LotOS Framework

Getting Started Guide for
Banana Pi

Copyright (C) 2015-2016 ilbers GmbH
Revision 1.3, 2016-06-14

Mango hypervisor and LotOS framework are copyright (C) 2014 – 2016 ilbers GmbH. All rights reserved.

There is no warranty for the software, to the extent permitted by applicable law, except when otherwise stated in writing. The copyright holders 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 the users of the software. Should the program prove defective, the users of the software assumes the cost of all necessary servicing, repair or correction.

In no event unless required by applicable law or agreed to in writing will any copyright holder be liable to the users of the software 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 the users of the software 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.

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.

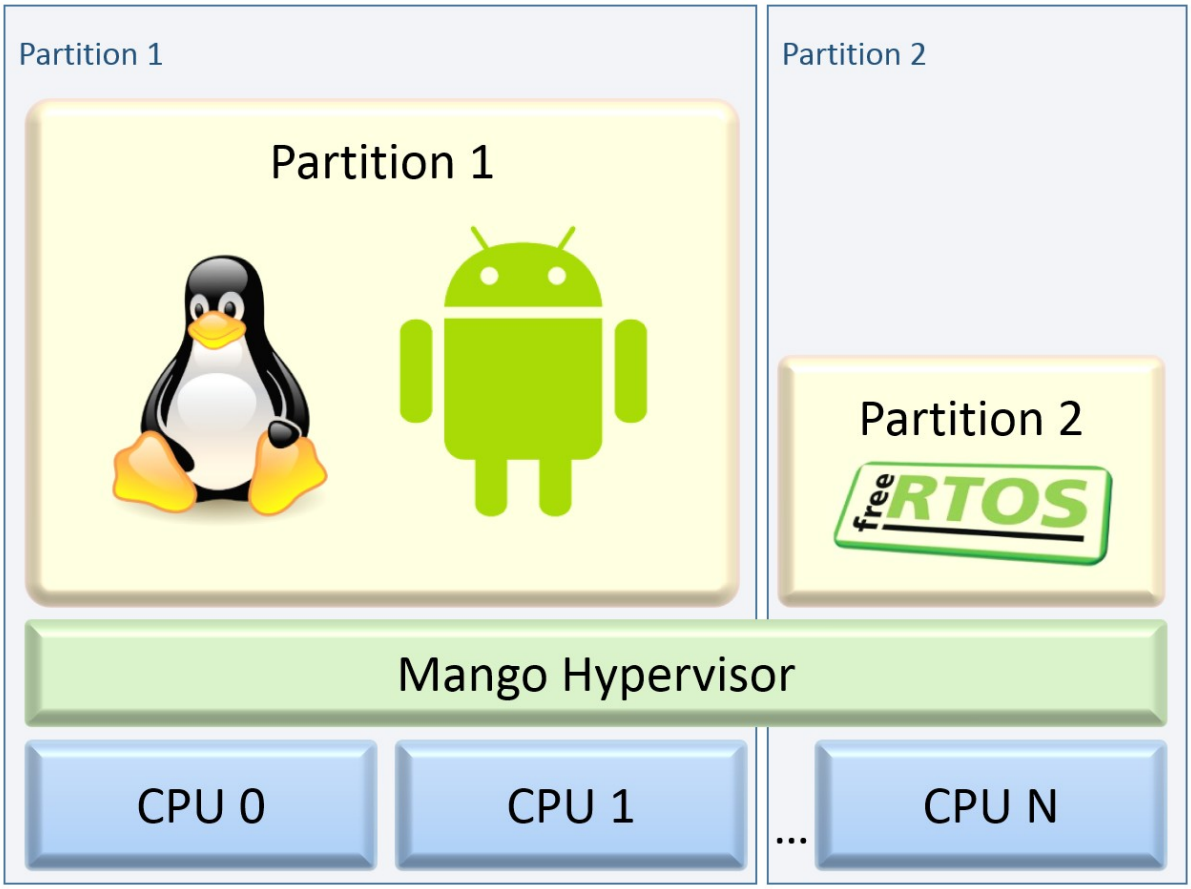

Illustration 1: LotOS Framework

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
- Cross-partition networking

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 v1.4 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.

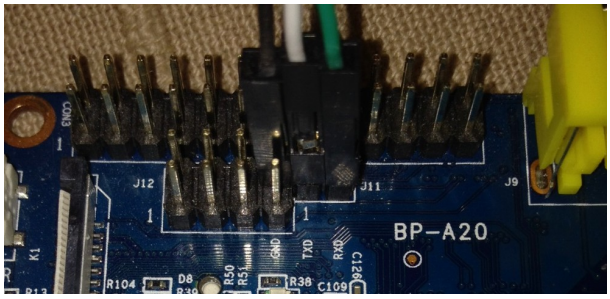

Illustration 2: Banana Pi serial connection

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/core-image-lotos.sdimg.gz

MD5Sum: `da29c24a8df9c09f8798f49430517989`

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

Unzip the image:

```
$ gunzip 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.

6.1.1. 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=core-image-lotos.sdimg of=${card} bs=1M && sync
```

6.1.2. 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.

NOTE: to get LotOS framework a `repo` tool has to be installed on the build host. It can be downloaded here.

Get LotOS framework release:

```
$ repo init -u https://github.com/ilbers/lotos-manifest.git
$ repo sync
```

Prepare build configuration:

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

Edit conf/local.conf: line 37

```
MACHINE ??= "bpi-lotos"
```

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

Now the environment is ready to start the build. There are several Linux images available:

- `core-image-minimal` – console Linux image with minimal features set.
- `core-image-base` – console Linux image with typical features set.
- `core-image-lotos` – graphical Linux image with Qt5 support.

The following command starts building of graphical Linux image:

```
$ bitbake core-image-lotos
```

When build is complete, the SD card image can be found in folder `tmp/deploy/images/bpi-lotos/*.bpi-sdimg`. 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 automatically.

If system has been modified and it needs 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
```

NOTE: this setup is done by Mango init scripts, so if the unmodified system is used, there is no need to configure networking manually.

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
- Partition GPIO pins assignment
- Partition allocator state

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 for the Banana Pi board. But also there is a way to modify partitions layout using a configuration file. As provided in this framework, it defines memory regions and physical IRQ lines assigned to every partition. In `mango-utils` package, there are two examples for BPi board with single and dual partitions layout.

5.2.1 File Format

The file has the following format:

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

```
        irq     N
        ...
        gpio    A
        gpio    A..B
        ...
endp
```

Description:

- `partition`: Partition X definition
- cpus: Assign Y CPU cores to this partition
- `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
- `gpio`: the consequent number of the pin in the GPIO interface, or the range of pins
- `endp`: End of partition section

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

```
$ cd 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 UART ports. In ARM architecture, the minimum mappable page size is 4 KiB. Unfortunately, some SoCs provides several peripherals 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 evaluation kit contains a virtual drivers for the UART, DMA and GPIO. The same registers sets are used by several partitions, without guest OS modifications.

6 Demo Applications

This framework provides a set of demo applications for framework evaluation.

Linux Applications

6.1 Overview

Demo applications intended to demonstrate the following Mango features.

6.1.1 Key Combinations

The LotOS framework hooks the following key combinations, pressed in serial console:

`CTRL+t` – switch console to the next partition

`CTRL+r` – reset the partition

6.1.2 Networking

Mango hypervisor provides a cross-partition networking support, so the guest partitions can communicate each other using TCP/IP. The default LotOS configuration brings up networking between partitions and assigned the following IP addresses:

- `Linux` – 192.167.20.1
- `FreeRTOS` – 192.167.20.2

The easiest way to check network connectivity is try to ping the second partition from Linux. The following command can be used:

```
$ ping 192.167.20.2
```

6.1.3 Data Channel

There is another way to communicate between partition – serial data channel. The default LotOS configuration provides 2 independent data channels. This interface intended to use for small data bursts, or if there is no networking stack in guest operating system.

6.1.4 Watchdog

Mango hypervisor provides individual watchdog timer for each partition. The guest operating system has to trigger the timer, otherwise it will be reset when timer expired. The default timeout value is 60 seconds.

6.1.5 Individual Partition Management

In LotOS framework both operating systems work in isolated partitions, so the activity in one partition doesn't affect the other. Each partition can be reset independently.

6.1.6 Fast Boot

Mango hypervisor caches the guest operating system binaries in RAM. So after the partition reset, there is no need to use slow disk I/O operations to load binaries. This feature significantly speeds up partition start time.

6.1.7 Mango Frame Buffer

LotOS framework provides a cross-partition graphical output. `FreeRTOS` port for Mango contains library to render the following:

- Text
- Graphical objects: pixel, line, rectangle
- Pixmap

The graphical output from `FreeRTOS` can be displayed in Linux using `mangofb` application.

6.2 Linux Applications

6.2.1 Watchdog

LotOS framework includes the watchdog daemon application, that starts automatically during boot time. To demonstrate the watchdog reset by timeout, the daemon can be killed in Linux partition by using the following command:

```
# killall wd_daemon
```

After this command, if the timeout value hasn't been changed, the partition will be reset during 1 minute.

6.2.2 Networking

To demonstrate networking functionality, a simple TCP client-server included to LotOS framework. The server is implemented in `FreeRTOS`, so the `Linux` partition contains the client application. The following command can be used to start the application:

```
# tcp_client
```

The program will ask you to enter a message. After pressing enter key, the `FreeRTOS` partition should print this message. Press `CTRL+t` and check this.

6.2.3 Data Channel

There is an application, that demonstrates usage of data channel. The default `FreeRTOS` application contains data channel echo server. So it listens for messages from `Linux` and send them back. The following command can be used in Linux to test data channel:

```
# hello_dc
```

This application sends the string to the second partition and then receives the reply. Both strings are printed to the console.

6.2.4 Mango Frame Buffer

As mentioned above, LotOS framework provides Linux application to display graphical output from `FreeRTOS` partition. The following command can be used to display Mango frame buffer content:

```
# mango_fb
```

Below is the screenshot from this application:

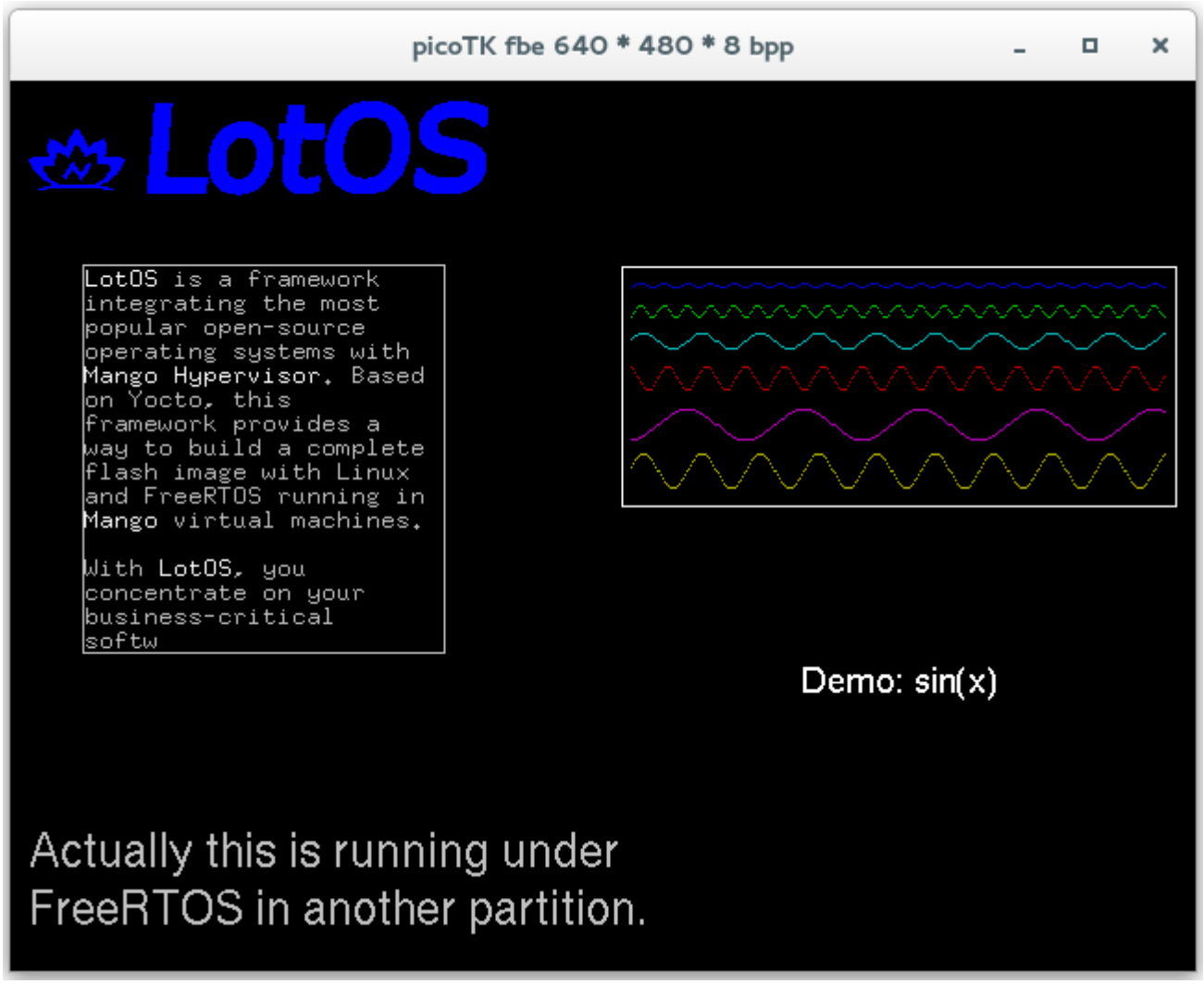

Illustration 3: Mango Frame Buffer

6.3 FreeRTOS Applications

`FreeRTOS` is built-in operating system, so all the applications are integrated to single image. The default LotOS `FreeRTOS` configuration contains several threads that implements simple demo applications:

- `time`: This `FreeRTOS` thread is printing partition uptime
- `tcp_server`: This thread starts TCP server and listens for connections from `Linux` partition on network port `3310`
- `watchdog`: This thread performs triggering of partition watchdog to prevent reset by timeout
- `dc`: This thread implements data channel echo server
- `mangofb`: This thread displays three graphical demos in loop: color scheme, partition running time and sine curves

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