

MORIN Module. B2B-Module.

Manual

FCC/IC Statements.....	4
FCC-WARNING.....	4
IC-WARNING.....	4
Setup Hardware	6
Setup via Web-Interface:.....	6
Optional: Setup via Serial-USB-Interface:.....	6
Rescue Mode:.....	6
Overview Software	9
OpenWrt build system – Installation.....	10

Documentation

Prerequisites	10
Install procedure on GNU/Linux.....	10
Luci feed	14
OpenWrt build system – Usage	15
Prerequisites	15
Procedure	15
Updating Sources with Git.....	15
Updating Feeds.....	16
Image Configuration	16
Make menuconfig.....	16
Explanations	17
Kernel configuration (optional)	17
Source Mirrors.....	18
Configure using config diff file.....	18
Using diff file	19
Patches	19
Custom files	19
Defconfig	19
Building Images	20
Make Tips	20
Building in the background.....	20
Building single Packages	20

Documentation

Spotting build errors	21
Getting beep notification	21
Skipping failed packages.....	21
Locating Images.....	22
Cleaning Up	23
Clean.....	23
Dirclean	23
Distclean.....	23
Examples	25
Troubleshooting	26
Missing source code file, due to download problems.....	26
Compilation errors	26
WARNING: skipping <package> -- package not selected.....	26
*-factory.bin and *-sysupgrade.bin images for my device do not get generated.....	26
Notes	27

Documentation

FCC/IC Statements

FCC-WARNING

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures: Reorient or relocate the receiving antenna. Increase the separation between the equipment and receiver. Connect the equipment into an outlet on a circuit different from that to which the receiver is connected. Consult the dealer or an experienced radio/TV technician for help.

The antenna used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other transmitter.

Labeling requirement

The Host device of OEM integrator must be labeled with „Contains FCC ID: 2AJY6MORIN01“

IC-WARNING

This device complies with Part 15 of the FCC Rules and with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes : (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Documentation

Under Industry Canada regulations, this radio transmitter may only operate using the following antenna(s):

Number	Characteristic	Certification Name	Gain
1	Omni	PCB-inverted F	2.1 dBi

To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (e.i.r.p.) is not more than that necessary for successful communication.

Labeling requirement

The Host device of OEM integrator must be labeled with „Contains IC: 22172-MORIN01“

Documentation

Setup Hardware

Setup via Web-Interface:

Connect your Morin-Board with a standard USB TypeA/B Cable to a standard USB power supply or your PC USB-port.

Use Ethernet switch jacks to connect your PC/Switch to the Morin Board.

Morin static IP-Address: 192.168.1.1 Mask: 255.255.255.0

Use Web Browser or telnet to connect.

Note: Use ssh instead when Morin Evaluation-Board admin password is configured. Telnet is switched to ssh by password configuration.

Optional: Setup via Serial-USB-Interface:

Connect the Morin Board with a standard USB TypeA/B Cable to the USB-port of your PC. Based on your specific operating system you are able to open a serial connection with a terminal application of your choice.

Configure serial settings: 115200-8-N-1

Note: If needed install driver:

USB COM DRIVER: <http://www.ftdichip.com/Drivers/VCP.htm>

Rescue Mode:

If you need to reset to the originally generated firmware, because Morin-Board doesn't boot anymore, rescue mode is the way to load Firmware via WEB Browser:

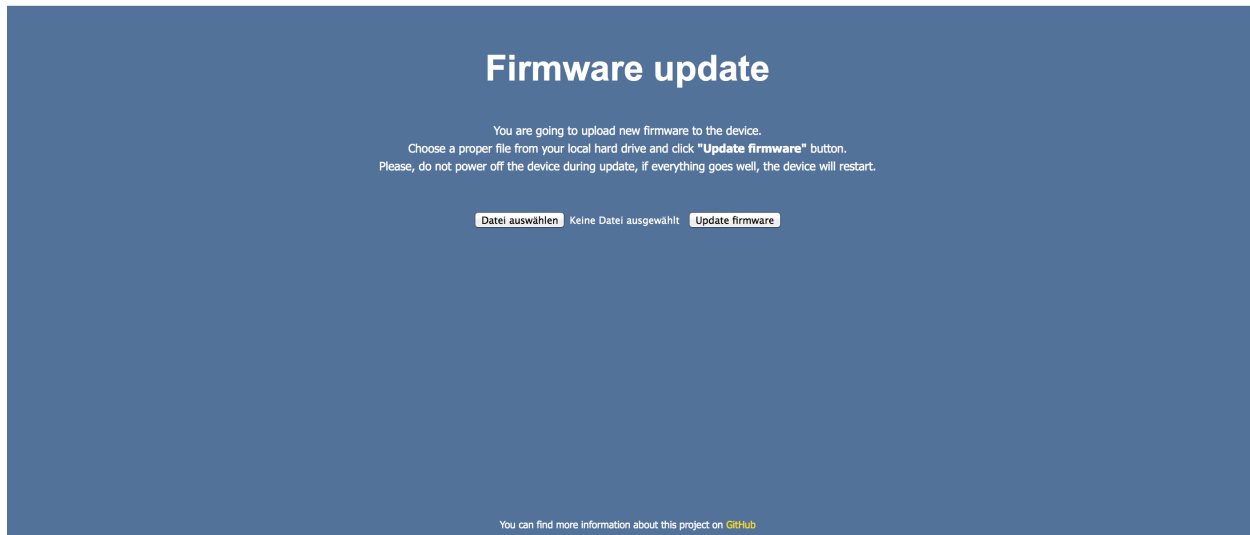
Power cycle the Morin-Board while Jumpstart button is pressed. Release Jumpstart button after 1s.

Put a firmware file via WEB Browser to the Morin Evaluation-Board. <http://192.168.1.1/index.html> Mask: 255.255.255.

- 1.) <http://192.168.1.1/index.html>. Maybe you have to use „private surf“. Depends on your browser vendor.

Documentation

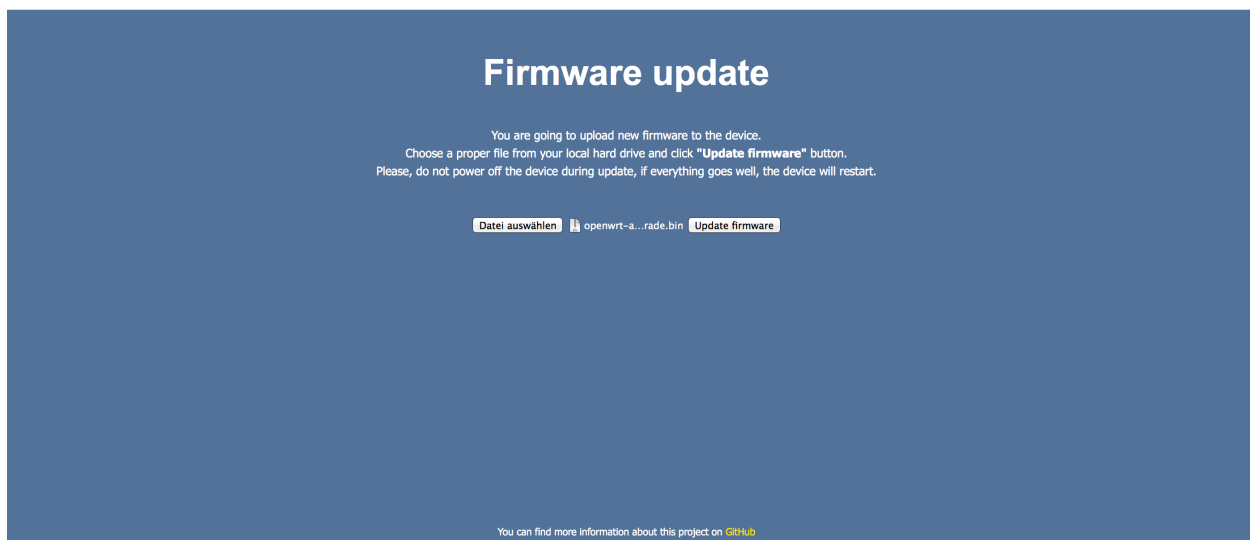
embedded wireless



2.) Press Button „Datei auswählen“.

3.) Point to your firmware (z.B.: openwrt-ar71xx-generic-...-16M-squashfs-sysupgrade.bin)

embedded wireless



4.) Press Button „Update firmware“.

Documentation

Update in progress

Your file was successfully uploaded! Update is in progress and you should wait for automatic reset of the device.
Update time depends on image size and may take up to a few minutes. You can close this page.



You can find more information about this project on [GitHub](#)

Documentation

Overview Software

Morin Module Software is OpenWrt Operating System. Most recent Version is V 15.01 Chaos Calmer.

Building Firmware is in the responsibility of the customer. Firmware is always customer specific and build by the customer of the product. Changes or modifications of the software not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Documentation

OpenWrt build system – Installation

OpenWrt build system is the buildsystem for the OpenWrt Linux distribution. OpenWrt build system works on GNU/Linux, BSD or MacOSX operating system. A case-sensitive filesystem is required. It is recommended that you use a GNU/Linux distribution (Debian), either a standalone installation or one running in a virtual environment (VMware or Qemu).

Ubuntu under Windows Subsystem for Linux is not an officially supported environment, but it appears to produce good builds. Cygwin(Windows) will not be supported because of the lack of case sensitivity in the file system.

Outdated information for old Buildroot versions, old GNU/Linux variants is archived at: buildroot.exigence.org

Prerequisites

To generate an installable OpenWrt firmware image file with a size of e.g. 8MB, you need:

- ca. 200 MB of hard disk space for OpenWrt build system
- ca. 300 MB of hard disk space for OpenWrt build system + OpenWrt Feeds
- ca. 2.1 GB of hard disk space for source packages downloaded during build from OpenWrt Feeds
- ca. 3-4 GB of available hard disk space to build (i.e. cross-compile) OpenWrt and generate the firmware file
- ca. 1-4 GB of RAM to build OpenWrt. (build x86's img need 4GB RAM)
-

Install procedure on GNU/Linux

- 
1. Do everything as non-root user!
 2. Issue all OpenWrt build system commands in the <buildsystem root> directory, e.g. ~/openwrt/trunk
 3. Do not build in a directory that has spaces in its full path
 4. Change ownership of the directory where you downloaded the OpenWrt to other than root user (sudo chown -R user:user /openwrt/)

- 1 Install git , to conveniently download the OpenWrt source code, and build tools to do the cross-compilation process: sudo apt-get update
- 2 sudo apt-get install git-core build-essential libssl-dev libncurses5-dev unzip gawk zlib1g-dev Some feeds might not available over git but only via subversion (short: svn) or mercurial. If you want

Documentation

to obtain their source-code, you need to install svn and mercurial as well:
`sudo apt-get install subversion mercurial`

- for information about the build tools see make and build-essential
 - for information about git see git(7)
 - for information about the subversion tool see svn and subversion documentation (multiple languages)
- 3 Download the OpenWrt bleeding edge(trunk Version) with git (see Downloading Sources for more options!):
`git clone https://github.com/openwrt/openwrt.git`
this creates a directory 'openwrt', which is the OpenWrt build system build-directory
the OpenWrt toolchain "OpenWrt build system" is included
 - 4 (optional) Download and install all available "feeds" (see OpenWrt Feeds for more options!):
`cd openwrt`
 - 5 `./scripts/feeds update -a`
 - 6 `./scripts/feeds install -a`
 - 7 Make OpenWrt build system check for missing packages on your build-system using ONE of the following commands:
`make menuconfig` (most likely you would like to use this)
 - 8 `make defconfig`
 - 9 `make prereq`
There you will need to select what you want to compile.
 - 10 Proceed with build (i.e. cross-compile the downloaded sources to binaries)
After the cross-compilation process the trunk-directory contained over 240000 files with a total size of above 3GiB!

Table of known prerequisites and their corresponding packages

Prerequisite	Debian	SUSE	OS X (via MacPorts)	Fedora
asciidoc	asciidoc	asciidoc	asciidoc	asciidoc
GNU Bash	bash	bash	bash	bash
GNU bc	bc	bc		bc
GNU Binutils	binutils	binutils	binutils	binutils
bzip2	bzip2	bzip2	bzip2	bzip2
fastjar	fastjar	fastjar	fastjar	libgcj
flex	flex	flex	flex	flex
git	git-core	git-core	git-core	git
GNU C++ Compiler	g	gcc-c	gcc	sys-devel/gcc
GNU C Compiler	gcc	gcc	?	gcc
getopt	util-linux	util-linux	util-linux	getopt
GNU awk	gawk	gawk	gawk	gawk

Documentation

gtk2.0-dev	libgtk2.0-dev	gtk2-devel	gtk2-devel	gtk2
intltool-update	intltool	intltool	intltool	intltool
jikes	jikespg	jikes	?	jikes
libz, libz-dev	zlib1g-dev	zlib-devel	zlib-devel	zlib
Mercurial / hg		mercurial		
make	make	make	make	gmake
mkisofs	genisoimage	genisoimage	?	?
ncurses	libncurses5-dev	ncurses-devel	ncurses-devel	ncurses
openssl/ssl.h	libssl-dev	libopenssl-devel	openssl-devel	openssl
patch	patch	patch	patch	patchutils
perl-ExtUtils-MakeMaker	perl-modules	perl-ExtUtils-MakeMaker	perl-ExtUtils-MakeMaker	p5-extutils-makemaker
python2.6-dev	python2.6-dev	python-devel	?	python26
rsync	rsync	rsync	rsync	rsync
ruby	ruby	ruby	?	ruby
sdcc	sdcc	sdcc	sdcc	sdcc
unzip	unzip	unzip	unzip	unzip
GNU Wget	wget	wget	wget	wget
xgettext	gettext	gettext-tools	gettext	gettext
xsltproc	xsltproc	libxslt-tools	?	libxslt
zlib, zlib-static	zlib1g-dev	zlib-devel	zlib-devel	zlib-devel

Examples of Package Installations

- **Debian 7 Wheezy:** `apt-get install libncurses5-dev zlib1g-dev gawk`
- **Debian 8 Jessie:** `sudo apt-get install build-essential libncurses5-dev gawk git subversion libssl-dev gettext unzip zlib1g-dev file python`
- **Fedora 24 - 64Bit :** `dnf install -y subversion binutils bzip2 gcc gcc-c++ gawk gettext git-core flex ncurses-devel ncurses-compat-libs zlib-devel zlib-static make patch unzip perl-ExtUtils-MakeMaker perl-Thread-Queue \`
- `glibc glibc-devel glibc-static quilt sed sdcc intltool sharutils`

Documentation

- `bison wget openssl-devel`
- **openSUSE 13.2**`zypper install asciidoc bash bc binutils bzip2 fastjar flex git-core gcc-c++ gcc util-linux gawk gtk2-devel intltool jikes zlib-devel mercurial make genisoimage ncurses-devel libopenssl-devel patch perl-ExtUtils-MakeMaker python-devel rsync ruby sdcc unzip wget gettext-tools libxslt-tools zlib-devel subversion`
- **Ubuntu 12.04 LTS**`sudo apt-get install build-essential subversion git-core libncurses5-dev zlib1g-dev gawk flex quilt libssl-dev xsltproc libxml-parser-perl mercurial bzip2 ecj cvs unzip`
- **Ubuntu 64bit**`sudo apt-get install build-essential subversion libncurses5-dev zlib1g-dev gawk gcc-multilib flex git-core gettext libssl-dev`

Downloading Sources

GIT

cloning the Git repository using one of the following commands. Note that the main source repos have all been moved to Github in 2016.

[trunk \(main development tree\)](#)

The development branch (trunk) contains everything from documentation to **experimental patches**.

[Main repository](#)

```
git clone git://github.com/openwrt/openwrt.git
```

Additional packages can be found in several feeds (Luci, packages, routing, management etc.). Let the Openwrt build system to clone the correct feeds into feeds/packages, feeds/luci etc.

[15.05 branch \(Chaos Calmer\)](#)

[Main repository](#)

```
git clone -b chaos_calmer git://github.com/openwrt/openwrt.git
```

Documentation

Additional packages can be found in several feeds (Luci, packages, routing, management etc.). Let the Openwrt build system to clone the correct feeds into feeds/packages, feeds/luci etc.

Luci feed

Note: The location of the Luci feed has not been corrected in feeds.conf.default after the move to Github, so you need to manually edit feeds.conf.default to pull Luci from Github:

```
src-git luci https://github.com/openwrt/luci.git;luci-0.11
```

Documentation

OpenWrt build system – Usage

Prerequisites

to generate an **installable** OpenWrt firmware image file with a size of e.g. 8MB:

- **Install OpenWrt build system and its prerequisites** on your OS.
- ca. 3-4 GB of available hard disk space
- environment variables:
 - SED should not be set. If it is, run ``unset SED`` before compiling.
 - GREP_OPTIONS should not have `-initial-tab` or other options affecting its output
- Add `<buildroot dir>/staging_dir/host/bin` and `<buildroot dir>/staging_dir/toolchain-<platform>-<gcc_ver>-<libc_ver>/bin` in front of your PATH variable in `~/.bashrc`. The staging directory is created shortly after starting the build and the toolchain directory is created when the toolchain build begins. The build spawns multiple shells, some of which expect the toolchain binaries to be present in the PATH.

Procedure



1. Do everything as *non-root* user
2. Issue all commands in the `<buildroot dir>` directory, e.g. `~/openwrt/trunk/`

- Update OpenWrt sources.
- Update and install package feeds.
- Configure the firmware image you want to obtain.
- Start the build. This will automatically compile toolchain, cross-compile sources, package packages, and finally generate an image ready to be flashed.
- Proceed to **Installing OpenWrt**

Stages 1 & 2 can be performed with a **Dockerfile**

Updating Sources with Git

 OpenWrt sources change frequently. It is recommended that you work with the latest sources.

Documentation

`git pull`

Updating Feeds

see [feeds](#)

 *Installing* in context of `./scripts/feeds` script means "making package available in `make menuconfig`" rather than really installing or compiling package.

11 Update feeds: `./scripts/feeds update -a`

12 Make downloaded package/packages available in `make menuconfig`:

- single package: `./scripts/feeds install <PACKAGENAME>`
- all packages: `./scripts/feeds install -a` This may take extra time and required if you to create own repository of the packages/

Image Configuration

Typical actions:

- run `make menuconfig` and set "Target System", "Subtarget", "Target Profile";
- run `make defconfig`;
- run `make menuconfig` and modify set of package;
- run `scripts/diffconfig.sh >mydiffconfig` (save your changes in the text file `mydiffconfig`);
- run `make V=s` (build OpenWrt with console logging, you will look where build failed.).

Make menuconfig

The **OpenWrt build system configuration interface** handles the selection of the target platform, packages to be compiled, packages to be included in the firmware file, some kernel options, etc.

Start the OpenWrt build system configuration interface by issuing the following command:

```
make menuconfig
```

This will update the dependencies of your existing configuration automatically, and you can now proceed to build your updated images.

You have three options: `y`, `m`, `n` which are represented as follows:

- pressing `y` sets the `<*>` built-in label This package will be compiled and included in

Documentation

the firmware image file.

- pressing **m** sets the **<M>** package label This package will be compiled, but **not** included in the firmware image file. (E.g. to be installed with **opkg** after **flashing** the firmware image file to the device.)
- pressing **n** sets the **< >** excluded label The source code will not be processed.

When you save your configuration, the file **<buildroot dir>/config** will be created according to your configuration.

Explanations

It has been the intention from the beginning, with the development of **menuconfig**, to create a simple, yet powerful, environment for the configuration of individual OpenWrt builds. **menuconfig** is more or less self-explanatory, and even the most specialized configuration requirements can be met by using it. Depending on the the particular target platform, package requirements and kernel module needs, the standard configuration process will include modifying:

- 1 Target system
- 2 Package selection
- 3 Build system settings
- 4 Kernel modules

Target system is selected from the extensive list of supported platforms, with the numerous target profiles – ranging from specific devices to generic profiles, all depending on the particular device at hand. Package selection has the option of either 'selecting all package', which might be un-practical in certain situation, or relying on the default set of packages will be adequate or make an individual selection. It is here needed to mention that some package combinations might break the build process, so it can take some experimentation before the expected result is reached. Added to this, the OpenWrt developers are themselves only maintaining a smaller set of packages – which includes all default packages – but, the feeds-script makes it very simple to handle a locally maintained set of packages and integrate them in the build-process. The final step before the process of compiling the intended image(s) is to exit **menuconfig** – this also includes the option to save a specific configuration or load an already existing, and pre-configured, version.

Exit the TUI, and choose to save your settings.

Kernel configuration (optional)

Note that **make kernel_menuconfig** modifies the Kernel configuration templates of

Documentation

the build tree and clearing the build_dir will not revert them:

While you won't typically need to do this, you can do it:

```
make kernel_menuconfig CONFIG_TARGET=subtarget
```

CONFIG_TARGET allows you to select which config you want to edit. possible options: target, subtarget, env.

: (git) (GIT) The changes can be reviewed with
`git diff target/linux/`
and reverted with (SVN)
`svn revert -R target/linux/`

Source Mirrors

The 'Build system settings' include some efficient options for changing package locations which makes it easy to handle a local package set:

- 1 Local mirror for source packages
- 2 Download folder

In the case of the first option, you simply enter a full URL to the HTTP or FTP server on which the package sources are hosted. Download folder would in the same way be the path to a local folder on the build system (or network). If you have a web/ftp-server hosting the tarballs, the OpenWrt build system will try this one before trying to download from the location(s) mentioned in the Makefiles. Similar if a local 'download folder', residing on the build system, has been specified.

The 'Kernel modules' option is required if you need specific (non-standard) drivers and so forth – this would typically be things like modules for USB or particular network interface drivers etc.

Configure using config diff file

Beside `make menuconfig` another way to configure is using a configuration diff file.

This file includes only the changes compared to the default configuration. A benefit is that this file can be version controlled with your OpenWRT based project. It's also less affected by OpenWRT updates, because it only contains the changes.

Creating diff file

This file is created using the `<buildroot dir>/scripts/diffconfig.sh` script.

```
./scripts/diffconfig.sh > diffconfig # write the changes to diffconfig
```

Documentation

Using diff file

These changes can form the basis of a config file (<buildroot dir>/<config>). By running `make defconfig` these changes will be expanded into a full config.

```
cp diffconfig .config # write changes to .config
make defconfig # expand to full config
```

These changes can also be added to the bottom of the config file (<buildroot dir>/<config>), by running `make defconfig` these changes will override the existing configuration.

```
cat diffconfig >> .config # append changes to bottom of .config
make defconfig # apply changes
```

Patches

OpenWrt build system integrates *quilt* for easy patch management:

→ [patches](#)

Custom files

In case you want to include some custom configuration files, the correct place to put them is:

- **<buildroot dir>/files/**

For example, let's say that you want an image with a custom `/etc/config/firewall` or a custom `etc/sysctl.conf`, then create this files as:

- **<buildroot dir>/files/etc/config/firewall**
- **<buildroot dir>/files/etc/sysctl.conf**

E.g. if your <buildroot dir> is `/openwrt/trunk` and you want some files to be copied into firmware image's `/etc/config` directory, the correct place to put them is `/openwrt/trunk/files/etc/config`.

Defconfig

select your target before issuing defconfig

```
make defconfig
```

will produce a general purpose configuration of the build system including a check of dependencies and prerequisites for the build environment etc

will check for dependencies. Install missing and run again.

Documentation

Building Images

Everything is now ready for building the image(s), which is done with one single command:

```
make  
or  
make world
```

This simple command will trigger a cascade of activity. As already stated, it will

- 1 compile the toolchain
- 2 then crosscompile the sources with this toolchain
- 3 create opkg-packages
- 4 generate a firmware image file ready to be **flashed**.

Make Tips

Building in the background

If you intend to use your system while building, you can have the build process use only idle I/O and CPU capacity like this (dualcore CPU):

```
ionice -c 3 nice -n19 make -j 2
```

Building single Packages

When developing or packaging software for OpenWrt, it is convenient to be able to build only the package in question (e.g. with package cups):

```
make package/cups/compile V=s
```

For a rebuild:

```
make package/cups/{clean,compile,install} V=s
```

Documentation

It doesn't matter what feed the package is located in, this same syntax works for any installed package.

Spotting build errors

If for some reason the build fails, the easiest way to spot the error is to do:

```
make V=s 2>&1 | tee build.log | grep -i error
```

The above saves a full verbose copy of the build output (with stdout piped to stderr) in /openwrt/trunk/build.log and only shows errors on the screen.

Another example:

```
ionice -c 3 nice -n 20 make -j 2 V=s CONFIG_DEBUG_SECTION_MISMATCH=y  
2>&1 | tee build.log | egrep -i '(warn|error)'
```

The above saves a full verbose copy of the build output (with stdout piped to stderr) in build.log and outputs only warnings and errors while building using only background resources on a dual core CPU.

Yet another way to focus on the problem without having to wade through tons of output from Make as described above is to check the corresponding log in 'logs' folder. IE: if the build fails at "make[3] -C package/kernel/mac80211 compile", then you can go to <buildroot>/logs/package/kernel/mac80211 and view the compile.txt found there.

Getting beep notification

Depending on your CPU, the process will take a while, or while longer. If you want an acoustic notification, you could use `echo -e '\a'`:

```
make V=s ; echo -e '\a'
```

Skipping failed packages

If you are building everything (not just packages enough to make a flashable image) and build stops on a package you don't care about you can skip failed packages by using `IGNORE_ERRORS=1`

```
IGNORE_ERRORS=1 make <make options>
```

Documentation

Locating Images

After a successful build, the freshly built image(s) can be found in the newly created `<buildroot_dir>/bin` directory. The compiled files are additionally classified by the target platform, so e.g. a firmware built for an ar71xx device will be located in `<buildroot_dir>/bin/ar71xx` directory.

E.g. if your `<buildroot_dir>` is `~/openwrt/trunk`, the binaries are in `~/openwrt/trunk/bin/ar71xx`.

Documentation

Cleaning Up

You might need to clean your *build environment* every now and then. The following make-targets are useful for that job:

Clean

```
make clean
```

deletes contents of the directories `/bin` and `/build_dir`. `make clean` does not remove the toolchain, it also avoids cleaning architectures/targets other than the one you have selected in your `.config`

Dirclean

```
make dirclean
```

deletes contents of the directories `/bin` and `/build_dir` and additionally `/staging_dir` and `/toolchain` (=the cross-compile tools) and `/logs`. 'Dirclean' is your basic "Full clean" operation.

Distclean

```
make distclean
```

nuke everything you have compiled or configured and also deletes all downloaded feeds contents and package sources.

CAUTION: In addition to all else, this will **erase your build configuration** (`<buildroot_dir>/config`), your toolchain and all other sources. Use with care! There are numerous other functionalities in the OpenWrt build system, but the above should have covered some of the fundamentals.

Clean small part

In more time, you may not want to clean so many objects, then you can use some of the commands below to do it.

Documentation

Clean linux objects.

```
make target/linux/clean
```

Clean package base-files objects.

```
make package/base-files/clean
```

Clean luci.

```
make package/luci/clean
```

Documentation

Examples

- <https://forum.openwrt.org/viewtopic.php?pid=129319#p129319>
- <https://forum.openwrt.org/viewtopic.php?id=28267>

Documentation

Troubleshooting

- Beware of unusual environment variables such as
 - `GREP_OPTIONS` which should not have `-initial-tab` or other options affecting its output
 - **SED should not be set. If it is, run ``unset SED`` before compiling.** (See [Ticket 10612](#).)
 - First get more information on the problem using the make option "`make V=sc`" or enable logging.
- ⚠ Read more about make options: [Buildroot Techref](#)
- ⚠ [Development FAQ](#)

Missing source code file, due to download problems

First check if the URL path in the make file contains a trailing slash, then try with it removed (helped several times). Otherwise try to download the source code manually and put it into "dl" directory.

Compilation errors

Try to update the main source and all the feeds (Warning! May result in other problems). Check for a related bug in ([TRAC](#)), use the filters to find it. Otherwise report the problem there, by mentioning the package, the target data (CPU, image, etc.) and the code revisions (main & package). Compiling with `make -j ...` sometimes gives random errors. Try compiling without `-j` first before reporting the problem.

WARNING: skipping <package> -- package not selected

Run `make menuconfig` and enable compilation for your package. It should be labeled with `<*>` or `<M>` to work correctly. Read [image.configuration](#) further up in this article.

*-factory.bin and *-sysupgrade.bin images for my device do not get generated

When you execute `make` to build an OpenWRT image for your device, both a `sysupgrade` and a `factory` image should be generated for every board that is linked to the device profile that you have selected via `make config` or `make menuconfig`. For example, when you select the [TP-Link WR710N profile](#), there should be a separate `sysupgrade` and a `factory` image generated for both the [WR710N-v1](#) and the [WR710N-](#)

Documentation

v2 (the main difference between the two devices is the size of the flash chip: the v1 comes with 8MiB of flash while the v2 only has 4MiB).

If running `make` does *not* yield images for one (or even all) of the boards linked to the device profile that you have selected, than you probably have selected/enabled too much stuff, resulting in an image that would be too big to be flashed onto your device. Please note that the OpenWRT buildroot will currently (september 2015) **not** display any warning or error message if an image cannot be created because it would be too big for its designated target board. Please keep in mind that in most cases, you will not be able to use all the flash memory in your device exclusively for your OpenWRT image, because there might be several separate **flash partitions** dedicated to things like the boot loader, the calibration data of the devices wifi card or the partition where your configuration data is stored.

Notes

- [OpenWrt Buildroot – Technical Reference](#)
- <http://downloads.openwrt.org/docs/buildroot-documentation.html>
- <https://dev.openwrt.org/browser/trunk/docs/working.tex>
- [Compiler Optimization Tweaks](#)

This Manual is created using documentation with several small changes from openwrt.org under CC license. Thanks to the openwrt team for their excellent work!

Full license is available here:

<https://creativecommons.org/licenses/by-nc-sa/3.0/legalcode>