WASP-OS

Nov 29, 2020

Contents:

CHAPTER 1

Watch Application System in Python

1.1 Introduction

Wasp-os is a firmware for smart watches that are based on the nRF52 family of microcontrollers, including hacker friendly watches such as the Pine64 PineTime. Wasp-os includes a digital clock, a stopwatch, a step counter and a heart rate monitor. All of these, together with access to the MicroPython REPL for interactive tweaking, development and testing.

Wasp-os includes a robust bootloader based on the Adafruit NRF52 Bootloader. It has been extended to make it robust for development on form-factor devices without a reset button, power switch, SWD debugger or UART. This allows us to confidently develop on sealed devices relying on Bluetooth Low Energy for over-the-air updates.

1.2 Documentation

Wasp-os is has extensive documentation which includes a detailed Application Writer's Guide to help you get started coding for wasp-os as quickly as possible.

1.3 Getting Started

Wasp-os can be installed without using any tools or disassembly onto the following devices:

- Pine64 PineTime
- Colmi P8
- Senbono K9

Use the Installation Guide to learn how to build and install wasp-os on these devices.

At the end of the install process your watch will show the time (03:00) together with a date and a battery meter. When the watch goes into power saving mode you can use the button to wake it again.

At this point you will also be able to use the Nordic UART Service to access the MicroPython REPL. You can use `tools/wasptool --console` to access the MicroPython REPL.

To set the time and restart the main application:

```
^C
watch.rtc.set_localtime((yyyy, mm, dd, HH, MM, SS))
wasp.system.run()
```

Or, if you have a suitable GNU/Linux workstation, just use:

```
./tools/wasptool --rtc
```

which can run these commands automatically.

As mentioned above there are many drivers and features still to be developed, see the *Roadmap* for current status.

1.4 Videos

Fig. 1: Open source heart rate monitoring for Pine64 PineTime

Fig. 2: An M2 pre-release running on Pine64 PineTime

Fig. 3: How to develop wasp-os python applications on a Pine64 PineTime

Fig. 4: Developing for Pine64 PineTime using wasp-os and MicroPython

1.5 Screenshots

(An older version of) the digital clock application running on a Pine64 PineTime:

Screenshots of the built in applications running on the wasp-os simulator (the "blank" screen is the torch application):


64 bpm


19:05
0:21 32
3 00:16.32
2 00:10.32
1 00:05.28

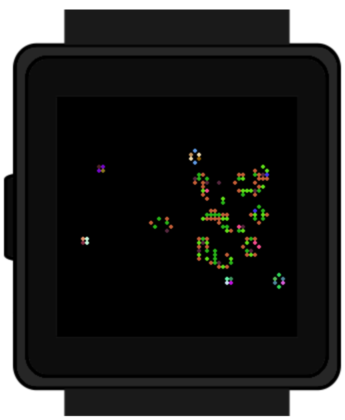

16:40
22


Self Test
Settings
Torch

wasp-os also contains a library of additional applications for you to choose. These are disabled by default but can be easily enabled by adding them using one of the techniques is the Application Writer's guide.


10 Oct 2020


Written in Python
Everything can be changed
You have the power




Dreams of Bamboo
Tasteless Brass Duck

CHAPTER 2

Installation Guide

2.1 Building wasp-os from source

Building wasp-os and launching the wasp-os simulator requires Python 3.6 (or later) and the following python modules: click, numpy, pexpect, PIL (or Pillow), pyserial, pysdl2.

On Debian Buster the required python modules can be obtained using the following commands:

```
sudo apt install \
  git build-essential libsdl2-2.0.0 python3-click python3-numpy \
  python3-pexpect python3-pil python3-pip python3-serial unzip
pip3 install --user cbor pysdl2
```

Additionally if you wish to regenerate the documentation you will require a complete sphinx toolchain:

```
sudo apt install sphinx graphviz python3-recommonmark
```

Alternatively, if your operating system does not package some or any of the above mentioned Python modules then you can install all of them with pip instead:

```
pip3 install --user cbor click numpy pexpect Pillow pyserial pysdl2
```

You will also need a toolchain for the Arm Cortex-M4. wasp-os is developed and tested using the GNU-RM toolchain (9-2019-q4) from Arm.

Note: There are known problems with toolchains older than gcc-7.3 when link time optimization is enabled during the MicroPython build (LTO is enabled by default).

Fetch the code from https://github.com/daniel-thompson/wasp-os and download the prerequisites:

```
git clone https://github.com/daniel-thompson/wasp-os
cd wasp-os
make submodules
make softdevice
```

To build the firmware select the command appropriate for your board from the list below:

```
make -j `nproc` BOARD=pinetime all
make -j `nproc` BOARD=k9 all
make -j `nproc` BOARD=p8 all
```

To rebuild the documentation:

```
make docs
```

2.2 Device Support

Wasp-os can run on multiple devices and, in time, will hopefully be ported to many more.

In terms of deciding which device to buy we can suggest two criteria to help.

The first is simply based on aesthetic appeal. A watch is something that you take everywhere and sits somewhere between clothing and jewellery. That means it is important to choose a device that feels good on the wrist and looks right when you glance at it. Aesthetics matter!

The second criteria is more subtle. In most cases, there is not really many important technical differences between the devices. They all use a Nordic chipset and have the same display controller running a 240x240 panel. So the second criteria is not technical, it is about community. The Pine64 PineTime is unique among the devices supported by wasp-os because it is intended that the watch be used to run a variety of different open source or free software operating systems. By manufacturing a watch with the intention that it be hacked every which way from Sunday then we get a bigger, stronger community focused on the PineTime. There is a vibrant support forum, multiple different OS developers (who share ideas and knowledge even when hacking on very different code bases) combined with a near complete set of hardware documentation.

There's definitely a lot of fun to be had buying something off-the-shelf and hacking it to become something the manufacturer never intended. We know this because we've done it! However there is also enormous benefit from participating in a community, especially if you enjoy working with or learning from other developers. Devices that can repurposed to run wasp-os are often only sold for short periods and may experience undocumented technical changes between manufacturing runs that can cause compatibility problems. This makes it hard for a large community to form around these devices.

Thus the second criteria it to think about your own needs and abilities. If you want to enjoy the social and community aspects of working together on open source watch development then you should look very closely at the PineTime.

2.2.1 Pine64 PineTime

Pine64 PineTime is a square smart watch based on an nRF52832 SoC and includes a 240x240 colour display with touch screen, a step counter and a heart rate sensor.

wasp-os can be installed directly from the factory default operating system using an over-the-air update with no tools or disassembly required. nRF Connect for Android can be used to install both the *wasp-bootloader* and the *main OS image*.

Note: The early adopter PineTime Developer Edition came pre-programmed with a proprietary test firmware rather than the current factory default OS. If you have an early adopter unit then it will appear in the device list as *Y7S*. In this case the process needed for an OTA update is different. Use DaFlasher for Android to install both the *wasp-bootloader* and the *main OS image*.

The developer edition comes without the case glued shut. This allows access to the Serial Wire Debug (SWD) pins which can make debugging easier. On developer edition devices it is also possible to install the wasp-bootloader using an *SWD programmer*.

2.2.2 The wasp-os simulator

The simulator allows you to run wasp-os programs using the Python interpreter included with your host operating system. The simulator provides a 240x240 colour display together with a touch screen and a physical button, all of which appears as a window on your host computer.

The simulator has large quantities of memory and, whilst useful for exploring wasp-os and testing your programs are syntactically correct, it is not a substitute for testing on real hardware. See *Testing on the simulator* for more details on how to use the simulator.

To launch the simulator try:

```
make sim
```

2.2.3 Colmi P8

The Colmi P8 is an almost square smart watch based on an nRF52832 SoC and includes a 240x240 colour display with touch screen, a step counter and a heart rate sensor.

> **Warning:** The P8 has multiple hardware revisions and the newest version (the one that includes a magnetic charger) uses a different and, currently, unsupported step counter module. The new models will boot wasp-os successfully but the step counter application will be disabled and cannot function.

DaFlasher for Android can be used to install both the *wasp-bootloader* and the *main OS image*. No tools or disassembly is required.

2.2.4 Senbono K9

The Senbono K9 is a circular smart watch based on an nRF52832 SoC and includes with a square 240x240 colour with a touch screen, a step counter and a heart rate sensor.

The wasp-os port for Senbono K9 does not, at this point, include a driver for the touch screen because the protocol has not yet been reverse engineered. The touch screen enumerates via I2C at address 70d (0x46) and the interrupt can be used to detect touch screen activity but the touch coordinates cannot be read from the hardware. Currently the touch screen can only act as a multi-function button and can be used to cycle through the quick ring and display notifications. This makes the device usable but not fully featured.

Note also that the to conceal the square display within the circular face this device has a heavily tinted filter over the display. This improves the look of the device but also significantly dims the backlight making it difficult to read the display in strong sunlight.

DaFlasher for Android can be used to install both the *wasp-bootloader* and the *main OS image*. No tools or disassembly is required.

2.3 Installing wasp-bootloader

2.3.1 nRF Connect for Android

For Pine64 PineTime devices running Infinitime then nRF Connect for Android can be used to install wasp-bootloader:

- Ensure the watch is fully charged before attempting to install the wasp-bootloader. Running out of power during this process can brick sealed devices.
- Copy `reloader-mcuboot.zip` (see *Building wasp-os from source*) to your Android device and download nRF Connect for Android if you do not already have it.
- Wake the device so that Infinitime is showing a watch face.
- Connect to the *Infinitime* device usnig nRF Connect, click the DFU button and send `reloader-mcuboot.zip` to the device.
- When the progress meter reaches 100% the nRF Connect will disconnect and the watch will reboot.
- The watch will boot the reloader application which draws a small blue pine cone in the centre of the screen. The pine cone acts a progress meter and will slowly become white. Once the update is complete the watch will show the wasp-os logo and an additional on-screen prompt.

Over-the-air update from Infinitime to wasp-os

Note: If you want to restore the PineTime factory firmware then you can use nRF Connect to do this. Perform a long press reset and then use nRF Connect to send `reloader-factory.zip` to the *PineDFU* device.

2.3.2 DaFlasher for Android

To install the bootloader using DaFlasher for Android:

- Ensure the watch is fully charged before attempting to install the wasp-bootloader. Running out of power during this process can brick sealed devices.
- Download and install DaFlasher and copy the DaFlasher bootloaders to your Android device. You will need DaFitBootloader23Hacked.bin and FitBootloaderDFU2.0.1.zip.
- Copy `bootloader-daflasher.zip` (see *Building wasp-os from source* above) to your Android device.
- Open the app and connect to the device (e.g. *Y7S* if you have a developer edition PineTime).
- Read the disclaimer carefully, then click **Ok**. PineTime).
- Click **Select file** and choose `DaFitBootloader23Hacked.bin`, then wait for the payload to be transferred and for the install process to complete on the watch (leaving three coloured squares on the display).
- Press the Back button to return to the scanner and connect to the device. The device name will have changed to *ATCdfu*.
- Click **Do DFU Update**.
- Click **Select DFU file** and select `FitBootloaderDFU2.0.1.zip`, then wait for the payload to transfer and the update to take place. The watch should be showing a single red square which is captioned *ATCnetz.de*.
- Click **Select DFU file** again and select `bootloader-daflasher.zip`. Once the update is complete the watch will show the wasp-os logo and some additional on-screen prompt.

It is important to ensure that both `bootloader-daflasher.zip` and `micropython.zip` match the device you are installing for. There are no runtime compatibility checks.

An end-to-end video of the above process (and the final install of wasp- os) is also available:

Installing MicroPython on a Colmi P8 smart watch using DaFlasher

Warning: The first step cannot be reversed. Once `DaFitBootloader23Hacked.bin` has been installed the factory firmware will be permanently removed from the device.

Although it is not possible to restore the factory firmware it is possible to switch back to Softdevice 5.0.1 and/or Softdevice 2.0.1 on order to run alternative firmwares such as ATCwatch. The zip updates in DaFlasherFiles cannot be applied directly but we can return to the DaFlasher bootloaders by installing DS-D6-adafruit-back-to-desay-sd132v201.zip followed by ATCdfuFromSD2toSD5.zip

2.3.3 Using an SWD programmer

There are many different SWD programmers that can be used to install wasp-bootloader. Use the PineTime SWD programming guide to lookup the specific instructions for your programmer.

Use the SWD programmer to install `bootloader.hex` to the device. This file is an Intel HEX file containing both the bootloader and the Nordic SoftDevice. Once the bootloader is installed the watch will boot, display a logo and wait for a OTA update.

Note: If you have a new device then it may have been delivered with flash protection enabled. You must disable the flash protection before trying to program it.

Be careful to disconnect cleanly from the debug software since just pulling out the SWD cable will mean the nRF52 will still believe it is being debugged (which harms battery life because the device won't properly enter deep sleep states).

2.4 Installing wasp-os

2.4.1 nRF Connect for Android

To install the main firmware using nRF Connect for Android:

- Copy `micropython.zip` (see *Building wasp-os from source*) to your Android device and download nRF Connect for Android if you do not already have it.

- Connect to the device (e.g. *PineDFU* if you have a PineTime) using nRFConnect, click the DFU button and send `micropython.zip` to the device.
- When the upload is complete the watch will reboot and launch the digital clock application.

2.4.2 DaFlasher for Android

To install the main firmware using DaFlasher for Android:

- Copy `micropython.zip` (see *Building wasp-os from source*) to your Android device and download DaFlasher if you do not already have it.
- Open the app and connect to the device (e.g. *PineDFU* if you have a PineTime).
- Click **Do DFU Update**.
- Click **Select DFU file** and select `micropython.zip`.
- When the upload is complete the watch will reboot and launch the digital clock application.

2.4.3 wasptool for GNU/Linux

To install the main firmware from a GNU/Linux workstation:

- Look up the MAC address for your watch (try: `sudo hcitool lescan`).
- Use ota-dfu to upload `micropython.zip` (see *Building wasp-os from source*) to the device. For example: `tools/ota-dfu/dfu.py -z micropython.zip -a A0:B1:C2:D3:E3:F5 --legacy`

2.5 Troubleshooting

There are three boot modes of the device: OTA update mode, safe mode and normal operation. Understanding these modes is useful to help troubleshoot installation and boot problems.

2.5.1 OTA update mode

Bootloader mode is entered automatically of the boot image is invalid or if the watchdog fires when running in another operating mode. OTA update mode can also be can also be entered manually by holding a physical button on the device for five seconds until the boot logo re-appears. When running in OTA update mode pressing the physical button will attempt to launch the application.

Note: To remain in OTA update mode it is import to release the button as soon as the boot logo appears otherwise you may acidentally request the bootloader restart the application!

When the bootloader starts it will display a boot logo for two seconds and will then either boot the application or enter OTA update mode. OTA update mode is easily recognised by the Bluetooth logo in the bottom right hand corner of the display.

When the device is in OTA update mode then it will enumerate with a name ending in `DFU` (Device Firmware Update). This device can be used to update the application image.

2.5.2 Safe mode

Safe mode is a special boot mode of the application that does not execute `main.py` automatically (and hence that the watch will not fully boot). This ensures the Python REPL is accessible for debugging. Safe mode also causes the watch to show it's boot activity on the screen which can be useful for fixing hardware problems.

Safe mode is entered if the physical button is held down when the boot logo disappears and the application first starts. The simplest way to enter safe mode is to hold down the physical button until `Init button` appear on the screen, then release it.

A device running in safe mode will display the message `Safe mode` on the display. To exit safe mode return to OTA update mode by holding down the physical button for five seconds and from there a short press of the button will return the device to Normal operation.

2.5.3 Normal operation

Underneath the covers normal operation is near identical to safe mode. There are only two differences:

- the boot messages will not appear unless a fault is detected (in which case `FAILED` will appear on the display)
- it will execute whatever it finds in `/flash/main.py`

A default version of `main.py` is installed automatically when wasp-os initially formats the external flash as a file system.

Most problems with normal mode operation occur either because `main.py` is missing, out-of-date or corrupt. These issues most commonly result in an entirely black screen when running the watch is running in normal mode.

Note: If the system reports FAILED at boot, in either safe mode or normal operation, then the best troubleshooting approach is to review the issue tracker. Initially look through the open issues and see if your problem is similar, if so there may be useful advice in the comments on the ticket. Otherwise if you cannot find anything similar then please raise a new issue.

2.5.4 main.py

By default main.py includes the following commands and, in normal operation, these will be executed to boot the watch:

```python
# SPDX-License-Identifier: LGPL-3.0-or-later
# Copyright (C) 2020 Daniel Thompson

import wasp
from gadgetbridge import *
wasp.system.schedule()
```

One of the most powerful troubleshooting techniques (and one that is usually effective in debugging "black screen" issues) is to switch to safe mode and run the contents of `main.py` by hand using a bluetooth console (typically either `wasptool --console` or an Android tool such as Serial Bluetooth Terminal). Either the watch will start running when started by hand or it will issue diagnostics via the console which can be captured and shared via the issue tracker.

If the watch can be successfully started by hand then it is likely the copy of `main.py` on your watch is broken, missing or out of date. You can explore the watch's filesystem using the shell module:

```python
from shell import *
cd('/flash')
ls
cat('main.py')
```

If your copy of `main.py` needs to be updated you can use wasptool to upload a new version:

```
tools/wasptool --upload wasp/main.py
```

Note: If you are not able to run wasptool on your system but have another means to access to the python REPL you can also use `shell.upload()` to manually upload a new version of main.py.

CHAPTER 3

Application Library

3.1 Built-in

The built-in application are summarised below but because these apps are treated as examples they are described in detail as part of the *Wasp-os Reference Manual*:

- `ClockApp`
- `FlashlightApp`
- `LauncherApp`
- `PagerApp`
- `TestApp`
- `TemplateApp``

3.2 Watch faces

3.2.1 Fibonacci clock

The Fibonacci sequence is a sequence of numbers created by the Italian mathematician Fibonacci in the 13th century. This is a sequence starting with 1 and 1, where each subsequent number is the sum of the previous two. For the clock I used the first 5 terms: 1, 1, 2, 3 and 5.

Fig. 1: Screenshot of the fibonacci clock application

The screen of the clock is made up of five squares whose side lengths match the first five Fibonacci numbers: 1, 1, 2, 3 and 5. The hours are displayed using red and the minutes using green. When a square is used to display both the hours and minutes it turns blue. White squares are ignored.

To tell time on the Fibonacci clock you need to do some calculations. To read the hour, simply add up the corresponding values of the red and blue squares. To read the minutes, do the same with the green and blue squares. The minutes are displayed in 5 minute increments (0 to 12) so you have to multiply your result by 5 to get the actual number.

This app is enabled by default in the simulator. The app is also frozen into the firmware image but it is disabled by default in order to keep RAM available for user developed applications. It can be enabled by modifying `main.py`.

3.3 Games

3.3.1 Conway's Game of Life

The Game of Life is a "no player game" played on a two dimensional grid where the rules interact to make interesting patterns.

The game is based on four simple rules:

1. Death by isolation: a cell dies if has fewer than two live neighbours.
2. Death by overcrowding: a cell dies if it has more than three live neighbours.
3. Survival: a living cell continues to survive if it has two or three neighbours.
4. Reproduction: a dead cell comes alive if it has exactly three neighbours.

Fig. 2: Screenshot of the Game of Life application

On 11 April 2020 John H. Conway who, among many, many other achievements, devised the rule set for his Game of Life, died of complications from a COVID-19 infection.

The Game of Life is the first "toy" program I ever recall seeing on a computer (running in a mid 1980s Apple Macintosh). It sparked something even if "toy" is perhaps an underwhelming description of the Game of Life. Either way it occupies a special place in my childhood. For that, this application is dedicated to Professor Conway.

This app is enabled by default in the simulator. The app is also frozen into the firmware image but it is disabled by default in order to keep RAM available for user developed applications. It can be enabled by modifying `main.py`.

3.4 Integration

3.4.1 Music Player for GadgetBridge

Fig. 3: Screenshot of the Music Player application

Music Player Controller:

- Touch: play/pause
- Swipe UPDOWN: Volume down/up

- Swipe LEFTRIGHT: next/previous

This app is enabled by default in the simulator. The app is also frozen into the firmware image but it is disabled by default in order to keep RAM available for user developed applications. It can be enabled by modifying `main.py`.

CHAPTER 4

Application Writer's Guide

4.1 Introduction

Wasp-os, the Watch Application System in Python, has one pervasive goal that influences almost everything about it, from its name to its development roadmap: make writing applications easy (and fun).

Applications that can be loaded, changed, adapted and remixed by the user are what **really** distinguishes a smart watch from a "feature watch"[1]. In other words if we want a watch built around a tiny microcontroller to be "smart" then it

[1] The fixed function mobile phones that existed before iOS and Android took over the industry were retrospectively renamed "feature phones"

has to be all about the applications.

This guide will help you get started writing applications for wasp-os. Have fun!

4.1.1 Hello World for wasp-os

Let's start by examining a simple "Hello, World!" application for wasp-os.

```python
# SPDX-License-Identifier: MY-LICENSE
# Copyright (C) YEAR(S), AUTHOR

import wasp

class HelloApp():
    """A hello world application for wasp-os."""
    NAME = "Hello"

    def __init__(self, msg="Hello, world!"):
        self.msg = msg

    def foreground(self):
        self._draw()

    def _draw(self):
        draw = wasp.watch.drawable
        draw.fill()
        draw.string(self.msg, 0, 108, width=240)
```

Some of the key points of interest in this example application are:

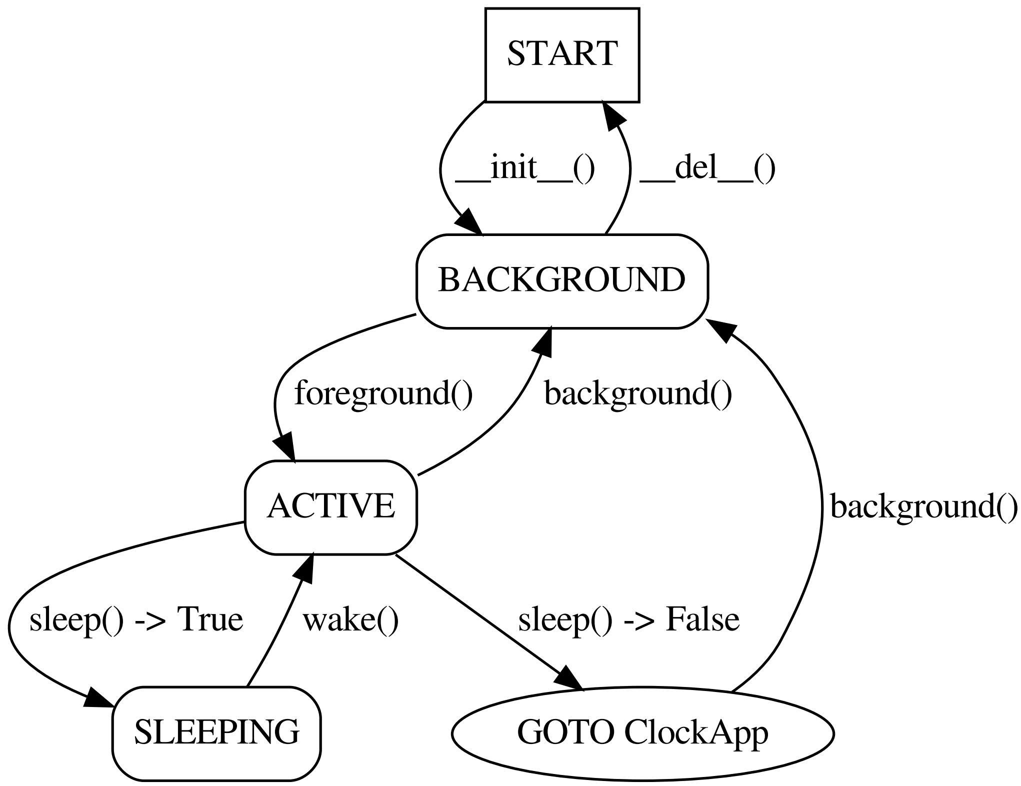
1. Applications have a `NAME`, which is shown in the launcher. Most applications also provide an `ICON` but a default will be displayed if this is omitted.
2. This example uses *`__init__()`* to initialize the state of the application, these variables are used to remember the state of the application when when it is deactivated.
3. *`foreground()`* is the only mandatory application entry point and it is responsible for redrawing the screen. This application does not implement *`background()`* because there is nothing for us to do!
4. The use of *`_draw()`* is optional. We could just do the work in *`foreground()`* but this application follows a common wasp-os idiom that is normally used to pattern to distinguish a full refresh of the screen from an fast update (a redraw).

4.2 Application life-cycle

Applications in wasp-os are triggered by and do all their processing from calls their entry points. The entry points can be coarsely categorized event notifications, timer callbacks (the application tick) and system actions.

System actions control the application life-cycle and that lifecycle is shown below. The system actions are used to tell the application about any change in its lifecycle.

to distinguish them from newer devices. Many of them were superficially similar to early Android devices but is was the application ecosystem that really made smart phones into what they are today.



When an application is initialized is enters the `BACKGROUND` state. A backgrounded application will not execute but it should nevertheless maintain its user visible state whilst deactivated. To conserve memory wasp-os does not permit two applications to run simultaneously but because each application remembers its state when it is not running then it will appear to the user as though all applications are running all the time.

For example, a stopwatch application should record the time that it was started and remember that start time, regardless of whether it is running or not so that when it restarted is can continue to run as the user expects.

A backgrounded application enters the `ACTIVE` state via a call to *`foreground()`*. When it is active the application owns the screen and must draw and maintain its user interface.

If the system manager wants to put the watch to sleep then it will tell the active application to *`sleep()`*. If the application returns True then the application will remain active whilst the watch is asleep. It will receive no events nor the application tick whilst the system is asleep and, instead, must wait for a *`wake()`* notification telling the application that the device is waking up and that it may update the screen if needed.

If an application does not support sleeping then it can simply not implement *`sleep()`* or *`wake()`*. In this case the system manager will automatically return to the default application, typically the main clock face.

Some applications may support sleeping only under certain circumstances. For example a stopwatch may choose to remain active when the watch sleeps only if the stopwatch is running. This type of application must implement *`sleep()`* and return False when it does not want to remain active when the system resumes.

Note: Most applications should not implement *`sleep()`* since it is often a better user experience for the watch to return to the default application when they complete an interaction.

4.3 API primer

This API primer introduces some of the most important and frequently used interfaces in wasp-os. For more comprehensive coverage see the *Wasp-os Reference Manual* which contains extensive API documentation covering the entire of wasp-os, including its drivers.

4.3.1 System management

The system management API provides a number of low-level calls that can register new applications and navigate between them. However most applications do not need to make these low level calls and will use a much smaller set of methods.

Applictions must call a couple of functions from their `foreground()` in order to register for event notifications and timer callbacks:

- `request_event()` - register for UI events such as button presses and touch screen activity.
- `request_tick()` - register to receive an application tick and specify the tick frequency.

Additionally if your application is a game or a similar program that should not allow the watch to go to sleep when it is running then it should arrange to call `keep_awake()` from the application's `tick()` method.

4.3.2 Drawing

Most applications using the drawing toolbox, `wasp.watch.drawable`, in order to handle the display. Applications are permitted to directly access `wasp.watch.display` if they require direct pixel access or want to exploit specific features of the display hardware (inverse video, partial display, etc) but for most applications the drawing toolbox provides convenient and optimized drawing functions.

- `blit()` - blit an image to the display at specified (x, y) coordinates, image type is detected automatically
- `fill()` - fill a rectangle, without arguments the default is a black rectangle covering the entire screen which is useful to clear the screen prior to an update
- `string()` - render a string, optionally centring it automatically
- `wrap()` - automatically determine where to break a string so it can be rendered to a specified width

Most applications run some variant of the following code from their `foreground()` or `_draw()` methods in order to clear the display ready for a redraw.

```
draw = wasp.watch.drawable
draw.fill()
# now use draw to render the rest of the screen
```

Some applications customize the above code slightly if they require a custom background colour and it may even be omitted entirely if the application explicitly draws every pixel on the display.

Finally, wasp-os provides a small number of widgets that allow common fragments of logic and redrawing code to be shared between applications:

- `BatteryMeter`
- `ScrollingIndicator`

4.3.3 MicroPython

Many of the features of wasp-os are inherited directly from MicroPython. It is useful to have a basic understanding of MicroPython and, in particular, put a little time into learning the best practices when running MicroPython on microcontrollers.

4.4 How to run your application

4.4.1 Testing on the simulator

wasp-os includes a simulator that can be used to test applications before downloading them to the device. The simulator is useful for ensuring the code is syntactically correct and that there are not major runtime problems such as misspelt symbol names.

Note: The simulator does not model the RAM or code size limits of the real device. It may still be necessary to tune the application for minimal footprint after testing on the simulator.

To launch the simulator:

```
sh$ make sim
PYTHONDONTWRITEBYTECODE=1 PYTHONPATH=.:wasp/boards/simulator:wasp \\
python3 -i wasp/main.py
MOTOR: set on
BACKLIGHT: 2
Watch is running, use Ctrl-C to stop
```

From the simulator console we can register the application with the following commands:

```
1 ^C
2 Traceback (most recent call last):
3 ...
4 KeyboardInterrupt
5 >>> from myapp import MyApp
6 >>> wasp.system.register(MyApp())
7 >>> wasp.system.run()
8 Watch is running, use Ctrl-C to stop
```

When an application is registered it does not start automatically but it will have been added to the launcher and you will be able to select in the simulator by swiping or using the Arrow keys to bring up the launcher and then clicking on your application.

The application can also be registered automatically when you load the simulator if you add it to `wasp/main.py`. Try adding lines 5 and 6 from the above example into this file (between `import wasp` and `wasp.system.run()`).

The simulator accepts gestures such as up/down and left/right swipes but the simulator also accepts keystrokes for convenience. The arrow keys simulate swipes and the Tab key simulates the physical button, whilst the ‘s’ key can be used to capture screen shots to add to the documentation for your application.

4.4.2 Testing on the device

When an application is under development it is best to temporarily load your application without permanently stored on the device. Providing there is enough available RAM then this can lead to a very quick edit-test cycles.

Try:

```
sh$ tools/wasptool \\
        --exec myapp.py \\
        --eval "wasp.system.register(MyApp())"
Preparing to run myapp.py:
[##################################################] 100%
```

Like the simulator, when an application is registered it is added to the launcher and it does not start automatically.

Note: If the progress bar jams at the same point each time then it is likely your application is too large to be compiled on the target. You may have to adopt the frozen module approach from the next section.

To remove the application simply reboot the watch by pressing and holding the physical button until the watch enters OTA mode (this takes around five seconds). Once the watch is in OTA mode then press the phyiscal button again to return to normal mode with the application cleared out.

4.4.3 Making it permanent

To ensure you application survives a reboot then we must copy it to the device and ensure it gets launched during system startup.

Note: Applications stored in external FLASH have a greater RAM overhead than applications that are frozen into the wasp-os binary. If you app does not fix then see next section for additional details on how to embed your app in the wasp-os binary itself..

To copy your application to the external FLASH try:

```
sh$ ./tools/wasptool --upload myapp.py
Uploading myapp.py:
[##################################################] 100%
```

At this point your application is stored on the external FLASH but it will not automatically be loaded. This requires you to update the `main.py` file stored in the external FLASH. When wasp-os runs for the first time it automatically generates this file (using `wasp/main.py` as a template) and copies it to the external FLASH. After this point `main.py` is user modifiable and can be used to tweak the configuration of the watch before it starts running.

Edit `wasp/main.py` to add the following two lines between `import wasp` and the `wasp.system.run()`:

```
from myapp import MyApp
wasp.system.register(MyApp())
```

Having done that we can use `wasptool` to upload the modified file to the watch:

```
sh$ ./tools/wasptool --upload wasp/main.py
Uploading wasp/main.py:
[##################################################] 100%
```

Note: If the new code on the watch throws an exception (including an out-of-memory exception) then your watch will display a black screen at startup. If that happens, and you don't know how to debug the problem, then you can use wasptool to restore the original version of main.py .

4.4.4 Freezing your application into the wasp-os binary

Freezing your application causes it to consume dramatically less RAM. That is because they can execute directly from the internal FLASH rather than running from RAM. Additionally the code is pre-compiled, which also means we don't need any RAM budget to run the compiler.

Freezing your application requires you to modify the `manifest.py` file for your board (e.g. `wasp/boards/pinetime/manifest.py`) to include your application and then the whole binary must be re-compiled as normal.

After that you an use the same technique described in the previous section to add an import and register for you application from `main.py`

Note: The micropython import path "prefers" frozen modules to those found in the external filesystem. If your application is both frozen and copied to external FLASH then the frozen version will be loaded.

In many cases it is possible to avoid rebuilding the binary in order to test new features by directly parsing the code in the global namespace (e.g. using `wasptool --exec` rather than `wasptool --upload` combined with `import`). With the code in the global namespace it can then be patched into the system. For example the following can be used to adopt a new version of the CST816S driver:

4.5 Application entry points

Applications provide entry points for the system manager to use to notify the application of a change in system state or an user interface event.

The complete set of wasp-os application entry points are documented below as part of a template application. Note that the template does not rely on any specific parent class. This is because applications in wasp-os can rely on *duck typing* making a class hierarchy pointless.

class `apps.template.`**`TemplateApp`**

Template application.

The template application includes every application entry point. It is used as a reference guide and can also be used as a template for creating new applications.

`NAME = 'Template'`

Applications must provide a short `NAME` that is used by the launcher to describe the application. Names that are longer than 8 characters are likely to be abridged by the launcher in order to fit on the screen.

`ICON = RLE2DATA`

Applications can optionally provide an icon for display by the launcher. Applications that expect to be installed on the quick ring will not be listed by the launcher and need not provide any icon. When no icon is provided the system will use a default icon.

The icon is an opportunity to differentiate your application from others so supplying an icon is strongly recommended. The icon, when provided, must not be larger than 96x64.

`__init__``()`

Initialize the application.

`__weakref__`

list of weak references to the object (if defined)

`_draw``()`

Draw the display from scratch.

_update()
Update the dynamic parts of the application display.

background()
De-activate the application.

foreground()
Activate the application.

press(*button*, *state*)
Notify the application of a button-press event.

sleep()
Notify the application the device is about to sleep.

swipe(*event*)
Notify the application of a touchscreen swipe event.

tick(*ticks*)
Notify the application that its periodic tick is due.

touch(*event*)
Notify the application of a touchscreen touch event.

wake()
Notify the application the device is waking up.

CHAPTER 5

Wasp-os Reference Manual

5.1 System

5.1.1 Wasp-os system manager

wasp.**system**

wasp.system is the system-wide singleton instance of `Manager`. Application must use this instance to access the system services provided by the manager.

wasp.**watch**

wasp.watch is an import of `watch` and is simply provided as a shortcut (and to reduce memory by keeping it out of other namespaces).

class wasp.**EventMask**

Enumerated event masks.

class wasp.**EventType**

Enumerated interface actions.

MicroPython does not implement the enum module so EventType is simply a regular object which acts as a namespace.

class wasp.**Manager**

Wasp-os system manager

The manager is responsible for handling top-level UI events and dispatching them to the foreground application. It also provides services to the application.

The manager is expected to have a single system-wide instance which can be accessed via `wasp.system` .

brightness

Cached copy of the brightness current written to the hardware.

cancel_alarm(*time*, *action*)

Unqueue an alarm.

keep_awake()

Reset the keep awake timer.

navigate(*direction=None*)

Navigate to a new application.

Left/right navigation is used to switch between applications in the quick application ring. Applications on the ring are not permitted to subscribe to :py:data‘EventMask.SWIPE_LEFTRIGHT‘ events.

Swipe up is used to bring up the launcher. Clock applications are not permitted to subscribe to :py:data‘EventMask.SWIPE_UPDOWN‘ events since they should expect to be the default application (and is important that we can trigger the launcher from the default application).

Parameters **direction** (`int`) – The direction of the navigation

register(*app*, *quick_ring=False*)

Register an application with the system.

Parameters **app** (`object`) – The application to regsister

request_event (*event_mask*)
Subscribe to events.

Parameters event_mask (*int*) – The set of events to subscribe to.

request_tick (*period_ms=None*)
Request (and subscribe to) a periodic tick event.

Note: With the current simplistic timer implementation sub-second tick intervals are not possible.

run (*no_except=True*)
Run the system manager synchronously.

This allows all watch management activities to handle in the normal execution context meaning any exceptions and other problems can be observed interactively via the console.

schedule (*enable=True*)
Run the system manager synchronously.

set_alarm (*time*, *action*)
Queue an alarm.

Parameters

- **time** (*int*) – Time to trigger the alarm (use time.mktime)
- **action** (*function*) – Action to perform when the alarm expires.

sleep ()
Enter the deepest sleep state possible.

switch (*app*)
Switch to the requested application.

wake ()
Return to a running state.

class wasp.**PinHandler** (*pin*)
Pin (and Signal) event generator.

TODO: Currently this driver doesn't actually implement any debounce but it will!

get_event ()
Receive a pin change event.

Check for a pending pin change event and, if an event is pending, return it.

Returns boolean of the pin state if an event is received, None otherwise.

5.1.2 Watch driver instances

watch.**backlight**
Backlight driver, typically a board specific driver with a single `set()` method.

watch.**battery**
Battery driver, typically the generic metering driver, *drivers.battery.Battery*.

watch.**button**
An instance of machine.Pin (or a signal) that an application can use to poll the state of the hardware button.

watch.**display**
Display driver, typically *drivers.st7789.ST7789_SPI*.

watch.**drawable**
Drawing library for *watch.display*. It will be adapted to match the bit depth of the display, *draw565.Draw565* for example.

watch.**rtc**
RTC driver, typically *drivers.nrf_rtc.RTC*.

watch.**touch**
Touchscreen driver, for example *drivers.cst816s.CST816S*.

watch.**vibrator**
Vibration motor driver, typically *drivers.vibrator.Vibrator*.

5.1.3 RGB565 drawing library

class draw565.**Draw565**(*display*)
Drawing library for RGB565 displays.

A full framebufer is not required although the library will 'borrow' a line buffer from the underlying display driver.

__init__(*display*)
Initialise the library.

Defaults to white-on-black for monochrome drawing operations and 24pt Sans Serif text.

blit(*image*, *x*, *y*, *fg=65535*, *c1=19049*, *c2=31727*)
Decode and draw an encoded image.

Parameters

- **image** – Image data in either 1-bit RLE or 2-bit RLE formats. The format will be autodetected
- **x** – X coordinate for the left-most pixels in the image
- **y** – Y coordinate for the top-most pixels in the image

fill(*bg=None*, *x=0*, *y=0*, *w=None*, *h=None*)
Draw a solid colour rectangle.

If no arguments a provided the whole display will be filled with the background colour (typically black).

Parameters

- **bg** – Background colour (in RGB565 format)
- **x** – X coordinate of the left-most pixels of the rectangle
- **y** – Y coordinate of the top-most pixels of the rectangle
- **w** – Width of the rectangle, defaults to None (which means select the right-most pixel of the display)
- **h** – Height of the rectangle, defaults to None (which means select the bottom-most pixel of the display)

reset()
Restore the default colours and font.

Default colours are white-on-block (white foreground, black background) and the default font is 24pt Sans Serif.

rleblit (*image*, *pos=(0, 0)*, *fg=65535*, *bg=0*)
Decode and draw a 1-bit RLE image.

Deprecated since version M2: Use `blit()` instead.

set_color (*color*, *bg=0*)
Set the foreground and background colours.

The supplied colour will be used for all monochrome drawing operations. If no background colour is provided then the background will be set to black.

Parameters

- **color** – Foreground colour
- **bg** – Background colour, defaults to black

set_font (*font*)
Set the font used for rendering text.

Parameters **font** – A font module generated using `font_to_py.py`.

string (*s*, *x*, *y*, *width=None*)
Draw a string at the supplied position.

Parameters

- **s** – String to render
- **x** – X coordinate for the left-most pixels in the image
- **y** – Y coordinate for the top-most pixels in the image
- **width** – If no width is provided then the text will be left justified, otherwise the text will be centred within the provided width and, importantly, the remaining width will be filled with the background colour (to ensure that if we update one string with a narrower one there is no need to "undraw" it)

wrap (*s*, *width*)
Chunk a string so it can rendered within a specified width.

Example:

```
draw = wasp.watch.drawable
chunks = draw.wrap(long_string, 240)

# line(1) will provide the first line
# line(len(chunks)-1) will provide the last line
def line(n):
    return long_string[chunks[n-1]:chunks[n]]
```

Parameters

- **s** – String to be chunked
- **width** – Width to wrap the text into

Returns List of chunk boundaries

5.1.4 Widget library

The widget library allows common fragments of logic and drawing code to be shared between applications.

class `widgets.`**`BatteryMeter`**

Battery meter widget.

A simple battery meter with a charging indicator, will draw at the top-right of the display.

`draw``()`

Draw from meter (from scratch).

`update``()`

Update the meter.

The update is lazy and won't redraw unless the level has changed.

class `widgets.`**`Clock`**`(`*enabled=True*`)`

Small clock widget.

`draw``()`

Redraw the clock from scratch.

The container is required to clear the canvas prior to the redraw and the clock is only drawn if it is enabled.

`update``()`

Update the clock widget if needed.

This is a lazy update that only redraws if the time has changes since the last call *and* the clock is enabled.

Returns An time tuple if the time has changed since the last call, None otherwise.

class `widgets.`**`NotificationBar`**`(`*x=2*, *y=8*`)`

Show BT status and if there are pending notifications.

`draw``()`

Redraw the notification widget.

For this simple widget `draw()` is simply a synonym for `update()` because we unconditionally update from scratch.

`update``()`

Update the widget.

This widget does not implement lazy redraw internally since this can often be implemented (with less state) by the container.

class `widgets.`**`ScrollIndicator`**`(`*x=222*, *y=216*`)`

Scrolling indicator.

A pair of arrows that prompted the user to swipe up/down to access additional pages of information.

`draw``()`

Draw from scrolling indicator.

For this simple widget `draw()` is simply a synonym for `update()`.

`update``()`

Update from scrolling indicator.

class `widgets.`**`Slider`**`(`*steps*, *x=10*, *y=90*, *color=14847*`)`

A slider to select values.

`draw``()`

Draw the slider.

class `widgets.`**`StatusBar`**

Combo widget to handle notification, time and battery level.

clock
True if the clock should be included in the status bar, False otherwise.

draw()
Redraw the status bar from scratch.

update()
Lazily update the status bar.

Returns An time tuple if the time has changed since the last call, None otherwise.

5.2 Device drivers

5.2.1 Generic lithium ion battery driver

class drivers.battery.**Battery**(*battery*, *charging*, *power=None*)
Generic lithium ion battery driver.

__init__(*battery*, *charging*, *power=None*)
Specify the pins used to provide battery status.

Parameters

- **battery** (*Pin*) – The ADC-capable pin that can be used to measure battery voltage.
- **charging** (*Pin*) – A pin (or Signal) that reports the charger status.
- **power** (*Pin*) – A pin (or Signal) that reports whether the device has external power, defaults to None (which means use the charging pin for power reporting too).

charging()
Get the charging state of the battery.

Returns True if the battery is charging, False otherwise.

level()
Estimate the battery level.

The current the estimation approach is extremely simple. It is assumes the discharge from 4v to 3.5v is roughly linear and 4v is 100% and that 3.5v is 5%. Below 3.5v the voltage will start to drop pretty sharply to we will drop from 5% to 0% pretty fast… but we'll live with that for now.

Returns Estimate battery level in percent.

power()
Check whether the device has external power.

Returns True if the device has an external power source, False otherwise.

voltage_mv()
Read the battery voltage.

Assumes a 50/50 voltage divider and a 3.3v power supply

Returns Battery voltage, in millivolts.

5.2.2 Hynitron CST816S touch contoller driver

class drivers.cst816s.**CST816S**(*bus*, *intr*, *rst*, *schedule=None*)
Hynitron CST816S I2C touch controller driver.

__init__(*bus*, *intr*, *rst*, *schedule=None*)
Specify the bus used by the touch controller.

Parameters **bus** (`machine.I2C`) – I2C bus for the CST816S.

get_event()
Receive a touch event.

Check for a pending touch event and, if an event is pending, prepare it ready to go in the event queue.

Returns An event record if an event is received, None otherwise.

get_touch_data(*pin_obj*)
Receive a touch event by interrupt.

Check for a pending touch event and, if an event is pending, prepare it ready to go in the event queue.

reset_touch_data()
Reset touch data.

Reset touch data, call this function after processing an event.

sleep()
Put touch controller chip on sleep mode to save power.

wake()
Wake up touch controller chip.

Just reset the chip in order to wake it up

5.2.3 nRF-family RTC driver

class drivers.nrf_rtc.**RTC**(*counter*)
Real Time Clock based on the nRF-family low power counter.

__init__(*counter*)
Wrap an RTCounter to provide a fully fledged Real Time Clock.

If the PNVRAM is valid then we use it to initialize the RTC otherwise we just make something up.

Parameters **counter** (`RTCounter`) – The RTCCounter channel to adopt.

get_localtime()
Get the current time and date.

Returns Wall time formatted as (yyyy, mm, dd, HH, MM, SS, wday, yday)

get_time()
Get the current time.

Returns Wall time formatted as (HH, MM, SS)

get_uptime_ms()
Return the current uptime in milliseconds.

set_localtime(*t*)
Set the current wall time.

Parameters **t** (`sequence`) – Wall time formatted as (yyyy, mm, dd, HH, MM, SS), any additional elements in sequence will be ignored.

time()
Get time in the same format as time.time

update()
Check for counter updates.

Returns True of the wall time has changed, False otherwise.

uptime
Provide the current uptime in seconds.

5.2.4 Inverting pin wrapper

class `drivers.signal.`**Signal**(*pin*, *invert=False*)
Simplified Signal class

Note: The normal C implementation of the Signal class used by MicroPython doesn't work on the nRF family. This class provides a temporary workaround until that can be addressed.

__init__(*pin*, *invert=False*)
Create a Signal object by wrapping a pin.

off()
Deactivate the signal.

on()
Activate the signal.

value(*v=None*)
Get or set the state of the signal.

Parameters **v** – Value to set, defaults to None (which means get the signal state instead.

Returns The state of the signal if v is None, otherwise None.

5.2.5 Sitronix ST7789 display driver

Note: Although the ST7789 supports a variety of communication protocols currently this driver only has support for SPI interfaces. However it is structured such that other serial protocols can easily be added.

class `drivers.st7789.`**ST7789**(*width*, *height*)
Sitronix ST7789 display driver

__init__(*width*, *height*)
Configure the size of the display.

Parameters

- **width** (*int*) – Display width, in pixels
- **height** (*int*) – Display height in pixels

fill(*bg*, *x=0*, *y=0*, *w=None*, *h=None*)
Draw a solid colour rectangle.

If no arguments a provided the whole display will be filled with the background colour (typically black).

Parameters

- **bg** – Background colour (in RGB565 format)

- **x** – X coordinate of the left-most pixels of the rectangle
- **y** – Y coordinate of the top-most pixels of the rectangle
- **w** – Width of the rectangle, defaults to None (which means select the right-most pixel of the display)
- **h** – Height of the rectangle, defaults to None (which means select the bottom-most pixel of the display)

init_display()
Reset and initialize the display.

invert(*invert*)
Invert the display.

Parameters invert (`bool`) – True to invert the display, False for normal mode.

mute(*mute*)
Mute the display.

When muted the display will be entirely black.

Parameters mute (`bool`) – True to mute the display, False for normal mode.

poweroff()
Put the display into sleep mode.

poweron()
Wake the display and leave sleep mode.

rawblit(*buf*, *x*, *y*, *width*, *height*)
Blit raw pixels to the display.

Parameters

- **buf** – Pixel buffer
- **x** – X coordinate of the left-most pixels of the rectangle
- **y** – Y coordinate of the top-most pixels of the rectangle
- **w** – Width of the rectangle, defaults to None (which means select the right-most pixel of the display)
- **h** – Height of the rectangle, defaults to None (which means select the bottom-most pixel of the display)

set_window(*x=0*, *y=0*, *width=None*, *height=None*)
Set the clipping rectangle.

All writes to the display will be wrapped at the edges of the rectangle.

Parameters

- **x** – X coordinate of the left-most pixels of the rectangle
- **y** – Y coordinate of the top-most pixels of the rectangle
- **w** – Width of the rectangle, defaults to None (which means select the right-most pixel of the display)
- **h** – Height of the rectangle, defaults to None (which means select the bottom-most pixel of the display)

class drivers.st7789.**ST7789_SPI**(*width*, *height*, *spi*, *cs*, *dc*, *res=None*, *rate=8000000*)

quick_write(*buf*)
Send data to the display as part of an optimized write sequence.

Parameters **buf** (*bytearray*) – Data, must be in a form that can be directly consumed by the SPI bus.

quick_end()
Complete an optimized write sequence.

quick_start()
Prepare for an optimized write sequence.

Optimized write sequences allow applications to produce data in chunks without having any overhead managing the chip select.

reset()
Reset the display.

Uses the hardware reset pin if there is one, otherwise it will issue a software reset command.

write_cmd(*cmd*)
Send a command opcode to the display.

Parameters **cmd** (*sequence*) – Command, will be automatically converted so it can be issued to the SPI bus.

write_data(*buf*)
Send data to the display.

Parameters **buf** (*bytearray*) – Data, must be in a form that can be directly consumed by the SPI bus.

5.2.6 Generic PWM capable vibration motor driver

class drivers.vibrator.**Vibrator**(*pin*, *active_low=False*)
Vibration motor driver.

__init__(*pin*, *active_low=False*)
Specify the pin and configuration used to operate the motor.

Parameters

- **pin** (*machine.Pin*) – The PWM-capable pin used to driver the vibration motor.
- **active_low** (*bool*) – Invert the resting state of the motor.

pulse(*duty=25*, *ms=40*)
Briefly pulse the motor.

Parameters

- **duty** (*int*) – Duty cycle, in percent.
- **ms** (*int*) – Duration, in milliseconds.

5.3 Applications

5.3.1 Digital clock

Shows a time (as HH:MM) together with a battery meter and the date.

class apps.clock.**ClockApp**

Simple digital clock application.

Fig. 1: Screenshot of the clock application

ICON = 'Default digital clock icon'

NAME = 'Clock'

foreground()

Activate the application.

Configure the status bar, redraw the display and request a periodic tick callback every second.

sleep()

Prepare to enter the low power mode.

Returns True, which tells the system manager not to automatically switch to the default application before sleeping.

tick(*ticks*)

Periodic callback to update the display.

wake()

Return from low power mode.

Time will have changes whilst we have been asleep so we must udpate the display (but there is no need for a full redraw because the display RAM is preserved during a sleep.

5.3.2 Flashlight

Shows a pure white screen with the backlight set to maximum.

class apps.flashlight.**FlashlightApp**

Trivial flashlight application.

ICON = 'Default torch or flashlight icon'

NAME = 'Torch'

background()

De-activate the application (without losing state).

draw()

Redraw the display from scratch.

Fig. 2: Screenshot of the flashlight application

foreground()
Activate the application.

tick(*ticks*)

5.3.3 Application launcher

class apps.launcher.**LauncherApp**
An application launcher application.

Fig. 3: Screenshot of the application launcher

ICON = 'Default application icon'

NAME = 'Launcher'

foreground()
Activate the application.

swipe(*event*)

touch(*event*)

5.3.4 Pager applications

The pager is used to present text based information to the user. It is primarily intended for notifications but is also used to provide debugging information when applications crash.

class apps.pager.**CrashApp**(*exc*)

Crash handler application.

This application is launched automatically whenever another application crashes. Our main job it to indicate as loudly as possible that the system is no longer running correctly. This app deliberately enables inverted video mode in order to deliver that message as strongly as possible.

background()

Restore a normal display mode.

Conceal the display before the transition otherwise the inverted bombs get noticed by the user.

foreground()

Indicate the system has crashed by drawing a couple of bomb icons.

If you owned an Atari ST back in the mid-eighties then I hope you recognise this as a tribute a long forgotten home computer!

swipe(*event*)

Show the exception message in a pager.

class apps.pager.**NotificationApp**

NAME = 'Notifications'

foreground()

Activate the application.

class apps.pager.**PagerApp**(*msg*)

Show a long text message in a pager.

ICON = 'Default application icon'

NAME = 'Pager'

background()

De-activate the application.

foreground()

Activate the application.

swipe(*event*)

Swipe to page up/down.

5.3.5 Self Tests

class apps.testapp.**TestApp**

Simple test application.

ICON = 'Default application icon'

NAME = 'Self Test'

foreground()

Activate the application.

press(*button, state*)

Fig. 4: Screenshot of the self test application

swipe (*event*)

touch (*event*)

5.4 Bootloader

The bootloader implements a couple of protocols that allow the bootloader and payload to communicate during a reset or on handover from bootloader to application.

5.4.1 GPREGRET protocol

`GPREGRET` is a general purpose 8-bit retention register that is preserved in all power states of the nRF52 (including System OFF mode when SRAM content is destroyed).

It can be used by the application to request specific bootloader behaviours during a reset:

Name	Value	Description
OTA_APPJUM	0xb1	Bootloader entered (without reset) from application.
OTA_RESET	0xa8	Enter OTA (Bluetooth) recovery mode
SERIAL_ONLY_RESET	0x4e	Enter UART recovery mode (if applicable)
UF2_RESET	0x57	Enter USB recovery mode (if applicable)
FORCE_APP_BOOT	0x65	Force direct application boot (no splash screen)

5.4.2 PNVRAM protocol

The pseudo non-volatile RAM is a small block of regular static RAM that, once initialized, can be used to share information.

The PNVRAM starts at 0x200039c0 and is 32 bytes long.

Address	Description
0x200039c0	Guard value. Must be set to 0x1abe11ed .
0x200039c4	Course grained RTC value (bootloader must preserve but can ignore).
0x200039c8	RTC millisecond counter (bootloader must increment this).
0x200039cc	Reserved
0x200039d0	Reserved
0x200039d4	Reserved
0x200039d8	Reserved
0x200039cc	Guard value. Must be set to 0x10adab1e .

Note: *The PNVRAM protocol allows up to 28 bytes to be transfered (compared to 2 bytes via GPREGRET and GPREGRET2) but it is less versatile. For example FORCE_APP_BOOT cannot be implmented using PNVRAM.*

The RTC millisecond counter is incremented whenever the bootloader is active (during splash screen or early UART recovery mode, during an update). It can be consumed by the application to prevent the current time being lost during an update.

5.4.3 Watchdog protocol

Form-factor devices such as smart watches and fitness trackers do not usually have any hardware mechanism to allow the user to force a failed device into bootloader mode. This makes them difficult to develop on because opening the case to access a SWD or reset pins may compromise their waterproofing.

wasp-os uses a watchdog timer (WDT) combined with a single hardware button in order to provide a robust mechanism to allow the user to force entry into a over-the-air firmware recovery mode that allows the buggy application to be replaced.

The software responsibilities to implement this are split between the bootloader and the application, although the application responsibilities are intentionally minimal.

The bootloader implements an over-the-air recovery mode, as well as handling normal boot, where it's role is to display the splash screen.

Additionally the bootloader implements several watchdog related features necessary for robust reboot handling:

1. The bootloader configures the watchdog prior to booting the main application. This is a simple, single channel reload request, watchdog with a 5 second timeout.
2. The bootloader checks the reset reason prior too booting the main application. If it detects a watchdog reset the bootloader switches automatically to DFU mode.
3. The bootlaoder initialized the pinmux allowing the hardware button state to be observed.
4. The bootloader monitors the hardware button and switches back to the main application when it is pressed.

From this list #1 and #2 are needed to ensure robust WDT handling whilst #3 and # 4 ensure the user can switch back to application from the device itself if they ever accidentally trigger entry to recovery mode.

The application's role is to carefully pet the watchdog so that it will trigger automatically if the hardware button is held down for five seconds. Key points for application robustness include:

1. Unlike a normal watchdog we can be fairly reckless about where in the code we pet the dog. For example petting the dog from a timer interrupt is fine because we only need the dog to bark if the hardware button is pressed.
2. The routine to pet the dog is predicated on the hardware button not being pressed.
3. The routine to pet the dog is also predicated on the hardware button still being correctly configured.

To avoid mistakes the application should contain no subroutines that unconditionally pet the dog; they should all implement #2 and #3 from the above list.

Note: *nRF52 microcontrollers implement a distributed pin-muxing mechanism meaning most peripheral can acidentally "steal" a pin if the pin is requested by the peripheral. This requires a fully robust implementation of #3 to visit the PSEL registers of every peripheral that can control pins. The code currently used in wasp-os does not yet meet this criteria.*

CHAPTER 6

Contributor's Guide

6.1 Introduction

Anyone can contribute to the wasp-os project. Contributions are typically made via github using the typical fork-and-pull-request approach. Contributors who do not wish to use github are welcome to share patches using `git format-patch --to wasp-os@redfelineninja.org.uk` and `git send-email`. In both cases, the code will be reviewed by a project maintainer, so please anticipate review comments and requests for changes. Typically pull requests will not be merged if there are open questions or requests for changes that have not been acted on.

All contributions must include a `Signed-off-by` tag added by the contributor who submits the patch or patches.

The `Signed-off-by` tag is added at the end of the patch description and certifies that the contributor either wrote the patch or has the right to share the code under the open source license appropriate for the file being modified.

A `Signed-off-by` tag is an explicit statement that your contribution comes under one of (a), (b), (c), or (d) from the list below so please be sure to read carefully what you are certifying by adding your Signed-off-by.

Additionally please be aware that that contributors, like all other members of the wasp-os community, are expected to meet the community guidelines described in the project's code of conduct when interacting within all community spaces (including the wasp-os github presence).

6.2 Developer Certificate of Origin

```
Developer Certificate of Origin
Version 1.1

Copyright (C) 2004, 2006 The Linux Foundation and its contributors.
1 Letterman Drive
Suite D4700
San Francisco, CA, 94129

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


Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
    have the right to submit it under the open source license
    indicated in the file; or

(b) The contribution is based upon previous work that, to the best
    of my knowledge, is covered under an appropriate open source
    license and I have the right under that license to submit that
    work with modifications, whether created in whole or in part
    by me, under the same open source license (unless I am
    permitted to submit under a different license), as indicated
    in the file; or

(c) The contribution was provided directly to me by some other
    person who certified (a), (b) or (c) and I have not modified
    it.

(d) I understand and agree that this project and the contribution
    are public and that a record of the contribution (including all
    personal information I submit with it, including my sign-off) is
    maintained indefinitely and may be redistributed consistent with
    this project or the open source license(s) involved.
```

This procedure is the same one used by the Linux kernel project. To sign off a patch append an appropriate line at the end of the commit message:

```
Signed-off-by: Random Developer <r.developer@example.org>
```

Adding a sign-off can be automated by using git features such as `git commit --signoff`. Please use your real name, anonymous and pseudonymous contributions will not be accepted.

6.3 Git Hints and Tricks

6.3.1 Quick fixes

The most common review feedback for contributions to wasp-os is a request that the contributor include their sign-off. For a single patch at the head of the current branch (and shared as a github pull request) this can be handled fairly easily:

```
git commit --amend --signoff
git push <myfork> HEAD
```

Additionally, please be aware that github will not send out automatic notifications to let the maintainer know that you have pushed an update to the pull-request. Follow up the above with a comment on the pull request thread saying that your contribution has been updated and is ready for another look.

6.4 Code of Conduct

6.4.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

6.4.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

6.4.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

6.4.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

6.4.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at wasp-os@redfelineninja.org.uk . All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

6.4.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

6.4.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.0, available at https://www.contributor-covenant.org/version/2/0/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

CHAPTER 7

Roadmap

7.1 0.4: Integration, Fit and finish

For 0.4 we focus on improving the watch/phone integration whilst also taking steps to improve the general fit and finish.

7.1.1 Bootloader

- [ ] Stay in bootloader after battery run down
- [ ] Implement power off support (no splash screen)

7.1.2 Micropython

- [ ] Use SoftDevice sleep logic
- [ ] Rebase on later version of MicroPython

7.1.3 Wasp-os

- [ ] Watch/phone integration with GadgetBridge

- [ ] Music player support
- [ ] Set date/time
- [ ] Fully fledged wasp-os device class

7.1.4 wasptool

- [ ] Integrate a more powerful minifier into the wasptool paste() method

7.2 0.3 (a.k.a. M3): Smartwatch

At M3 we start to build out full fitness tracking and notification functionality.

7.2.1 Reloader

- [X] Pre-flash image verification
- [X] Post-flash image verification
- [X] Board identity check
- [X] UICR update support
- [X] Improve linker map (everything except linker table at +256K)
- [X] mcuboot
  - [X] Reconfigurable entry point (allow reloader to run from mcuboot)
  - [X] Allow reloader to install mcuboot and flash app (from wasp-bootloader)
  - [X] Allow reloader to install wasp-os (from mcuboot)

7.2.2 Wasp-os

- [X] Enable heart rate sensor
  - [X] HRS3300 driver
  - [X] HRS data post-processing
  - [X] Heart rate counter app
- [X] Notifications
  - [X] BLE notification protocol
  - [X] Notification popups
  - [X] Notification app (show notification history)
  - [X] Add (out-of-tree) Gadgetbridge support
- [X] Step counting
  - [X] BMA421 driver
  - [X] Step counter app
- [X] Automatically enter SPI flash power saving mode

- [X] Documentation
  - [X] Contributors guide (and code of conduct)
  - [X] Debugging and troubleshooting guide
  - [X] Screenshots for bootloader and all applications
  - [X] Improve the install guide
- [X] Simulator
  - [X] Add a simple skin for better screenshots
  - [X] Full swipe detection (avoid keyboard)

7.3 0.2 (a.k.a. M2): Great developer experience

The focus for M2 is to make development faster and easier by providing a file system and file transfer code. This allows much faster development cycles compared to full downloads of frozen modules. Additionally support for multiple event-driven applications will be added during M2 to further help developers by providing example applications.

7.3.1 Bootloader

- [X] OTA bootloader update
- [X] RTC time measurement whilst in bootloader

7.3.2 MicroPython

- [X] SPI FLASH driver
- [X] Enable LittleFS on SPI FLASH (at boot)
- [X] BLE file transfer

7.3.3 Wasp-os

- [X] Add dd/mm/yyyy support to RTC
- [X] Button driver (interrupt based)
- [X] Touch sensor driver
- [X] Event driven application framework
- [X] Stopwatch app
- [X] Settings app
- [X] PC-hosted simulation platform
- [X] Documentation
  - [X] Sphinx framework and integration with github.io
  - [X] Document bootloader protocols
  - [X] Application writer's guide

- [X] Write full docstring documentation for all wasp-os components

- [X] Application Launcher
- [X] Debug notifications
- [X] Multi-colour RLE images
  - [X] Optimized “2-bit” RLE encoder and decoder
  - [X] Logarithmic RBG332 <-> RGB56516bit color space conversion

7.4 M1: Dumb watch feature parity

The focus for M1 is to get wasp-os both to meet feature parity with a dumb watch and to have a bootloader and watchdog strategy that is robust enough to allow a PineTime case to be confidently glued shut.

7.4.1 Bootloader

- [X] Basic board ports (PineTime, DS-D6, 96Boards Nitrogen)
- [X] OTA application update
- [X] Enable watchdog before starting the application
- [X] Splash screen
- [X] Ignore start button for first few seconds

7.4.2 MicroPython

- [X] Basic board ports (PineTime, DS-D6, 96Boards Nitrogen)
- [X] Long press reset (conditional feeding of the watchdog)
  - [X] Feed dog from REPL polling loop
  - [X] Feed dog from a tick interrupt

7.4.3 Wasp-os

- [X] Display driver
  - [X] Display initialization
  - [X] Bitmap blitting
  - [X] RLE coder and decoder
  - [X] Optimized RLE inner loops
- [X] Backlight driver
- [X] Button driver (polling)
- [X] Battery/charger driver
- [X] Simple clock and battery level application
- [X] Basic (WFI) power saving

- [X] Implement simple RTC for nrf52

CHAPTER 8

Licensing

wasp-os is licensed to you under the GNU Lesser General Public License, as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

wasp-os 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 and the GNU Lesser General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

Notwithstanding the above some essential components of wasp-os, such as the MicroPython distribution, are licensed under under different open source licenses. The licensing for these components is clearly indicated and reinforced by the directory and sub-module structure.

Additionally binary releases of wasp-os include a binary copy of the Nordic Softdevice which is licensed under the 5-clause Nordic license.

8.1 GNU Lesser General Public License

Version 3, 29 June 2007 Copyright © 2007 Free Software Foundation, Inc <http://fsf.org/>

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

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

8.1.1 0. Additional Definitions

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

8.1.2 1. Exception to Section 3 of the GNU GPL

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

8.1.3 2. Conveying Modified Versions

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

- **a)** under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- **b)** under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- **a)** Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- **b)** Accompany the object code with a copy of the GNU GPL and this license document.

8.1.4 4. Combined Works

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- **a)** Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- **b)** Accompany the Combined Work with a copy of the GNU GPL and this license document.

- **c)** For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- **d)** Do one of the following:
  - **0)** Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
  - **1)** Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that **(a)** uses at run time a copy of the Library already present on the user's computer system, and **(b)** will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
- **e)** Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option **4d0**, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option **4d1**, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

8.1.5 5. Combined Libraries

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 that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

- **a)** Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
- **b)** Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8.1.6 6. Revised Versions of the GNU Lesser General Public License

The Free Software Foundation may publish revised and/or new versions of the GNU 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 as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

8.2 GNU General Public License

Version 3, 29 June 2007 Copyright © 2007 Free Software Foundation, Inc <http://fsf.org>

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

8.2.1 Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. 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 for them 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 prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. 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.

Developers that use the GNU GPL protect your rights with two steps: **(1)** assert copyright on the software, and **(2)** offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

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

8.2.2 TERMS AND CONDITIONS

0. Definitions

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that **(1)** displays an appropriate copyright notice, and **(2)** tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that **(a)** is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and **(b)** serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies

You may convey 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; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- **a)** The work must carry prominent notices stating that you modified it, and giving a relevant date.
- **b)** The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- **c)** You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- **d)** If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- **a)** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- **b)** Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either **(1)** a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or **(2)** access to copy the Corresponding Source from a network server at no charge.
- **c)** Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- **d)** Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- **e)** Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either **(1)** a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or **(2)** anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- **a)** Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- **b)** Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- **c)** Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- **d)** Limiting the use for publicity purposes of names of licensors or authors of the material; or
- **e)** Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- **f)** Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated **(a)** provisionally, unless and until the copyright holder explicitly and finally terminates your license, and **(b)** permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either **(1)** cause the Corresponding Source to be so available, or **(2)** arrange to deprive yourself of the benefit of the patent license for this particular work, or **(3)** arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license **(a)** in connection with copies of the covered work conveyed by you (or copies made from those copies), or **(b)** primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom

If 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 convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License

The Free Software Foundation may publish revised and/or new versions of the GNU 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 that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty

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

16. Limitation of Liability

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

17. Interpretation of Sections 15 and 16

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

8.2.3 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 state 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 a brief idea of what it does.> Copyright (C) <year> <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 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

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

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

> <program> Copyright (C) <year> <name of author> This program 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, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU 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. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

8.3 The MIT License (MIT)

Copyright © *<year>* *<copyright holders>* (see individual files for copyright information)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

8.4 5-Clause Nordic License

Copyright (c) 2007 - 2018, Nordic Semiconductor ASA All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

d

f

i

l

w

Index

Symbols

A

B

C

D

E

F

T

U

V

W