Linux Driver Development for Embedded Processors

ST STM32MP1 Practical Labs Setup

Building a Linux embedded system for the ST STM32MP1 processor

The STM32MP1 microprocessor series is based on a heterogeneous single or dual Arm Cortex-A7 and Cortex-M4 cores architecture, strengthening its ability to support multiple and flexible applications, achieving the best performance and power figures at any time. The Cortex-A7 core provides access to open-source operating systems (Linux/Android) while the Cortex-M4 core leverages the STM32 MCU ecosystem.

You can check all the info related to this family at
https://www.st.com/en/microcontrollers-microprocessors/stm32mp1-series.html#overview

For the development of the labs the **STM32MP157C-DK2** Discovery kit will be used. The documentation of this board can be found at
https://www.st.com/en/evaluation-tools/stm32mp157c-dk2.html

Connect and set up hardware

To set up the STM32MP15 Discovery kit connections follow the steps indicated in the STM32 MPU wiki section located at
https://wiki.st.com/stm32mpu/wiki/Getting_started/STM32MP1_boards/STM32MP157x-DK2

Creating the structure for the STM32MPU embedded software distribution

The STM32MPU embedded software distribution for STM32 microprocessor platforms supports three software packages.

- The **Starter Package** to quickly and easily start with any STM32MPU microprocessor device. The Starter Package is generated from the Distribution Package.
- The **Developer Package** to add your own developments on top of the STM32MPU Embedded Software distribution, or to replace the Starter Package pre-built binaries. The Developer Package is generated from the Distribution Package.
- The **Distribution Package** to create your own Linux® distribution, your own Starter Package and your own Developer Package.

Create your <working directory> and assign a unique name to it (for example by including the release name).

```
PC:~$ mkdir STM32MP15-Ecosystem-v2.0.0
```

```
PC:~$ cd STM32MP15-Ecosystem-v2.0.0
```

Create the first-level directories that will host the software packages delivered through the STM32MPU embedded software distribution release note.

```
PC:~/STM32MP15-Ecosystem-v2.0.0$ mkdir Starter-Package
PC:~/STM32MP15-Ecosystem-v2.0.0$ mkdir Developer-Package
PC:~/STM32MP15-Ecosystem-v2.0.0$ mkdir Distribution-Package
```

Populate the target and boot the image

To populate the STM32MP15 Discovery kit with the Starter Package follow the steps indicated in the STM32 MPU wiki section located at
https://wiki.st.com/stm32mpu/wiki/Getting_started/STM32MP1_boards/STM32MP157x-DK2/Let%27s_start/Populate_the_target_and_boot_the_image

Installing the SDK for the developer package

To download the STM32MP1 Developer Package SDK for the STM32MP15-Ecosystem-v2.0.0 release follow the steps indicated in the STM32 MPU wiki section located at
https://wiki.st.com/stm32mpu/wiki/STM32MP1_Developer_Package

Follow the next steps to install the SDK:

1. Uncompress the tarball file to get the SDK installation script and make it executable.

```
PC:~$ mdir -p $HOME/STM32MPU_workspace/tmp
PC:~$ mkdir -p $SDK_ROOT/SDK
PC:~/STM32MPU_workspace/tmp$ tar xvf en.SDK-x86_64-stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24.tar.xz
PC:~/STM32MPU_workspace/tmp$ chmod +x stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sdk/st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-openstlinux-5.4-dunfell-mp1-20-06-24.sh
```

2. Add the following line to .bashrc.

```
PC:~$ echo "export SDK_ROOT=$HOME/STM32MP15-Ecosystem-v2.0.0/Developer-Package" >> $HOME/.bashrc
```

3. Install the SDK.

```
PC:~/STM32MPU_workspace/tmp$ ./stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sdk/st-image-weston-openstlinux-weston-stm32mp1-x86_64-toolchain-3.1-openstlinux-5.4-dunfell-mp1-20-06-24.sh -d $SDK_ROOT/SDK
ST OpenSTLinux - Weston - (A Yocto Project Based Distro) SDK installer version 3.1-openstlinux-5.4-dunfell-mp1-20-06-24
```

```
================================================================================
========================================
You are about to install the SDK to "/home/alberto/STM32MP15-Ecosystem-
v2.0.0/Developer-Package/SDK". Proceed [Y/n]? y
Extracting
SDK.....................................................................
.....................................................................
.....................................................done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

4. Each time you wish to use the SDK in a new shell session, you need to source the environment setup script.

```
PC:~$ source $SDK_ROOT/SDK/environment-setup-cortexa7t2hf-neon-vfpv4-ostl-
linux-gnueabi
```

Installing and compiling the Linux kernel for the developer package

To download the STM32MP1 Linux kernel for the STM32MP15-Ecosystem-v2.0.0 release follow the steps indicated in the STM32 MPU wiki section located at
https://wiki.st.com/stm32mpu/wiki/STM32MP1_Developer_Package

Follow the next steps to install and compile the Linux kernel:

1. Extract the kernel source code.

```
PC:~$ cd $SDK_ROOT
```

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package$ tar xvf en.SOURCES-kernel-
stm32mp1-openstlinux-5-4-dunfell-mp1-20-06-24.tar.xz
```

```
PC:~$ cd $SDK_ROOT/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-
ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0
```

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-
dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0$
tar xvf linux-5.4.31.tar.xz
```

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-
dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0$ cd
linux-5.4.31
```

2. To initialize a pad in GPIO mode with a bias (internal pull-up, pull-down..), it is needed to disable the strict mode of pinctrl. You have to change the strict variable of the struct pinmux_ops to false. You can find within the kernel sources the struct pinmux_ops structure; it is included in the /drivers/pinctrl/stm32/pinctrl-stm32.c file.

```
static const struct pinmux_ops stm32_pmx_ops = {
        .get_functions_count  = stm32_pmx_get_funcs_cnt,
        .get_function_name    = stm32_pmx_get_func_name,
```

```
        .get_function_groups  = stm32_pmx_get_func_groups,
        .set_mux              = stm32_pmx_set_mux,
        .gpio_set_direction   = stm32_pmx_gpio_set_direction,
        .strict               = false,
};
```

3. Prepare and configure kernel source code.

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ for p in `ls -1 ../*.patch`; do patch -p1 < $p; done
```

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ make multi_v7_defconfig fragment*.config
```

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ for f in `ls -1 ../fragment*.config`; do
scripts/kconfig/merge_config.sh -m -r .config $f; done
```

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ yes '' | make ARCH=arm oldconfig
```

4. Configure the following kernel settings that will be needed during the development of the drivers.

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ make ARCH=arm menuconfig
```

```
Device drivers >
       <*> Industrial I/O support  --->
              -*-   Enable buffer support within IIO
              -*-   Industrial I/O buffering based on kfifo
              <*>  Enable IIO configuration via configfs
              -*-   Enable triggered sampling support
              <*>   Enable software IIO device support
              <*>   Enable software triggers support
                     Triggers - standalone  --->
                            <*> High resolution timer trigger
                            <*> SYSFS trigger
```

```
 Device drivers >
       <*> Userspace I/O drivers  --->
              <*>   Userspace I/O platform driver with generic IRQ handling
```

```
Device drivers >
       Input device support  --->
              -*- Generic input layer (needed for keyboard, mouse, ...)
```

```
<*>   Polled input device skeleton
```

5. Compile kernel source code and kernel modules.

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ make -j4 ARCH=arm uImage vmlinux dtbs LOADADDR=0xC2000040

PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ make ARCH=arm modules

PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ mkdir -p $PWD/install_artifact/

PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ make ARCH=arm INSTALL_MOD_PATH="$PWD/install_artifact" modules_install
```

6. Boot the STM32MP1 target and open a new terminal on the host, for example "minicom". Set the following configuration: "115.2 kbaud, 8 data bits, 1 stop bit, no parity".

```
PC:~$ minicom -D /dev/ttyACM0
```

7. Connect Ethernet cable between host and eval board and verify the connection.

```
root@stm32mp1:~# ifconfig eth0 down

root@stm32mp1:~# ifconfig eth0 up

root@stm32mp1:~# ifconfig eth0 10.0.0.10

root@stm32mp1:~# ping 10.0.0.1
```

8. Deploy the compiled Linux kernel image and the kernel modules to the target STM32MP1 device.

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ scp arch/arm/boot/uImage root@10.0.0.10:/boot

PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ rm install_artifact/lib/modules/5.4.31/build install_artifact/lib/modules/5.4.31/source
```

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ find install_artifact/ -name "*.ko" | xargs $STRIP --strip-debug --remove-section=.comment --remove-section=.note --preserve-dates

PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ scp -r install_artifact/lib/modules/* root@10.0.0.10:/lib/modules
```

9. Re-generate the list of module dependencies (modules.dep) and the list of symbols provided by modules (modules.symbols), synchronize data on disk with memory and reboot the board.

```
root@stm32mp1:~# /sbin/depmod -a

root@stm32mp1:~# sync

root@stm32mp1:~# modinfo vivid

root@stm32mp1:~# reboot
```

Compile and deploy the Linux kernel drivers

Download the linux_5.4_stm32mp1_drivers.zip file from the github of the book and unzip it in the STM32MP15-Ecosystem-v2.0.0 folder of the Linux host:

```
~$ cd ~/STM32MP15-Ecosystem-v2.0.0/
```

Compile and deploy the drivers to the **STM32MP157C-DK2** Discovery kit:

```
~/STM32MP15-Ecosystem-v2.0.0/linux_5.4_stm32mp1_drivers$ make
~/STM32MP15-Ecosystem-v2.0.0/linux_5.4_stm32mp1_drivers$ make deploy

scp *.ko root@10.0.0.10:
adxl345_stm32mp1.ko                                   100%   12KB  12.3KB/s   00:00
adxl345_stm32mp1_iio.ko                               100%   12KB  12.4KB/s   00:00
hellokeys_stm32mp1.ko                                 100% 7024     6.9KB/s   00:00
helloworld_stm32mp1.ko                                100% 4008     3.9KB/s   00:00
helloworld_stm32mp1_char_driver.ko                    100% 6184     6.0KB/s   00:00
helloworld_stm32mp1_class_driver.ko                   100% 7724     7.5KB/s   00:00
helloworld_stm32mp1_with_parameters.ko                100% 4604     4.5KB/s   00:00
helloworld_stm32mp1_with_timing.ko                    100% 5688     5.6KB/s   00:00
i2c_stm32mp1_accel.ko                                 100% 7216     7.1KB/s   00:00
int_stm32mp1_key.ko                                   100% 7812     7.6KB/s   00:00
int_stm32mp1_key_wait.ko                              100%   10KB   9.9KB/s   00:00
io_stm32mp1_expander.ko                               100% 9664     9.4KB/s   00:00
keyled_stm32mp1_class.ko                              100%   16KB  16.2KB/s   00:00
ledRGB_stm32mp1_class_platform.ko                     100% 9524     9.3KB/s   00:00
ledRGB_stm32mp1_platform.ko                           100%   11KB  10.9KB/s   00:00
```

```
led_stm32mp1_UIO_platform.ko                    100% 6912     6.8KB/s   00:00
linkedlist_stm32mp1_platform.ko                 100% 9460     9.2KB/s   00:00
ltc2422_stm32mp1_dual.ko                        100% 7344     7.2KB/s   00:00
ltc2422_stm32mp1_trigger.ko                     100% 9840     9.6KB/s   00:00
ltc2607_stm32mp1_dual_device.ko                 100% 8056     7.9KB/s   00:00
ltc3206_stm32mp1_led_class.ko                   100%   11KB  11.1KB/s   00:00
misc_stm32mp1_driver.ko                         100% 5780     5.6KB/s   00:00
sdma_stm32mp1_m2m.ko                            100%   12KB  11.7KB/s   00:00
sdma_stm32mp1_mmap.ko                           100%   12KB  11.7KB/s   00:00
```

```
~/STM32MP15-Ecosystem-v2.0.0/linux_5.4_stm32mp1_drivers$
```

Verify that the drivers are now in the STM32MP157C-DK2 Discovery kit:

```
root@stm32mp1:~# ls
adxl345_stm32mp1.ko                    keyled_stm32mp1_class.ko
adxl345_stm32mp1_iio.ko                ledRGB_stm32mp1_class_platform.ko
hellokeys_stm32mp1.ko                  ledRGB_stm32mp1_platform.ko
helloworld_stm32mp1.ko                 led_stm32mp1_UIO_platform.ko
helloworld_stm32mp1_char_driver.ko     linkedlist_stm32mp1_platform.ko
helloworld_stm32mp1_class_driver.ko    ltc2422_stm32mp1_dual.ko
helloworld_stm32mp1_with_parameters.ko ltc2422_stm32mp1_trigger.ko
helloworld_stm32mp1_with_timing.ko     ltc2607_stm32mp1_dual_device.ko
i2c_stm32mp1_accel.ko                  ltc3206_stm32mp1_led_class.ko
int_stm32mp1_key.ko                    misc_stm32mp1_driver.ko
int_stm32mp1_key_wait.ko               sdma_stm32mp1_m2m.ko
io_stm32mp1_expander.ko                sdma_stm32mp1_mmap.ko
```

```
root@stm32mp1:~#
```

The stm32mp15xx-dkx.dtsi and stm32mp15-pinctrl.dtsi files with all the needed modifications to run the drivers are stored in the device_tree folder inside the linux_5.4_stm32mp1_drivers.zip file. During the development of the drivers you will modify these device tree files, then build and copy them to the STM32MP1 board.

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ make dtbs
```

```
PC:~/STM32MP15-Ecosystem-v2.0.0/Developer-Package/stm32mp1-openstlinux-5.4-dunfell-mp1-20-06-24/sources/arm-ostl-linux-gnueabi/linux-stm32mp-5.4.31-r0/linux-5.4.31$ scp arch/arm/boot/dts/stm32mp157c-dk2.dtb root@10.0.0.10:/boot
```

Hardware and device tree descriptions for the ST STM32MP1 labs

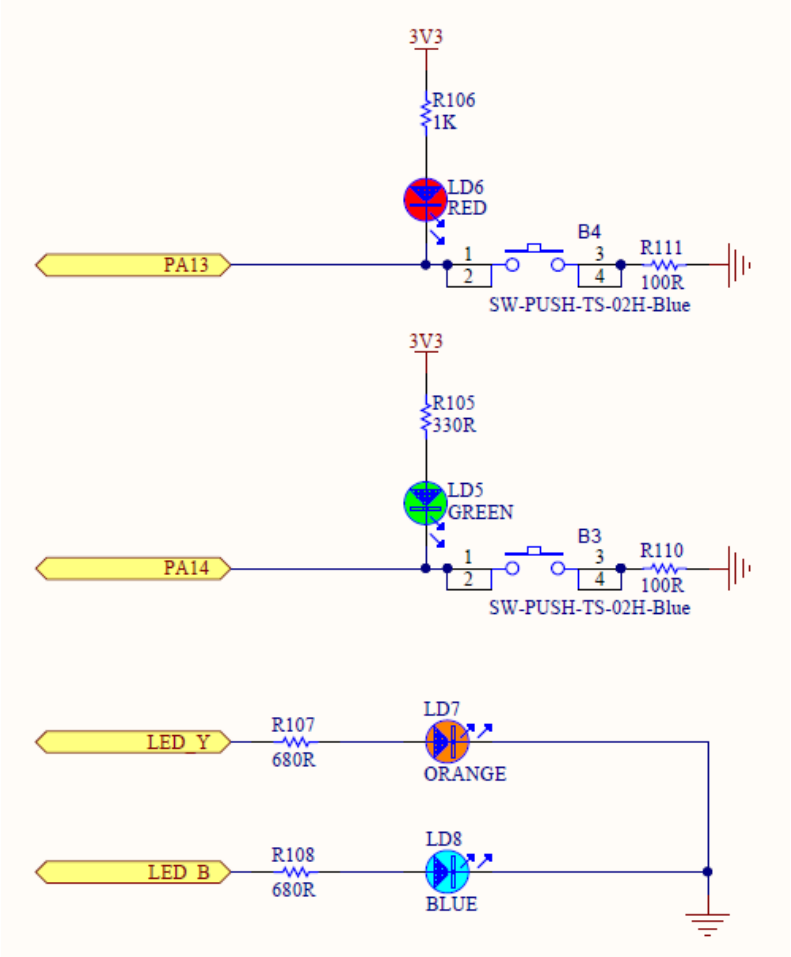
In the next sections it will be described the different hardware and device tree configurations for the labs where external hardware connected to the processor is controlled by the drivers. The

schematic of the STM32MP157C-DK2 Discovery kit is included inside the linux_5.4_stm32mp1_drivers.zip file that can be downloaded from the github of the book.

LAB 5.2, 5.3 and 5.4 hardware and device tree descriptions

During the development of these drivers you will use the LD6 RED, LD5 GREEN and LD8 BLUE leds included in the STM32MP157C-DK2 Discovery kit. Go to the pag.13 of the schematic to see them. Each LED is individually controlled by a processor pin programmed as GPIO output. The pins are PA13, PA14, and PD11. The PD11 pin is used by the "gpio-leds" driver, therefore you'll have to disable it in the stm32mp15xx-dkx.dtsi file to avoid conflicts with your developed drivers.

```
/*led {
           compatible = "gpio-leds";
           blue {
                  label = "heartbeat";
                  gpios = <&gpiod 11 GPIO_ACTIVE_HIGH>;
                  linux,default-trigger = "heartbeat";
                  default-state = "off";
           };
};*/
```



This is the device tree node that should be included in the stm32mp15xx-dkx.dtsi file to run the driver for the LAB 5.2:

```
ledRGB {
            compatible = "arrow,RGBleds";
            clocks = <&rcc GPIOA>,
                     <&rcc GPIOD>;

            clock-names = "GPIOA", "GPIOD";

            red {
                   label = "ledred";
```

```
            };

            green {
                    label = "ledgreen";
            };

            blue {
                    label = "ledblue";
            };
    };
```

This is the device tree node that should be included in the stm32mp15xx-dkx.dtsi file to run the driver for the LAB 5.3:

```
ledclassRGB {
            compatible = "arrow,RGBclassleds";
            reg = <0x50002000 0x400>,
                  <0x50005000 0x400>;

            clocks = <&rcc GPIOA>,
                     <&rcc GPIOD>;

            clock-names = "GPIOA", "GPIOD";

            red {
                    label = "ledred";
            };

            green {
                    label = "ledgreen";
            };

            blue {
                    label = "ledblue";
            };
    };
```

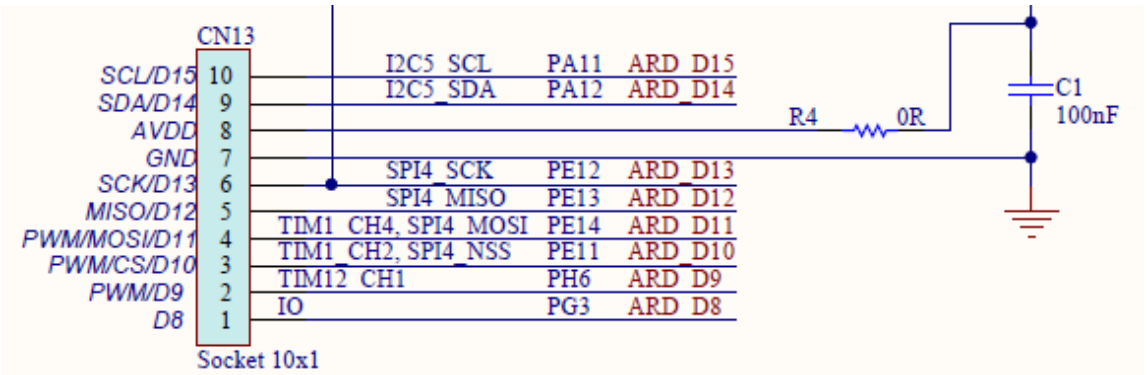
This is the device tree node that should be included in the stm32mp15xx-dkx.dtsi file to run the driver for the LAB 5.4:

```
UIO {
   compatible = "arrow,UIO";
   reg = <0x50002000 0x1000>;
   clocks = <&rcc GPIOA>;
};
```

LAB 6.1 hardware and device tree descriptions

In this lab the driver will able to manage several PCF8574 I/O expander devices connected to the I2C bus. You can use one of the multiples boards based on this device to develop this lab, for example, the next one https://www.waveshare.com/pcf8574-io-expansion-board.htm.

You will take the I2C5 bus from the CN13 connector of the STM32MP157C-DK2 Discovery kit. Go to the pag.10 of the STM32MP157C-DK2 schematic to see the connector.



You can take the 3V3 and GND signals from the CN16 connector of the STM32MP157C-DK2 board. Go to the pag.10 of the STM32MP157C-DK2 schematic to see the connector.

This is the device tree node that should be included in the stm32mp15xx-dkx.dtsi file to run the driver for the LAB 6.1:

```
&i2c5 {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&i2c5_pins_a>;
    pinctrl-1 = <&i2c5_pins_sleep_a>;
    i2c-scl-rising-time-ns = <185>;
    i2c-scl-falling-time-ns = <20>;
    clock-frequency = <400000>;
    /delete-property/dmas;
    /delete-property/dma-names;
    status = "okay";

    ioexp@38 {
            compatible = "arrow,ioexp";
            reg = <0x38>;
    };
```

```
};
```

LAB 6.2 hardware and device tree descriptions

To test this driver you will use the DC749A - Demo Board (http://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/dc749a.html).

In this lab you will use the I2C5 pins of the STM32MP157C-DK2 CN13 connector to connect to the DC749A - Demo Board. Connect the pin 9 (I2C5_SDA) of the CN13 connector to the pin 7 (SDA) of the DC749A J1 connector and the pin 10 (I2C5_SCL) of the CN13 connector to the pin 4 (SCL) of the DC749A J1 connector. Connect the 3.3V pin from the STM32MP157C-DK2 CN16 connector to the DC749A Vin J2 pin and to the DC749A J20 DVCC connector. Connect the pin 1 (PG3 pad) of the CN13 connector to the pin 6 (ENRGB/S) of the DC749A J1 connector. Do not forget to connect GND between the two boards.

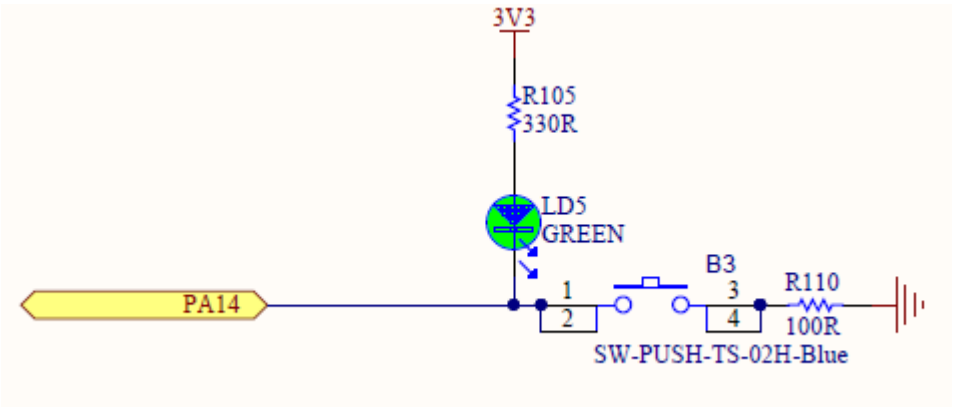
This is the device tree node that should be included in the stm32mp15xx-dkx.dtsi file to run the driver for the LAB 6.2:

```
&i2c5 {
   pinctrl-names = "default", "sleep";
   pinctrl-0 = <&i2c5_pins_a>;
   pinctrl-1 = <&i2c5_pins_sleep_a>;
   i2c-scl-rising-time-ns = <185>;
   i2c-scl-falling-time-ns = <20>;
   clock-frequency = <400000>;
   /delete-property/dmas;
   /delete-property/dma-names;
   status = "okay";

   ltc3206: ltc3206@1b {
          compatible = "arrow,ltc3206";
          reg = <0x1b>;
          gpios = <&gpiog 3 GPIO_ACTIVE_LOW>;

          led1r {
                 label = "red";
          };

          led1b {
                 label = "blue";
          };

          led1g {
                 label = "green";
```

```
            };

            ledmain {
                    label = "main";
            };

            ledsub {
                    label = "sub";
            };
    };
};
```

LAB 7.1 and 7.2 hardware and device tree descriptions

In these two labs you will use the "USER" button (B3) of the STM32MP157C-DK2 board. The button is connected to PA14 pin. The pin will be programmed as an input generating an interrupt. You will also have to ensure the mechanical key is debounced. Open the STM32MP157C-DK2 schematic and find the button B3 in pag.13.



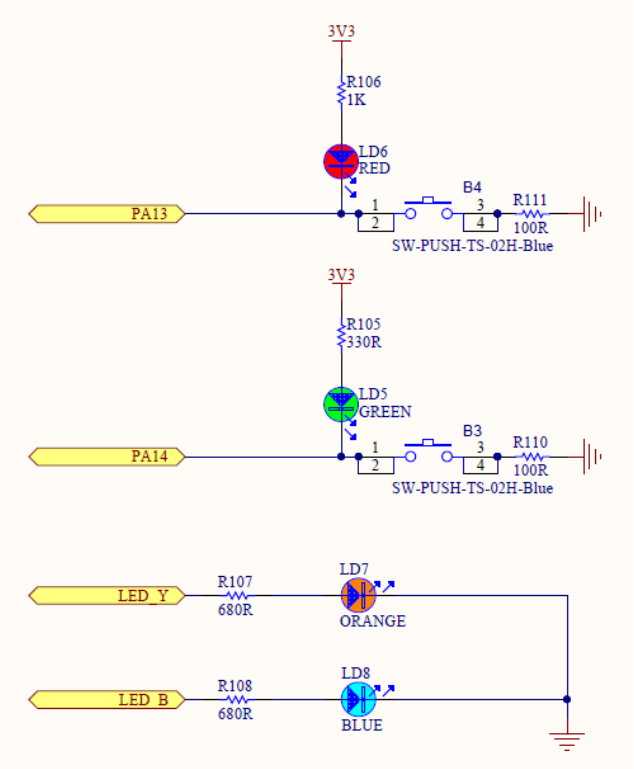
These are the device tree nodes that should be included in the stm32mp15xx-dkx.dtsi file to run the drivers for the LAB 7.1 and the LAB 7.2:

```
    int_key {
                compatible = "arrow,intkey";
                pinctrl-names = "default";
                pinctrl-0 = <&key_pins>;
                label = "PB_USER";
                gpios = <&gpioa 14 GPIO_ACTIVE_LOW>;
                interrupt-parent = <&gpioa>;
                interrupts = <14  IRQ_TYPE_EDGE_FALLING>;
```

```
};

int_key_wait {
        compatible = "arrow,intkeywait";
        pinctrl-names = "default";
        pinctrl-0 = <&key_pins>;
        label = "PB_USER";
        gpios = <&gpioa 14 GPIO_ACTIVE_LOW>;
        interrupt-parent = <&gpioa>;
        interrupts = <14  IRQ_TYPE_EDGE_FALLING>;
};
```

LAB 7.3 hardware and device tree descriptions

In this lab you will use the LD7 ORANGE and the LD8 BLUE leds included in the STM32MP157C-DK2 Discovery kit. Go to the pag.13 of the STM32MP157C-DK2 schematic to see them. Each LED is individually controlled by a processor pin programmed as GPIO output. The pins are PH7 and PD11. Currently the PD11 pin is used by by the “gpio-leds” driver, therefore you’ll have to disable it in the stm32mp15xx-dkx.dtsi file. In this lab you will also use the buttons B4 and B3. The button B4 is connected to PA13 pin and the button B3 is connected to the PA14 pin. Both pins will be programmed as an input generating an interrupt. You will also have to ensure the mechanical key is debounced. Open the STM32MP157C-DK2 schematic and find the B4 and B3 buttons in pag.13.



This is the device tree node that should be included in the stm32mp15xx-dkx.dtsi file to run the driver for the LAB 7.3:

```
ledpwm {
            compatible = "arrow,ledpwm";

            pinctrl-names = "default";
            pinctrl-0 = <&keyleds_pins>;

            bp1 {
                    label = "KEY_1";
                    gpios = <&gpioa 13 GPIO_ACTIVE_LOW>; // B4:USER2
                    trigger = "falling";
            };

            bp2 {
                    label = "KEY_2";
                    gpios = <&gpioa 14 GPIO_ACTIVE_LOW>; //B3:USER1
```

```
                trigger = "falling";
        };

        ledorange {
                label = "led";
                colour = "orange";
                gpios = <&gpioh 7 GPIO_ACTIVE_LOW>;
        };

        ledblue {
                label = "led";
                colour = "blue";
                gpios = <&gpiod 11 GPIO_ACTIVE_LOW>;
        };

};
```

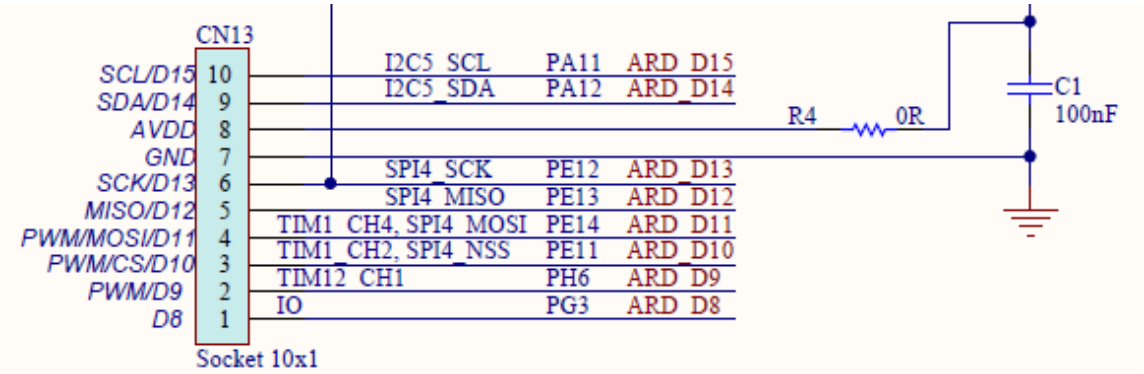
This is the device tree node that should be included in the stm32mp15-pinctrl.dtsi file to run the driver for the LAB 7.3:

```
keyleds_pins: keyleds-0 {
                        pins1 {
                                pinmux = <STM32_PINMUX('H', 7, GPIO)>,
                                         <STM32_PINMUX('D', 11, GPIO)>;
                                drive-push-pull;
                                bias-pull-down;
                        };

                        pins2 {
                                pinmux = <STM32_PINMUX('A', 13, GPIO)>;
                                drive-push-pull;
                                bias-pull-up;
                        };

                        pins3 {
                                pinmux = <STM32_PINMUX('A', 14, GPIO)>;
                                drive-push-pull;
                                bias-pull-up;
                        };
};
```

LAB 10.1,10.2 and 12.1 hardware and device tree descriptions

In these labs you will control an accelerometer board connected to the I2C and SPI buses of the processor. You will use the ADXL345 Accel click mikroBUS™ accessory board to develop the drivers; you will access to the schematic of the board at http://www.mikroe.com/click/accel/.

For the LAB 10.1 you will connect the accelerometer board to the I2C5 pins of the STM32MP157C-DK2 CN13 connector. For the LAB 10.2 and the LAB 12.1 you will connect the accelerometer board to the SPI4 pins of the CN13 connector.



The pin 1 of the CN13 connector (PG3 pad) will be programmed as an input generating an interrupt for the LAB 10.2 and the LAB 12.1.

This is the device tree node that should be included in the stm32mp15xx-dkx.dtsi file to run the driver for the LAB 10.1:

```
&i2c5 {
   pinctrl-names = "default", "sleep";
   pinctrl-0 = <&i2c5_pins_a>;
   pinctrl-1 = <&i2c5_pins_sleep_a>;
   i2c-scl-rising-time-ns = <185>;
   i2c-scl-falling-time-ns = <20>;
   clock-frequency = <400000>;
   /delete-property/dmas;
   /delete-property/dma-names;
   status = "okay";

   adxl345@1c {
           compatible = "arrow,adxl345";
           reg = <0x1d>;
   };
};
```

This is the device tree node that should be included in the stm32mp15xx-dkx.dtsi file to run the drivers for the LAB 10.2 and the LAB 12.1:

```
&spi4 {
   pinctrl-names = "default", "sleep";
```

```
        pinctrl-0 = <&spi4_pins_a>;
        pinctrl-1 = <&spi4_sleep_pins_a>;
        cs-gpios = <&gpioe 11 0>;
        status = "okay";

        Accel: ADXL345@0 {
                compatible = "arrow,adxl345";
                pinctrl-names ="default";
                pinctrl-0 = <&accel_pins>;
                spi-max-frequency = <5000000>;
                spi-cpol;
                spi-cpha;
                reg = <0>;
                int-gpios = <&gpiog 3 GPIO_ACTIVE_LOW>;
                interrupt-parent = <&gpiog>;
                interrupts = <3 IRQ_TYPE_LEVEL_HIGH>;
        };
};
```

This is the device tree node that should be included in the stm32mp15-pinctrl.dtsi file to run the drivers for the LAB 10.2 and the LAB 12.1:

```
accel_pins: accel-0 {
          pins {
                 pinmux = <STM32_PINMUX('G', 3, GPIO)>;
                 drive-push-pull;
                 bias-pull-down;
          };
};
```

LAB 11.1 hardware and device tree descriptions

In this lab you will control the Analog Devices LTC2607 internal DACs individually or both DACA + DACB in a simultaneous mode. You will use the DC934A evaluation board; you can download the schematics at
https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/dc934a.html

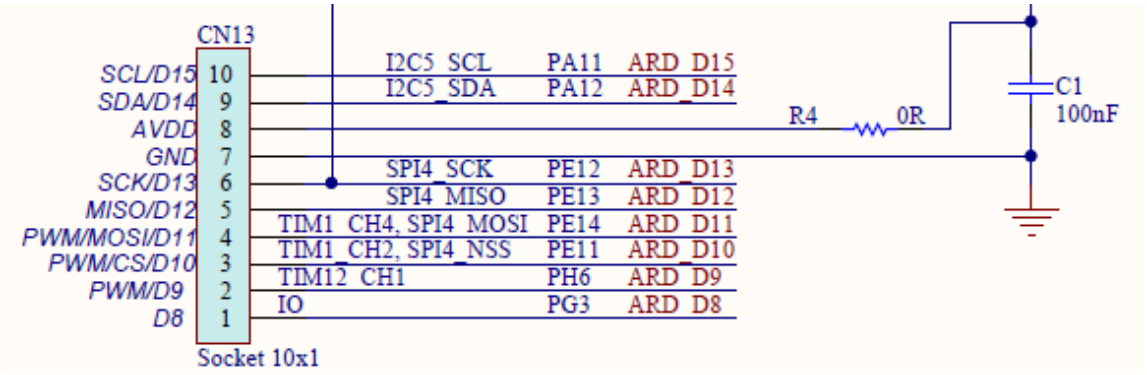
For this LAB 11.1 you will connect the I2C5 pins of the STM32MP157C-DK2 CN13 connector to the SDA and SCL pins of the LTC2607 DC934A evaluation board. You are going to power the LTC2607 with the 3.3V pin of the STM32MP157C-DK2 CN16 connector, connecting it to V+, pin 1 of the DC934A´s connector J1. Also connect GND between the DC934A (i.e., pin 3 of connector J1) and GND pin of the STM32MP157C-DK2 Discovery kit.

This is the device tree node that should be included in the stm32mp15xx-dkx.dtsi file to run the driver for the LAB 11.1:

```
&i2c5 {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&i2c5_pins_a>;
    pinctrl-1 = <&i2c5_pins_sleep_a>;
    i2c-scl-rising-time-ns = <185>;
    i2c-scl-falling-time-ns = <20>;
    clock-frequency = <400000>;
    /delete-property/dmas;
    /delete-property/dma-names;
    status = "okay";

    ltc2607@72 {
            compatible = "arrow,ltc2607";
            reg = <0x72>;
    };

    ltc2607@73 {
            compatible = "arrow,ltc2607";
            reg = <0x73>;
    };
};
```

LAB 11.2, LAB 11.3 and LAB 11.4 hardware and device tree descriptions

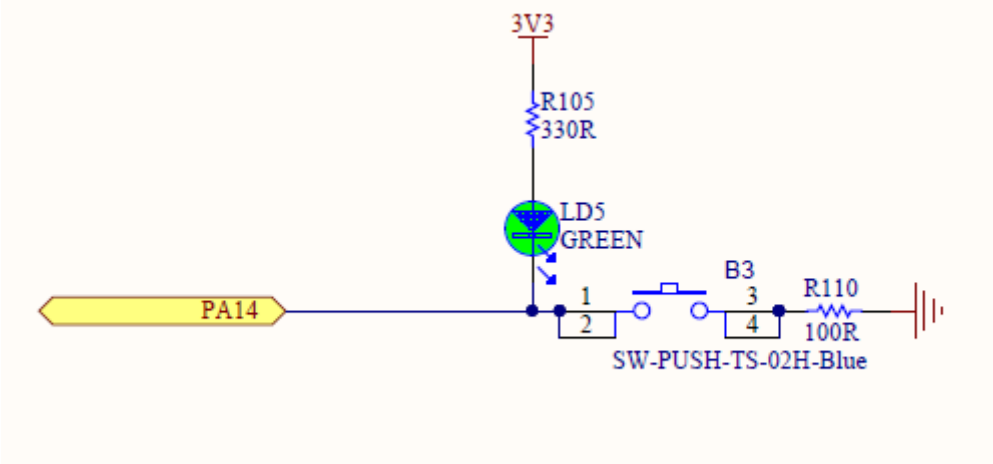
In these three labs you will reuse the hardware description of the LAB 11.1 and will use the SPI4 pins of the STM32MP157C-DK2 CN13 connector to connect to the LTC2422 dual ADC SPI device that is included in the DC934A board.



Open the STM32MP157C-DK2 schematic to see the CN13 connector and look for the SPI pins. The CS, SCK and MISO (Master In, Slave Out) signals will be used. The MOSI (Master out, Slave in) signal won´t be needed, as you are only going to receive data from the LTC2422 device. Connect the next CN13 SPI4 pins to the LTC2422 SPI ones obtained from the DC934A board J1 connector:

- Connect the STM32MP157C-DK2 **SPI4_NSS** (CS) to LTC2422 **CS**
- Connect the STM32MP157C-DK2 **SPI4_SCK** (SCK) to LTC2422 **SCK**
- Connect the STM32MP157C-DK2 **SPI4_MISO** (MISO) to LTC2422 **MISO**

In the lab 11.4 you will also use the "USER" button (B3). The button is connected to the PA14 pin. The pin will be programmed as an input generating an interrupt.



These are the device tree nodes that should be included in the stm32mp15xx-dkx.dtsi file to run the drivers for the LAB 11.2, LAB 11.3 and LAB 11.4:

```
&spi4 {
    pinctrl-names = "default", "sleep";
    pinctrl-0 = <&spi4_pins_a>;
    pinctrl-1 = <&spi4_sleep_pins_a>;
    cs-gpios = <&gpioe 11 0>;
    status = "okay";

    /* spidev@0 {
            compatible = "spidev";
            spi-max-frequency = <2000000>;
            reg = <0>;
```

```
    }; */

    ADC: ltc2422@0 {
            compatible = "arrow,ltc2422";
            spi-max-frequency = <2000000>;
            reg = <0>;
    };

    ADC: ltc2422@0 {
            compatible = "arrow,ltc2422";
            spi-max-frequency = <2000000>;
            reg = <0>;
            pinctrl-names ="default";
            pinctrl-0 = <&key_pins>;
            int-gpios = <&gpioa 14 GPIO_ACTIVE_LOW>;
    };
};
```

This is the device tree node that should be included in the stm32mp15-pinctrl.dtsi file to run the driver for the LAB 11.4:

```
key_pins: key-0 {
            pins {
                   pinmux = <STM32_PINMUX('A', 14, GPIO)>;
                            drive-push-pull;
                            bias-pull-up;
                 };
};
```

The kernel 5.4 modules developed for the STM32MP157C-DK2 board are included in the linux_5.4_STM32MP1_drivers.zip file and can be downloaded from the GitHub repository at https://github.com/ALIBERA/linux_book_2nd_edition.