LotOS Framework

Getting Started Guide for
Banana Pi

Linux is a registered trademark of Linus Torvalds.

Yocto is a registered trademark of Linux Foundation.

FreeRTOS is a registered trademark of Real Timer Engineers Ltd.

All other trademarks are the property of their respective owners.

LotOS framework contains open-source software. License texts and source code are delivered within the kit.

Contents

List of Figures

List of Tables

1 Introduction

LotOS framework is a set of open source operating systems, that can be run on the same hardware using Mango hypervisor.
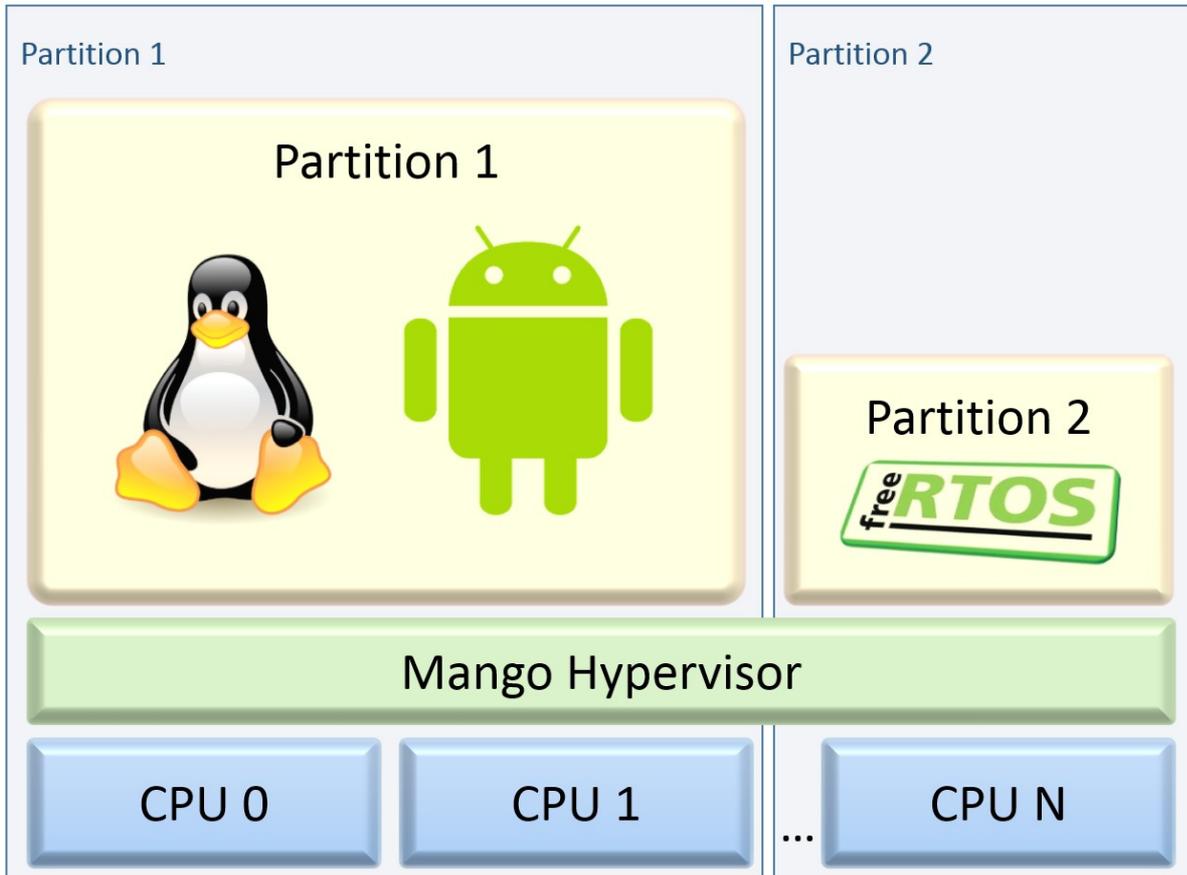
Illustration 1: LotOS Framework

Mango hypervisor is a piece of software that allows running several operating systems (guests) in full isolation on a single system. Hardware resources and peripheral devices are assigned statically to every guest.

Mango provides:

- Hardware-assisted virtualization
- Full guest isolation
- Static hardware resources assignment
- Bounded worst-case latency
- API for communication between guests
- Native operating system performance
- Guest extensions

This document describes LotOS framework. ilbers GmbH provides a free port of this

framework for evaluating it on a Banana Pi board. Banana Pi is a single-board computer; please refer to its official site for more information:

http://www.bananapi.org/

LotOS framework is based on Yocto project:

https://www.yoctoproject.org/

and implemented as a layer.

The build has been tested on an x86 host under the following operating systems:

- Fedora 20, 64-bit

- Debian 7, 32-bit

1.1 Document Organization

This document is intended for users who want to try LotOS framework on Banana Pi board. It is split into two parts:

- Quick start guide – using complete SD card image with LotOS framework components already installed

- Step-by-step manual installation

Please refer to appropriate chapter for more information.

1.2 Pre-requisites

The following hardware and software is needed to use the LotOS framework:

1. Banana Pi board.

2. A 2-GB SD card (4-GB for image with accelerated graphics drivers).

3. Optionally, power supply for Banana Pi: 5V, 2A, micro-USB (micro-B plug).

   Samsung ETA0U80E (5V, 1A) and PC USB port (5V, 0.5A) worked for tests in this framework.

4. USB to serial converter with 3.3V TTL signals.

   PL2303-based USB to serial converter was used for tests.

5. A PC with:

   - USB port for serial communication

   - USB port for board power if no power supply is used

   - SD card slot

   - Linux OS

   - Serial terminal program, such as minicom or kermit

6. Please refer to Yocto project documentation to get full list of host packages needed to be installed:

http://www.yoctoproject.org/docs/1.8/mega-manual/mega-manual.html#packages

1.3 LotOS Framework Contents

The LotOS layer contains the following components:

- Mango hypervisor binary image

- FreeRTOS v8.2.1

- Mango configuration file and configuration compiler

- U-Boot

- Linux kernel

- Linux kernel modules (for accessing certain Mango features)

- Various demo applications

2 Getting Started

2.1 Connect Serial Port

The board serial port are used by the software for debugging output and command input.

The UART pins are located in the upper left corner of the board. They are marked as TXD, RXD and GND on the PCB. Commercially available USB to serial converters with 3.3V TTL signals may be used.

WARNING: Do not connect RS-232 devices (e.g., PC serial port via null modem cable) to the board directly, since RS-232 signals may damage the board.

The following table shows the connection for a PL2303-based converter.

Board connector	Connector pin	Converter wire
J11	TXD	White
J11	RXD	Green
J12	7 or 8	Black
Do not connect	Do not connect	Red

Table 1: Banana Pi serial pinout

WARNING: Do not connect the converter's red wire (VCC or 3.3V / 5V), as that might damage the board.

The picture below shows how to connect the converter to the Banana Pi board.
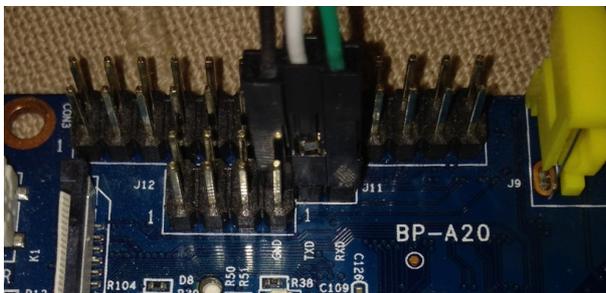
Illustration 2: Banana Pi serial connection

Connect the USB plug of the converter to the host PC. Look for the serial device name in the `dmesg` output (they are usually called like `/dev/ttyUSB0`, `/dev/ttyACM0`). Configure the serial terminal program for 115200 bps, 8 data bits, no parity, 1 stop bit, no flow control. Example `~/.kermrc` for kermit:

```
set line /dev/ttyUSB0
set speed 115200
set carrier-watch off
set flow-control none
connect
```

The serial connection will be tested after getting the software on the SD card.

For more information, please refer to http://linux-sunxi.org/UART.

3 LotOS Framework Quick Start

3.1 Prepare SD card

The full SD card image can be downloaded from the following link:

http://dl.ilbers.de/dl/lotos-bananapi.sdimg.zip

This image contains already installed and configured components for LotOS framework.

Unzip the image:

```
$ unzip lotos-bananapi.zip
```

Now, the image should be written to the card on the host PC.

WARNING: The image overwrites all data on the card, including partition table.

WARNING: Ensure that the image is written to the correct device; writing to a wrong device may result in data loss on other storage available on the host system.

3.1.1. For Linux

Determine the SD card device name using `dmesg` or `lsscsi` commands. Depending on the card reader connection type, devices may have names like `/dev/mmcblk0` or `/dev/sde`.

Many Linux distributions mount inserted devices automatically. Check whether the card device name is present in the `mount` output (if the card is partitioned, device names may be `/dev/mmcblk0p1` or `/dev/sde1`). If this is the case, unmount the SD card devices listed in the `mount` output as shown below. Replace `${part}` with the mounted partition name. Repeat for every mounted SD card partition.

```
$ sudo umount ${part}
```

Replace `${card}` with the device name of your card and write the image:

```
$ sudo dd if=mf_sd_bananapi.img of=${card} bs=1M && sync
```

3.1.2. For Windows

To write SD card image in windows the Win32DiskImager can be used.

3.2 Boot LotOS Framework

Now Banana Pi board is ready to boot. By default, the framework is configured to start two guest operating systems:

- FreeRTOS

- Linux

The board will boot automatically after power supply is enabled.

To switch serial console between guest partition a `CTRL+T` key combination can be used.

Please refer to Mango Hypervisor User Interface chapter for more information about hypervisor user interface.

4 Build LotOS Framework

4.1 Build Framework Components

LotOS framework can be also built from the scratch. As already mentioned, it is based on Yocto project. Please refer to [Yocto Complete Documentation Set](#) for more information about the project and host system requirements.

The following steps describe how to build LotOS from the scratch.

Get Yocto poky release:

```
$ git clone git://git.yoctoproject.org/poky.git
```

Get `sunxi` layer. This is official Yocto layer for Allwinner SoCs support :

```
$ cd poky
$ git clone https://github.com/linux-sunxi/meta-sunxi.git
```

Get LotOS layer:

```
$ git clone https://github.com/ilbers/meta-lotos
```

Prepare build configuration:

```
$ source oe-init-build-env lotos_build_dir
```

Edit conf/local.conf: line 37

```
MACHINE ??= "bananapi-lotos"
```

Edit conf/bblayers.conf and add layers `meta-sunxi` and `meta-lotos`.

Start build :

```
$ bitbake core-image-sato
```

When build is complete, the SD card image can be found in folder `tmp/deploy/images/bananapi`. Instructions how to flash this image can be found in previous [chapter](#).

4.2 Insert Linux Kernel Modules

This framework provides several drivers for Linux to enable extra hypervisor features:

- Individual partition watchdog
- Cross-partition data channels
- Cross-partition virtual network adapter

LotOS framework provides `init.d` script, which loads all the Linux extension for Mango support.

To load the modules manually the following commands can be used:

```
# modprobe mango_core
# modprobe mango_data_channel
# modprobe mango_watchdog
# modprobe mango_net_iface
```

4.3   Virtual UART

The hypervisor is configured with virtual UART support, which allows to use one physical port for input / output from several partitions. At any given moment, the physical port is logically connected to one partition. The output from the other one is buffered. Ctrl+T connects the port to another partition. The buffered output (if any) is printed, and the input is logically connected to the second partition.

4.4   Cross-Partition Networking

LotOS framework provides a possibility to communicate between partition over TCP/IP. FreeRTOS port, shipped with the framework has built-in Mango networking support. The default IP address for FreeRTOS network adapter is `192.167.20.2`. To enable this feature in Linux, the following commands have to be executed:

```
# modprobe mango_net_iface
# ifconfig mango0 192.167.20.1
```

The networking can be quickly tested by using ping command:

```
# ping -I mango0 192.167.20.2
```

5 Mango Hypervisor User Interface

5.1 Mango Management Console

Mango hypervisor provides a textual console to check and change current partition configuration. Each guest partition has its own console instance, so the console settings can differ between partitions. Please note, that console settings are volatile, so board power cycle will restore the default ones.

NOTE: console support is optional and can be disabled during compilation time. Please check Mango binary release notes to identify whether console is included to your image.

5.1.1 Partition Information

To retrieve the information about current partition the following command can be used:

```
=> partinfo
```

This command displays the following information:

- Current partition identifier
- CPU cores assigned to the partitioned
- Guest operating system identifier
- Partition boot cycles
- Partition memory layout
- Partition IRQ lines assignment

5.1.2 Mango Watchdog

Mango hypervisor provides a watchdog to monitor the partitions state. It's implemented individually for each partition, so each partition counter should be triggered individually by guest operating system running in this partition.

There is one main watchdog command in console, which accepts different parameters. To see the information about parameters the only command should be entered:

```
=> wd
```

This command displays the supported parameters.

To get information about current watchdog state the following command should be used:

```
=> wd info
```

This command displays the following information:

- Current watchdog state: enabled or disabled
- Watchdog timeout value

To enable watchdog counter, the following command should be entered:

```
=> wd enable
```

To change the default watchdog timeout, the following command should be used

```
=> wd timeout VAL
```

Where **VAL** is new timeout value in seconds. This parameter can't be set to zero.

NOTE: once enabled, the watchdog timer can't be disabled.

5.1.3 Console Commands List

To retrieve the list of commands supported by the console, the following command should be used:

```
=> help
```

5.1.4 Start Guest Operating System

To exit the console and start guest operating system the following command should be used:

```
=> boot
```

5.1.5 Console Settings

The console can be disabled for the next guest restarts by using the following command:

```
=> console
```

To enable the console for the next partition boot-cycles, run the command:

```
=> console on
```

To disable the console for the next partition boot-cycles, run the command:

```
=> console off
```

To see the current console state, run the command:

```
=> console info
```

5.2 Mango Configuration File

This framework provides Mango hypervisor binary with the built-in partition configuration for the Banana Pi board. But also there is a way to modify this layout using a configuration file. As provided in this framework, it defines memory regions and physical IRQ lines assigned to every partition.

5.2.1 File Format

The file has the following format:

```
; This line is a comment and is ignored by the converter
partition X
        mem     IPA     PA     Size     Type
        ...
        irq     N
        ...
endp
```

Description:

- `partition`: Partition X definition
- `mem`: Memory region to be mapped to partition:
  - `IPA`: Physical address of the region as it is seen by the guest
  - `PA`: Physical address of the region
  - `Size`: Size of the region (should be multiple of 4 KiB)
  - `Type`: Memory type: "Device" or "normal"
- `irq`: IRQ number to be routed to the guest
- `endp`: End of partition section

To compile the configuration, `mng2bin` tool should be used:

```
$ cd mango-framework/components/mango-utils/mng2bin
$ ./mng2bin bananapi.mng bananapi.bin
```

5.2.2 Device Sharing

In some situations, it may be desirable to use a certain device in several partitions. There are two ways to do that:

- Allow physical access for both partitions and implement arbitration in the guest systems.

- Disallow physical access for both partitions and use a hypervisor-level virtual device driver for full isolation.

5.2.3 Shared Hardware

Mango configuration file supports hardware sharing through mapping the respective I/O memory region to several partitions. One use case could be using of GPIO pins. For example, the first partition uses a set of GPIO A pins, and the second one – a set of GPIO B pins.

Another example are UART ports. In ARM architecture, the minimum mappable page size is 4 KiB. Unfortunately, Allwinner A20 SoC used in Banana Pi provides all UARTs (from 0 to 3) in a single page. This makes it impossible to assign different UARTs to different guests exclusively. With hardware sharing approach, all UARTs are mapped to both partitions. The first partition can use e.g. UART0, the second – UART1. Non-interference must be ensured by the guests.

5.2.4 Virtual Hardware

If full isolation is required, a virtual device driver has to be implemented in Mango. The guests are not aware that the device is shared. Mango manages the access to the physical device. This framework contains a virtual driver for the UART. The same hardware port is used for serial I/O by several partitions, without guest OS modifications.

6 Demo Applications

This framework provides a set of demo scenarios for evaluation.

6.1   Linux Applications

• `watchdog`: A simple watchdog daemon for Linux partition.

• `hello`: This application demonstrates the usage of cross-partition networking. The Linux sends the string "Hello, world!" to the FreeRTOS partition and it displays it on serial console. The TCP/IP sockets are used for communication.

6.2   FreeRTOS Applications

• `time`: This FreeRTOS thread is printing partition uptime, calculated using timer interrupts. The application may be used to assess hypervisor's bounded interrupt latency.

• `tcp_server`: This thread starts TCP server and listens for connections from Linux partition on network port `3310`.

7 Support

ilbers GmbH provides services around the whole life-cycle of your product development, including:

- Mango maintenance, new features, porting to other CPU architectures

- Hardware and software evaluation

- Initial product setup, integration with Mango

- Driver and application development (hardware control, Linux, real-time OS)

- Feature development and maintenance of open-source software

- Testing, test-driven development, automated testsuites, continuous integration

- Product and process documentation for certification

For any questions, please contact: mango-support@ilbers.de