Supporting LVGL GUI in i.MX RT uClinux BSP

Detailed Requirements and Design

rm6919-drad-1_2.doc

TABLE OF CONTENTS

1. Overview

This application note explains how to run the LVGL GUI in uClinux running on the i.MX RT devices. LVGL is the most popular free and open-source embedded graphics library to create powerful UIs for any MCU, MPU and display type. Refer to https://lvgl.io/ for detailed information on the LVGL GUI.

The Emcraft BSP includes a port and integration of the LVGL GUI, specifically for the i.MX RT devices running uClinux. The LVGL GUI support is integrated on top of the Linux `framebuffer` device driver (for the display) and touch screen devices driver, providing seamless integration with the Linux display and `/dev/input` input devices frameworks.

Full sources of the LVGL GUI, as well as pre-built LVGL binaries are provided.

2. Understanding Implementation

2.1. Understanding Integration and Build Framework

The LVGL sources, as well as some pre-built binaries, are integrated to the Emcraft uCLinux distribution in the `A2F` directory. The following are the key LVGL directories available in the distribution:

- `A2F/lvgl` - source code of the LVGL library itself (clone of the original https://github.com/lvgl/lvgl / tag v8.3.6)
- `A2F/lv_drivers`- source code of the LVGL drivers (clone of the original https://github.com/lvgl/lv_driver / tag v8.3.0)
- `A2F/lvgl_bins/IMXRT105X_NXPEVK/` - pre-built binaries for the NXP IMXRT1050-EVKB board
- `A2F/lvgl_bins/IMXRT105X_NXPEVK/liblvgl.so` - LVGL shared library, pre-built for the NXP IMXRT1050-EVKB board kit and display
- `A2F/lvgl_bins/IMXRT105X_NXPEVK/benchmark` - a standard LVGL demo application, pre-built for the NXP IMXRT1050-EVKB board kit and display. Other pre-built LVGL demos included in this directory are: `ebike`, `music`, `stress`, `widgets`.

The LVGL is built by the user in context of the standard `projects/rootfs/` project available in the Emcraft distribution. Specifically, the LVGL configuration and build files, along with some LVGL demo applications can be found in the `projects/rootfs/lvgl` directory. The key files in that directory are as follows:

- `lv_drv_conf.h` - header file with configuration parameters for the Linux `framebuffer` and touchscreen
- `lv_conf.h` - header file with configuration parameters for various LVGL options
- `main.c` - C source file containing a typical `main()` for an LVGL application
- `Makefile` - build file allowing to build it all together.

The core of the LVGL is built as a shared library and is available on the target board as `/usr/lib/liblvgl.so`. Any LVGL demos are built as separate Linux applications linked with the `liblvgl.so` library.

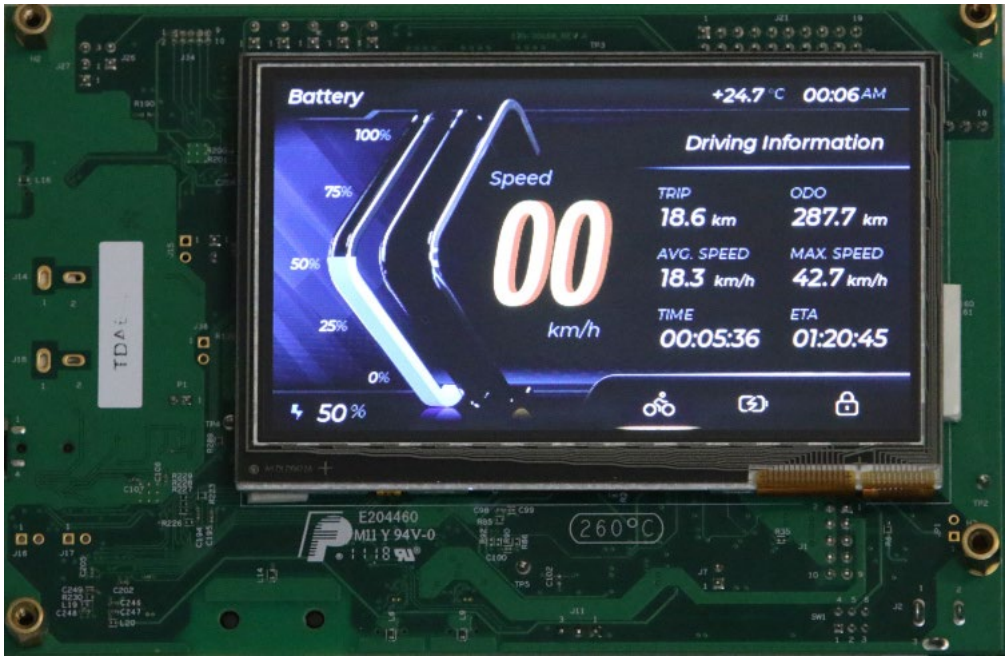
Even though, as noted above, the LVGL library as well as some standard LVGL demos prebuilt for a specific reference kit, are available from the Emcraft distribution, it is important to understand that in a typical situation one will have to rebuild the library and application binaries from scratch. The reason for that is that re-configuring the LVGL for a specific I/O configuration, as well as to port to a new display and / or a new input device, implies making custom changes to the `lv_drv_conf.h` and `lv_conf.h` configuration header files. Once an update has been made to those files, the LVGL library and any LVGL applications need to be rebuilt. One rebuild the library and applications by running the `Makefile` in the `projects/rootfs/lvgl` directory.

The Emcraft distribution includes 4 standard LVGL demos, which are built as part of the `projects/rootfs/` project and included to the target root filesystem:

- `benchmark_gui_demo`
- `music_gui_demo`
- `stress_gui_demo`
- `widgets_gui_demo`.

2.2. Implementing I/O interactions

In addition to the 4 standard LVGL demos, Emcraft provides one more to demonstrate interactions of the GUI with various IO interfaces. The GUI is based on the Futuristic Ebike example form the SquareLine Studio (https://squareline.io/).



The Original Futuristic Ebike project was modified by Emcraft in the SqureLine Studio to assign a `pin_clicked` C-function callback to the virtual keyboard button click events in the `Group Pin` component. Then the UI was then exported to the `projects/rootfs/lvgl/ebike_ui` directory. The `pin_clicked` callback was implemented in `projects/rootfs/lvgl/ebike_ui/ui_event.c` so that when the user enters a 4-digits PIN-code and presses the `v` button on the virtual keyboard the PIN-code in printed out to the Linux shell terminal from when the application is being run.

There is a separate thread implemented by Emcraft in the `projects/rootfs/lvgl/main.c` to monitor the state of the `USER` button (the `SW8` button on the back side of the NXP IMXRT1050-EVKB board). If the `USER` button is pressed the application assumes that the bike accelerates, and if the button is released the bike slows down. A dedicated LVGL timer was implemented to update the Speed, Trip, Odometer and other on-screen labels depending on the current button state. The user can see that the bike speed is increasing if the `USER` button is kept pressed, and decreasing if one releases the button.

The integration and build provisions for `ebike_gui_demo` are similar to the ones for the 4 standard LVGL demos described above. Once the demo is built, it is available on the target from the Emcraft standard `rootfs` project.

2.3. Understanding Interface to Linux I/O Frameworks

The LVGL implementation in the Emcraft BSP is configured to use the standard Linux `framebuffer` and input frameworks to support the display graphics and input for the LCD panel.

In case of the NXP i.MX RT1050-EVK board, this is the 4.3" LCD Panel RK043FN02H-CT.

Support for the `framebuffer` is provided by the `mxs-lcdif` driver (`drivers/video/fbdev/mxsfb.c`). The driver is configured to a 480x272-pixels resolution with a 16-bit color depth. Access to the display graphics is available via the `/dev/fb0` device node:

```
/ # ls -al /dev/fb0
crw-------    1 root     root       29,   0 Jan  1 00:00 /dev/fb0
/ # cat /sys/class/graphics/fb0/name
mxs-lcdif
/ #
```

Support for the touch screen is provided by the `ft5x0x_ts` driver (`drivers/input/touchscreen/ft5x46_ts.c`). The input is available via the `/dev/input/event0` device node:

```
/ # ls -al /dev/input/event0
crw-------    1 root     root       13,  64 Jan  1 00:00 /dev/input/event0
/ # cat /sys/class/input/event0/device/name
ft5x0x ts
/ #
```

Both the Linux frame buffer and the LVGL internal rendering buffers are allocated in the on-chip SDRAM. This supports a 25 FPS, as measured with `ebike` and other standard LVGL demos.

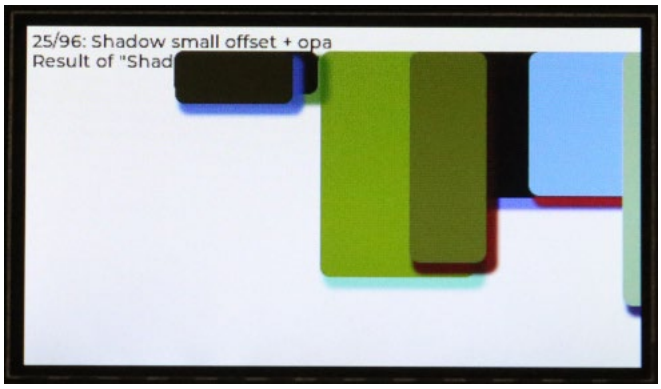
3. Running LVGL Demos

3.1. Running Standard LVGL Demos

Step through the following procedure to run the standard LVGL demos:

1. From the Linux shell, type the **benchmark_gui_demo** command to run the `benchmark` demo

```
/ # benchmark_gui_demo
```



Type **Ctrl-C** to finish the demo:

```
/ # benchmark gui demo
^C
/ #
```

2. Type the **music_gui_demo** command to run the `music` demo. Click to widgets icons on the touch panel to navigate the demo:

```
/ # music_gui_demo
```

Type **Ctrl-C** to finish the demo:

```
/ # music_gui_demo
^C
/ #
```

3. Type the **stress_gui_demo** command to run the `stress` demo

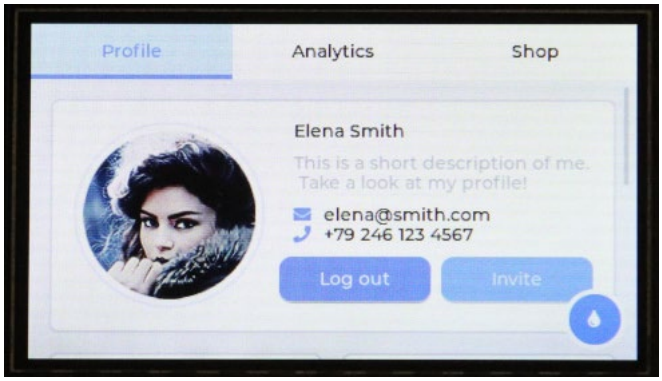
```
/ # stress_gui_demo
```

Type **Ctrl-C** to finish the demo:

```
/ # stress gui demo
^C
/ #
```

4. Type the **widgets_gui_demo** command to run the `widgets` demo. Click to widgets icons on the touch panel to navigate the demo:

```
/ # widgets_gui_demo
```



Type **Ctrl-C** to finish the demo:

```
/ # widgets_gui_demo
^C
/ #
```

3.2. Running Emcraft Ebike Demo

Step through the following procedure to run the Emcraft Ebike demo.

1. From the Linux shell, type the **ebike_gui_demo** command to run the `ebike` demo

```
/ # ebike_gui_demo
```

2. Click to the padlock icon in the bottom right corner of the screen to switch to the "Unlock Your Bike" group.

3. Click 4 any digits and then `v`. Make sure the correct PIN-code in printed out to the Linux shell terminal:

```
/ # ebike gui demo
entered pin: 4 7 1 2
```

4. Click to the bike icon to switch back to the "Driving Information" group.
5. Press and hold the `SW8` button which resides on the opposite side of the i.MXR1050-EVK board to LCD. Make sure that the Speed, Trip, Odometer and other values are increasing on the corresponding widgets on the LCD. If release the `SW8` button the Speed reading is decreasing:

4. Building LVGL

4.1. Obtaining LVGL Add-on

Emcraft support the LVGL port to the i.MX RT as a paid add-on.

Once you have purchased the add-on from Emcraft, you can obtain the LVGL patch from the following location:

https://www.emcraft.com/imxrtaddon/imxrt1050/lvgl

The page is protected as follows:

- Login: *CONTACT EMCRAFT FOR DETAILS*
- Password: *CONTACT EMCRAFT FOR DETAILS*

4.2. Building LVGL

Step through the following procedure to apply the LVGL add-on and build the LVGL binaries:

1. From the top of the Linux installation on the development host, go to the `projects` directory

```
$ cd projects/
```

2. Apply the patch:

```
$ patch -p1 < ../../projects-lvgl-gui.patch
```

3. Build the `roofts` project:

```
$ cd rootfs/
$ make
```

The built LVGL binaries will be included in the bootable target image. They can be run on the target as described in the previous sections.