *applfs_a"*.
- If the watchdog counter is greater or equal three, the emergency image is started. This image does not use an overlay filesystem.

5.1.3 System Maintenance

The maintenance script used for the action described below is `sysupgrade`. It must be executed as *root* or using *sudo*.

Given that the system operates from a SquashFS file and uses an overlay filesystem, updating it is a seamless process with the added benefit of easily preserving and restoring the current state of the filesystem overlay.

5.1.3.1 Update the system software

System software updates entail the replacement of the SquashFS file.

In standard mode this file is mounted, so it can not be replaced directly.

To address this, a workaround is implemented: the new image is copied to the eMMC, and the emergency mode is initiated to facilitate file replacement.

This update process leverages a *Rauc-Bundle* (available at https://github.com/rauc/rauc), with most of the intricacies managed behind the scenes.

As first step, the update must be *prepared*.

As the initial step, the update must be prepared. This necessitates copying the *Rauc-Bundle* to the eMMC, where the *interim* partition is designated for this purpose. You can use tools like *scp* to copy the bundle to the system.

Afterwards, execute the command `sysupgrade --prepare /media/interim/mcr-v22.YY.raucb` to initiate the update preparation process. This step simply involves moving the file to the directory `/media/interim/job`.

Following this, a system reboot is required. During the boot-up sequence, the bootloader verifies the presence of a Rauc-Bundle and triggers the emergency system if found.

Within the emergency system, the standard SquashFS is no longer mounted, allowing for its replacement.

`sysupgrade --upgrade` is called automatically, then the update image is removed, a file named *update-image.[applied|failed|unknown]* is created instead and the system is rebooted automatically.

5.1.3.2 Save the current configuration

All configuration settings, like display resolution, additional packages, customer scripts are stored in persistent partition *applfs_a*.

To make a snapshot of the current application configuration partition, the script *sysupgrade* is used. The backup archive is stored in */media/interim/backup* and can be downloaded to the host from there.

The `sysupgrade --backup <tag>` backup function takes an option *tag*, which can freely by chosen (avoid blanks).

> **Hint**
>
> The backup archive file name matches following convention and must not be altered. `backup-mcr-<BACKUP_TAG>-<date>-<part_size>-<part_name>.fsa` This naming convention is necessary for the `sysupgrade --restore` function.

```
root@CyBoxMCR:~# sysupgrade --backup V1_0
```

```
root@CyBoxMCR:~# sysupgrade -b V1.0
Partition layout and mount locations
        /dev/mmcblk1p1 (efi_boot)       is NOT mounted
        /dev/mmcblk1p2 (grub_boot)      is mounted on   /boot
        /dev/mmcblk1p3 (rootfs_a)       is mounted on   /usr/lib/live/mount/persistence/mmcblk1p3
        /dev/mmcblk1p4 (rootfs_b)       is NOT mounted
        /dev/mmcblk1p5 (applfs_a)       is mounted on   /usr/lib/live/mount/persistence/mmcblk1p5
        /dev/mmcblk1p6 (applfs_b)       is NOT mounted
        /dev/mmcblk1p7 (interim)        is mounted on   /media/interim
Starting backup of application partition: applfs_a
Statistics for filesystem 0
* files successfully processed:....regfiles=42, directories=45, symlinks=1, hardlinks=0, specials=0
* files with errors:..............regfiles=0, directories=0, symlinks=0, hardlinks=0, specials=0
Successfully stored backup at:
/media/interim/backup/backup-cymcr-V1.0-2021.03.18-1000M-applfs_a.fsa
root@CyBoxMCR:~#
```

5.1.3.3 Restore a configuration

Restoring an overlay configuration has the same constraint as the system update (see: 5.1.3.1 Update the system software). The overlay is mounted when the system runs the standard image.

To restore the configuration, use the “prepare” mechanism. Copy the configuration archive to */media/interim*, then execute the command `sysupgrade --prepare <backup archive>` and proceed with a reboot.

Upon reboot, the emergency system is started, and the command `sysupgrade --restore <backup-archive>` is automatically executed. Subsequently, the system undergoes another reboot, transitioning back to the standard mode.

5.1.3.4 Factory reset

Executing a *factory reset* operation clears the overlay, resulting in the loss of all changes stored in the overlay filesystem.

To initiate a factory reset, simply run the command `sysupgrade --factory-reset`. This action clears the overlay filesystem and triggers an *immediate* system reboot.

5.2 Westermo Eltec Packages

5.2.1 Kernel Modifications

The Linux kernel utilized on the Hyrax-1000 is essentially a vanilla kernel sourced from http://kernel.org. It has been enhanced with approximately 20 specific patches contributed by Westermo Eltec.

You can locate these patches within the linux/packages directory when unpacking the source delivery.

These patches are essential for the internal CPU module and various Hyrax-1000 modules.

5.2.1.1 Watchdog handling

An essential addition to the vanilla kernel is the watchdog driver `mcri2c_wdt`. This driver is needed to manage the watchdog functionality within the board management controller.

> **Warning**
>
> The watchdog cannot be deactivated through software, making the presence of this driver imperative. Without this driver and a user-mode application to initiate the watchdog, the system will automatically reboot after a 2-minute interval.

If you need to install a different Linux kernel, please ask your sales representative for support.

The watchdog driver `mcri2c_wdt` is used to provide access to the BMC watchdog.

The watchdog sys filesystem entries are located under */sys/class/watchdog/watchdog0*.

The following output shows the kernel messages written into the system log during boot.

```
$ journalctl --utc -k -g watchdog
Nov 02 11:26:26 PCEyeMCR kernel:  (null): Detected Westermo Eltec watchdog mcri2c_wdt
Nov 02 11:26:26 PCEyeMCR kernel: mcri2c_wdt 0-0070: Westermo Eltec Watchdog device enabled. Firmware version: 1. VendorID: 0x1433. Bootstatus: 0x0, Timeout: 120 s.
Nov 02 11:26:26 PCEyeMCR systemd[1]: Using hardware watchdog 'mcri2c_wdt', version 1, device /dev/watchdog
Nov 02 11:26:26 PCEyeMCR systemd[1]: Set hardware watchdog to 2min.
```

The *bootstatus* is “0” in case of a clean boot, “0x01” if the system was shut down due to an overheat event and “0x20” if the system was shut down to a watchdog reset.

The value can be read out from the SYS-FS.

```
$ cat /sys/class/watchdog/watchdog0/bootstatus
0
```

While the Linux operating system is in operation, the watchdog is periodically triggered by `systemd`. The timeout parameters can be set in */etc/systemd/system.conf*.

Excerpt from *system.conf*:

```
RuntimeWatchdogSec=120
#RebootWatchdogSec=10min
#KExecWatchdogSec=0
#WatchdogDevice=
```

Additionally there is a script `/usr/sbin/watchdog-bootstatus`, which checks the watchdog boot status and changes the boot loader environment file, when the last reset was initiated by the watchdog. See: 5.1.2 Boot Concept .

5.2.2 Status LED handling

The status LED is located on the base CPU system.

The following LED colors indicate various system states:

Color	Meaning
off	System is off
red	System is not booted yet
green	System booted in standard mode
yellow (orange)	System booted in emergency mode

The script `status-led-service` handles the status LED. In the unpacked source delivery, the script can be found in the subdirectory “tools/reset_button_service”.

The script is installed as a *Systemd* service and configures the LED to reflect which Linux operating system is running. In normal mode - standard image started - it is green after system startup, in emergency mode - rescue image started - it is orange.

Additionally the LED can be controlled by external programs using a DBUS interface. Through the DBUS interface, customer applications can utilize the status LED, allowing for tasks such as displaying error messages.

5.2.2.1 Status LED DBUS interface

The script `tools/status_led_service/doc/status-led-client-example.py` shows an example how to control the status LED.

To indicate an error state, the DBUS method `SetStatus` can be used. The example below uses *dbus-send* on the command line

```
$ dbus-send --system --print-reply \
  --dest=com.eltec.StatusLedService \
  /StatusLedObject \
  com.eltec.StatusLedInterface.SetStatus array:string:"eth0-check","error"
```

Errors can occur in different scopes. For example there may be an error in a customer application, or setting up the network.

For handling customer specific LED indications the concept of “scope” and “state” is used. “scope” is a string chosen by the application, “state” is one of “error”, “success”, “”.

The string array *“eth0-check”,”error”* contains the command.

The LED will be indicate red after calling “SetStatus”.

To reset the error state for *“eth0-check”* “SetStatus” must be called with *“eth0-check”,”success”* (or *“eth0-check”,””*).

“com.eltec.StatusLedInterface.SetStatus” can be called multiple times with different scopes.

The LED indicates red as long as there are scopes with the state “error”.

Additionally the LED can be set to a blinking mode, which will override any existing LED color settings.

```
dbus-send --system --print-reply \
    --dest=com.eltec.StatusLedService \
    /StatusLedObject \
    com.eltec.StatusLedInterface.SetupBlinking array:string:"500","alternate"
```

“com.eltec.StatusLedInterface.SetupBlinking” uses the parameter string array *“500”,”alternate”* in the example.

The syntax is `com.eltec.StatusLedInterface.SetupBlinking array:string:"frequency","blink_mode"`.

The *frequency* parameter shows the blinking frequency in milliseconds. If set to “0,” blinking is halted, and the last state established by “SetStatus” is restored.

The blinking modes are:

StatusLedInterface.SetupBlinking parameters

Blink_mode	Meaning
red	LED is blinking red
green	LED is blinking green
all	LED is blinking red and green (yellow)
alternate	red and green LED blink alternately

5.2.3 Activity LEDs for WiFi and Broadband Modules

The “pcmcr-tools” package contains scripts and a database that activate the LEDs on all Hyrax-1000 devices, when a network connection is established.

5.2.4 Reset button handling

The reset button is located at the tiny hole above “LAN 1” on the base CPU system. Use e.g. a paper-clip to press it.

Pressing the button sets a GPIO, which is handled by a Python script called `reset-button-service`.

This script is part of the delivery software. The script can be found in the subdirectory “tools/reset_button_service” of the unpacked source delivery.

The script is installed as *Systemd* service and checks the GPIO periodically.

When the reset button is pressed the LED starts to blink and stops after a while again.

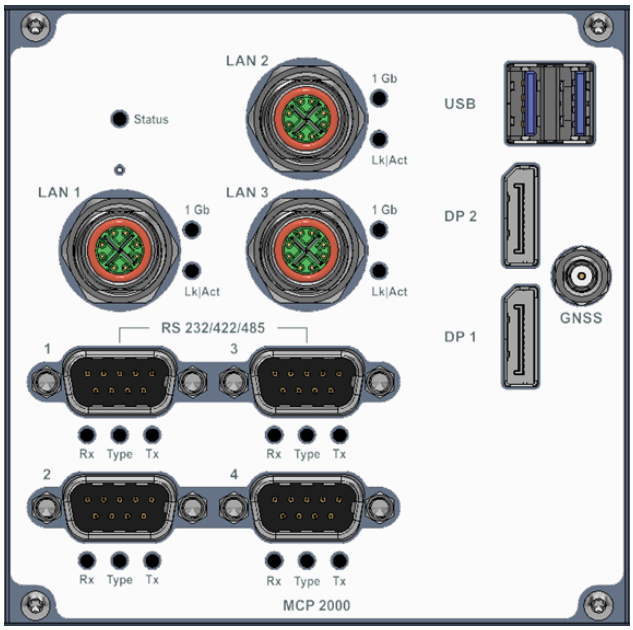
Releasing the button when the LED blinks cause either a reboot or a factory reset, releasing the button when blinking has stopped causes no action.

The following table explains the different blink modes.

Reset button handling

Button handling	Status LED state	Meaning
Press the button and release it within 2 sec.	LED blinking	System reboot initiated
Press the button for more than 2 sec. and release it before 5 sec.	LED blinking stops	No action
Press the button for more than 5 sec. and release it before 10 sec.	LED blinking fast	Factory reset initiated
Press the button for more than 10 sec. and release it	LED blinking stops	No action

5.2.5 Serial Port Configuration



Serial ports on the CPU module

There are 4 serial ports on the CPU module, which can be used by the customer. They are numbered “1” to “4”.

The device node of the first port is */dev/ttyS0*.

The other three are provided by a USB to serial adapter.

As it is not guaranteed that the indexes are continuous, a UDEV rule [1] and helper script [2] are used to create links named */dev/ttyFront[1-4]*.

1 The UDEV rule is */usr/lib/udev/rules.d/79-mcr-front-tty.rules*.

2 The helper script is */usr/sbin/rename_mcr_front_tty*.

The serial ports support different modes:

Serial port modes

Port	RS232	RS422	RS485
1	yes	yes	no
2	yes	yes	yes
3	yes	yes	yes
4	yes	yes	yes

To setup each port, the USB to serial converter must be programmed.

The programming tools and libraries are part of the system and located at source delivery.

The directory “tools/librs232” contains a library and “tools/cy210xsmt” the scripts and programs needed.

To program the serial ports the helper script */usr/sbin/cp210x_serial_config.lua* is used. It must be executed using *sudo*.

The following snippet shows the output of `sudo cp210x_serial_config.lua --help`:

```
sudo cp210x_serial_config.lua --help
Usage: cp210x_serial_config [-v] [--dry-run] [--no-cleanup]
      --port-1 <port_1> --port-2 <port_2> --port-3 <port_3>
      --port-4 <port_4> [-h]

Script for setting up the the serial ports

Options:
  -v, --verbose         Sets verbosity level. Use for more output.
  --dry-run             Just create the config file, do not write the eeprom.
  --no-cleanup          Do not remove temporary files. Use '-v' to see the temporary files name.
  --port-1 <port_1>     Serial port 1 setting (RS232, RS422). Mandatory.
  --port-2 <port_2>     Serial port 2 setting (RS232, RS422, RS485). Mandatory.
  --port-3 <port_3>     Serial port 3 setting (RS232, RS422, RS485). Mandatory.
  --port-4 <port_4>     Serial port 4 setting (RS232, RS422, RS485). Mandatory.
  -h, --help            Show this help message and exit.

After setting up the ports, the system must be rebooted.
```

The command line switches “-v”, “–dry-run” and “–no-cleanup” are for development purposes.

> **Hint**
>
> After setting up the ports, the system must be rebooted. It is not possible to change the settings “on the fly”.

5.2.5.1 Cabling hints and hardware details

5.2.5.1.1 RS422 with Hyrax-1000

RS422 is a point to point connection between two nodes.

Due to differential operation it may be faster and more robust than RS232.

The drawback is that RS422 does not support hardware handshaking, because the pins that are normally used for this are occupied by RxD- and TxD-. The TxD signals need to be connected with the RxDs on the other side. Also a ground connection is required.

At higher speed (>115kBaud) or larger length (>10m) a termination resistor of 100..150Ω is recommended between RxD- and RxD+ at the receiver. For short distances an ordinary null modem cable can be used.

5.2.5.1.2 RS485 with Hyrax-1000

Note: Due to hardware restrictions the „ttyFront1“ (upper left) connector is not capable to operate in RS485 mode.

The other three ports are utilizing a CP2108 USB UART.

This device automatically enables the RS485 transmitter. As long as a port is transmitting it’s receiver is disabled by the hardware. I.e. a node does not see what it is transmitting.

RS485 operation requires some arbitration mechanism to manage the permission to transmit on the bus.

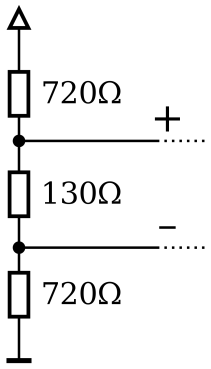
5.2.5.1.2.1 Cabling

For RS485 operation the TxD and RxD Signals need to be connected to each other (i.e. pins 2 with 3 and 6 with 7 of the DB9 connector).

Since the Interface is isolated a ground connection is necessary (pin 5). This will abrogate isolation if one or more of the connected devices is not isolated. In this case the Hyrax-1000 is only isolated if there is no other ground path (e.g. via shield).

5.2.5.1.2.2 Termination

The RS485 bus requires a termination on both ends. This also maintains idle state when there is no transmitting actions. The termination voltage may be supplied externally. If the Hyrax-1000 is the last node it may be provided on pin 9 of the DB9 connector. To achieve this, a bridge in the serial controller device must be closed (i.e. pin 9 is open by default). Since the termination voltage and the resistor values are not specified by the standard, it is recommended to use +5V and the resistor values in the image below. The exact values are not critical.



RS485 termination

5.2.6 Revision EEProm

The system has a revision EEProm to store e.g. order number and serial number.

The revision EEProm can be read using the helper script REV_EEPROM.

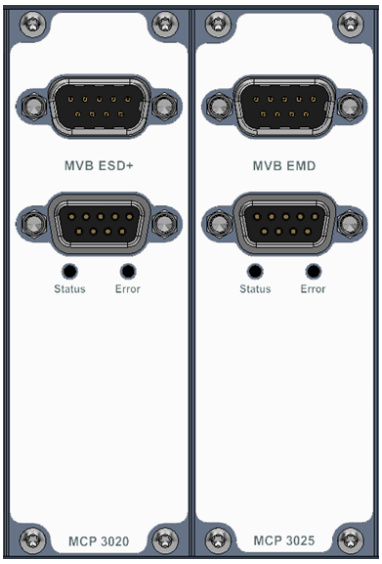
The script and the corresponding program `rev_eeprom` are part of the system and source delivery.

The output below shows an example.

```
$ sudo REV_EEPROM
  EEProm rev.   : 4
  Revision      : 1A
  Board Name    : PCMCX-1000V0
  Serial        : 12345678
  Ethernet ID   : 00005b123456
  Ethernet ID 2 : 00005b123457
  Component 1   :
  Component 2   :
  Country Code  : 00
  DTB Index     : 0
  Text Section  :
  Watchdog Cnt. : 0
  Manu. date    : 2022-06-09
```

5.2.7 MVB Bus Support

The image below shows the MVB ESD+ and MVB EMD modules. See the *Installation Manual* for details.



MVB ESD+ and EMD module

The MVB modules inlcude an software package with a Linux driver, a C library, and a set of example programs.

The library sources are part of the source code shipment.

When the source archive is unpacked, the directory *tools/mvbc_lib* contains the C sources of the library.

Using Doxygen, the documentation can be build in the subdirectory *tools/mvbc_lib/doc*.

The generated HTML documentation shows the functions and an introduction.

5.2.8 SIM card switching and Cellular Network Setup

If the Hyrax-1000 is equipped with a LTE module, it has 4 SIM slots.

Switching between SIM slots requires to change the so called primary modem slot and a GPIO.

For creating a cellular network connection, parameters as the APN, PIN, username, password and eventually others must be supplied.

To simplify the task for a user, there is a script `pcmcx-wwan-setup`.

5.2.8.1 Script `pcmcx-wwan-setup`

The script takes a configuration file containing all the parameters as input.

The program must be called with root privileges for some commands.

Here are the possible command line parameters.

```
$ pcmcx-wwan-setup -h
usage: pcmcx-wwan-setup [-h] [--verbosity] [--log-to-stderr] [--log-cfg-file LOG_CFG_FILE] [--cfg-file CFG_FILE] [--hw-cfg-file HW_CFG_FILE] [--hw-index HW_INDEX]
                        {version,sim_info,restore_sim_gpio,reset_modem,sim_switch,wwan_switch} ...

SIM selection and WWAN setup tool for PCEyeMCR. For detailed help see the manual.

positional arguments:
  {version,sim_info,restore_sim_gpio,reset_modem,sim_switch,wwan_switch}
```

```
                        Sub command help
    version             Display the current version.
```

```
    sim_info            Display current SIM information.
    restore_sim_gpio    Restore the SIM GPIOs for all WWAN devices.
    reset_modem         Reset the modem device (normally not used).
    sim_switch          Switch the SIM slot.
    wwan_switch         Start up a network connection for a given SIM slot

optional arguments:
  -h, --help            show this help message and exit
  --verbosity, -v       Increase verbosity.
  --log-to-stderr       Log to stderr instead of to syslog.
  --log-cfg-file LOG_CFG_FILE
                        Supply the path of a logging configuration.
  --cfg-file CFG_FILE   Supply the path of an application config file.
  --hw-cfg-file HW_CFG_FILE
                        Supply the path of an alternate hardware config file.
  --hw-index HW_INDEX   Supply the hardware index to use instead of reading the revision eeprom.
```

Every command has help output.

The “optional arguments” are common for all positional arguments. So the verbosity can be increase by using “-v” or even “-vv”.

5.2.8.1.1 Parameter “sim_info”

This command show the SIM information for a given device. It can be used to verify e.g.: the “iccid” of the selected SIM.

```
$ pcmcx-wwan-setup sim_info --help
usage: pcmcx-wwan-setup sim_info [-h] -d [0-3]

optional arguments:
  -h, --help            show this help message and exit
  -d [0-3], --device [0-3]
                        Choose a mobile device index.

$ pcmcx-wwan-setup sim_info -d 0 | jq .
{
  "sim": {
    "dbus-path": "/org/freedesktop/ModemManager1/SIM/0",
    "properties": {
      "active": "yes",
      "eid": "--",
      "emergency-numbers": [],
      "esim-status": "--",
      "gid1": "--",
      "gid2": "--",
      "iccid": "89492026196028919743",
      "imsi": "--",
      "operator-code": "--",
      "operator-name": "--",
      "removability": "--",
      "sim-type": "--"
    }
  }
}
```

5.2.8.1.2 Parameter “sim_switch”

This command switches the SIM slot.

```
$ pcmcx-wwan-setup sim_switch -h
usage: pcmcx-wwan-setup sim_switch [-h] -d [0-3] -s [1-4] [--force-switching] [-t TIMEOUT]
```

```
optional arguments:
  -h, --help            show this help message and exit
  -d [0-3], --device [0-3]
                        Choose a mobile device index.
  -s [1-4], --slot [1-4]
                        Choose a SIM slot.
  --force-switching, --force
                        Force switching, even if the destination SIM slot seems to be the current one.
  -t TIMEOUT, --timeout TIMEOUT
                        Timeout for wait busy waiting for ModemManager in sec (default: 180).
```

This command can be used, if you want setup the cellular connections yourself and only SIM switching is required.

A large timeout may be needed, because connecting to a network provider may last very long. The script does not return before the SIM is switched or the timeout is reached.

5.2.8.1.3 Parameter “restore_sim_gpio”

This should not be used manually. It is meant to be called automatically during the boot process.

5.2.8.1.4 Parameter “wwan_switch”

This command is used to switch the SIM slot and open a cellular network connection.

```
$ pcmcx-wwan-setup wwan_switch -h
usage: pcmcx-wwan-setup wwan_switch [-h] -d [0-3] -s [1-4] [--force-switching] [-t TIMEOUT] [--net-cfg-file NET_CFG_FILE]

optional arguments:
  -h, --help            show this help message and exit
  -d [0-3], --device [0-3]
                        Choose a mobile device index.
  -s [1-4], --slot [1-4]
                        Choose a SIM slot.
  --force-switching, --force
                        Force switching, even if the destination SIM slot seems to be the current one.
  -t TIMEOUT, --timeout TIMEOUT
                        Timeout for wait busy waiting for ModemManager in sec (default: 180).
  --net-cfg-file NET_CFG_FILE
                        Supply the path of the WWAN network config file.
```

Under the hood the script uses NetworkManager to set up the cellular network connection.

The SIM configuration is stored in `/etc/pcmcx/pcmcx-wwan-setup.yaml`. Here APN, PIN and eventually username, password and authentication type can be entered.

5.2.8.1.4.1 Configuration file `/etc/pcmcx/pcmcx-wwan-setup.yaml`

```
# First (or only) modem device
devices:
  -
    index: 0
    # base name for the connection
    # the full name will be "wwan_0"
    connection_name_base: wwan
    slot_desc:
      - sim_slot: 1 # SIM slot 1
        apn: "web.vodafone.de"
        pin: 1234
        username: "" # not needed for most providers
        password: "" # not needed for most providers
        noauth: true # authorization not needed for most providers
        initial-bearer-configure: false # false for most providers
        refuse-chap: false
        refuse-mschap: false
        refuse-mschapv2: false
        refuse-pap: false
        refuse-eap: false
        autoconnect: true
      # - sim_slot: 2 # 4 slots are possible
      # ...
```

5.3 Standard Linux functionality

5.3.1 Audio Support

The base system (see 3.1 Base CPU system) has connectors for microphone, line in and line out.

Software support is supplied by the *Pulse Audio* system and the corresponding *Pulse Audio*-plugin for the XFCE display manager.

5.3.2 GNSS Support

The GNSS device is connected to */dev/ttyS1* and “gpsd” is installed.

The program *gpsmon* can be used to see if GPS is working:

```
/dev/ttyS1                    NMEA0183>
------------------------------------------------------------------------------
|Time: 2022-07-12T10:05:12.000Z   Lat: 49 57.599000' N   Lon:   8 15.500700' E |
---------------------------------- Cooked TPV ----------------------------------
------------------------------------------------------------------------------
| GPRMC GPGGA GNGSA GPGSV GLGSV                                              |
---------------------------------- Sentences -----------------------------------
------------------------------------------------------------------------------
| SVID  PRN  Az El SN HU|Time:     100512.000     |Time:      100512.000       |
| GP 10   10 127 59 29  Y|Latitude:    4957.5990 N |Latitude:  4957.5990        |
| GP 16   16 192 43 35  Y|Longitude:  00815.5007 E |Longitude: 00815.5007       |
| GP 26   26 179 18 29  Y|Speed:    0.16           |Altitude:  178.53           |
| GL  1   65 108 37 19  Y|Course:   0.0023         |Quality:   1   Sats: 05     |
| GL 14   78 220 24 29  Y|Status:   A        FAA:A |HDOP:      3.5              |
| GP  7    7 302 11  0  N|MagVar:                  |Geoid:     47.9             |
| GP  8    8 298 43  0  N---------- RMC ---------------------- GGA ------------
| GP 13   13  10  2  0  N------------------------------------------------------
| GP 15   15  39 11  0  N|Mode: A3 Sats: 10 16 26 +|UTC:           RMS:        |
| GP 18   18  65 17  0  N|DOP H=3.5  V=3.4  P=4.9  |MAJ:           MIN:        |
| GP 21   21 254 22  0  N|TOFF:  0.127466383       |ORI:           LAT:        |
| GP 23   23  70 49  0  N|PPS: N/A                 |LON:           ALT:        |
--------- GSV ----------------- GSA + PPS -------------------- GST -------------
```

```
(64) $GLGSV,3,1,09,88,64,038,,81,44,292,,79,42,281,,65,37,108,19*62
(64) $GLGSV,3,2,09,72,31,045,,78,24,220,29,87,22,075,,66,05,156,*63
```

5.3.3 Temperature Sensors

The Hyrax-1000 has several temperature sensors.

1. There are two *LM75* sensors. One is placed nearby the center of the CPU board and used by the board controller (see: 3.1.1 Board management controller). The other is located at the edge of the CPU board and currently not used by the software.
2. The CPU has one sensor per core and one additional sensor for the CPU package.

All temperatures can be easily read out via the program *sensors* on the command line.

```
$ sensors
coretemp-isa-0000
Adapter: ISA adapter
Package id 0:  +32.0°C  (high = +110.0°C, crit = +110.0°C)
Core 0:        +32.0°C  (high = +110.0°C, crit = +110.0°C)
Core 2:        +32.0°C  (high = +110.0°C, crit = +110.0°C)

lm75-i2c-4-48
Adapter: i2c-ocores
temp1:        +45.0°C  (high = +80.0°C, hyst = +75.0°C)

acpitz-acpi-0
Adapter: ACPI interface
temp1:        +24.0°C  (crit = +125.0°C)

lm75-i2c-0-48
Adapter: SMBus I801 adapter at 4040
temp1:        +45.5°C  (high = +105.0°C, hyst = +82.0°C)
```

The CPU temperature and the temperature of the optional SATA disk are checked automatically by the script */usr/sbin/temperature-check*.

This script is started every 2 minutes by *systemd*. The maximum temperatures are set in the unit file *temperature-check.service*:

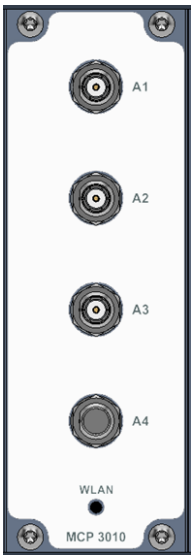
```
[Unit]
Description=CPU and HDD temperature check

[Service]
ExecStart=/usr/sbin/temperature-check --hdd-device=/dev/sda --max-cpu-temp=100 --max-hdd-temp=85 --log-level=error
```

If one of the temperatures is to high, the script sends an overheat event to the board controller. The board controller powers the system off for at least 5 minutes.

See 3.1.1 Board management controller for details.

5.3.4 WiFi Support

The image below shows a WiFi module. See the *Installation Manual* for details.



WiFi module

The WiFi module supports the client mode, access point mode and ad-hoc networks. In this manual we describe client network only, but as the operating system is derived from a standard Linux, other modes can be set up by the customer.

A client network can be set up easily using the Network-Manager.

Run `sudo nmtui` in a console window and select *Activate a connection*:

```
----NetworkManager TUI ----
|                          |
| Please select an option  |
|                          |
| Edit a connection        |
| Activate a connection    |
| Set system hostname      |
|                          |
| Quit                     |
|                          |
|                    <OK>  |
---------------------------
```

Then choose a WiFi network - in this case *example_net*. (If there are more than one WiFi cards in the system, there will be more WiFi entries.):

```
-------------------------------------------------
| ---------------------------------              |
| | Ethernet (enp13s0)            | <Activate>   |
| | * Wired connection 1          |              |
| |                               |              |
| | Ethernet (enp3s0)             |              |
| | * Wired connection 2          |              |
| |                               |              |
| | Ethernet (enp4s0)             |              |
| |   Wired connection 3          |              |
| |                               |              |
| | Wi-Fi                         |              |
```

```
| |   example_net         ****    | <Back>       |
| --------------------------------               |
-------------------------------------------------
```

Enter the password:

```
--------- Authentication required by wireless network ----------
|                                                                |
| Passwords or encryption keys are required to access the      |
| wireless network 'example_net'.                                |
|                                                                |
|     Password _____________________________                     |
|                                                                |
|                                              <Cancel> <OK> |
------------------------------------------------------------------
```

The configuration files are stored in the directory */etc/NetworkManager/system-connections/*. The directory is in the overlay filesystem so they outlast a reboot.

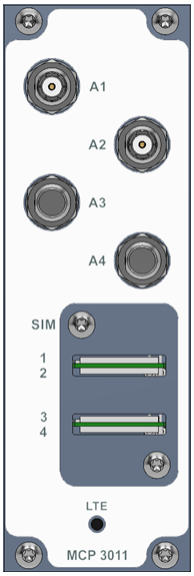
In this case the configuration file will be called *example_net.nmconnection* and contains the following data (password is left out):

```
[connection]
id=example_net
uuid=30fb35aa-79a5-4664-a7f8-76b68207adc1
type=wifi
interface-name=wlp8s0
permissions=

[wifi]
mac-address-blacklist=
mode=infrastructure
ssid=example_net

[wifi-security]
auth-alg=open
key-mgmt=wpa-psk
psk=******************

[ipv4]
dns-search=
method=auto

[ipv6]
addr-gen-mode=stable-privacy
dns-search=
method=auto

[proxy]
```

5.3.5 Cellular Radio Support

The image below shows a Cellular Radio module. The cover of the SIM slots is opened to show the slots.

See the *Installation Manual* for details.



Cellular Radio module

The simplest method to establish a broadband connection is by using the command-line interface of *Network Manager* known as *nmcli*.

The following block shows an example.

- "thename" is the name of the connection
- we assume "1234" as SIM card PIN
- In the example the APN "internet.telekom" of our SIM-Card carrier provider is used. Please enter/use the correct APN of your carrier provider.
- To find out which "ifname" we need for the connection, *mmcli* can be used. We look for the *primary port* in the output of *mmcli*.

```
# Find out which is the modem number
$ mmcli -L
/org/freedesktop/ModemManager1/Modem/0 [Sierra Wireless, Incorporated] MC7421

# Look for the primary qmi port
$ mmcli -m 0
-----------------------------
General  |         dbus path: /org/freedesktop/ModemManager1/Modem/0
         |         device id: 8ccede40d9d9213b44d67bad5f95f761b78badc4
-----------------------------
# output omitted
-----------------------------
System   |           device: /sys/devices/pci0000:00/0000:00:15.0/usb1/1-4/1-4.1
         |          drivers: qmi_wwan, qcserial
         |           plugin: sierra
         |      primary port: cdc-wdm1
         |            ports: cdc-wdm1 (qmi), ttyUSB6 (qcdm), ttyUSB7 (gps),
         |                   ttyUSB8 (at), wwan0 (net)
-----------------------------
# output omitted
```

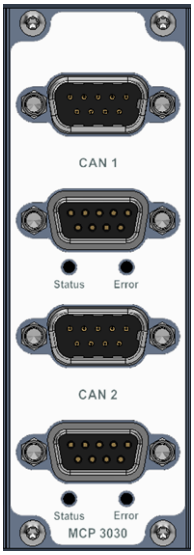
Now set up the connection:

```
# Set up the connection
sudo nmcli connection add type gsm ifname cdc-wdm1 con-name thename apn internet.telekom gsm.pin 1234
    Connection 'thename' (e66043fa-4220-4f2a-8be0-9e12594c7fef) successfully added.
# Show the connection state
sudo nmcli connection show thename
# Shut down the connection
sudo nmcli connection down thename
# Start the connection again
sudo nmcli connection up thename
# Revove the connection including the configuration file
sudo nmcli connection delete thename
```

The configuration files are stored in the directory */etc/NetworkManager/system-connections/*. The directory is in the overlay filesystem so they outlast a reboot.

5.3.6 CAN Bus Support

The image below shows a CAN module. See the *Installation Manual* for details.



CAN module

The CAN module provides two SocketCan ports being capable of CAN FD mode. For an overview of SocketCan see e.g.: https://en.wikipedia.org/wiki/SocketCAN or the documentation in the Linux kernel: https://www.kernel.org/doc/html/latest/networking/can.html.

The interfaces *can0* and *can1* can be configured manually on the command line or using *systemd-networkd*.

To set up the the CAN port manually, the command `ip` is used.

```
$ sudo ip link set can0 up type can dbitrate 5000000 bitrate 2000000 fd on
```

For startup at boot time, create (as *root*) a file named *canX.network* in */etc/systemd/network* ('X' denotes the CAN device index 0,1,…).

See the documentation of systemd-networkd.

```
# /etc/systemd/network/can0.network
[Match]
Name=can0
```

```
[CAN]
# Same as: sudo ip link set can0 up type can dbitrate 5000000 bitrate 2000000 fd on
BitRate=2000000
DataBitRate=5000000
FDMode=yes
```

To show the current setting of the CAN connections, *ip* can be used again.

```
$ ip -details link show can0
9: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 72 qdisc pfifo_fast state UP mode DEFAULT group default qlen 10
    link/can  promiscuity 0 minmtu 0 maxmtu 0
    can <FD> state ERROR-ACTIVE (berr-counter tx 0 rx 0) restart-ms 0
        bitrate 2000000 sample-point 0.750
        tq 25 prop-seg 7 phase-seg1 7 phase-seg2 5 sjw 1
        mcp251xfd: tseg1 2..256 tseg2 1..128 sjw 1..128 brp 1..256 brp-inc 1
        dbitrate 5000000 dsample-point 0.750
        dtq 25 dprop-seg 2 dphase-seg1 3 dphase-seg2 2 dsjw 1
        mcp251xfd: dtseg1 1..32 dtseg2 1..16 dsjw 1..16 dbrp 1..256 dbrp-inc 1
        clock 40000000 numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
```

5.4 Build Software from Source

This chapter describes the contents of the binary and source shipment and how to build the binaries from source. The source code for the complete Hyrax-1000 operating system is available.

To build everything from source, in addition to the source archive a Debian mirror or a copy of the Debian mirror is needed.

See 1.1.4.3 Obtaining License Texts and Sources for details.

5.4.1 Shipment Details

The sources are shipped in a compressed TAR archive.

The binary shipment consists of an ISO image of the installation software, a *RAUC* bundle for updates and the release notes.

The ISO image can be copied to a USB stick. Please use e.g.: Rufus under Windows in “file mode”. Do not use the “dd” mode copying the image to a USB stick.

The *RAUC* bundle can be used to update an existing installation.

The license texts for all programs contained in the OS are available in the running system. For convenience the license texts are added to the shipment too.

The following files are taken as an example.

Debian-11-amd64-mcr-xfce-v22.44.iso

This is an ISO filesystem which can be copied to an USB drive.

Use e.g. Unetbootin under Linux or Rufus under Windows. In any case use the *ISO image* mode, not the *DD* mode.

Press “ESC” during boot to change the systems boot order to boot from this USB drive.

Upon system startup, the standard operating system launches and operates as it typically does on the Hyrax-1000. In a console window, the command `sudo sysupgrade --setup` can be used to install the OS on the Hyrax-1000.

mcr-v22.44.raucb

The RAUC bundle is used to update an existing system. See 5.1.3.1 Update the system software for details.

pcmcr_release_notes_v22.44.pdf

These are the release notes.

licenses.txt.gz

The licenses of all software can be found on the running system in the files "/usr/share/doc//*copyright*", *where* "" denotes the package name.

This file is gathered only for convenience.

mcr_source-v22.44.tar.gz

This archive contains all sources to build the RAUC bundle and ISO image.

Additionally access to a Debian mirror is needed (see 5.4.2 Build the Binaries from Source).

5.4.2 Build the Binaries from Source

As development operating system Debian 11 or Ubuntu 20.04 is fine to build from source. Windows 10 or 11 running WSL may work too.

There are some packages to be installed (*build-essential* and others).

There is a script `scripts/check_prerequisites.sh` in the source archive which checks, whether all packages are installed.

5.4.2.1 Short version of installation hints

1. Be sure the development PC is connected to the internet.
2. Unpack the source archive on the development PC.
3. Open a console window and change to the unpacked sources directory.
4. Check for prerequisites using *MIRROR_CONFIG=debian-de ./scripts/check_prerequisites.sh*. For *MIRROR_CONFIG=debian-de* see the long version.
5. Install all packages that are missing.
6. Run the build script *MIRROR_CONFIG=debian-de scripts/build_binary_from_deliv.sh -a -f path/file.tar.gz*. "path/file.tar.gz" is the complete path and filename of the archive unpacked prior.
7. The script *build_binary_from_deliv.sh* now build the tools in the subdirectory *tools*, the Linux kernel in the subdirectory *linux*, copies all Debian packages created to the directory *filesystem* and creates the ISO image and RAUC update bundle.
8. All files created are copied to the subdirectory *deliv* in the unpacked sources directory.

5.4.2.2 Longer version of installation hints

5.4.2.2.1 Contents of the source archive

Assuming we have unpacked the archive `mcr_source-v22.44.tar.gz`, the first directory level looks as follows:

```
$ tree -L 1 --charset=ascii
.
|-- README.md
|-- VERSION
|-- doc
|-- filesystem
|-- linux
|-- scripts
`-- tools
```

README.md

This README include some information about the archive.

VERSION

This file contains the version number of the system. In this case it is “v22.44”.

doc

This directory is a leftover and not needed. It may be removed in the future.

scripts

This directory contains the scripts needed to build the system and will be explained below.

tools

This directory contains different tools and programs, which are used in the final image.

linux

This directory contains everything to build the Linux kernel used for the system.

filesystem

This directory contains everything to build the final ISO image and Rauc update bundle.

5.4.2.2.1.1 Subdirectory “tools”

The subdirectory “tools” contains several scripts and programs needed on the final system:

```
$ tree -L 1 --charset=ascii tools/
tools/
|-- BUILD_TOOLS.sh
|-- cp210xsmt
|-- custom_eeprom
|-- eltec_revision_eeprom
|-- librs232
|-- mvbc_lib
|-- reset_button_service
|-- status_led_service
`-- upd72020x-load
```

Each one is compiled, packed as Debian archive and copied to the folder “filesystem/packages/eltec_packages” during the build process.

Each directory contains a file “README.md” explaining the purpose of the tools.

BUILD_TOOLS.sh

This script can be used to build all tools manually.

5.4.2.2.1.2 Subdirectory “linux”

The directory “linux” contains everything to compile the Linux kernel and create Debian packages for installation:

```
$ tree -L 2 --charset=ascii linux
linux
|-- BUILD_LINUX.sh
|-- COMMON.sh
|-- INSTALL_MODULES.sh
|-- MAKE_SELECTED_MODULES.sh
|-- README.md
|-- VERSION
|-- config
|   `-- mcr-5_10.config
|-- download
|   `-- linux-5.10.142.tar.xz
```

```
`-- patches
    `-- linux-5.10
```

VERSION

This file contains the sub version number of the kernel. It is added to - in this case - “5.10.142”, so the final version is e.g.: “5.10.142-mcr-v22.44”

BUILD_LINUX.sh

This script is used to build the kernel manually.

INSTALL_MODULES.sh, MAKE_SELECTED_MODULES.sh

These scripts are useful only during the Westermo Eltec development process.

COMMON.sh

This script is included by *BUILD_LINUX.sh*. It contains the definition of some environment variables concerning the version and name for the Linux kernel used.

If another version than “5.10.142” is used, this file needs to be adapted.

config

This directory contains the Linux kernel configuration.

download

This directory contains the Linux kernel source archive downloaded from http://kernel.org.

patches/linux-5.10

There are several patches applied to the vanilla kernel before it is built.

Only the patches in *linux-5.10* are used, there may be subdirectories with unused patches.

The script *BUILD_LINUX.sh* performs the following steps.

1. The archive (e.g.: linux-5.10.142.tar.xz) is unpacked and so a directory “linux-5.10.142” is created.
2. The file “VERSION” is used to create the file “.scmversion” in “linux-5.10.142”.
3. A new git repository is created in this directory.
4. The patches are copied to the directory and applied using `git am *.patch`
5. The configuration file is copied as “.config” and `make oldconfig` is called.
6. The build process is started using `make deb-pkg` to build the kernel and the corresponding Debian packages.

5.4.2.2.1.3 Subdirectory “filesystem”

The directory “filesystem” contains a lot of scripts and subdirectories. Only a few are described here. See 5.4.2.2.2 Filesystem build details for details concerning the build process of the filesystem Squashfs file.

Normally the scripts in this directory do not need to be used directly. The main build script uses them (see 5.4.2.2.1.4 Subdirectory “scripts”).

```
$ tree -L 1 --charset=ascii filesystem
filesystem
|-- CHECK_INSTALL.sh
|-- CHROOT_FILESYSTEM.sh
|-- CLEANUP.sh
|-- COMMON.cfg
|-- COMMON.sh
|-- CREATE_ISOIMAGE.sh
|-- CREATE_SQUASHFS.sh
|-- DEBOOTSTRAP.sh
```

```
|-- FUNCTIONS.sh
|-- GATHER_LICENSES.sh
|-- MAKE_PARTITIONS.sh
|-- RAUC_CREATE_BUNDLE.sh
|-- REBUILD.sh
|-- UNETBOOTIN.sh
|-- config
|-- log
|-- packages
|-- part_efi_boot
|-- part_grub_boot
|-- part_img
|-- part_iso
|-- rauc
`-- scripts
```

CHECK_INSTALL.sh

This script checks and shows, if all packages needed for development are installed.

REBUILD.sh

This script calls all other required scripts step by step. As some scripts need *root* access *sudo* is used.

Be sure to have copied the Debian packages for tools and Linux kernel to *packages/eltec_packages* before starting this script.

The script creates the installation ISO image and Rauc bundle in the subdirectory *images*.

COMMON.sh

This script is included by most other scripts. It sets a lot of environment variables which are normally not changed.

If another Linux kernel version is used, the entry `DEBIAN_KERNEL_PREFIX="5.10.142-mcr"` must be changed!

COMMON.cfg

This file is likely to be changed by customers. It contains the location of the Debian mirror used. In our case it is "http://ftp.de.debian.org/debian", which may be unsuitable in different countries.

5.4.2.2.1.4 Subdirectory "scripts"

```
$ tree -L 1 --charset=ascii scripts
scripts
|-- GETVERSION.sh
|-- build_binary_from_deliv.sh
|-- build_changelog.sh
|-- build_kernel.sh
|-- build_tools.sh
|-- check_prerequisites.sh
|-- create_bundle.sh
|-- create_source_archive.sh
|-- gendeliv.cfg
`-- slog.sh
```

build_changelog.sh, create_bundle.sh, create_source_archive.sh

Used for Westermo Eltec delivery only.

check_prerequisites.sh

Checks, if all packages needed for development are installed.

GETVERSION.sh

Get the version of the sources. Uses the Westermo Eltec git repository tag of the file *VERSION*.

build_kernel.sh

Builds the kernel only. Used by `build_binary_from_deliv.sh`.

build_tools.sh

Builds the tools only. Used by `build_binary_from_deliv.sh`.

build_binary_from_deliv.sh

Main build script. Builds everything. It needs the complete path of the still packed source archive as parameter: `MIRROR_CONFIG=debian-de scripts/build_binary_from_deliv.sh -a -f path/file.tar.gz`.

All files created are copied to the subdirectory *deliv* in the unpacked sources directory. There is a subdirectory *log* containing the files created by the script. These log files may be useful in case of errors.

5.4.2.2.2 Filesystem build details

The systems filesystem is build using the Debian program debootstrap.

Normally all scripts are called by *build_binary_from_deliv.sh*, but for test purposes, they can be called manually one by one. Have a look at *filesystem/REBUILD.sh*.

The first script *filesystem/DEBOOTSTRAP.sh* uses a two step approach to create the filesystem.

First *debootstrap* creates a minimal image, downloading packages from the Debian - or hard disk - mirror. The resultant directory is *filesystem/images/default_target/debian_fs*.

When the first step is finished, the files from *filesystem/config/default_target/debootstrap_overlay* and the Debian packages from *filesystem/packages/eltec_packages* are copied to *…default_target/debian_fs*.

In a second step *chroot* is used to enter the directory created. A second stage script *debootstrap-stage2.sh* installs additional packages described in *filesystem/config/default_target/dpkg-selections*.

When *filesystem/DEBOOTSTRAP.sh* has finished, the directory *filesystem/images/default_target/debian_fs* contains the system root file system.

Now *filesystem/CREATE_SQUASHFS.sh* creates the final Squashfs image archive. The Squashfs archive serves as read only file system on the Hyrax-1000.

filesystem/MAKE_PARTITIONS.sh gathers all files needed for the ISO image.

filesystem/CREATE_ISOIMAGE.sh creates the ISO image suitable for booting from an USB stick.

filesystem/RAUC_CREATE_BUNDLE.sh creates the Rauc bundle suitable for updates.

5.4.3 Add own applications

The cleanest way to add an application is to create a Debian package.

A Debian package has the advantage, that it can be installed onto a running system using *dpkg -i package-name.deb*.

The disadvantage of this approach is, that the package does not survive a factory reset.

To add a Debian archive persistently to the system, it must be added during the build process of the image. The package must be copied to *filesystem/packages/eltec_packages* and the ISO image and Rauc update bundle must be recreated using the script *filesystem/REBUILD.sh*.

For example see the source archives subdirectory “tools”.

The script “BUILD_TOOLS.sh” recursively enters all subdirectories and builds the corresponding package. All projects are built using `CMake`, so all subdirectories can be build in the same manner.

Afterwards the resultant Debian packages are copied to *filesystem/packages/eltec_packages*.

When the software is rebuild using *filesystem/REBUILD.sh*, all packages are automatically installed.

6 Appendix

6.1 Open Issues

7 Frequently asked Questions

7.1 Which is the system time, if the power CAP supply is discharged?

The system has no battery for backing up the clock, but a power capacitor. After some time, the power cap is empty.

When booting with an empty power cap, the system comes up with a default time which is the last access time of the file */usr/lib/clock-epoch*.

When the system image is build, this files time is set to the build date and time. When the system is running, the file is touched periodically by systemd.

As the file is stored in the overlay filesystem, the system comes up with a reasonable time and date.

After a factory reset, the file access time is the build time again.

7.2 What are the default values set after factory reset?

The default user is *user* with password *user*.

The root password is *root*.

LAN 1 has the static IP address *192.168.100.1*. *LAN 2* and *LAN 3* use DHCP as default.

7.3 How can the hardware revision be detected?

The system has a revision EEprom where hardware revision, serial number and other data are stored.

See 5.2.6 Revision EEProm for details.

7.4 How can the systems software revision be detected?

To read the operating system version, dump the file */etc/os-version*.

```
$ cat /etc/os-release | grep IMAGE_VERSION
IMAGE_VERSION="v22.44"
```

The Linux kernel version can be printed using *uname -r*.

```
$ uname -r
5.10.142-mcr-v22.44
```

7.5 How can the default keyboard layout be changed?

The system comes up with an English keyboard layout.

This can be changed using e.g. *setxkbmap de* for a German keyboard in a console window. After reboot, the keyboard layout is switched to English again.

To change this permanently (at least until next factory reset), use the command *localectl set-keymap de* for example for a German keyboard.

This changes the file */etc/default/keyboard*. The change is effective after next login.

```
$ cat /etc/default/keyboard
XKBMODEL=pc105
XKBLAYOUT=de
XKBOPTIONS=terminate:ctrl_alt_bksp
BACKSPACE=guess
```

7.6 How can we use serial port 1 for console redirection?

Serial port 1 can be used as debug port, if no monitor is connected.

This is normally not enabled, to be able to serial port 1 to gather data.

Even the BIOS output can be redirected to the serial port. See 4.3 Redirect BIOS output to serial port 1 for details.

For Linux output to be redirected to the serial port, the following parameters must be added to the kernel command line:

```
console=ttyS0,115200n8
```

To see this in action without changing your system, boot from the installation USB stick (4.2 Setup USB boot).

Choose “USB stick boot with serial output”.

USB boot grub menu

You may even select a line and hit ‘e’ to see the difference between the two boot commands.

When booting a shell is opened over the serial port.

The same temporary change can be made with the Linux on the Hyrax-1000.

Stop the boot loader and edit the command line before booting.

To make this change permanently the file “filesystem/config/default_target/grub-rauc-ar.cfg.tmpl” must be changed when building the system from source (see 5.4 Build Software from Source).

7.7 Can the wireless country code be changed?

This depends on the setting of the *country code* in the 5.2.6 Revision EEProm .

If the country code is “00”, a valid country code (e.g.: ‘US’) can be entered in the file */etc/default/crda* as “regdomain” (regulatory domain).

In the example below, a German regdomain is added.

```
# Set REGDOMAIN to a ISO/IEC 3166-1 alpha2 country code so that iw(8) may set
# the initial regulatory domain setting for IEEE 802.11 devices which operate
# on this system.
#
# Governments assert the right to regulate usage of radio spectrum within
# their respective territories so make sure you select a ISO/IEC 3166-1 alpha2
# country code suitable for your location or you may infringe on local
# legislature. See `/usr/share/zoneinfo/zone.tab' for a table of timezone
# descriptions containing ISO/IEC 3166-1 alpha2 country codes.

REGDOMAIN=DE
```

If the country code in the revision-eeprom is not “00”, this value will be written into */etc/default/crda* every time the wireless drivers are loaded, thus overwriting any other value.

Index

A

Activity LEDs

B

BIOS console redirection

Boot process

C

Cellular radio client

Central Regulatory Domain Agent

Console redirection

Crda

D

Debug port

Default IP

Default password

F

Factory reset

Factory reset defaults

H

Hardware revision

I

Image creation

M

MVB software

O

Open Source Licenses

P

pcmcx-wwan-setup.yaml

R

Reset button handling

Restore filesystem configuration

S

Save filesystem configuration

Serial port configuration

Software version

Status led

System time without timeserver

System update

U

USB boot setup

W

Watchdog and Systemd

WiFi client

Wireless country code