www.vscom.de

OnRISC
User Manual
Edition: May 2022

Vision Systems GmbH
Tel: +49 40 528 401 0
Fax: +49 40 528 401 99
Web: www.visionsystems.de
Support: faq.visionsystems.de

The software described in this manual is furnished under a license agreement and may be used only in accordance with the terms of that agreement.

Copyright Notice

Trademarks

VScom is a registered trademark of Vision Systems GmbH. All other trademarks and brands are property of their rightful owners.

Disclaimer

Vision Systems reserves the right to make changes and improvements to its product without providing notice.

Vision Systems provides this document “as is”, without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. Vision Systems reserves the right to make improvements and/or changes to this manual, or to the products and/or the programs described in this manual, at any time.

Information provided in this manual is intended to be accurate and reliable. However, Vision Systems assumes no responsibility for its use, or for any infringements on the rights of third parties that may result from its use.

This product might include unintentional technical or typographical errors. Changes are periodically made to the information herein to correct such errors, and these changes are incorporated into new editions of the publication.

Contents

List of Figures

List of Tables

1. Introduction

1.1. OnRISC Family

The OnRISC is an ARM-based RISC industrial embedded computer family. The great variety of interfaces like LAN, CFast, microSD/SD, USB, CAN, I²C, serial interface, and digital I/O makes it easy to connect various industrial devices to the OnRISC. Some new OnRISC devices provide graphical display (VS-860 has built-in 8" LCD display).

Compact dimensions and DIN Rail mounting capability make the OnRISC to a space saving and flexible mounting industrial computer. It is feasible to be installed even in space limited environments.

Due to RISC based architecture, the OnRISC has very small power consumption, so fanless heat dissipation is possible. Working in an extended temperature range, the OnRISC can be used under harsh industrial conditions. Therefore the OnRISC is downright designed for industrial automation. Refer to the Hardware Manual for exact characteristics.

The embedded computer runs full-featured Debian GNU/Linux on ARM operating system. With Debian's repository database it is easy to install and update the free software on the OnRISC. The OnRISC is capable to act directly as a software development host, Web, Mail, Print and Database server or as a desktop computer with the X11 window manager and many more.

1.2. How to Read the Manual?

Hardware Manual should be consulted first to understand, how the device is going to be powered. Section Getting Started will get you from power on till log on and basic network configuration. Section Debian Maintenance Notes describes how to search/install/remove Debian packages. Section Network Services and Tools Provided by OnRISC shows how to set up various network services like SSH, ser2net, socketcand, etc.

Software development related sections are spread over the manual and touching different aspects of embedded software development for OnRISC devices. Section Software Development gives step-by-step instructions, how to set up a development environment, get source code for the Linux kernel, etc. Section OnRISC Hardware API gives an overview of hardware interfaces and how they can be accessed in software. Section Recommended Books provides a list of books about Linux administration and programming.

Sections Buildroot, Yocto and OpenWrt introduce other embedded Linux distributions built from scratch, that can be used instead of Debian.

OnRISC Model	Baltos iR 5221 Baltos iR 3220	Baltos iR 2110	Baltos 1080
CPU	AM335x (ARM Cortex-A8)	AM335x (ARM Cortex-A8)	AM335x (ARM Cortex-A8)
RAM	256MB DDR3	256MB DDR3	256MB DDR3
Flash Memory on Board	256MB NAND	256MB NAND	256MB NAND
Serial Interfaces	2 x RS232/RS422/RS485 (16C750)	1 x RS232/RS422/RS485 (16C750)	8 x RS232 (FTDI)
CAN Interface on board	up to 1 x	N.A.	N.A.
Digital I/O channels	4 x inputs, 4 x outputs	N.A.	8 x inputs, 8 x outputs
CFast slot	N.A.	N.A.	N.A.
SD-Slot	1 x external slot	1 x external microSD-slot	1 x internal microSD-slot
USB	2 x USB 2.0 as Host, up to 1 x as OTG	1 x USB 2.0 as Host	N.A.
Expansion Slot	Mini PCI Express	N.A.	N.A.
Ethernet	1 x Gbit, up to 4 x 100Mbit switch	1 x Gbit, 1 x 100Mbit	1 x Gbit (WAN)
I²C bus	1 x	N.A.	N.A.
RTC	1 x	1 x	1 x
Watchdog Timer	1 x	1 x	1 x
Display	N.A.	N.A.	N.A.

Table 1: OnRISC Products Based on OMAP3 am335x SoC

OnRISC Model	OnRISC 113 OnRISC 213	OnRISC 413 OnRISC 813	OnRISC CAN
CPU	AM335x (ARM Cortex-A8)	AM335x (ARM Cortex-A8)	AM335x (ARM Cortex-A8)
RAM	256MB DDR3	256MB DDR3	256MB DDR3
Flash Memory on Board	256MB NAND	256MB NAND	256MB NAND
Serial Interfaces	up to 2 x RS232/RS422/RS485 (16C750)	up to 8 x RS232/RS422/RS485 (FTDI)	N.A.
CAN Interface on board	N.A.	N.A.	1 x
Digital I/O channels	N.A.	N.A.	N.A.
CFast slot	N.A.	N.A.	N.A.
SD-Slot	1 x internal microSD-slot	1 x internal microSD-slot	1 x internal microSD-slot
USB	1 x USB 2.0 as Host	1 x USB 2.0 as Host	1 x USB 2.0 as Host
Expansion Slot	N.A.	N.A.	N.A.
Ethernet	1 x Gbit (WAN)	1 x Gbit (WAN)	1 x Gbit (WAN)
I²C bus	N.A.	N.A.	N.A.
RTC	1 x	1 x	1 x
Watchdog Timer	1 x	1 x	1 x
Display	N.A.	N.A.	N.A.

Table 2: OnRISC Products Based on OMAP3 am335x SoC

OnRISC Model	VS-860
CPU	AM3517 (ARM Cortex-A8)
RAM	256MB DDR2
Flash Memory on Board	256MB NAND
Serial Interfaces	2 x RS232/RS422/RS485 (FTDI)
CAN Interface on board	1 x
Digital I/O channels	N.A.
CFast slot	1 x
SD-Slot	1 x external slot
USB	2 x USB 2.0 as Host, 1 x as OTG
Expansion Slot	Mini PCI Express
Ethernet	2 x
I²C bus	N.A.
RTC	1 x
Watchdog Timer	1 x
Display	built in LCD

Table 3: OnRISC Products Based on OMAP3 am3517 SoC

2. Getting Started

2.1. Connect to OnRISC via Serial Link

Connect the OnRISC to the serial port of your PC and start a terminal software (HyperTerminal, ZOC[1], minicom etc) with 115200,8,n,1 settings (no hardware/software handshake is needed. Set the terminal type according to Section 2.2). Insert a SD/microSD-card with a preinstalled system (refer to Section 4). Power your OnRISC according to the Hardware Manual. You'll see Linux booting. After the boot procedure you'll be asked to log in. As only super user (root) is available use following credentials to login:

```
Debian login:  root
Password:  linux
```

2.2. Terminal Type

Terminal type is defined in the environment variable `TERM` and is set to `TERM=linux` by default. The terminal type of your terminal application (HyperTerminal, ZOC, minicom etc.) should be set to the same type to interact correctly with the OnRISC console. If `linux` terminal type is not available in your software, `vt100` can be used instead. To do this add following line to the `~/.bashrc`:

```
export TERM=vt100
```

[1] www.emtec.com

2.3. Configure Network

2.3.1. Legacy Configuration

Network interfaces can be configured by editing `/etc/network/interfaces`. The IP addresses for `eth0`, `eth1` and `wlan0`[2] are statically assigned by default (see the Listing below). Please refer to Section 5.1 for detailed hardware network interface mapping, i.e. what network device name corresponds to what physical connector.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.254.254
        netmask 255.255.255.0

# The secondary network interface
auto eth1
iface eth1 inet static
        address 192.168.253.254
        netmask 255.255.255.0

# The wireless interface
#auto wlan0
iface wlan0 inet static
        address 192.168.127.254
        netmask 255.255.255.0
        wpa-driver nl80211
        wpa-conf /etc/wpa_supplicant.conf
```

Listing 1: /etc/network/interfaces

The wireless interface `wlan0` will be configured with the `wpa_supplicant` utility (see Section 5.2).

Setup gateway and DNS server Assuming your router has IP address 192.168.254.1 the OnRISC will be configured in the following way:

1. change `eth1` section of `/etc/network/interfaces` file
   ```
   auto eth1
   iface eth1 inet static
   address 192.168.254.254
   netmask 255.255.255.0
   gateway 192.168.254.1
   ```
2. insert following line to the `/etc/resolv.conf`[3]
   ```
   nameserver 192.168.254.1
   ```

[2] to activate wlan0 uncomment the `#auto wlan0` line in `/etc/networking/interfaces`
[3] see `man resolv.conf` for explanations

3. execute `/etc/init.d/networking restart`

2.3.2. Configuration via systemd-networkd

`systemd-networkd`[4] is systemd's network configuration service. It is disabled by default and can be started with following commands:

```
systemctl enable systemd-networkd
systemctl start systemd-networkd
systemctl enable systemd-resolved
systemctl start systemd-resolved
echo "nameserver 127.0.0.53" > /etc/resolv.conf
```

Following example shows how to configure `eth1` to get IP address via DHCP. In this case `systemd-networkd` will also take care of `eth1` link state, i.e. if you replug the network cable `systemd-networkd` will automatically restart DHCP client. Create `/etc/systemd/network/eth.network` wit this content:

```
[Match]
Name=eth1
[Network]
DHCP=yes
```

Now restart `systemd-networkd`:

```
systemctl restart systemd-networkd
```

DNS name resolution will be handled by `systemd-resolved.service`[5]. The example above shows how to use this service together with `/etc/resolv.conf` file. The service will be configured via `/etc/systemd/resolved.conf`[6] file. In this configuration file you can for example configure, which DNS servers will be configured in cases where no DNS server information is supplied. To disable automatic configuration of the fallback DNS servers you must uncomment the line with *FallbackDNS* and leave the list empty:

```
FallbackDNS=
```

See Arch-Linux wiki[7] for more examples.

[4] https://www.freedesktop.org/software/systemd/man/systemd-networkd.html
[5] https://www.freedesktop.org/software/systemd/man/systemd-resolved.service.html
[6] https://www.freedesktop.org/software/systemd/man/resolved.conf.html
[7] https://wiki.archlinux.org/index.php/Systemd-networkd

3. Supported Linux Distributions

Following Linux distributions are provided for OnRISC devices:

- Debian
- Buildroot
- Yocto
- OpenWrt

Each distribution has its own advantages and disadvantages, so one has to consider, what properties are important for the project in question[8]. Below you'll find comparison of these distributions and suggestions for bringing your project on OnRISC.

3.1. Debian

Debian is well known and established Linux distribution in the desktop/server world. So working with its ARM port doesn't really differ from working with your PC. Debian for ARM provides the same packages and tools. So it is best suited to start developing your software with it. Open Source dependency installation from Debian repositories is very fast and can be done at any time. And if you design your software to create a *.deb package, you'll get an update mechanism right away. But this flexibility has its price: your root file system exceeds the size of 1GB very quickly depending on required software dependencies (libraries, frameworks). This makes it almost unsuitable for NAND flash based projects.

3.2. Buildroot

Buildroot is a set of Makefiles and patches, that compiles root file system from scratch. So initial stage can take more than hour to compile depending on hardware setup of your PC. Buildroot doesn't provide binary package management system like Debian, so adding new packages will require rewriting root file system on target medium (NAND or SD/microSD-card) again. But particularly this approach leads to relative small core system. Depending on what software dependencies your software has, you can get an initramfs image (kernel and rootfs) of about 10MB. Such approach makes firmware update procedure quite straight forward: start the system and just replace `kernel-fit.itb`. That's all.

3.3. Yocto

Yocto is also a set of Makefiles and patches, that compiles root file system from scratch. But compared to Buildroot it also creates re-distributable binary packages (rpm ,deb), so that one can easily update a rootfs in production.

[8] These slides provided by Mind compare Debian and Buildroot as embedded Linux distributions: https://www.yumpu.com/en/document/view/39437530/presentation-emdedded-distro-shootout-buildroot-vs-debian-mind

3.4. OpenWrt

OpenWrt is network agnostic embedded Linux OS, that will be built from scratch as Buildroot and Yocto do. Compared to the above mentioned Linux distributions OpenWrt is based around a web interface and related infrastructure and will be often used in routers and other network related devices.

3.5. Distribution Choice

First of all a decision must be made, what medium to use NAND flash or SD/microSD-card. If NAND flash, then Buildroot, Yocto or OpenWrt are the right choices. For SD/microSD-card you still have alternative and other factors play role. But regardless of distribution choice, it is easier to start developing with Debian. Because you can experiment with different libraries/frameworks and so determine required set of software dependencies. This set can then be used either to configure Buildroot, Yocto, OpenWrt or Debian BSPs (script, that creates custom Debian root file system on your host PC).

4. Software Configuration

The OnRISC comes with a preinstalled Debian GNU/Linux on ARM[9] operating system. The system image (see Section 4.5), that can be either downloaded or obtained as part of the Starter kit, provides necessary tools and services to start with application development, various services such as ssh, RFC2217 etc.

This image can be downloaded from our FTP server[10] and can then be copied to the SD/microSD-card as described in Appendix D.1.

4.1. Init System

Debian uses `systemd`[11] as the default init system. `systemd` differs in many points from System V like init system, but it maintains a compatibility layer so that legacy start-up scripts still can be used. For example, you can still configure network interfaces via `/etc/network/interfaces` (default for our Debian image), but the new approach is to remove `/etc/network/interfaces` and configure the network via `networkd` (refer to Section 2.3.2).

4.2. Boot Device Sequence

Baltos systems will first check NAND device, if it contains a valid boot image, i.e. the first NAND partition is not empty. If the first partition is not empty, the system starts bootloader from NAND, if it is empty, MMC device will be searched for MLO file. Regardless of what device has started the bootloader, the final decision where to start the kernel from, will be taken by the bootloader's script.

4.2.1. "Emergency" Jumper

Baltos devices provide an "emergency" jumper JP1 close to the SODIMM module. If it is set, the system will ignore NAND and boot from MMC directly. Power off the system before setting or removing this jumper as it won't be evaluated by the SoC otherwise.

4.2.2. Booting from SD/microSD-card

The system images for OMAP3 based devices have two partitions:

1. FAT partition having files need to initialize and boot the system (`MLO`, `u-boot.img`, `uEnv.txt` and `uImage` or `kernel-fit.itb`). This partition must have bootable flag set
2. ext4 partition having Debian root file system

The OMAP CPU automatically loads SPL[12] (`x-loader/MLO`) from the FAT partition and then the code in SPL takes over and loads U-Boot[13] (`u-boot.img`), that takes care of Linux kernel (`uImage` or `kernel-fit.itb`).

[9] http://www.debian.org/ports/arm/
[10] ftp://ftp.visionsystems.de/pub/multiio/OnRISC/Baltos/
[11] https://en.wikipedia.org/wiki/Systemd
[12] Secondary Program Loader
[13] http://www.denx.de/wiki/U-Boot

4.2.3. Booting from NAND

Baltos has following NAND partitions:

```
#define MTDPARTS_DEFAULT    "mtdparts=omap2-nand.0:128k(SPL)," \
                            "128k(SPL.backup1)," \
                            "128k(SPL.backup2)," \
                            "128k(SPL.backup3)," \
                            "1920k(u-boot)," \
                            "-(UBI)"
```

Listing 2: NAND Partition Table

```
#define NANDARGS \
        "mtdids=" MTDIDS_DEFAULT "\0" \
        "mtdparts=" MTDPARTS_DEFAULT "\0" \
        "nandargs=setenv bootargs console=${console} " \
                "${optargs} " \
                "${mtdparts} " \
                "root=${nandroot} " \
                "rootfstype=${nandrootfstype}\0" \
        "nandroot=ubi0:rootfs rw ubi.mtd=5\0" \
        "nandrootfstype=ubifs rootwait=1\0" \
        "nandboot=echo Booting from nand ...; " \
                "run nandargs; " \
                "setenv loadaddr 0x81000000; " \
                "ubi part UBI; " \
                "ubifsmount ubi0:kernel; " \
                "ubifsload $loadaddr kernel-fit.itb; " \
                "ubifsumount; " \
                "bootm $loadaddr\0"
```

Listing 3: U-Boot: NAND Settings

Partitions SPL - SPL.backup3 have the same MLO as on the SD/microSD-card, u-boot holds U-Boot, UBI MTD partition holds further UBIFS[14] formatted partitions like kernel and rootfs (see Section 9.5.2 for details).

[14] http://en.wikipedia.org/wiki/UBIFS

4.2.4. Booting from USB

Baltos doesn't support direct booting from USB, but the bootloader started from either MMC or NAND is configured to search for a USB drive and check whether it is bootable. U-Boot checks for `uEnv.txt` file and barebox searches for `kernel-fit.itb`. As long as you don't use a kernel with embedded initramfs, you will have to make sure it has all needed USB drivers compiled in:

```
CONFIG_USB_STORAGE
CONFIG_USB_MUSB_HDRC
CONFIG_USB_MUSB_DSPS
CONFIG_AM335X_PHY_USB
```

You'll need to perform the following actions to start our Debian image from a USB drive:

1. prepare kernel as explained above
2. burn Debian image on the USB drive (see Section Section D.1 for the reference)
3. mount second (ext4) partition and copy prepared `kernel-fit.itb` and extract `modules.tar`
4. mount first (FAT) partition and change `uEnv.txt` to the content shown below (skip this step, if you're using barebox bootloader)

```
bootcmd=setenv loadaddr 0x82000000; run usbargs; ext4load usb 0:2 ${loadaddr} boot/kernel-fit.itb; bootm
${loadaddr}#conf${board_name}; if test $?  -ne 0; then echo "Using default FIT configuration"; bootm
${loadaddr}; fi;
uenvcmd=boot
```

4.3. Swapping and Logging

Due to the fact that the flash memory has a finite number of erase-write cycles it is very important to reduce them. Many applications use logging for information, recovery and debugging purposes, this can lead to frequent flash usage. There are several possibilities to avoid this:

1. use external HDD attached via USB and redirect swapping and logging to it
2. disable swapping[15] (remove swap entry in the `/etc/fstab`) and logging where it is possible
3. redirect the log stream via network[16]

To receive the log messages under Linux you can use your existing `syslog` utility, for Windows you can use Kiwi Syslog Daemon[17] or any other Syslog daemon.

4.4. Activating and Deactivating Services

Some services will be started as daemons at system startup and hence reduce the amount of free memory and increase the boot time. The ways to activate/deactivate services at boot time are extensively covered in Sections System Boot and The inetd Super-Server of "The Debian Administrator's Handbook".

[15] To list swapping devices execute `cat /proc/swaps`
[16] The Debian Administrator's Handbook
[17] www.kiwisyslog.com

4.5. System Image

The complete system image contains lots of programs and libraries. It contains a development environment consisting of the gcc toolchain and vim-tiny text editor. This image was created with vsdebootstrap (see Section 8.2 for details).

4.5.1. Program Overview

The complete image provides among others the following utilities:

- Software Development
  - gcc
  - CMake
  - git
- Network
  - ssh (server and client)
  - netcat
  - ser2net RFC2217 server
  - socketcand
- Desktop:
  - XFCE desktop

5. Network Services and Tools Provided by OnRISC

The OnRISC can be accessed via Ethernet for remote usage and file sharing. For this purpose there are several services such as `telnet` or `ssh` installed and preconfigured. For WLAN configuration `wpa_supplicant` and `iw` are included in the distribution.

5.1. LAN Configuration

5.1.1. Baltos

Baltos systems have following network interfaces:

1. WAN (`eth1`) - all devices
2. LAN (`eth0`) - iR devices and VS-860

WAN is connected to a single port PHY (Atheros 8035). Linux kernel automatically shows cable insertion/removal status on the console.

```
cpsw 4a100000.ethernet eth1:  Link is Down
```

or

```
cpsw 4a100000.ethernet eth1:  Link is Up - 1Gbps/Full - flow control off
```

LAN configuration differs across devices. In iR 5221/3220 devices LAN (`eth0`) is connected to a switch chip (ICPlus IP175D). As shown in Figure 1 on page 23 `eth0` is connected to one of the switch ports, hence has always cable inserted status. The LAN of iR 2110 is connected to a single port PHY (ICPlus IP101A/G) and behaves like WAN interface. The only difference is the maximum speed of 100Mbit/s. And finally Baltos 1080, OnRISC 113/213/413/813/CAN have WAN interface only i.e. `eth1`.

VLAN The network controller in Baltos systems is implemented as a switch device (CPSW). To use both slaves as separate network interfaces special feature called Dual Emac is used. Dual Emac uses VLAN IDs 1 and 2 to separate `eth0` and `eth1`, hence you should not add these VLAN IDs via `vconfig` or `iproute2`.

For example, when creating a bridge, use the following commands to avoid conflicts:

```
ip link add name br0 type bridge
ip link set br0 up
ip link set eth0 up
ip link set br0 type bridge vlan_default_pvid 3
ip link set eth0 master br0
```

If you are forced to use VLAN ID 1 and/or 2, you'll have to change default VLAN IDs in the DTS file corresponding to your device, recompile and install the kernel. In the Listing below you can see the field `dual_emac_res_vlan` specifying default VLAN IDs. Just change 1 and 2 to for example 4093 and 4094.

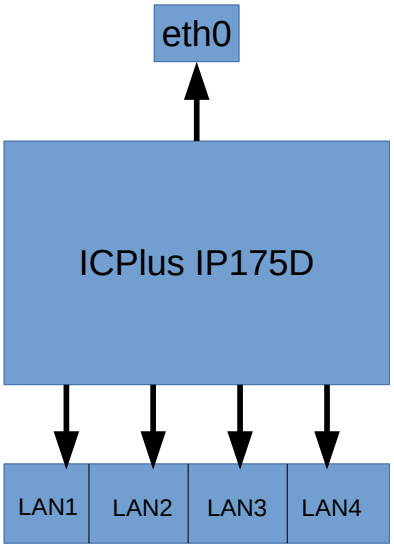

Figure 1: iR 5221 LAN Configuration

```
&cpsw_emac0 {
        phy-mode = "rmii";
        dual_emac_res_vlan = <1>;
        fixed-link {
                speed = <100>;
                full-duplex;
        };
};

&cpsw_emac1 {
        phy-mode = "rgmii-id";
        dual_emac_res_vlan = <2>;
        phy-handle = <&phy1>;
};
```

Listing 4: DTS: Dual Emac Configuration

5.2. WLAN Configuration

Wpa_supplicant[18] is a WPA Supplicant for Linux, BSD, Mac OS X, and Windows with support for WEP, WPA and WPA2 (IEEE 802.11i / RSN). Supplicant is the IEEE 802.1X/WPA component that is used in the client stations. It implements key negotiation with a WPA Authenticator and it controls the roaming and IEEE 802.11 authentication/association of the WLAN driver.

Further information about configuring the WLAN interface can be taken from wpa_supplicant's documentation[19].

5.2.1. Regulatory Domains

By default Linux kernel uses regulatory domain 00. This domain restricts some WLAN channels (12-14). In order to comply to your country's regulations you'll need `crda` package. With this packages installed you can set your regulatory domain using `iw` utility. Following command sets regulatory domain to Germany (DE):

```
iw reg set DE
```

For other countries see this article[20].

5.2.2. Managed Wireless Network (Wpa_supplicant)

Wpa_supplicant uses `/etc/wpa_supplicant.conf` file for its configuration (see the Listing below).

WLAN interface can be automatically configured on system's startup (see Section 2.3.1). To test the configuration just run:

```
/etc/init.d/networking restart
```

Wpa_supplicant can also be called manually to investigate configuration problems:

```
wpa_supplicant -iwlan0 -c/etc/wpa_supplicant.conf -dd -Dnl80211
```

On a successful connection/authentication the log should show `CTRL-EVENT-CONNECTED` event. Following command reveals some AP connection statistics:

5.2.3. Ad-hoc Wireless Network

Ad-hoc connections will be managed via iw tool. Following example shows, how to connect to an unencrypted Ad-hoc host "foo":

```
iw wlan0 connect foo
```

To connect to a WEP encrypted host invoke:

```
iw wlan0 connect foo keys 0:abcde d:1:0011223344
```

[18] http://hostap.epitest.fi/wpa_supplicant/
[19] http://w1.fi/cgit/hostap/plain/wpa_supplicant/README
[20] https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

```
ap_scan=1

# no encryption
network={
  ssid="TEST"
  key_mgmt=NONE
}
# WEP encryption
network={
  ssid="TESTWEP"
  key_mgmt=NONE
  wep_key0=xxxxxxxxxxxxxxxxxxxxx
  wep_tx_keyidx=0
  auth_alg=SHARED
}
# WPA/WPA2 encryption
network={
  ssid="TESTWPA2"
  proto=WPA RSN
  key_mgmt=WPA-PSK
  pairwise=CCMP TKIP
  group=CCMP TKIP
  psk="xxxxxxxxxxxxx"
}
```

Listing 5: /etc/wpa_supplicant.conf

```
root@onrisc:~# iw wlan0 link
Connected to 84:ea:96:ee:6d:5e (on wlan0)
        SSID: ExampleAP
        freq: 2462
        RX: 4449784 bytes (23438 packets)
        TX: 773784 bytes (2735 packets)
        signal: -76 dBm
        tx bitrate: 72.2 MBit/s MCS 7 short GI

        bss flags:      short-preamble short-slot-time
        dtim period:    3
        beacon int:     100
```

5.2.4. Supported WLAN Hardware

Model	Interface	Kernel Symbols	Firmware (Buildroot Symbols)
Jorjin WG7833	SDIO	WL18XX, WLCORE_SDIO	BR2_PACKAGE_LINUX_FIRMWARE_TI_WL18XX
TP-LINK TL-WN272N V3	USB	RT2800USB_RT53XX	BR2_PACKAGE_LINUX_FIRMWARE_RALINK_RT2XX
SparkLAN WPER-172GN	USB	RT2800USB_RT53XX	BR2_PACKAGE_LINUX_FIRMWARE_RALINK_RT2XX

Table 4: Supported WLAN Hardware

5.3. GSM Support

Using the USB connector or the PCI Express Mini Card adapter described in the Hardware Manual, you can attach a GSM card. The system was tested with the following GSM/HSPA/UMTS/LTE cards:

Vendor	Model	3G	4G	GPS type	Connector	VID/PID
Huawei	MU609	yes	no	passive antenna	mPCIe	12d1:1573
Huawei	MU709S-6	yes	no	none	mPCIe	12d1:1c25
Huawei	ME909u-521	yes	yes	passive antenna	mPCIe	12d1:1573
Sierra Wireless	AirPrime MC7304	yes	yes	active antenna	mPCIe	1199:68c0
Sierra Wireless	AirPrime WP7607	yes	yes	passive antenna	mPCIe	1199:68c0
Quectel	UC20	yes	no	passive antenna	mPCIe	05c6:9003
SIMcom	SIM5360E	yes	no	none	mPCIe	05c6:9000
SIMcom	SIM7600x	yes	yes	passive antenna	mPCIe	1e0e:9001
Huawei	E353	yes	no	none	USB	12d1:1506

Table 5: Supported Modems

Please read Section 7.3. It describes, how mPCIe slot can be turned on and off.

Baltos iR5221/3220 devices provide an LED labelled "3G". It is directly connected to the installed modem. Hence, the information about the behaviour of this LED can be found in the modem's documentation. For example, Huawei MU609[21] has the following blink codes:

Operating Status	LED_WWAN#
No service/Restricted service	Outputs: low(0.1s)-high (0.1s)-low (0.1s)-high (1.7s) 2s cycle
Register to the network	Outputs: low (0.1s)-high (1.9s) 2s cycle
Dial-up successfully	Outputs: low

Table 6: Huawei MU609 Blink Codes

5.3.1. PPP

Special udev rules in `/etc/udev/rules.d/98-vscom.rules` create `/dev/atcmd0` symlink to card's command port, that is used in the scripts below. This allows the user to use any supported card without changing its device name in the scripts. The following configuration files were prepared for modem usage:

- `/etc/chatscripts/gprs` (your provider's Access Point Name (APN) is stored here)
- `/etc/ppp/peers/gprs`

To activate a GSM connection configure proper APN and invoke

```
pppd file /etc/ppp/peers/gprs
```

You can find the supported AT command reference[22] on the CD .

[21] "HUAWEI MU609 Mini PCIe Module: Hardware Guide" from 2013-07-24
[22] documentation/2130617_Supported_AT_Command_Reference-v2.4.pdf

5.3.2. ModemManager

Starting from version 8 Debian provides ModemManager[23] with QMI[24]/MBIM[25] support. These protocols/drivers provide higher data throughput than PPP. Below you'll find some usage examples:

List detected modems:

```
# mmcli -L

Found 1 modems:  /org/freedesktop/ModemManager1/Modem/0 [Huawei Technologies Co., Ltd.]  MU609
```

Show modem's status:

```
# mmcli -m 0
/org/freedesktop/ModemManager1/Modem/0 (device id '4a1d302560babc0bc24d6a46e1d8c56740045b8f')
-------------------------
Hardware | manufacturer:  'Huawei Technologies Co., Ltd.'
| model:  'MU609'
| revision:  '12.105.29.00.00'
| supported:  'gsm-umts'
| current:  'gsm-umts'
| equipment id:  '323432046447031'
-------------------------
System | device:  '/sys/devices/ocp/47400000.usb/47401400.usb/musb-hdrc.1.auto/usb1/1-1/1-1.3'
| drivers:  'option1, cdc_ether'
| plugin:  'Huawei'
| primary port:  'ttyUSB2'
| ports:  'ttyUSB0 (at), eth2 (net), ttyUSB2 (at), ttyUSB3 (gps), ttyUSB4 (at), ttyUSB1 (qcdm)'
-------------------------
Numbers | own :  '+xxxxxx'
-------------------------
Status | lock:  'none'
| unlock retries:  'sim-pin (3), sim-pin2 (2), sim-puk (10), sim-puk2 (10)'
| state:  'disabled'
| power state:  'on'
| access tech:  'unknown'
| signal quality:  '0' (cached)
-------------------------
Modes | supported:  'allowed:  3g; preferred:  none
| allowed:  2g; preferred:  none
| allowed:  2g, 3g; preferred:  none'
| current:  'allowed:  2g, 3g; preferred:  none'
-------------------------
Bands | supported:  'unknown'
| current:  'unknown'

-------------------------
IP | supported:  'ipv4'
-------------------------
3GPP | imei:  '11122222111'
| enabled locks:  'none'
| operator id:  'unknown'
| operator name:  'unknown'
| subscription:  'unknown'
| registration:  'unknown'
-------------------------
SIM | path:  '/org/freedesktop/ModemManager1/SIM/0'
```

[23] https://www.freedesktop.org/wiki/Software/ModemManager/
[24] https://www.freedesktop.org/wiki/Software/libqmi/
[25] https://www.freedesktop.org/wiki/Software/libmbim/

```
------------------------
Bearers | paths:  'none'
```

Enable modem:

```
# mmcli -m 0 -e
successfully enabled the modem
```

Show available networks:

```
# mmcli -m 0 --3gpp-scan --timeout=300
---------------------
3GPP scan | networks:  26201 - TDG (gsm, available)
| 26202 - Vodafone (lte, available)
| 26201 - TDG (umts, available)
| 26201 - TDG (lte, available)
| 26202 - Vodafone (gsm, available)
| 26202 - Vodafone (umts, available)
| 26203 - E-Plus (gsm, available)
| 26203 - E-Plus (umts, available)
| 26203 - MEDIONmobile (lte, current)
```

Create a simple connection:

```
# mmcli -m 0 --simple-connect="apn=internet.eplus.de"
successfully connected the modem
```

After the successful connection, you can see the assigned network interface and its IP configuration using the Bearer object. Execute `mmcli -m 0` again and search for the Bearer line:

`/org/freedesktop/ModemManager1/Bearer/2`

Take the Bearer's index (in the example above, it is 2) and execute the following command:

```
root@onrisc:/tmp# mmcli -b 2
  ------------------------------------
  General             |       dbus path: /org/freedesktop/ModemManager1/Bearer/2
                      |            type: default
  ------------------------------------
  Status              |       connected: yes
                      |       suspended: no
                      |       interface: eth2
                      |     ip timeout: 20
  ------------------------------------
  Properties          |             apn: internet.eplus.de
                      |         roaming: allowed
  ------------------------------------
  IPv4 configuration  |          method: static
                      |         address: 10.208.60.65
                      |          prefix: 30
                      |         gateway: 10.208.60.66
                      |             dns: 62.109.121.17, 62.109.121.18
                      |             mtu: 1500
  ------------------------------------
  Statistics          |        duration: 120
                      |        attempts: 1
                      | total-duration: 120
```

Now, you can obtain an IP address using the `dhclient` and network interface mentioned in the Bearer configuration (in the case of Huawei MU609, it is `eth2`) or assign the IP configuration statically:

```
# dhclient eth2
```

Some modems like, for example, SIM7600 utilize the so-called "raw-ip" mode that `dhclient` doesn't support. In this case, you can use `udhcpc` instead:

```
# udhcpc -i eth2
```

Disconnect:

```
# mmcli -m 0 --simple-disconnect
successfully disconnected all bearers in the modem
```

5.3.3. Disabling PIN Request

To disable the PIN request, you have to execute the following command to discover the SIM card path and index:

```
mmcli -m 0 | grep SIM
```

The path to the SIM card looks as follows:

```
SIM | path:  '/org/freedesktop/ModemManager1/SIM/0'
```

The number after the last slash is the `SIM_INDEX` (it is 0 in this case). Now execute the following command to disable the PIN request:

```
mmcli --sim=SIM_INDEX --pin=PIN --disable-pin
```

5.3.4. Connect Modem On Startup

There are many ways to activate the modem on startup. Below, a method using `incron`[26] will be shown. First of all, you'll have to install `incron`:

```
apt install incron
```

Open `/etc/incron.allow` and enter root, save and exit. Now root is allowed to perform incron jobs. To create modem job invoke:

```
incrontab -e
```

This command will open an editor, where you will enter following line, save and exit:

```
/dev/ IN_CREATE /usr/local/bin/startmodem.sh
```

After configuring `incron` create `/usr/local/bin/startmodem.sh` with following code:

```
#!/bin/sh

if [ -e /dev/atcmd0 ]; then
        sleep 60
        # enter PIN, if required
        #echo -e "at+cpin="1234"\r" > /dev/atcmd0
        /usr/sbin/pppd file /etc/ppp/peers/gprs
fi
```

[26] http://inotify.aiken.cz/?section=incron&page=about&lang=en

Make file executable `chmod +x /usr/local/bin/startmodem.sh`

Now every time the system boots and mPCIe slot is activated `/usr/local/bin/startmodem.sh` will be started. `/usr/local/bin/startmodem.sh` waits for a minute to give ModemManager time to find the modem and then starts `pppd`. You can replace `pppd` invocation with `mmcli` as described in Section 5.3.2.

5.3.5. Controlling a Modem with Python

You can write a Python script to control your modem via the ModemManager. ModemManager can be accessed either via DBus directly or using GObject introspection for libmm. DBus reference can be found here[27]. GObject introspection examples can be found in the `examples` folder of the ModemManager git repository[28].

5.4. Remote VPN Access with viaVPN

Sometimes it is required to access OnRISC device from a remote location via a VPN tunnel. Depending on the network infrastructure this can be a difficult task, because at least on side of this tunnel (VPN server) must be accessible via Internet. This requires either making a hole in your company's/customer's firewall or setting up a dedicated VPN rendezvous server with a public IP address.

Our viaVPN service provides a convenient solution for this task. It offers a complete set of software/services to establish a VPN tunnel with no or minimal changes to your network infrastructure. viaVPN has following components:

- rendezvous server
- viaVPN daemon for Baltos (Debian and Buildroot versions)
- a viaVPN client utility for PC (Windows and Linux versions)

Both viaVPN daemon and client utility make a VPN connection to the rendezvous server via standard HTTPS port, so in most cases such connections are allowed by company's firewall and hence no changes are needed to be made by a network administrator.

In order to use the service you'll need to make a viaVPN account and each device must be registered and associated with this account. Please refer to the "viaVPN Daemon for Baltos" Manual for further instructions.

5.5. GPS

You can use various devices to get GPS data. So far following devices were tested:

1. VersaLogic VL-MPEu-G2E - dedicated mPCIe card with active GPS antenna support
2. GlobalSat BU-353S4 - USB GPS mouse
3. Sierra Wireless AirPrime MC7304 - mPCIe LTE modem with active GPS antenna support

[27] https://www.freedesktop.org/software/ModemManager/api/latest/ref-dbus.html
[28] https://cgit.freedesktop.org/ModemManager/ModemManager/tree/examples

5.5.1. GPS Device Setup

VersaLogic VL-MPEu-G2E VL-MPEu-G2E exports /dev/ttyACM0 device. Baudrate 9600b/s:

```
stty -F /dev/ttyUSB0 ispeed 9600 ospeed 9600
```

GlobalSat BU-353S4 BU-353S4 exports /dev/ttyUSB0 device provided by pl2303 driver. Baudrate 4800b/s:

```
stty -F /dev/ttyUSB0 ispeed 4800 ospeed 4800
```

Sierra Wireless AirPrime MC7304 MC7304 provides GPS data on the `/dev/ttyUSB1` device. You'll have to active the GPS port prior to reading data:

```
echo 'AT!GPSTRACK=1,255,100,1000,5' > /dev/ttyUSB2
stty raw -F /dev/ttyUSB1; echo \$GPS_START >/dev/ttyUSB1
```

5.5.2. Reading Raw GPS Data

After configuring the required device you can just invoke `cat device` in order to get raw data:

```
cat /dev/ttyUSB0
```

You'll get similar output:

```
$GPGSA,A,3,12,15,25,32,10,,,,,3.4,2.3,2.5*33
$GPRMC,160059.000,A,5340.0274,N,00959.2111,E,0.48,38.52,090316,,A*5D
$GPGGA,160100.000,5340.0274,N,00959.2116,E,1,05,2.3,10.4,M,45.9,M,,0000*61
$GPGSA,A,3,12,15,25,32,10,,,,,3.4,2.3,2.5*33
$GPRMC,160100.000,A,5340.0274,N,00959.2116,E,0.47,38.52,090316,,A*58
$GPGGA,160101.000,5340.0273,N,00959.2114,E,1,04,4.1,10.5,M,45.9,M,,0000*61
```

The meaning of these messages can be read here.

5.5.3. gpsd

gpsd project[29] provides various tools to manage GPS receivers and to provide their data over network.

gpsmon is a real-time GPS packet monitor and control utility. Execute following command to see GPS coordinates, satellite information etc.

```
gpsmon /dev/ttyUSB0
```

gpsd is interface daemon for GPS receivers. To start a daemon in foreground and listening on TCP port 5050 execute:

```
gpsd -b -G -N -S 5050 /dev/ttyUSB0
```

[29] http://www.catb.org/gpsd/

5.6. SSH

To access the OnRISC via SSH from Linux execute:

```
ssh root@192.168.254.254
```

To access OnRISC via SSH from Windows you need a ssh-client such as PuTTY[30]. To exchange files several tools could be used:

- scp (Linux) - secure copy tool
- pscp (Windows) - secure copy tool included in PuTTY distribution
- WinSCP[31] (Windows) - secure copy tool with graphical interface

For further information see:

```
man sshd
```

5.7. RFC2217

Internal serial interfaces of the OnRISC can be made accessible over network via RFC2217 protocol. `ser2net`[32] is a daemon, that provides such functionality. On the client side (PC) you need either a virtual COM port driver or an application/library communicating RFC2217 directly. For MS Windows OS you can use the VScom driver of NetCom Mini[33].

The installation requires following components to be installed:

- RFC2217 driver and NetCom UPnP Manager (is bundled with the driver)
- `vsupnpd` (provided with our system images, enabled by default)
- `ser2net` (provided with our system images, disabled by default)

`ser2net` will be configured via special configuration file (default `/etc/ser2net.conf`). This file contains following information:

```
SIGNATURE:sign_vs:VScom NetCom:111S:(C) VS Vision Systems GmbH 2010:1
5000:telnet:0:/dev/ttyS1:115200 sign_vs
5001:telnet:0:/dev/ttyS2:115200 sign_vs
```

SIGNATURE is a string that will be sent on RFC2217 driver's request to identify the device. Don't change this setting. After that you'll find the per serial interface configuration strings in following format:

```
<TCP port>:<state>:<timeout>:<device>:<options>
```

In the example the `/dev/ttyS1` device will be used in RFC2217 mode (telnet), listens on TCP port 5000 and the timeout function is disabled.

`vsupnpd` daemon parses `/etc/ser2net.conf` file and extracts TCP port numbers. This information will be announced via SSDP broadcasts, so that NetCom UPnP Manager can find and install the

[30] http://www.chiark.greenend.org.uk/~sgtatham/putty/
[31] http://winscp.net/eng/index.php
[32] https://sourceforge.net/projects/ser2net/
[33] http://www.vscom.de/download/multiio/Windows7/driver/NCMini-Install-7z.exe

serial interfaces. If OnRISC is behind a router and broadcasts are not allowed, you can manually add the device in the NetCom UPnP Manager. See NetCom Mini Manual for more details.

Please note that `ser2net` service is disabled by default. You need to activate using `rcconf` (see Section 4.4).

5.8. Socketcand

`socketcand` is a daemon that provides access to CAN interfaces on a machine via a network interface. The communication protocol uses a TCP/IP connection and a specific protocol to transfer CAN frames and control commands. The protocol specification can be found in `./doc/protocol.md`.[34]

The project provides Java API to develop your own client applications for `socketcand`. There is also a CAN sniffer software Kayak[35], that uses socketcand protocol, so that you can use OnRISC as a network based CAN sniffer at once.

Please note that `socketcand` service is disabled by default. You need to activate using `rcconf` (see Section 4.4).

5.9. GPIO over Modbus/TCP

Baltos GPIO can be made accessible over network via `modbusgpio` daemon. `modbusgpio` can be invoked with following parameter:

1. TCP port number - port number `modbusgpio` is listening on (required parameter)
2. IP address to bind daemon to (optional parameter)
3. debug enable option (optional parameter)

Example:

```
modbusgpio 502 debug
```

This command would start `modbusgpio`, that listens on port 502 and also prints debug messages on `stderr`.

The daemon implementation supports Discrete Inputs, Coils, Input Registers and Holding Registers. These are the most useful function codes (see Table 7 on page 35).

Mapping of addresses to Digital-I/O signals are shown in following tables:

For initial testing you can use QModBus[36] software to set/get GPIO values. Figure 2 shows how to set OUT2 to high and Figure 3 shows the result provided OUT2 is connected to IN2.

[34] https://github.com/linux-can/socketcand
[35] https://dschanoeh.github.io/Kayak/
[36] http://qmodbus.sourceforge.net/

	Function	Decimal Code
Single bit access	Read Discrete Inputs	02
	Read Coils	01
	Write Single Coil	05
	Write Multiple Coils	15
16 bit access	Read Holding Register	03
	Read Input Register	04
	Write Single Register	06
	Write Multiple Registers	16

Table 7: Function Codes Modbus/TCP for Digital-I/O

Coil Address	Register Address	Signals	Interpretation
0 .. 7	0	Input 1 .. 8	External status 0: Input open 1: Input closed
8 .. 15		Output 1 .. 8	External status 0: Connect <n> to Ö 1: Connect <n> to S
32 .. 47	2	Data Direction	0: Input signal 1: Output signal
64 .. 79	4	Direction Option	0: Changeable 1: Fixed
(80 .. 96)	6	Number of I/Os	16

Table 8: Mapping Modbus addresses to Input-/Output-signals for Baltos 1080

Coil Address	Register Address	Signals	Interpretation
0 .. 3	0	IN0 .. IN3	External status 0: low 1: high
4 .. 7		OUT0 .. OUT3	External status 0: low 1: high
32 .. 47	2	Data Direction	0: Input signal 1: Output signal
64 .. 79	4	Direction Option	0: Changeable 1: Fixed
(80 .. 96)	6	Number of I/Os	8

Table 9: Mapping Modbus addresses to Input-/Output-signals for Baltos 5221/3220

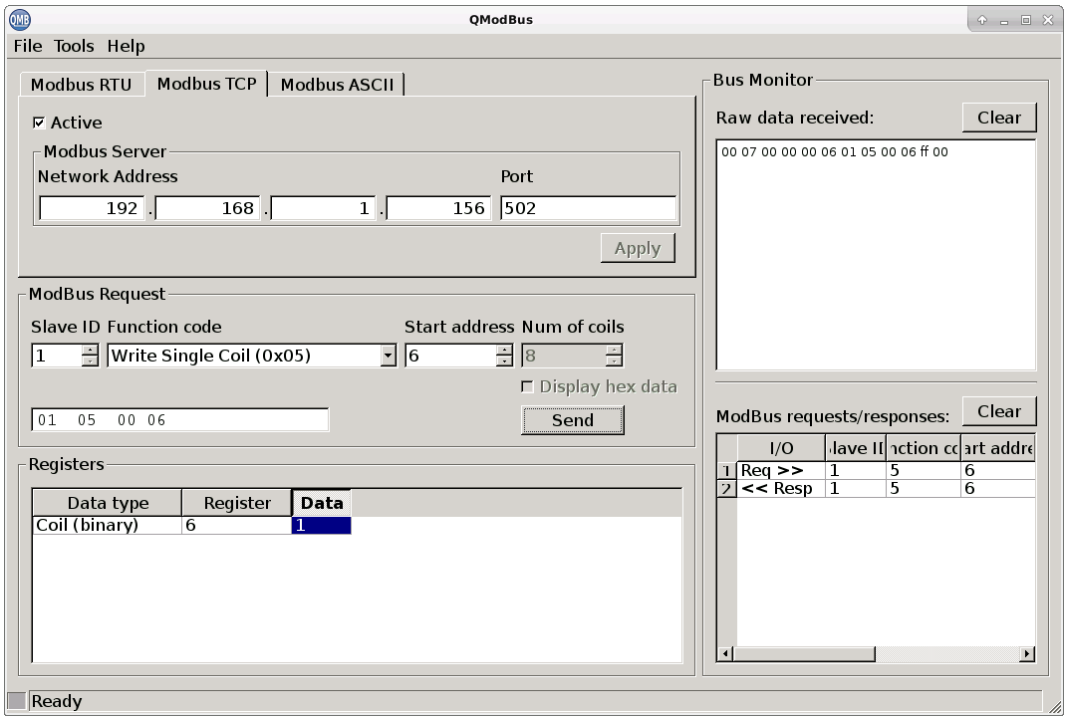

Figure 2: QModBus: Baltos iR 5221: Set OUT2 to High

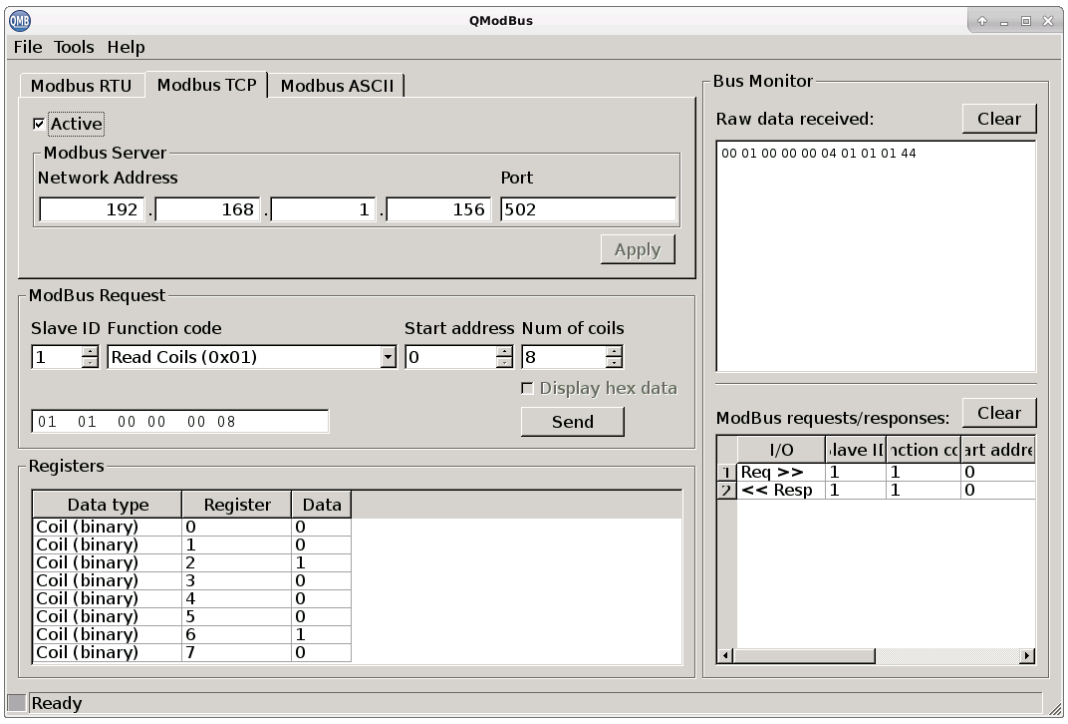

Figure 3: QModBus: Baltos iR 5221: Get all GPIOs

6. Software Development

6.1. Environment

6.1.1. Compile your software directly on the OnRISC

You can start programming directly on the OnRISC. The toolchain is already installed and GCC compiler will be invoked in the same way it is done on a desktop Linux. To modify files you can use `vi` or some other editor. This method is preferred, if your software has many dependencies.

6.1.2. Cross-compile your software on the PC

Debian 9 and 10 will be supplied with OMAP3 based devices. It is based on an arm-linux-gnueabihf 6.3.x (Debian 9) or arm-linux-gnueabihf 8.3.x (Debian 10) toolchains. Provided you use the same Debian version on both your host and Baltos you can take advantage of Debian's multiarch feature. Below you'll find cross-toolchain installation instructions based on your Debian version:

Debian 9 and 10 Install following packages from Debian's own repository:

```
apt install binutils-arm-linux-gnueabihf g++-arm-linux-gnueabihf build-essential git
```

Multiarch Usage Example If your software has dependencies on other libraries provided by Debian you can get their cross-compiled version via `apt-get install package:armhf`. For example to install `libsocketcan` you need to invoke following command:

```
apt-get install libsocketcan-dev:armhf
```

After installation you'll see following packages appearing in your package data base:

```
# dpkg -l | grep libsocketcan
```

```
ii libsocketcan-dev 0.0.9+git20140207-1 armhf library to control some basic functions in SocketCAN from
userspace
```

```
ii libsocketcan2 0.0.9+git20140207-1 armhf library to control some basic functions in SocketCAN from
userspace
```

Following code sample uses `libsocketcan` to shutdown `can0` interface:

```
#include <stdio.h>
#include <libsocketcan.h>

int main(int argc, char **argv)
{
        printf("Hello␣world!\n");

        if (can_do_stop("can0")) {
                perror("Stop␣CAN␣device:␣");
                return -1;
        }

        return 0;
}
```

Listing 6: Debian Multiarch Example

In order to build this program invoke:

```
arm-linux-gnueabihf-gcc -o test test.c -lsocketcan
```

Then copy resulting binary to the OnRISC device. On the device execute `ldd` and you'll see linked libraries including `libsocketcan`:

```
# ldd test
libsocketcan.so.2 => /usr/lib/arm-linux-gnueabihf/libsocketcan.so.2 (0xb6f1b000)
libc.so.6 => /lib/arm-linux-gnueabihf/libc.so.6 (0xb6e2c000)
/lib/ld-linux-armhf.so.3 (0xb6f31000)
```

See https://wiki.debian.org/CrossToolchains for more information about multiarch support in Debian.

Alternatively you can use Buildroot or Yocto as your development environment.

6.2. Linux Kernel

6.2.1. Getting Source Code

Baltos Kernel config, patches and Device Tree[37] source files are part of Buildroot's BSP described in Section 9. To get the source code first build Buildroot's default image, then you can change kernel configuration invoking `make linux-menuconfig` from Buildroot's folder.

Buildroot creates a so called FIT image (`*.itb`). This file combines both zImage and DTB blobs in one file. Please refer to these slides[38] and `board/vscom/baltos/custom-FIT.sh` for more details. To install the kernel copy `kernel-fit.itb` to the FAT partition of your SD/microSD-card. See Section 6.2.2 for kernel modules handling. You are advised to always install `kernel-fit.itb` together with kernel modules from the same build to avoid versioning issues.

6.2.2. Install Kernel Modules

Compilation Outside Buildroot Linux kernel provides means to create installation package. The most portable packaging type is a tar package. After kernel compilation invoke:

```
make tar-pkg
```

This will create `linux-x.y.z.tar` file in the root of your kernel tree.

Copy this package to the OnRISC device via `scp` and extract:

```
scp linux-x.y.z.tar root@192.168.254.254:/tmp
```

On the device perform:

```
tar xf /tmp/linux-x.y.z.tar -C /
```

You'll now get `/lib/modules/x.y.z` folder created. Remove unneeded files from `/boot` like `System.map-x.y.z` and `vmlinux-x.y.z` to save space.

Compilation Using Buildroot Buildroot's `baltos_defconfig` automatically invokes `modules.sh`. This script creates tar file and copies it to `output/images/modules.tar`.

6.2.3. Creating Kernel *.deb Package from Binary Files

After successfully building your kernel for example using Buildroot, create following directory structure somewhere (`/home/user/debs`) on your Debian host:

```
mkdir -p /home/user/debs/kernel_3.18.32-2/boot
mkdir -p /home/user/debs/kernel_3.18.32-2/DEBIAN
mkdir -p /home/user/debs/kernel_3.18.32-2/lib
```

Copy `output/images/kernel-fit.itb` to `/home/user/debs/kernel_3.18.32-2/boot`.

Copy `output/target/lib/modules` to `/home/user/debs/kernel_3.18.32-2/lib`.

Create `/home/user/debs/kernel_3.18.32-2/DEBIAN/control` with following content:

[37] http://elinux.org/Device_Tree
[38] http://elinux.org/images/f/f4/Elc2013_Fernandes.pdf

```
Package: kernel
Version: 3.18.32-2
Section: kernel
Priority: optional
Architecture: armhf
Depends: kmod
Maintainer: Your Name <you@yoursite.com>
Description: Linux Kernel
  Linux Kernel with OnRISC related patches.
```

Replace *Maintainer* field with your name and e-mail address. As you can see *Version* field and package folder are the same. The "2" the end of the version shows *.deb package version number, i.e. kernel 3.18.32 but the second version of it (for example because of some changes in the kernel configuration or the source code).

Now you're ready to compile a *.deb package. Invoke:

1. `cd /home/user/debs/`
2. `dpkg-deb --build kernel_3.18.32-2`

As a result you'll get `kernel_3.18.32-2.deb` file.

6.2.4. Creating a Standard Kernel *.deb Package

Starting from Debian 10, we provide the standard Kernel Debian package consisting of the following components[39]:

- `linux-headers-5.4.61-elbe_5.4.61-1_armhf.deb` - Linux header files
- `linux-image-5.4.61-elbe_5.4.61-1_armhf.deb` - Linux binary image, i.e. `kernel-fit.itb`
- `linux-libc-dev-5.4.61-elbe_5.4.61-1_armhf.deb` - Linux support headers for userspace development
- `linux-5.4.61-elbe_5.4.61-1.dsc` - Debian package description
- `linux-5.4.61-elbe_5.4.61.orig.tar.xz` - original source code
- `linux-5.4.61-elbe_5.4.61-1.debian.tar.xz` - custom patches and debianization files

To build your own version of this kernel (for example, with different configuration), perform the following steps:

1. `apt install quilt libelf-dev libssl-dev flex bison bc lzop u-boot-tools debhelper`
2. `tar xJf linux-5.4.61-elbe_5.4.61.orig.tar.xz`
3. `cd linux-5.4.61`
4. `tar xJf linux-5.4.61-elbe_5.4.61-1.debian.tar.xz`

Now you have the kernel source with the `debian` folder. Next , we will apply the patches from `debian/patches` directory:

1. `export QUILT_PATCHES=debian/patches`

[39] ftp://ftp.visionsystems.de/pub/multiio/OnRISC/Baltos/deb/buster/

2. `quilt push -a`

To cross-compile the kernel and generate the packages invoke:

- `DEB_BUILD_OPTIONS=nostrip dpkg-buildpackage --no-sign -a armhf`

6.3. Bootloader

6.3.1. U-Boot

Baltos Baltos systems use U-Boot as a default bootloader. Buildroot BSP provides a patch[40], that adds support for Baltos devices. In order to modify U-Boot (change NAND partition table etc.) you need to make following steps:

1. download U-Boot version, that is specified in `baltos_defconfig`
2. apply related patch from `board/vscom/u-boot-patches/`
3. add an entry to `local.mk` pointing to U-Boot's folder
4. execute `make` in Buildroot's folder

You'll need following files after compilation process:

- `output/images/MLO`
- `output/images/u-boot.img`

If you're booting from SD card, then just copy them to the first partition (FAT) or just burn `sdcard.img` file directly to your SD card. In the case of NAND burn them to following partitions:

- `MLO` to `/dev/mtdblock0`
- `u-boot.img` to `/dev/mtdblock4`

6.3.2. Barebox

Baltos Buildroot BSP also provides support for barebox[41] bootloader. barebox provides bash like shell environment (hush[42]), SquashFS[43] support as also support for TCA6416A I2C I/O expander. To build barebox instead of U-Boot invoke:

```
make baltos_barebox_defconfig
make clean
make
```

You'll need the following files after compilation process:

- `output/images/barebox-am33xx-baltos-mlo.img` - needs to be renamed to MLO when writing to a FAT partition
- `output/images/barebox-am33xx-baltos.img` - needs to be renamed to `barebox.bin` when writing to a FAT partition

[40] `board/vscom/u-boot-patches/2014.07/uboot-0001-Add-support-for-Baltos-system.patch`
[41] http://www.barebox.org/
[42] http://www.barebox.org/doc/latest/user/hush.html
[43] https://en.wikipedia.org/wiki/SquashFS

If you're booting from SD card, then just copy them to the first partition (FAT) or just burn `sdcard.img` file directly to your SD card. In the case of NAND burn them to following partitions:

- `barebox-am33xx-baltos-mlo.img` to `/dev/mtdblock0`
- `barebox-am33xx-baltos.img` to `/dev/mtdblock4`

barebox boot script[44] is stored in the Buildroot BSP repository. In this file you can change boot order or check DIP switches, I/O etc. After changing the `init` script you'll need to rebuild barebox. To do so invoke:

```
make barebox-rebuild
```

and you'll get new `output/images/barebox-am33xx-baltos.img` file.

6.4. Other Programming Languages Than C/C++

You can also use other programming languages like Python, Perl etc. The system image already provides Python and Perl. Basically you can install whatever programming language Debian itself provides. To install for example Java invoke:

```
apt-get install openjdk-7-jdk
```

or

```
apt-get install openjdk-8-jdk
```

6.5. Recommended Books

A list of books about Linux programming/usage and related topics:

- "The Debian Administrator's Handbook" by Raphaël Hertzog, Roland Mas (http://debian-handbook.info/)
- "Building Embedded Linux Systems, 2nd Edition " by Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, Philippe Gerum
- "Linux Device Drivers, Third Edition" by Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman (http://lwn.net/Kernel/LDD3/)
- "C: The Complete Reference" by Herbert Schildt
- "Dive Into Python 3" (https://diveintopython3.problemsolving.io)
- "Pro Git" by Scott Chacon (http://git-scm.com/book)

[44] https://github.com/visionsystemsgmbh/onrisc_br_bsp/blob/master/board/vscom/baltos/barebox/env/bin/init

7. OnRISC Hardware API

7.1. libonrisc

Such hardware as digital I/O, serial interfaces, USB OTG will be controlled through sysfs GPIO entries. To simplify this task `libonrisc` provides a convenient access to these hardware from user space. Following features are implemented:

- get system's hardware info like serial number. MACs etc.
- control LEDs
- control UART's RS232/422/485 driver
- control digital I/O
- read DIP switch
- control mPCIe slot

The library is hosted on GitHub (https://github.com/visionsystemsgmbh/libonrisc) and is included into Debian image and is also part of other BSPs. API documentation can be found in both `README.md` as also `include/onrisc.h`. `libonrisc` also provides Python bindings generated via SWIG[45].

Since Debian 9 `libonrisc` provides following packages:

- `libonrisc1` - provides `*.so` files and other binaries like `onrisctool` etc.
- `libonrisc-dev` - development files, i.e. header files
- `python-onrisc` - Python 2.x bindings
- `python3-onrisc` - Python 3.x bindings

`libonrisc` provides a command line tool called `onrisctool` (see Section B). It uses most of the API routines provided by the library and thus its source code is a good starting point to understand API programming. In addition `examples` folder[46] provides dedicated source code samples (both C and Python) for different API calls.

[45] http://www.swig.org/
[46] https://github.com/visionsystemsgmbh/libonrisc/tree/master/examples

7.2. Digital I/O

Digital I/O is made via TCA6416A I²C I/O expander. GPIOs will be exported via sysfs. See Kernel documentation[47].

For testing purposes one can either export and manipulate I/Os via sysfs directly or use `onrisctool` to control digital I/O (refer to Section B.2). For software development `libonrisc` provides a convenient API (refer to Section 7.1).

7.2.1. Baltos iR 5221/3220

Baltos iR 5221/3220 provides 4 inputs (496-499) and 4 outputs (500 - 503). Inputs and outputs have fixed direction, that cannot be changed. `/sys/class/gpio` will have following entries for 8 GPIOs:

```
gpio496 -> ../../devices/ocp.3/4802a000.i2c/i2c-1/1-0020/gpio/gpio496
gpio497 -> ../../devices/ocp.3/4802a000.i2c/i2c-1/1-0020/gpio/gpio497
gpio498 -> ../../devices/ocp.3/4802a000.i2c/i2c-1/1-0020/gpio/gpio498
gpio499 -> ../../devices/ocp.3/4802a000.i2c/i2c-1/1-0020/gpio/gpio499
gpio500 -> ../../devices/ocp.3/4802a000.i2c/i2c-1/1-0020/gpio/gpio500
gpio501 -> ../../devices/ocp.3/4802a000.i2c/i2c-1/1-0020/gpio/gpio501
gpio502 -> ../../devices/ocp.3/4802a000.i2c/i2c-1/1-0020/gpio/gpio502
gpio503 -> ../../devices/ocp.3/4802a000.i2c/i2c-1/1-0020/gpio/gpio503
```

7.2.2. Baltos 1080

Baltos 1080 provides 8 inputs (496-503) and 8 outputs (504-511). Inputs and outputs have fixed direction, that cannot be changed.

[47] https://www.kernel.org/doc/Documentation/gpio/sysfs.txt

7.3. mPCIe On/Off Switching

In Baltos devices, the mPCIe slot can be switched on/off to enable/disable or reset the installed mPCIe card. Switching can be made either via `onrisctool` or via `libonrisc`. For `onrisctool` please refer to Section B.6. For `libonrisc` usage see a programming example[48] on GitHub.

Please note, that cutting USB device's power is not a standard operation, so please disable the device and unload its driver before switching mPCIe off to avoid unexpected Linux kernel behaviour.

Baltos systems with U-Boot Version "U-Boot 2014.07 (May 11 2016 - 15:38:41)" and Debian image with Kernel 3.18.32 and newer will enable mPCIe already in the bootloader, so that depending on modem's firmware a modem is already available when Kernel performs USB device enumeration. Bootloader on older systems can be easily updated using our latest Debian image:

1. boot into Debian
2. invoke `update_bootloader.sh`

You'll get the following output:

```
Updating bootloader
Bootloader updated successfully
```

7.4. Serial Interfaces

Depending on the model used OnRISC devices provide serial interfaces based on different UART types. Each UART type has it's own handling. Below you'll find a table describing devices and used UARTs.

Device	UART
Baltos iR 5221/3220/2110, OnRISC 113/213	16C750 (TI OMAP)
Baltos 1080, OnRISC 413/813	FT4232H (FTDI)
VS-860	FT2232D (FTDI)

Table 10: Used UARTs

7.4.1. RS422/RS485 and GND Connection

With RS422/485 serial interfaces you have to connect the logic Ground between all connected devices. This is necessary because the specifications of RS485 require this. Unfortunately many demonstration graphics focus on the data lines, and omit the ground line for simplicity. In order to clarify this issue we have created a comprehensive FAQ entry[49].

[48] https://github.com/visionsystemsgmbh/libonrisc/blob/master/examples/mpcie.c
[49] http://faq.visionsystems.de/index.php?action=artikel&cat=12&id=91&artlang=en

7.4.2. RS Mode Switching

Two methods are provided to switch between RS232/RS422/RS485 modes (termination inclusive): via software (**recommended**) and via DIP-switch (DIP-switch for direct RS mode switching is implemented only on Baltos iR 5221/3220 and VS-860). Each port has its own DIP-switch named SW1 and SW2. Table 11 shows possible DIP-switch settings.

S1	S2	S3	S4	Mode
off	off	x	x	RS-232 Loopback Test
on	off	x	x	RS-232 MODE
on	on	on	off	RS-422 MODE
on	on	on	on	RS-422 MODE RxD +/- with 120 ohm Term
on	on	off	off	RS-485 Full Duplex Mode
on	on	off	on	RS-485 Full Duplex Mode RxD +/- with 120 ohm Term
off	on	off	off	RS-485 Half Duplex Mode without Echo
off	on	off	on	RS-485 Half Duplex Mode without Echo RxD +/- with 120 ohm Term

Table 11: Serial Modes (hardware switching for Baltos iR 5221/3220 and VS-860 only)

Software configuration can be done either via invoking `onrisctool` or using `libonrisc` routines. Below you'll find some `onrisctool` usage examples:

- `onrisctool -p 1 -t rs232` (set first serial port to RS232 mode)
- `onrisctool -p 2 -t rs422` (set second serial port to RS422 mode without termination)
- `onrisctool -p 2 -r -t rs422` (set second serial port to RS422 mode with termination)
- `onrisctool -p 1 -r -t rs485-hd` (set first serial port to RS485 half-duplex mode with termination)

For `libonrisc` usage see a programming examples in C[50] and Python[51] on GitHub.

Baltos 1080 has RS232 only fixed serial ports, so `onrisctool` will report error, if you try to switch RS modes.

7.4.3. FTDI Based UARTs

FTDI based UARTs have following device naming scheme: `/dev/ttyUSBx`. If the system has two UARTs, they appear in Linux as `/dev/ttyUSB0` and `/dev/ttyUSB1`.

7.4.4. 16C750 Based UARTs

16C750 based UARTs have the following device naming scheme: `/dev/ttySx`. If the system has two UARTs, they appear in Linux as `/dev/ttyS1` and `/dev/ttyS2`. In this case the numbering begins with '1' because `/dev/ttyS0` is a console device. These UARTs have also special symlinks:

```
/dev/ttyVS0 -> ttyS1
/dev/ttyVS1 -> ttyS2
```

[50] https://github.com/visionsystemsgmbh/libonrisc/blob/master/examples/rs485.c
[51] https://github.com/visionsystemsgmbh/libonrisc/blob/master/examples/rs485.py

RS modes will be switched the same way as described in Section 7.4.2. Example below shows, how to configure RS485 mode with termination from software (Please note, that you'll need `libonrisc` installed in order to compile and link this example). To build this example in Debian on OnRISC execute following command:

```
gcc -o rs485 rs485.c -l onrisc
```

`-l onrisc` tells linker to link against `libonrisc`.

```
#include <stdio.h>
#include <stdlib.h>
#include <linux/serial.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <termios.h>
#include <onrisc.h>

#define TEST_BAUDRATE   B9600

int main(int argc, char **argv)
{
        int rc = EXIT_FAILURE;
        int fd = -1, ret;
        struct termios ser_termios;
        onrisc_uart_mode_t mode;
        char buf[32];

        if (argc != 3) {
                printf("Wrong␣parameter␣count.␣\nUsage:\nrs485␣
                   device␣port-number\nExmaple:\nrs485␣/dev/ttyS1␣1\
                   n");
                goto error;
        }

        if (onrisc_init(NULL) == EXIT_FAILURE) {
                perror("init␣libonrisc:␣");
                goto error;
        }

        mode.rs_mode = TYPE_RS485_HD;
        mode.termination = 1;

        if (onrisc_set_uart_mode(atoi(argv[2]), &mode) ==
           EXIT_FAILURE) {
                perror("failed␣to␣set␣serial␣port␣mode:␣");
                goto error;
        }

        /* open serial port */
```

```
        fd = open(argv[1], O_RDWR);
        if (fd < 0) {
                perror("open: ");
                goto error;
        }

        ret = tcgetattr(fd, &ser_termios);
        if (ret < 0) {
                perror("Getting attributes");
                goto error;
        }
        cfmakeraw(&ser_termios);

        /* configure local flags */
        ser_termios.c_lflag = 0;

        /* configure input flags */
        ser_termios.c_iflag = IGNPAR;

        /* configure output flags */
        ser_termios.c_oflag = 0;

        /* configure control flags */
        ser_termios.c_cflag = CS8 | CLOCAL | CREAD;

        ser_termios.c_cc[VMIN] = 0;
        ser_termios.c_cc[VTIME] = 0;

        ret = cfsetispeed(&ser_termios, TEST_BAUDRATE);
        ret = cfsetospeed(&ser_termios, TEST_BAUDRATE);
        ret = tcsetattr(fd, TCSANOW, &ser_termios);
        if (ret < 0) {
                perror("Setting attributes");
                goto error;
        }

        /* send test string */
        sprintf(buf, "hello");
        ret = write(fd, buf, strlen(buf));

        rc = EXIT_SUCCESS;
 error:
        if (fd > 0) {
                close(fd);
        }

        return rc;
}
```

Listing 7: RS485 Half-Duplex with Termination

7.5. CAN

SocketCAN provides access to CAN controller on all CAN capable OnRISC devices. SocketCAN[52] is a set of open source CAN drivers and a networking stack contributed by Volkswagen Research to the Linux kernel. Formerly known as Low Level CAN Framework (LLCF).

7.5.1. CAN Interface Configuration

Special script `/etc/init.d/can_if`[53] is provided to setup `can0` interface. It is preconfigured to use `can0` at 1000000b/s. These settings are stored in the `CAN_IF` variable:

```
CAN_IF="can0@1000000,200"
```

To start the CAN interface issue:

```
/etc/init.d/can_if start
```

to stop CAN interface issue:

```
/etc/init.d/can_if stop
```

Further information regarding programming and configuration can be found on the SocketCAN's project site.

7.5.2. CAN Usage Examples

To send a CAN frame execute:

```
cansend can0 123#1234
```

To receive CAN frames execute:

```
candump can0
```

7.5.3. CANopen

It is possible to use CANopen[54] library from CanFestival[55] or any other open source or proprietary CANopen stack, that has SocketCAN support.

7.5.4. J1939

Since Linux kernel 5.4.x SocketCAN also supports J1939 protocol. Take a look at the related kernel documentation[56]. This documentation describes motivation for this approach, API, and some usage scenarios. This tutorial[57] describes the first steps in learning the J1939 framework under Linux. In our GitHub repository[58] you can find source code samples for J1939 protocol.

[52] http://elinux.org/CAN_Bus
[53] this script is maintained in this git repository: `https://github.com/linux-can/can-misc`
[54] https://www.can-cia.org/
[55] https://canfestival.org/
[56] https://www.kernel.org/doc/html/latest/networking/j1939.html
[57] https://github.com/linux-can/can-utils/blob/master/can-j1939-kickstart.md
[58] https://github.com/visionsystemsgmbh/programming_examples/tree/master/CAN/c/socketcan

7.6. I²C

I²C[59] (Inter-Integrated Circuit) is a multi-master serial computer bus. In the OnRISC integrated I²C controller is already supported by the mainline kernel. A good starting point on how to use I²C is Linux kernel documentation[60]. `i2c-tools`[61] project provides user space utilities to work with I²C.

To list all available I²C buses issue following command:

```
# i2cdetect -l
i2c-1   i2c           OMAP I2C adapter              I2C adapter
```

Knowing the bus you can now search for devices connected to it. As can be seen from the output below `i2cdetect` has found all devices registered in iR5221 DTS file (0x20 - TCA6416, 0x2d - PMIC and 0x50 - EEPROM):

```
# i2cdetect -r 1
i2cdetect: WARNING! This program can confuse your I2C bus
Continue? [y/N] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: UU -- -- -- -- -- -- -- -- -- -- -- -- UU -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

7.7. Watchdog Timer

The OnRISC provides a watchdog timer (WDT). The access to the WDT occurs via `/dev/watchdog` device. After starting the WDT it should be turned off or reloaded after the critical part otherwise the system will reboot. Table 12 shows the most important IOCTLs to control WDT.

IOCTL	Description
`WDIOC_SETTIMEOUT`	set timeout and start the timer
`WDIOC_GETTIMEOUT`	get timeout
`WDIOC_SETOPTIONS`	enable (`WDIOS_ENABLECARD`) disable (`WDIOS_DISABLECARD`) the timer
`WDIOC_KEEPALIVE`	reload the timer

Table 12: Watchdog Timer IOCTLs

The WDT driver provides a special option called "Disable watchdog shutdown on close" to prevent stopping the timer on WDT descriptor close (see Figure 4). This is a compile-time option. It is deactivated by default, so you have to build your own kernel if you need it.

[59] http://en.wikipedia.org/wiki/I2C
[60] https://www.kernel.org/doc/Documentation/i2c/dev-interface
[61] https://i2c.wiki.kernel.org/index.php/I2C_Tools

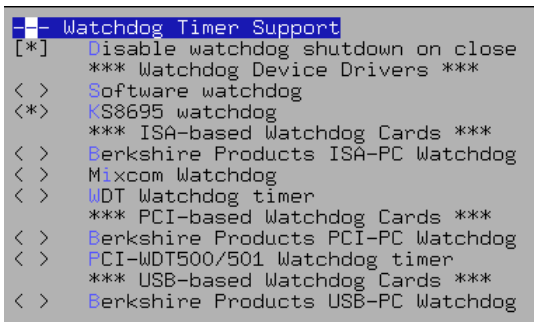

Figure 4: Watchdog Timer Support

See https://www.kernel.org/doc/Documentation/watchdog/watchdog-api.txt for WDT usage example.

7.8. Read Hardware Parameters like MAC Address, Serial Number etc.

Use `onrisctool -s` and you'll get a list of all parameters stored in EEPROM. For software development `libonrisc` (refer to Section 7.1) should be used to get hardware parameters.

7.9. Built-in Touchscreen Calibration (VS-860 Only)

You can choose between two calibration methods:

- tslib[62] - suitable for Qt Embedded applications
- X11 evdev driver - as the name says suitable for X11

Tslib package provides ts_calibrate utility, to calibrate the touchscreen. New calibration values will be written to `/etc/pointercal`. Tslib requires following environment variables to use the touchscreen:

```
TSLIB_CALIBFILE=/etc/pointercal
TSLIB_CONFFILE=/etc/ts.conf
TSLIB_TSDEVICE=/dev/input/event1
```

Hwtest-qt utility provides special Python script (scripts/input_dev.py), that reads /proc/bus/input/devices and configures `TSLIB_TSDEVICE` together with Qt Embedded related environment variables.

X11 evdev driver will be calibrated via xinput_calibrator (must be started from X11 session). The calibration values will be stored in `/usr/share/X11/xorg.conf.d/10-evdev.conf`. Example:

```
Section "InputClass"
Identifier "evdev touchscreen catchall"
MatchIsTouchscreen "on"
MatchDevicePath "/dev/input/event*"
Option "Calibration" "28 999 988 41"
Driver "evdev"
EndSection
```

[62] https://github.com/kergoth/tslib

8. Debian BSPs

8.1. ELBE

E.mbedded L.inux B.uild E.nvironment (ELBE)[63] is a Debian based system to generate root-filesystems for embedded devices. It is a replacement for `vsdebootstrap`. As stated by the project, it takes the following approach to fulfil its objective:

- Standard packages are not self compiled. ELBE uses the Debian distributions binary packages.
- Own applications are not cross compiled. They are built natively on the target architecture in a `chroot` environment using `qemu`.
- The root-filesystem is a subset of the Debian system inside the `chroot` environment. This implicitly ensures, that the same toolchain is used on the development machine and on the target.
- Updating, adding or removing a package is done via Debian's package-management (`apt`) which also resolves package dependencies.

The main idea is to use the own Debian repository, where both VScom packages like `kernel`, `libonrisc` as also customer's own packages are hosted. This repository can be used during the image creation as also afterwards from the device itself. Our README.md on the GitHub site[64] illustrates this approach and guides you through the initial image creation process.

The whole root filesystem configuration is stored in a single XML files, that can be found under `configs` folder.

8.2. vsdebootstrap

Debian provides tools to create custom root file system directly on the host. One of these tools is `debootstrap`[65]. Our system image is created with a help of scripts wrapped around `debootstrap`. You can find the scripts and corresponding documentation on our GitHub account: vsdebootstrap. `vsdebootstrap` not only creates root file system with OSS packages like `openvpn`, `ssh` etc., but also installs our packages like `libonrisc` and `hwtest-qt`. In addition it provides facilities to embed your own packages and configuration files into resulting root file system. By default `vsdebootstrap` creates an image for the current stable Debian version.

9. Buildroot

Buildroot[66] is a set of Makefiles and patches that allows you to easily generate a cross-compilation toolchain, a root filesystem and a Linux kernel image for your target. Buildroot can be used for one, two or all of these options, independently. The benefits are:

- you don't have to supply the toolchain it will be built automatically

[63] https://elbe-rfs.org
[64] https://github.com/visionsystemsgmbh/vscom-elbe
[65] https://wiki.debian.org/Debootstrap
[66] https://buildroot.org/

- you can choose between various file systems for your target image
- you can build a very small and quick booting system that suits your needs
- drivers compiled as modules will be automatically installed on your file system image
- many more

Bootlin provides very informative slides[67] related to Buildroot usage. We recommend to go through them before you begin using our BSP.

9.1. Build Host Requirements

Before building your root file system please check BR's software requirements[68]. It is also recommended, but not mandatory to build BR on a machine with more than 1GB RAM. You'll also need at least 6GB of free disk space in order to download and build needed packages.

9.2. Downloading

`README.md` in `onrisc_br_bsp` specifies a minimal required Buildroot version, so you can either use this tag or just clone master branch as shown below. Buildroot provides BR2_EXTERNAL[69] feature, that lets you keep your configuration and packages outside of Buildroot itself. To get our BSP execute following steps:

1. git clone https://github.com/visionsystemsgmbh/onrisc_br_bsp.git
2. git clone git://git.buildroot.net/buildroot or git clone http://git.buildroot.net/git/buildroot.git (if you're behind a firewall)
3. cd buildroot
4. make BR2_EXTERNAL=../onrisc_br_bsp/ help

9.3. BSP Structure

- `board/` - this folder provides actual BSP, i.e. kernel patches, kernel config etc.
- `Config.in` - Buildroot's local package config file
- `configs/` - Buildroot's defconfigs per device like baltos_defconfig etc.
- `external.mk` - Buildroot's local package config file
- `local.mk`[70] - file needed to override package source directory
- `package/` - local packages like libonrisc and packages created by customer
- `README.md` - BSP documentation

BSP for Baltos provides following defconfigs:

[67] https://bootlin.com/doc/training/buildroot/buildroot-slides.pdf
[68] http://nightly.buildroot.org/manual.html#requirement
[69] http://nightly.buildroot.org/manual.html#outside-br-custom
[70] http://nightly.buildroot.org/manual.html#_using_buildroot_during_development

- `configs/baltos_defconfig` - default config with U-Boot bootloader, kernel and a root file system
- `configs/baltos_barebox_defconfig` - default config with barebox bootloader, kernel and a root file system
- `configs/baltos_initramfs_defconfig` - default config with U-Boot bootloader, kernel and a initramfs root file system embedded into `kernel-fit.itb`
- `configs/baltos_experimental_defconfig` - default config with barebox bootloader, recent kernel, a root file system and patches rebased against a vanilla kernel

9.4. Building the Image

After checking out the repository you can build the default image with U-Boot bootloader by issuing the following command:

```
make baltos_defconfig && make
```

For Barebox bootloader invoke following commands instead:

```
make baltos_barebox_defconfig && make
```

Buildroot will start to automatically download needed packages and compile them. At the end of this procedure you'll find built images under:

```
output/images
```

The rootfs image can now be burned on to the target media.

9.5. Copying the Created Image to the System

9.5.1. SD/microSD-card

Copy Genimage Created Image `onrisc_br_bsp` provides a script `post-image.sh`[71], that creates a ready-to-burn image for the SD/microSD-card including both FAT and ext4 partitions. After the build process is finished you'll find following file under `output/images`:

```
sdcard.img
```

This file can then be directly burned via `dd` or Win32 Disk Imager. See Section D.1 for further details.

[71] https://github.com/visionsystemsgmbh/onrisc_br_bsp/blob/master/board/vscom/baltos/post-image.sh

Copy Separate Images Manually On your SD/microSD-card you need to create two partitions as described in Section 4.2.2. Copy following files from output/images to the FAT partition:

- `MLO` - SPL
- `u-boot.img` - U-Boot
- `kernel-fit.itb` - zImage + DTB as FIT binary

After this copy `onrisc_br_bsp/board/vscom/baltos/uEnv.txt` to the same partition.

Root file system is packaged as a tarball `output/images/rootfs.tar.bz2`. All you need to do is to extract it to the second partition. Following steps will be needed on Debian/Ubuntu to burn rootfs assuming, that SD/microSD-card is assigned to `/dev/sdd`:

```
sudo mount /dev/sdd2 /mnt
sudo rm -fr /mnt/*
sudo tar xjf output/images/rootfs.tar.bz2 -C /mnt/
sudo umount /mnt
```

9.5.2. NAND

Copy `MLO`, `u-boot.img`, `kernel-fit.itb` and `rootfs.tar.bz2` to your SD/microSD-card, for example to `/root/`. Boot from this SD/microSD-card and perform following commands:

1. `cd /root/`
2. `cat MLO > /dev/mtdblock0`
3. `cat u-boot.img > /dev/mtdblock4`
4. `ubiformat -y /dev/mtd5`
5. `ubiattach -p /dev/mtd5`
6. `ubimkvol /dev/ubi0 -N kernel -s 56MiB`
7. `mount -t ubifs ubi0:kernel /mnt`
8. `cp kernel-fit.itb /mnt`
9. `umount /mnt`
10. `ubimkvol /dev/ubi0 -N rootfs -s 180MiB`
11. `mount -t ubifs ubi0:rootfs /mnt`
12. `tar xjf rootfs.tar.bz2 -C /mnt`
13. `umount /mnt`

The example above shows the universal method, as you can use the same `rootfs.tar.bz2` to populate to any already mounted file system. There is a faster way to write a UBIFS/SquashFS image to a UBI partition. Instead of steps 10-13 perform following steps to write a UBIFS image:

```
ubiupdatevol /dev/ubi0_1 rootfs.ubifs
```

For SquashFS image:

```
ubiupdatevol /dev/ubi0_1 rootfs.squashfs
```

Please refer to this article[72] about flashing UBIFS images for more details.

If the system doesn't start from NAND, you can always boot from the MMC device using the "emergency" jumper (see Section "Emergency" Jumper).

9.6. Customizing the Image

Buildroot uses Kconfig[73] subsystem to manage the process of configuration just like Linux kernel does. Following bigger parts can be configured:

- `make menuconfig` - configures Buildroot: toolchain, rootfs, packages
- `make busybox-menuconfig` - configures BusyBox[74]
- `make linux-menuconfig` - configures Kernel

First place to look for choosing new packages is BusyBox configuration. If the package is not available there, then it can be found in Buildroot itself, if it was included into its repository.

Buildroot provides a file system overlay facility to store files that should be added to the rootfs like configuration files (`/etc/network/interfaces`, etc.), SSH keys, etc. Aside from this, you can specify your scripts that will be invoked before or after the root file system is packaged:

- `BR2_ROOTFS_POST_BUILD_SCRIPT`
- `BR2_ROOTFS_POST_IMAGE_SCRIPT`

9.7. Compiling Your Own Software

There are following ways on how to get your software to be build and installed on the system using Buildroot:

- adding your software to the `onrisc_br_bsp/package` folder (recommended)
- using generated toolchain to compile your software and then manually copying it to `output/target`

The first way is described here Add packages and showed in the example below. The second way is described here Using toolchain. See this article on how to override source directory Source directory override.

You can also use Buildroot together with Eclipse to develop your application Eclipse integration.

[72] https://bootlin.com/blog/creating-flashing-ubi-ubifs-images/
[73] en.wikipedia.org/wiki/Kconfig
[74] www.busybox.net

Example This sample project will show, how to create a CMake package with for example libsocketcan dependency in BR. First create sample project folder:

```
mkdir /home/user/myprog
```

Then create `test.c` from Listing 6 and `CMakeLists.txt` with following content:

```
project(myprog C)
add_executable(test test.c)
target_link_libraries(test socketcan)
install(TARGETS test RUNTIME DESTINATION bin)
```

Now you need to create package recipe in `onric_br_bsp`:

```
mkdir package/myprog
touch package/myprog/Config.in
touch package/myprog/myprog.mk
```

`package/myprog/Config.in` will have following content:

```
config BR2_PACKAGE_MYPROG
        bool "myprog"
        select BR2_PACKAGE_LIBSOCKETCAN
        help
          Package example
```

`package/myprog/myprog.mk`:

```
################################################################################
#
# myprog
#
################################################################################
MYPROG_VERSION = 1.0.0
MYPROG_SITE = git://git.mygitsrv.com/myprog.git
MYPROG_DEPENDENCIES = libsocketcan
$(eval $(cmake-package))
```

`package/myprog/Config.in` must be included in central `Config.in`:

```
source "$BR2_EXTERNAL/package/libonrisc/Config.in"
source "$BR2_EXTERNAL/package/myprog/Config.in"
```

As `MYPROG_SITE` is not available and we want Buildroot to compile the working copy of myprog in `/home/user/myprog`, `local.mk` must be edited to point to myprog's folder:

```
MYPROG_OVERRIDE_SRCDIR=/home/user/myprog
```

You can now select `myprog` via `make menuconfig` and entering "User-provided options". You'll see `myprog` next to `libonrisc`. Select `myprog` and invoke `make myprog-rebuild`. This will rsync `/home/user/myprog` into `output/build/myprog-custom` and build/install it. So every time you make changes to `test.c`, just invoke `make myprog-rebuild` to rebuild it.

9.8. Setup SSH Server

SSH server will be activated via the BR2_PACKAGE_OPENSSH variable. After copying the new rootfs image to the media, you'll see RSA/DSA keys will be created. By default password is required to login via SSH. You can configure the system in the following ways:

- define root password via `passwd root`
- change `sshd` behavior to accept empty passwords (set `PermitEmptyPasswords` to `yes` in `/etc/sshd_config`)

9.9. Getting Help

If you encounter any non-kernel related problems, you should get in contact with Buildroot community. As usual it is recommended to read the mailing list archives before asking questions on the mailing list.

10. Yocto

The Yocto Project[75] is an open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of the hardware architecture. This section provides step-by-step instructions on how to add Baltos BSP layer to your project. It is not meant as a full Yocto tutorial. You'll find links to particular sections of the Yocto documentation describing variables and files used in this mini tutorial.

10.1. Downloading

Layers are the main Yocto concept and consist of a set of recipes matching a common purpose. To build an image having a kernel and `libonrisc` for Baltos you'll need to add layer `meta-baltos` to your `bblayers.conf` file. Due to `libonrisc` dependencies you'll have to use Yocto 2.2 (morty) or newer. If you build Yocto for the first time, you'll need to perform following steps:

1. `mkdir /home/user/yocto`
2. `cd /home/user/yocto`
3. `git clone -b morty git://git.yoctoproject.org/poky.git`
4. `cd poky`
5. `git clone -b morty git://git.openembedded.org/meta-openembedded`
6. `git clone https://github.com/visionsystemsgmbh/meta-baltos.git`
7. `source oe-init-build-env`

10.2. Build Configuration

After these steps you'll have a basic setup of all needed repositories and are ready to configure your rootfs image. First of all you'll need to add cloned layers to `conf/bblayers.conf`[76]. Open this file and add baltos and openembedded layers so it looks like this:

```
# LAYER_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"
BBPATH = "${TOPDIR}"
BBFILES ?= ""
BBLAYERS ?= " \
/home/user/yocto/poky/meta \
/home/user/yocto/poky/meta-poky \
/home/user/yocto/poky/meta-yocto-bsp \
/home/user/yocto/poky/meta-baltos \
/home/user/yocto/poky/meta-openembedded/meta-oe \
/home/user/yocto/poky/meta-openembedded/meta-python \
/home/user/yocto/poky/meta-openembedded/meta-networking \
"
```

[75] https://www.yoctoproject.org/
[76] http://www.yoctoproject.org/docs/2.1/ref-manual/ref-manual.html#structure-build-conf-bblayers.conf

```
BBLAYERS_NON_REMOVABLE ?= " \
/home/user/yocto/poky/meta \
/home/user/yocto/poky/meta-poky \
"
```

Add following lines to `conf/local.conf`[77] to include Baltos BSP into the build process:

- `MACHINE ??= "baltos"`[78]
- `IMAGE_INSTALL_append = " libonrisc"`[79]

Invoking `bitbake core-image-base` would start the build process. At the end you'll find all needed images under `tmp/deploy/images/baltos`[80]:

- `tmp/deploy/images/baltos/kernel-fit.itb` - zImage + DTB as a FIT binary
- `tmp/deploy/images/baltos/core-image-base-baltos.tar.bz2` - symbolic link to the rootfs
- `tmp/deploy/images/baltos/MLO` - symbolic link to SPL
- `tmp/deploy/images/baltos/u-boot-img` - symbolic link to U-Boot

Use steps from Copying the Created Image to the System in order to copy these files to your boot medium.

10.3. Meta-baltos Structure

Baltos BSP layer provides following files/recipes:

- `conf/machine/baltos.conf` - initial configuration for kernel and u-boot
- `recipes-bsp` - recipes for `libonrisc`, `vsopenvpnd` and other packages
- `recipes-kernel` - kernel recipe, defconfig and patches

[77] http://www.yoctoproject.org/docs/2.1/ref-manual/ref-manual.html#structure-build-conf-local.conf
[78] http://www.yoctoproject.org/docs/2.1/ref-manual/ref-manual.html#var-MACHINE
[79] http://www.yoctoproject.org/docs/2.1/ref-manual/ref-manual.html#var-IMAGE_INSTALL
[80] http://www.yoctoproject.org/docs/2.1/ref-manual/ref-manual.html#structure-build-tmp-deploy-images

11. OpenWrt

11.1. BSP Structure

Our OpenWrt BSP is hosted on GitHub and consists of following repositories:

- openwrt-setup - main repository providing `build_openwrt` script, that manages the whole build process
- onrisc-target - this repository provides kernel patches, configuration as also file system overlay files
- vscom-packages - this is a feed repository providing `libonrisc` and its dependencies

11.2. Initial Setup

You'll need to download only the first repository. Prior to doing so make sure you've installed all required packages as mentioned in README.md.

Perform following actions:

1. `git clone https://github.com/visionsystemsgmbh/openwrt-setup.git`
2. `cd openwrt-setup`
3. `./build_openwrt all`

The whole process can take more than one hour to finish. After this you'll get an `openwrt` folder with upstream OpenWrt and our patch applied as also an SD card image (`openwrt/bin/baltos/sdcard.img`). You can change to `openwrt` folder and work with its native build system. For further details and `build_openwrt` options refer to the above mentioned `README.md` file.

OpenWrt projects provides a comprehensive documentation[81], that covers many topics like OpenWrt usage, configuration and customization.

11.3. Accessing Web Interface

OpenWrt provides DHCP server accessible via LAN port. Attaching your PC to LAN port of your Baltos device would assign an IP address for 192.168.1.0/24 subnet. Baltos IP then is 192.168.1.1. Enter this IP in your browser and you'll get the initial OpenWrt login page. By default no password is set and clicking on "Login" button is sufficient.

11.4. Installing OpenWrt to NAND

You can use previously obtained SD card image to burn OpenWrt to NAND. Access OpenWrt's web interface and go to "System->Backup/Flash Firmware" menu item. There you'll find "Install OpenWrt to Flash" section (see Figure 5 on page 63). Read related information concerning configuration backup and you can proceed with installation via clicking on the "Flash" button.

[81] https://openwrt.org/docs/start

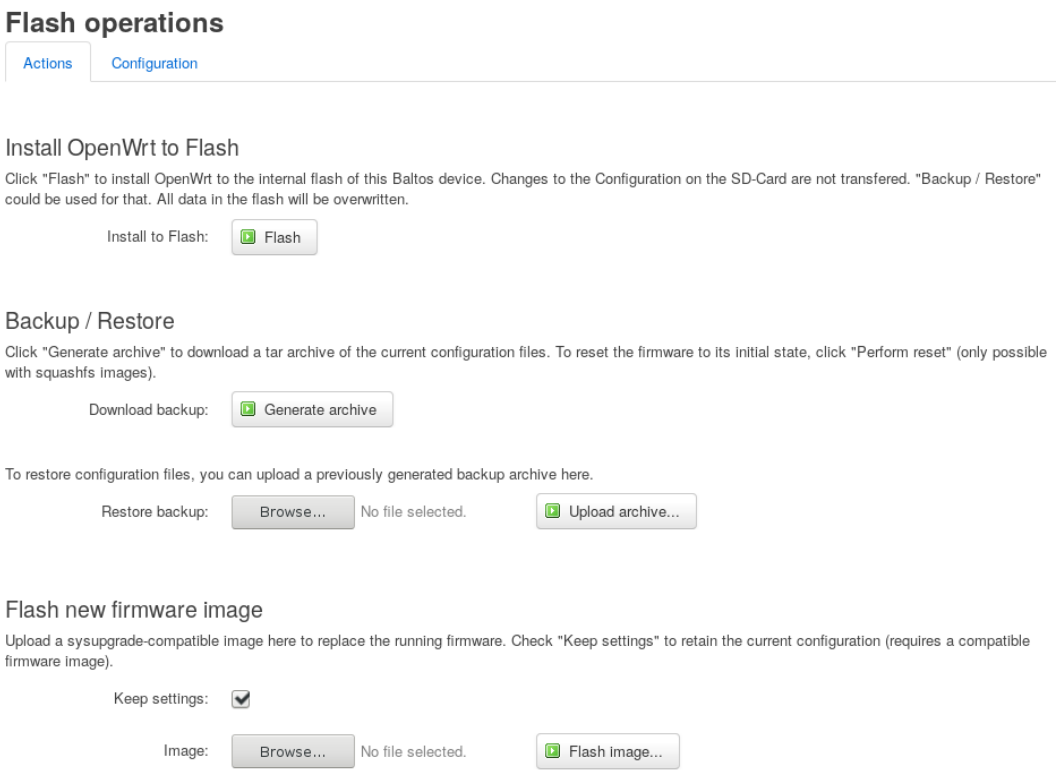

Figure 5: OpenWrt: Install to Flash

12. IoT

This Section tries to give insight into IoT universe and also to show how OnRISC can be used as an IoT Router. For deeper understanding we recommend to look at this free eBook: A Reference Guide To The Internet Of Things[82].

IoT or the Internet of Things is a technology about connecting physical devices like sensors and actuators to the cloud in order to analyse data collected from physical devices and control the actuators. So the role of OnRISC will be to talk to sensors/actuators via its interfaces like CAN, Ethernet, serial, GPIO, WLAN etc. and provide this data to a cloud service (see Figure 6 on page 64). OnRISC device can be either a gateway between the sensors and the cloud or maintain controlling logic.

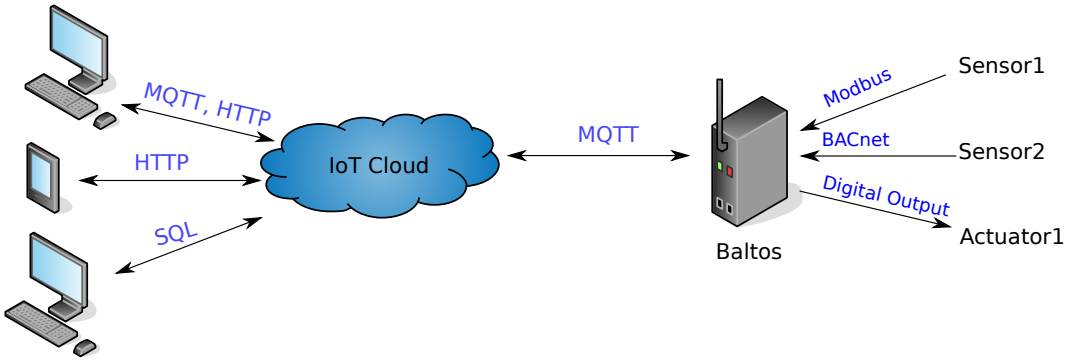

Figure 6: Baltos as IoT Router

12.1. IoT Related Protocols

There are many protocols used in IoT world: MQTT, HTTP, CoAP, AMQP etc. We will describe only MQTT protocol as it is wide spread and it is supported by the most IoT cloud providers like Amazon, IBM and Microsoft.

12.1.1. MQTT

As stated on the projects site[83] MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. As of version 3.1.1 MQTT became an OASIS Standard.

MQTT utilizes a “publish/subscribe” message transport model. The central part of this protocol is a MQTT broker, that receives, manages and routes messages among its nodes. Client authentication and authorization is also made by the MQTT broker. Table 13 on page 65 and Figure 7 on page 65 provide a MQTT example where three publishers send temperature sensor values (T1-T3) and three subscribers receiving only relevant values.

[82] https://bridgera.com/ebook/
[83] http://mqtt.org

Topic Name	Subscriber
T1	Sub1, Sub3
T2	Sub2
T3	Sub2, Sub3

Table 13: MQTT Scheme

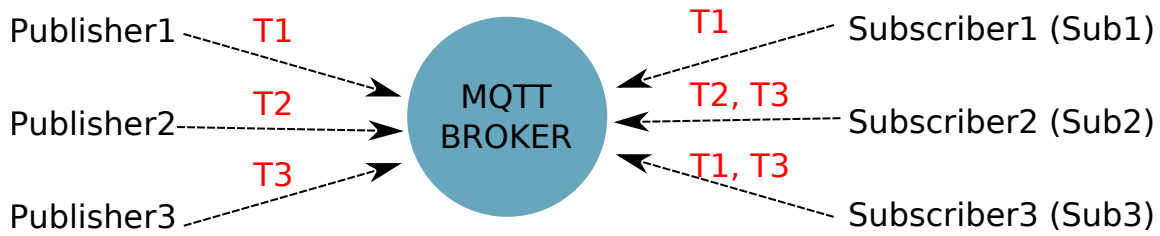

Figure 7: MQTT Scheme

Installation Buildroot already provides both C and Python protocol implementations:

- `BR2_PACKAGE_PAHO_MQTT_C`
- `BR2_PACKAGE_PYTHON_PAHO_MQTT`

If you need OpenSSL support, it must be activated prior to MQTT building.

Debian installation requires following steps for C protocol implementation:

1. `apt install libssl-dev debhelper fakeroot lsb-release`
2. `cd /usr/src/`
3. `git clone https://github.com/eclipse/paho.mqtt.c.git`
4. `cd paho.mqtt.c/`
5. `mkdir build`
6. `cd build`
7. `cmake .. -DPAHO_WITH_SSL=TRUE -DPAHO_BUILD_DEB_PACKAGE=TRUE`
8. `make`
9. `cpack`
10. `dpkg -i *.deb`

For Python implementation:

1. `apt install python-pip python3-pip`
2. `pip install paho-mqtt` - for Python 2.x environment
3. `pip3 install paho-mqtt` - for Python 3.x environment

Example: mqtt_gpio This program publishes Baltos digital inputs:

- *onrisc/gpio/input/0*
- *onrisc/gpio/input/1*
- *onrisc/gpio/input/2*
- *onrisc/gpio/input/3*

And subscribes to digital outputs:

- *onrisc/gpio/output/0*
- *onrisc/gpio/output/1*
- *onrisc/gpio/output/2*
- *onrisc/gpio/output/3*

If any output on the MQTT broker would change, Baltos would propagate this value to its digital outputs. And as soon as Baltos inputs change their state, these values will be propagated to the MQTT broker.

In order to build the C test example perform:

1. cd /usr/src/
2. git clone https://github.com/visionsystemsgmbh/programming_examples.git
3. cd programming_examples/mqtt/c/
4. mkdir build
5. cd build/
6. cmake ..
7. make

`mqtt_gpio` has following syntax:

```
mqtt_gpio IP address [port]
```

IP address indicates MQTT broker's IP address and *port* is an optional argument specifying broker's TCP port (default value is 1883).

`mqtt_gpio` requires following setup:

1. MQTT broker (either on Baltos or on your host)
2. connect IN0 with OUT0 using a 4,7k resistor

For our example we will use Node.js based broker Mosca[84] running on a desktop PC. Install Node.js according to the project's documentation[85]. After this invoke:

1. `npm install mosca pino -g` with super user permissions
2. `mosca -v | pino`

[84] http://www.mosca.io
[85] https://nodejs.org/en/download/package-manager

We assume, that your host has IP address 192.168.1.170 and MQTT broker works with the standard TCP port 1883. On Baltos invoke:

```
./mqtt_gpio 192.168.1.170
```

You'll get following output showing that Baltos sent its initial digital input state:

```
Waiting for up to 10 seconds for publication of 0
on topic onrisc/gpio/input/0 for client with ClientID: Baltos
Message with token value 2 delivery confirmed
Message with delivery token 2 delivered
Waiting for up to 10 seconds for publication of 0
on topic onrisc/gpio/input/1 for client with ClientID: Baltos
Message with token value 3 delivery confirmed
Message with delivery token 3 delivered
Waiting for up to 10 seconds for publication of 0
on topic onrisc/gpio/input/2 for client with ClientID: Baltos
Message with token value 4 delivery confirmed
Message with delivery token 4 delivered
Waiting for up to 10 seconds for publication of 0
on topic onrisc/gpio/input/3 for client with ClientID: Baltos
Message with token value 5 delivery confirmed
Message with delivery token 5 delivered
```

Mosca's output shows, that client with ID "Baltos" has made a connection and subscribed to the topic *"onrisc/gpio/output/#"* i.e. all published outputs:

```
[2017-09-13T15:25:52.565Z] INFO (mosca/16765 on debian9): client connected
    client: "Baltos"
[2017-09-13T15:25:52.571Z] INFO (mosca/16765 on debian9): subscribed to topic
    topic: "onrisc/gpio/output/#"
    qos: 1
    client: "Baltos"
```

Let's toggle OUT1:

```
onrisctool -a -0x10 -b 0x10
```

In reaction to this `mqtt_gpio` would produce following output:

```
Waiting for up to 10 seconds for publication of 1
on topic onrisc/gpio/input/0 for client with ClientID: Baltos
Message with token value 6 delivery confirmed
Message with delivery token 6 delivered
```

12.2. IoT Programming: Node-RED

Node-RED[86] is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

[86] https://nodered.org

Installation In Buildroot you'll have to activate Node.js package and provide a list of required modules:

- `BR2_PACKAGE_OPENSSL`
- `BR2_PACKAGE_NODEJS`
- `BR2_PACKAGE_NODEJS_MODULES_ADDITIONAL="node-red node-red-dashboard"`

For Debian 9 perform following steps:

1. `apt install -y apt-transport-https`
2. `echo "deb https://deb.nodesource.com/node_6.x stretch main" > /etc/apt/sources.list.d/nodesource.list`
3. `wget -qO- https://deb.nodesource.com/gpgkey/nodesource.gpg.key | apt-key add -`
4. `apt update`
5. `apt install -y nodejs`
6. `npm install -g node-red node-red-dashboard`

Example Node-RED example connects to modbusgpio daemon (see Section 5.9) via Modbus/TCP and hence will work only in Debian. Perform following actions:

1. `cd /usr/src/`
2. `git clone https://github.com/visionsystemsgmbh/programming_examples.git`
3. `cp programming_examples/node-red/gpio.json /root/.node-red/flows_onrisc.json`
4. `npm install -g --unsafe-perm node-red-contrib-modbus`
5. `modbusgpio 502&`
6. `node-red`

We assume that Baltos has its default IP address 192.168.254.254. As soon as you can see the output shown below, you can point your browser to *http://192.168.254.254:1880/*. You'll see the flow chart as shown in Figure 8 on page 69. Now open a new tab in your browser and point it to *http://192.168.254.254:1880/ui/#/0* to see the dashboard as shown in Figure 9 on page 69. You can click on the switches to change the digital output state and if you connect INs with OUTs via 4,7k resistors, you'll see, how digital inputs get changed accordingly.

```
Welcome to Node-RED
===================

15 Sep 08:53:43 - [info] Node-RED version: v0.17.5
15 Sep 08:53:43 - [info] Node.js  version: v6.11.3
15 Sep 08:53:43 - [info] Linux 3.18.32 arm LE
15 Sep 08:53:47 - [info] Loading palette nodes
15 Sep 08:54:05 - [info] Dashboard version 2.4.3 started at /ui
15 Sep 08:54:07 - [info] Settings file  : /root/.node-red/settings.js
15 Sep 08:54:07 - [info] User directory : /root/.node-red
15 Sep 08:54:07 - [info] Flows file     : /root/.node-red/flows_onrisc.json
15 Sep 08:54:07 - [info] Server now running at http://127.0.0.1:1880/
15 Sep 08:54:07 - [info] Starting flows
15 Sep 08:54:08 - [info] Started flows
```

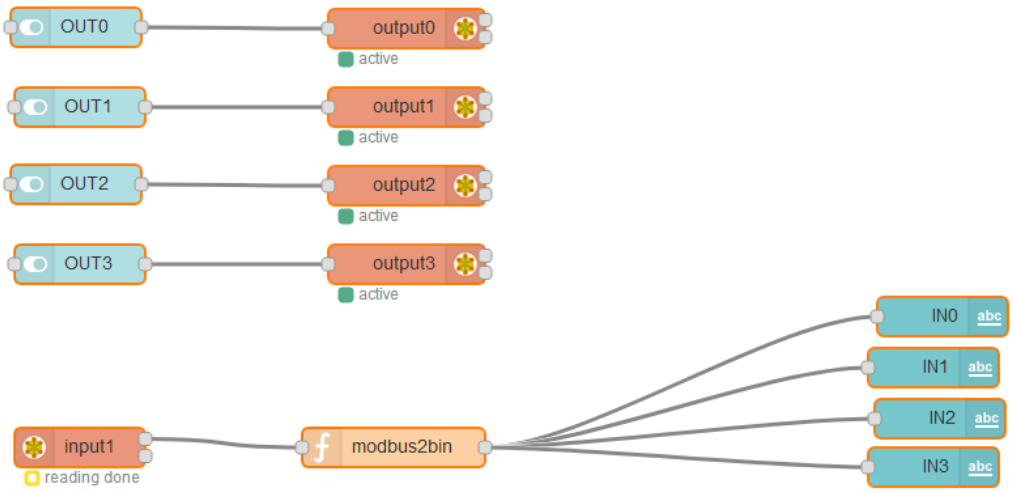

Figure 8: Node-RED: Modbus/GPIO Example Flow Chart

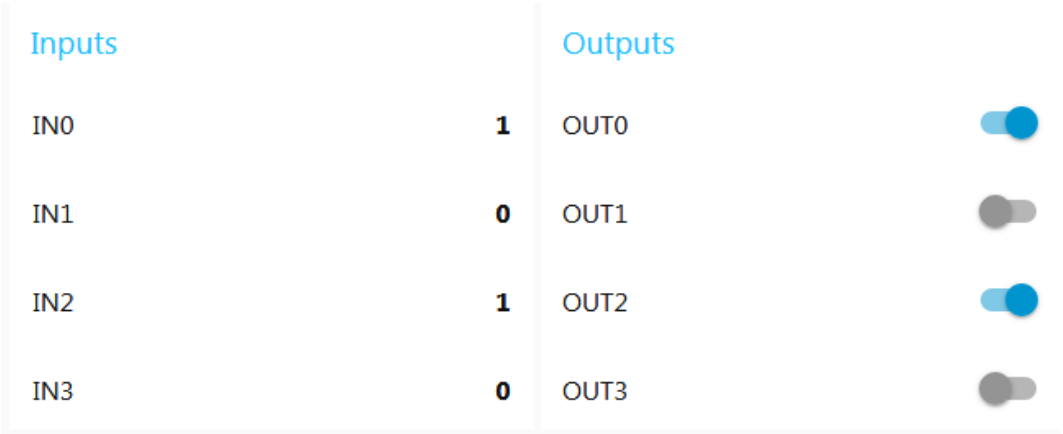

Figure 9: Node-RED Modbus/GPIO Example Dashboard

A. Debian Maintenance Notes

A.1. Debian Package Management

Debian uses following utilities for the package management[87] :

- dpkg - the main package management program.
- APT - the Advanced Package Tool. It provides the `apt-get` program. `apt-get` allows a simple way to retrieve and install packages from multiple sources using the command line. Unlike `dpkg`, `apt-get` does not understand .deb files, it works with the packages proper name and can only install .deb archives from a source specified in `/etc/apt/sources.list`. `apt-get` will call `dpkg` directly after downloading the .deb archives from the configured sources.
- aptitude - a package manager for Debian GNU/Linux systems that provides a frontend to the apt package management infrastructure. `aptitude` is a text-based interface using the curses library, it can be used to perform management tasks in a fast and easy way.

To find a package execute:

`apt-cache search` *`pkg name`*

To view info such as version, dependencies, installed size etc. execute:

`apt-cache show` *`pkg name`*

To install packages execute:

`apt-get install` *`pkg1 pkg2 ...`* (su rights needed)

To update the list of package known by your system execute:

`apt-get update` (su rights needed)

To upgrade all the packages on your system

`apt-get upgrade` (su rights needed)

To remove packages from your system execute:

`apt-get remove` *`pkg1 pkg2 ...`* (su rights needed)

To install a package that is not contained in the repository download the *.deb file and execute:

`dpkg -i` *`pkg file name`* (su rights needed)

[87] for detailed information visit https://wiki.debian.org/PackageManagementTools

B. onrisctool

OnRISC configuration utility provides following features to configure OMAP3 based devices:

- configure serial interfaces (RS232, RS422 etc.)
- configure digital I/O
- configure LEDs
- get EEPROM info
- set MACs from EEPROM
- enable/disable mPCIe slot

B.1. Configure Serial Interfaces

Baltos and VS-860 provide two UARTs working in RS232/RS422/RS485 modes. Invoking `onrisctool -j` will show which mode each port is configured to:

```
# onrisctool -j
Port 1:  mode:  rs232 termination:  off source:  DIP
Port 2:  mode:  rs232 termination:  off source:  DIP
```

As one can see both ports are configured in RS232 mode through DIP-switches. To switch the first port into RS422 mode execute:

```
onrisctool -t rs422 -p 1
```

All possible modes are show in the Table 14. Termination can be enabled via `-r` parameter.

Mode	Description
dip	DIP-switch settings are valid
loop	loopback mode
rs232	RS232
rs422	RS422
rs485-fd	RS485 full duplex
rs485-hd	RS485 half duplex

Table 14: Serial Modes (Software switching)

B.2. Configuring Digital I/O

Baltos iR 5221/3220 provides 4 digital inputs and 4 digital outputs, Baltos 1080 provides 8 digital inputs and 8 digital outputs. With `onrisctool` you can read and modify data (see Table 15 on page 72).

Following example for Baltos iR 5221/3220 turns pins 0 and 2 to high and clears pin 1:

```
onrisctool -a 0x70 -b 0x50
```

The same example but for Baltos 1080:

Parameter	Description
`-a`	Bit mask. Defines, what bits will be affected via the data
`-b`	Data bits, that will affect only the bits defined by `-a` parameter
`-g`	Show current state

Table 15: GPIO Parameters

```
onrisctool -a 0x0700 -b 0x0500
```

Show current state:

```
# onrisctool -g
0x00000505
```

Figure 10 on page 72 shows, how mask and value are mapped to input/output connector for Baltos iR 5221/3220. In this example IN0 is connected to OUT0 and OUT0 has high level. As one can see 4 bytes are reserved, so that devices with more digital inputs/outputs can be supported (for example Baltos 1080, that uses two bytes), but for Baltos iR 5221/3220 only the least significant byte is important. Baltos 1080 uses least significant byte for inputs and the next byte for outputs.

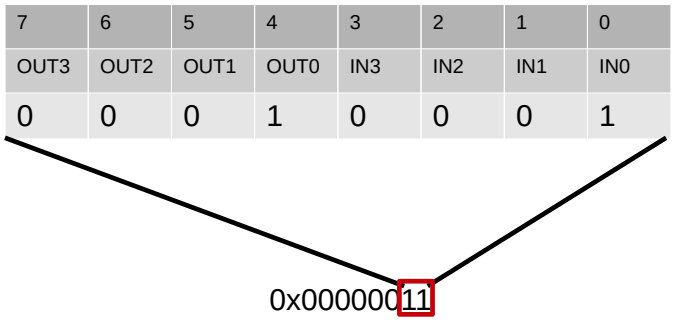

Figure 10: Baltos: Digital I/O Mapping

B.3. Configuring LEDs

Baltos devices provide up to 3 user configurable LEDs. LEDs will be configured via `-l` parameter. The call consists of an LED name (see Table 16 on page 72) and desired function (see Table 17 on page 73). In the example below power LED will be turned off:

```
onrisctool -l pwr:0
```

Name	Description	Color
pwr	power LED	red
app	application LED	green
wln	WLAN LED	blue

Table 16: LED Names

Function	Description
0	turn LED off
1	turn LED on
2	blink with different duty cycles for some seconds
3	get current LED state (on/off)

Table 17: LED Functions

B.4. Get EEPROM Info

`onrisctool -s` shows hardware related data like MACs, serial number etc. stored in EEPROM:

```
# onrisctool -s
Hardware Parameters
===================
Model:  210
HW Revision:  1.2
Serial Number:  550100239
Production date:  20.06.2015
MAC1:  746a8f0001f5
MAC2:  746a8f0001f6
MAC3:  746a8f0001f7
```

B.5. Setting LAN MACs from EEPROM

You can use in EEPROM stored MACs for internal LAN interfaces:

`onrisctool -m`

Executing this command will set MAC1 and MAC2 to `eth0` and `eth1/usb0`.

B.6. Controlling mPCIe Slot

The mPCIe slot in Baltos devices can be switched on and off via GPIO pin. This slot is off by default and must be enabled on demand. To do this invoke:

`onrisctool -k 1`

To disable the slot invoke:

`onrisctool -k 0`

To check current state invoke:

`onrisctool -k 3`

This would output following message, if the slot is on:

```
mPCIe switch:  on
```

C. hwtest-qt

Hwtest-qt provides various hardware tests in both console and graphical environment. To get GUI start `ghwtest-qt` instead of `hwtest-qt`. Hwtest-qt can be run using special configuration file.

```
hwtest-qt -qws -c /etc/hwtest.conf
```

If you want to execute the tests in parallel:

```
hwtest-qt -qws -c /etc/hwtest.conf --parallel-exec
```

Please see `hwtest-qt -qws --help` for further options.

C.1. Controller Area Network Test

CAN test requires USB-CAN device attached to one of the USB ports[88]. Connect internal CAN interface with USB-CAN via CAN cable and execute:

```
hwtest-qt -qws --test-can --cani=slcan0 --cano=can0
```

C.2. UART Test

Connect both internal serial ports via null-mode, cable and execute:

```
hwtest-qt -qws --test-serial --seri=/dev/ttyS1 --sero=/dev/ttyS2 --seropts="-t rs232"
```

C.3. Network Test

Connect both LAN insteraces via patch cable and execute:

```
hwtest-qt -qws --test-network --neti=eth0 --neto=eth1
```

C.4. RTC Test

```
hwtest-qt -qws --test-rtc
```

C.5. WLAN Test

WLAN test scans for available WLAN hosts (iw wlan0 scan). At least one AP must be running and antenna must be attached.

```
hwtest-qt -qws --test-wlan
```

[88] You'll need a slcan tools to bring USB-CAN up. See VSCAN SocketCAN FAQ

C.6. Bluetooth Test

Bluetooth test needs an active Bluetooth device like headset, Bluetooth dongle etc. to find during the scan.

```
hwtest-qt -qws --test-bt
```

C.7. Disk Test

Insert CFast card and execute:

```
hwtest-qt -qws --test-disk --drive=/dev/sda
```

C.8. Touch Test

If you start ghwtest-qt and you can precisely touch the buttons, then touch controller is functioning properly. You can calibrate it by executing

```
ts_calibrate
```

C.9. Button Test

For button test the user will be prompted to press buttons on the front panel from left to right, skipping the first button (power button).

```
hwtest-qt -qws --test-btn
```

C.10. Audio Test

Insert earphones into the left connector and execute:

```
hwtest-qt -qws --test-audio
```

You'll hear a woman voice saying "Front left"

C.11. Modem Test

Insert a mPCIe modem and a SIM card. If you only want to test both SIM and mPCIe slots, you don't need an antenna. In the other case, connect an appropriate antenna to the main connector on the mobile card.

```
hwtest-qt -qws --test-modem
```

For the real connection test execute:

```
hwtest-qt -qws --test-modem --apn=internet.eplus.de --site=www.google.com
```

D. Managing System Images

D.1. Flashing System Images

To copy a system image to a SD/microSD-card using your host PC just download the required system image from our FTP server[89]. The image is a self-extracting 7-Zip archive that looks like this:

```
20140805_debian_7_baltos.exe
```

In Windows, it can be extracted just with clicking on the file. In Linux you'll first need to install 7-Zip archiver (`p7zip-full` package in Debian) and then invoke:

```
7z e 20140805_debian_7_baltos.exe
```

D.1.1. Windows

Win32 Disk Imager[90] was developed to make/copy images from/to the flash drives connected to a Windows host (see Figure 11). You can use Etcher[91] as alternative. This application supports Linux, Windows and Mac OS.

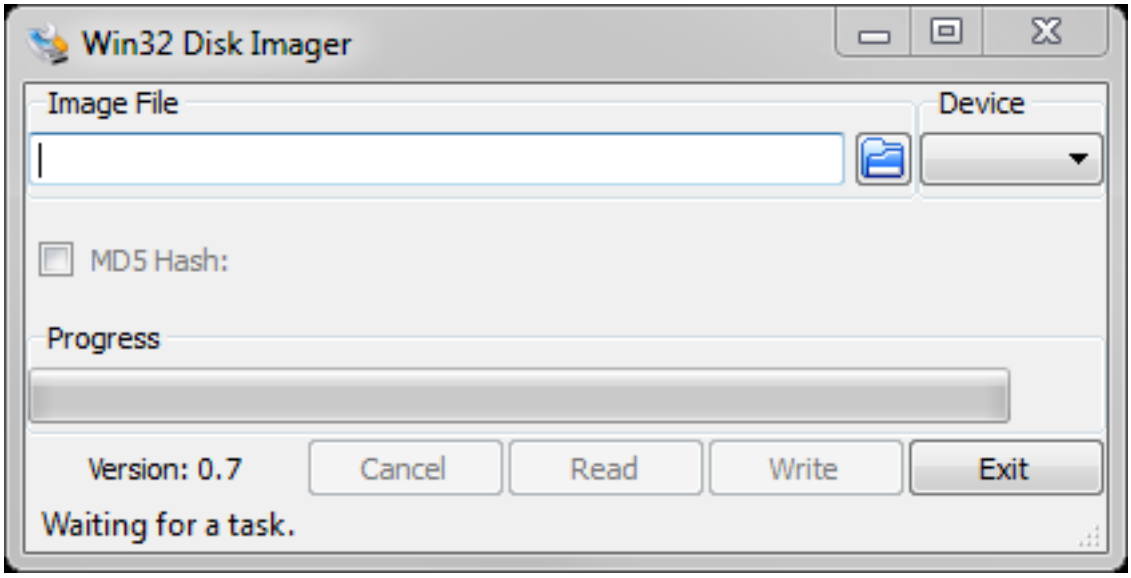

Figure 11: Win32 Disk Imager

[89] ftp://ftp.visionsystems.de/pub/multiio/OnRISC/Baltos/
[90] http://sourceforge.net/projects/win32diskimager/
[91] https://etcher.io/

D.1.2. Linux

From Host PC Copy already extracted `*.img` file (for example to `/home/user/`).

If USB card reader is used, the SD/microSD-card could be found under `/dev/sdx` devices (for example `/dev/sda`, `/dev/sdb` etc). Please note, that the system images are complete images of a SD/microSD-card not of the single partition and must be applied to the device like `/dev/sda` and not `/dev/sda1`.

For the example below it will be assumed that the SD/microSD-card is assigned to `/dev/sda` device and the system image is `20140805_debian_7_baltos.img`:

```
su
dd if=/home/user/20140805_debian_7_baltos.img of=/dev/sda bs=4096
```

To create an image from the SD/microSD-card execute following:

```
dd if=/dev/sda of=/home/user/image.img
```

D.2. Working with Partitions

It is also possible to write/extract only one partition at file system level. The advantage: you can write this partition even to a smaller medium. Please refer to this project FSArchiver

E. Frequently Asked Questions (FAQ)

There is dedicated website http://faq.visionsystems.de to handle the FAQ concerning OnRISC and other products provided by VScom. The customer question will be posted to the support team for approval and if approved appears in the corresponding category.

F. Licensing Information

Following precompiled images can be obtained for Baltos systems:

- Debian
- OpenWrt

The images provide software licensed under various Open Source licenses. The main licenses involved are the GNU General Public Licence version 2 (GPLv2) and the GNU Lesser General Public Licence version 2.1 (LGPLv2.1). You'll find the relevant text of those licenses below. In part there are similar public licenses involved.

Sections 9, 11, 10 and 8 describe how to compile these Open Source components and create an image from these components.

On Debian the texts of the common licenses can be found under /usr/share/common-licenses. License information for specific packages can be found on Debian's package data base[92].

OpenWrt provides all package sources on their FTP server[93]. The source file archives also provide related license information. Here is the main OpenWrt license[94].

To export the source code and licenses of all used packages in Buildroot invoke `make legal-info`. All required files can be found under `output/legal-info`.

Visit our Open Source related page[95] for more information about Open Source Software used in our products.

F.1. GNU GENERAL PUBLIC LICENSE

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge

[92] https://www.debian.org/distrib/packages
[93] https://sources.openwrt.org/
[94] https://openwrt.org/license
[95] http://www.visionsystems.de/opensource

for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the

major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this

License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, 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.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE 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.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and an idea of what it does.>

Copyright (C) < yyyy> <name of author>

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 2 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, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

F.2. GNU LESSER GENERAL PUBLIC LICENSE

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we

use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to

this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified

Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify

the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "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 LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY 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 LIBRARY (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 LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and an idea of what it does.>

Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library 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 Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in

the library 'Frob' (a library for tweaking knobs) written

by James Random Hacker.

<signature of Ty Coon>, 1 April 1990

Ty Coon, President of Vice

That's all there is to it!

Index