From	HEIG-VD / REDS Institute
Date	February 7, 2021
Subject	SO3 Reference Guide

Version	2021_1
Authors	Daniel Rossier
Status	*Working*
Issue date	-

Version	Date	Changed by	Comments / Nature of change
2021_1	28.01.2021	Daniel Rossier	Initial document based on various sources

TABLE OF CONTENT

1 Acronyms and abbreviations

REDS	Reconfigurable Embedded Digital System (Institute)
SOO	Smart Object Oriented

2 Introduction

2.1 SO3

The SO3 (*Smart Object Oriented Operating*) system is a lightweight and full featured operating system devoted to ARM devices exclusively which can be used in academic and industrial environment.

SO3 can also be used as a *ME* (*Mobile Entity*) which is a migrating (complete) execution environment in the SOO framework.

SO3 is targeted to ARM processors only and is supplied in two versions: the *open source* version which is the kernel and main core subsystems, and a proprietary version which contains some extensions to be used in the SOO framework.

2.2 Target readers

This document is a technical reference manual of the SO3 operating system and is intended to developers.

3 The device model

SO3 has its own specific device model based on “*dev_t*” and “*devclass_t*” structure.

to be completed

4 Light and versatile graphics library (LVGL)

SO3 has the support for LVGL and framebuffer in the *vExpress* environment.

to be completed

5 Networking with lightweight IP (LWIP)

A network support is available in SO3 with the lightweight IP (LWIP) subsystem.

5.1 Network subsystem

The network subsystem is initialized in a post-init *initcall* during the boot process of SO3.

to be completed

6 User Guide

6.1 Initial setup

6.1.1 Pre-requisite

The *Linaro arm-linux-gnueabihf toolchain*[1] must be installed. Version 6.4.1 has been successfully tested, but more recent versions should be fine.

6.2 Files and directory organization

The SO3 tree is organized in two main parts: **kernel** and **user** space files. All kernel space related files are located in "***so3/***" while the user space files resides in "***usr/***".

The SO3 user space libc is based on the *musl library*[2]

6.3 Quick setup & early test

The following commands is helpful to have quick up-and-running environment with SO3, i.e. a shell running on top of the kernel in the emulated *vExpress* environment.

6.3.1 Building Qemu

The emulator must be built in *qemu/* using the command line described in README.so3 followed by invoking make (-j8 means parallel building on 8 cores):

```
cd qemu
./configure --target-list=arm-softmmu --disable-attr --disable-werror --disable-docs
make -j8
```

6.3.2 Compiling U-boot

U-boot is used as initial bootloader. It will be possible to start an ITB image file containing the kernel, the device tree and an initrd filesystem. In u-boot/ directory:

```
cd u-boot
make vexpress_defconfig
make -j8
```

6.3.3 Creating the virtual disk image

In *filesystem/* directory, create a virtual disk image with the following script:

[1] https://releases.linaro.org/components/toolchain/binaries/latest-7/arm-linux-gnueabihf
[2] https://musl.libc.org

```
cd filesystem
./create_img.sh vexpress
```

6.3.4 Compiling the user space

To build the user space applications, go to *usr/* directory and simply do make:

```
cd usr
make
```

6.3.5 Compiling the kernel space

The kernel has to be compiled in *so3/* after choosing a configuration:

```
cd so3
make vexpress_mmc_defconfig
make
```

At this point, all necessary components have been built. Now comes the phase of deployment in the virtual disk. This done by means of the deploy.sh script located at the root tree.

Currently, you should only use option b and u to deploy the ITB image as well as the user apps.

```
./deploy.sh -bu
```

6.3.6 Starting SO3

Simply invoking the script st as following:

```
./st
```

and you should run into the shell…

6.4 Default configurations

This section describes the default configurations of the SO3 kernel which are present in "*so3/configs/*".

6.4.1 *vExpress* platform

vexpress_ramfs_defconfig	Basic environment with a rootfs in RAM (*ramfs*)
vexpress_mmc_defconfig	Basic environment with a separate *rootfs* needed to be stored in an MMC partition

vexpress_thread_defconfig	Basic environment with no process support, hence no *user space*.
vexpress_nommu_defconfig	The MMU is disabled and only threads are available (no process/*user space*).
vexpress_net_defconfig	Environment with networking support (*lwip*)
vexpress_fb_defconfig	Environment with LVGL and framebuffer support
vexpress_full_defconfig	Complete environment with networking and framebuffer support

6.4.2 *Raspberry Pi 4* platform

Currently, there is only one default configuration file called *rpi4_defconfig* which has a basic environment, without networking and framebuffer support.

The drivers required for networking and graphics are not available yet.

7 Changelog

7.1 v2020.4.1 *(29 jan 2021)*

This release includes support for graphics (LVGL) and networking based on “*lwip*”.

- Added support for LVGL (https://lvgl.io) (early stage)
- Added support for *gettimeofday()* and *clock_gettime()*
- New scheduling policy based on dynamic priorities
- Added support for networking based on *lwip* (early stage)

7.2 v2020.4.0 *(28 jan 2021)*

Major changes have been made at the system and device level:

- Improvements of fdt parsing and file organization
- Revisited device model based on devclass with private structure
- Improvements of initcall processing, adding post_initcall
- Fixed numerous bugs in cache handling especially on Cortex-A72
- Simplified model of the MMU configuration at bootstrap
- Added support of rpi 4 Sense-hat extension
- Added post-init initcall for late initialization
- 64 KB Stack size of all threads
- Fixed bugs related to process loading and page list management
- Fixed bugs in frame table management

8 References

[1] ARM Ltd, ARM920T Technical Reference Manual.