Linux Driver Development for Embedded Processors

Microchip SAMA5D27-SOM1 Practical Labs Setup

Building a Linux Embedded System for the Microchip SAMA5D27-SOM1 System-On-Module

The SAMA5D2-SOM1 is a small single-sided System-On-Module (SOM) based on the high-performance 32-bit Arm® Cortex®-A5 processor-based MPU SAMA5D27 running up to 500 MHz. The SAMA5D27 SOM1 is built on a common set of proven Microchip components to reduce time to market by simplifying hardware design and software development. The SOM also simplifies design rules of the main application board, reducing overall PCB complexity and cost. You can check all the info related to this device at
https://www.microchip.com/wwwproducts/en/ATSAMA5D27-SOM1

For the development of the labs the **SAMA5D27-SOM1-EK1** evaluation kit will be used. The user guide and design files of this board can be found at
https://www.microchip.com/developmenttools/ProductDetails/atsama5d27-som1-ek1

Introduction

To get the Yocto Project expected behavior in a Linux Host Machine, the packages and utilities described below must be installed. An important consideration is the hard disk space required in the host machine. For example, when building on a machine running Ubuntu, the minimum hard disk space required is about 50 GB for the X11 backend. It is recommended that at least 120 GB is provided, which is enough to compile all backends together.

Host Packages

A Yocto Project build requires that some packages be installed for the build that are documented under the Yocto Project. Please make sure your host PC is running Ubuntu 16.04 64-bit and install the following packages:

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
build-essential chrpath socat cpio python python3 python3-pip python3-pexpect \
xz-utils debianutils iputils-ping libsdl1.2-dev xterm

$ sudo apt-get install autoconf libtool libglib2.0-dev libarchive-dev python-git \
sed cvs subversion coreutils texi2html docbook-utils python-pysqlite2 \
help2man make gcc g++ desktop-file-utils libgl1-mesa-dev libglu1-mesa-dev \
mercurial automake groff curl lzop asciidoc u-boot-tools dos2unix mtd-utils pv \
libncurses5 libncurses5-dev libncursesw5-dev libelf-dev zlib1g-dev
```

Yocto Project Setup and Image Building

The Yocto Project is a powerful building environment. It is build on top of several components including the famous OpenEmbedded build framework for embedded Linux. Poky is the reference system for building a whole embedded Linux distribution. The support for SAMA5 ARM® Cortex-A5-based MPUs has been added permitting to offer a root file system with a wide range of applications.

The support for the SAMA5 family is included in a particular Yocto layer: meta-atmel. The source for this layer are hosted on Linux4SAM GitHub account at https://github.com/linux4sam/meta-atmel

You can see below the step by step build procedure:

Create a directory:

```
~$ mkdir sama5d2_sumo
~$ cd sama5d2_sumo/
```

Clone yocto/poky git repository with the proper branch ready:

```
~/sama5d2_sumo$ git clone git://git.yoctoproject.org/poky -b sumo
```

Clone meta-openembedded git repository with the proper branch ready

```
~/sama5d2_sumo$ git clone git://git.openembedded.org/meta-openembedded -b sumo
```

Clone meta-qt5 git repository with the proper branch ready:

```
~/sama5d2_sumo$ git clone git://code.qt.io/yocto/meta-qt5.git
~/sama5d2_sumo$ cd meta-qt5/
~/sama5d2_sumo/meta-qt5$ git checkout v5.9.6
~/sama5d2_sumo$ cd ..
~/sama5d2_sumo$
```

Clone meta-atmel layer with the proper branch ready:

```
~/sama5d2_sumo$ git clone git://github.com/linux4sam/meta-atmel.git -b sumo
```

Enter the poky directory to configure the build system and start the build process

```
~/sama5d2_sumo$ cd poky
~/sama5d2_sumo/poky$
```

Initialize build directory:

```
~/sama5d2_sumo/poky$ source oe-init-build-env
```

Add meta-atmel layer to bblayer configuration file:

```
~/sama5d2_sumo/poky/build$ gedit conf/bblayers.conf
```

```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BSPDIR := "${@os.path.abspath(os.path.dirname(d.getVar('FILE', True)) +
'/../../..')}"

BBLAYERS ?= " \
  ${BSPDIR}/poky/meta \
  ${BSPDIR}/poky/meta-poky \
  ${BSPDIR}/poky/meta-yocto-bsp \
  ${BSPDIR}/meta-atmel \
  ${BSPDIR}/meta-openembedded/meta-oe \
  ${BSPDIR}/meta-openembedded/meta-networking \
  ${BSPDIR}/meta-openembedded/meta-python \
  ${BSPDIR}/meta-openembedded/meta-ruby \
  ${BSPDIR}/meta-openembedded/meta-multimedia \
  ${BSPDIR}/meta-qt5 \
  "

BBLAYERS_NON_REMOVABLE ?= " \
  ${BSPDIR}/poky/meta \
  ${BSPDIR}/poky/meta-poky \
  "
```

Edit local.conf to specify the machine, location of source archived, package type (rpm, deb or ipk). Set MACHINE name to “sama5d27-som1-ek-sd”:

```
~/sama5d2_sumo/poky/build$ gedit conf/local.conf

[...]
MACHINE ??= "sama5d27-som1-ek-sd"
[...]
DL_DIR ?= "your_download_directory_path"
[...]
PACKAGE_CLASSES ?= "package_ipk"
[...]
USER_CLASSES ?= "buildstats image-mklibs"
```

To get better performance, use the "poky-atmel" distribution by also adding that line:

```
DISTRO = "poky-atmel"
```

Build Atmel demo image. We found that additional local.conf changes are needed for our QT demo image. You can add these two lines at the end of the file:

```
~/sama5d2_sumo/poky/build$ gedit conf/local.conf

[...]
LICENSE_FLAGS_WHITELIST += "commercial"
SYSVINIT_ENABLED_GETTYS = ""

~/sama5d2_sumo/poky/build$ bitbake atmel-qt5-demo-image
```

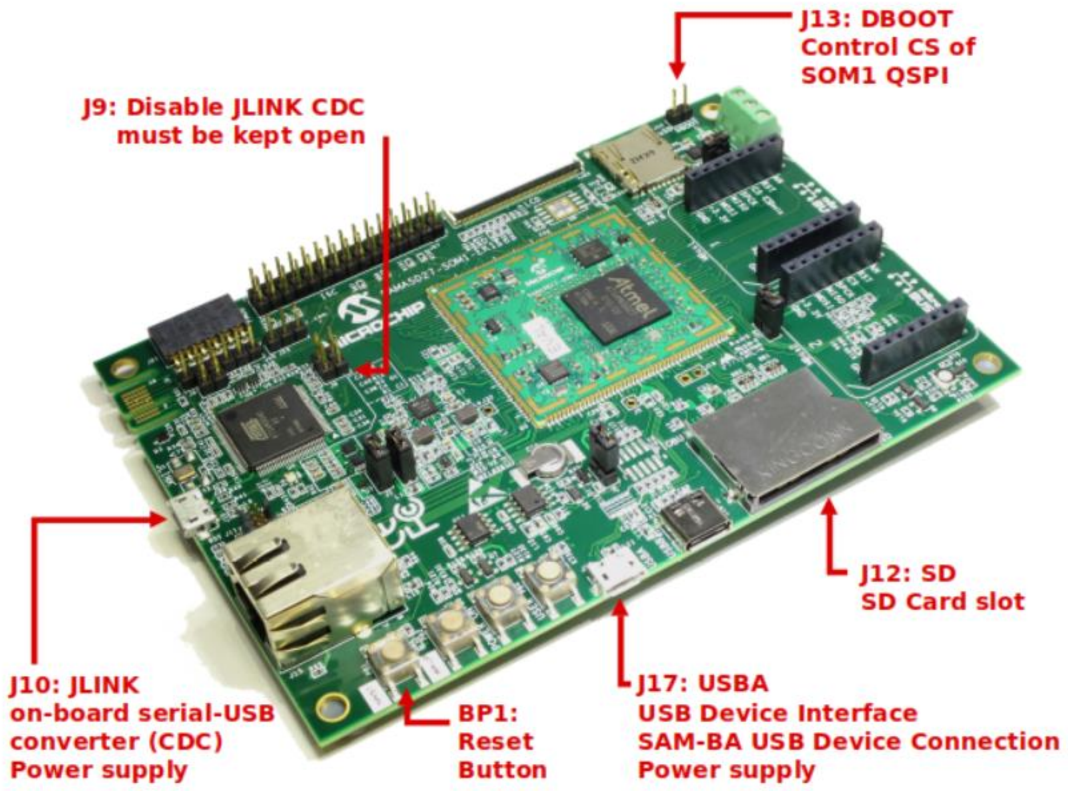
Enhancements are added on top of the official v4.14 Linux kernel tag where most of the Microchip SoC features are already supported. Note as well that Microchip re-integrate each and every stable kernel release on top of this Long Term Support (LTS) kernel revision. This means that each v4.14.x version is merged in Microchip branch. You will use the **linux4sam_6.0 tag** with integration of stable kernel updates up to **v4.14.73.** You can chek further info at https://www.at91.com/linux4sam/bin/view/Linux4SAM/LinuxKernel

You are going to create a SD demo image compiled from **tag linux4sam_6.0**. Go to https://www.at91.com/linux4sam/bin/view/Linux4SAM/DemoArchive6_0 and download linux4sam-poky-sama5d27_som1_ek_video-6.0.img.bz2 yocto demo image.

To write the compressed image on the SD card, you will have to download and install **Etcher**. This tool, which is an Open Source software, is useful since it allows to get a compressed image as input. More information and extra help is available on the Etcher website at https://etcher.io/ Follow the steps of the Create a SD card with the demo section at https://www.at91.com/linux4sam/bin/view/Linux4SAM/Sama5d27Som1EKMainPage

Connect and Set Up Hardware

Connect the SAMA5D27-SOM1-EK1 board to your host computer via the J10 JLink connector with the micro USB cable and open a serial console. Through this console, you can access the SAMA5D27-SOM1-EK1 running an embedded Linux distribution.

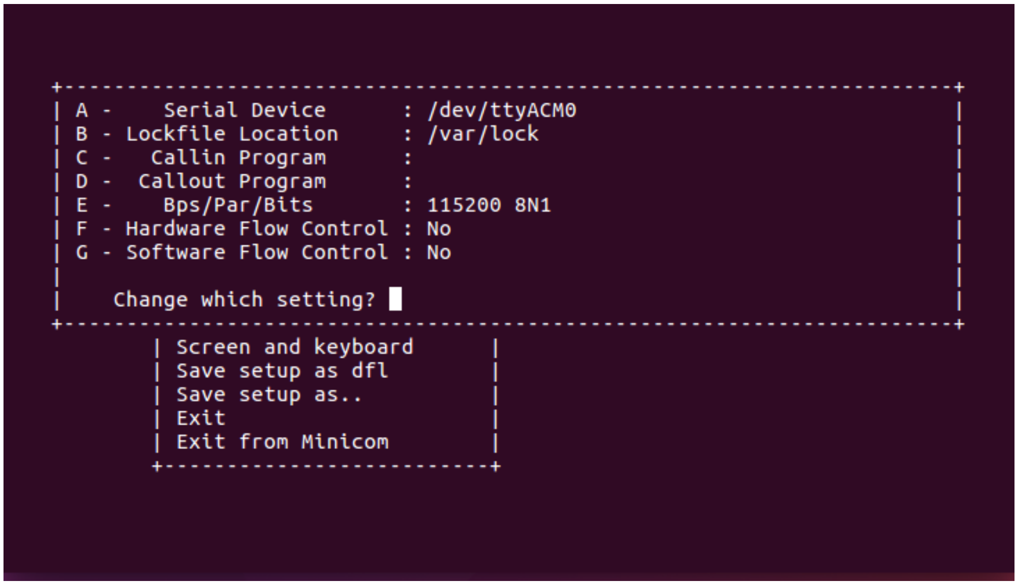


Launch a terminal on the host Linux PC by clicking on the Terminal icon. Type dmesg at the command prompt.

```
~$ dmesg
usb 3-1: New USB device found, idVendor=1366, idProduct=0106
usb 3-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
usb 3-1: Product: J-Link
usb 3-1: Manufacturer: SEGGER
usb 3-1: SerialNumber: 000483029459
cdc_acm 3-1:1.0: ttyACM0: USB ACM device
usbcore: registered new interface driver cdc_acm
cdc_acm: USB Abstract Control Model driver for USB modems and ISDN
adapters
```

In the log message you can see that the new USB device is found and installed as **ttyACM0**.

Launch and configure **minicom** in your host to see the booting of the system. Set the following configuration: “115.2 kbaud, 8 data bits, 1 stop bit, no parity”.



```
+-----------------------------------------------------------------------+
| A -    Serial Device      : /dev/ttyACM0                              |
| B - Lockfile Location     : /var/lock                                 |
| C -   Callin Program      :                                           |
| D -  Callout Program      :                                           |
| E -    Bps/Par/Bits       : 115200 8N1                                |
| F - Hardware Flow Control : No                                        |
| G - Software Flow Control : No                                        |
|                                                                       |
|    Change which setting?                                              |
+-----------------------------------------------------------------------+
        | Screen and keyboard      |
        | Save setup as dfl        |
        | Save setup as..          |
        | Exit                     |
        | Exit from Minicom        |
        +--------------------------+
```

Insert the SD card into the J12 SD slot on the SAMA5D27-SOM1-EK1 and reset the board (PB1/NRST). You should see Linux boot messages on the console.

Establish a network connection between the Linux host and the SAMA5D27-SOM1-EK target so that drivers built on the Linux Host PC can be easily transferred to the target for installation. Connect the Ethernet cable between your host PC and your SAMA5D27-SOM1-EK board. Set up the Linux host PC's IP Address to **10.0.0.1**. On the SAMA5D27-SOM1-EK board (the target), configure the eth0 interface with IP address **10.0.0.10** by editing the /etc/network/interfaces file. Open the file using vi editor:

```
root@sama5d27-som1-ek-sd:~# vi /etc/network/interfaces
```

Start editing the file by hitting **"i"**. You may find there is already an entry for eth0 as shown below. If not, configure eth0 with the following lines:

```
auto eth0
iface eth0 inet static
address 10.0.0.20
netmask 255.255.255.0
```

Exit the editing mode by hitting the ESC button. Save and exit the file by typing **":wq"**. If any time you want to exit the file without saving the changes you made, hit ESC followed by: **":q!".**

Then run following commands to make sure that the eth0 Ethernet interface is properly configured:

```
root@sama5d27-som1-ek-sd:~# ifdown eth0
root@sama5d27-som1-ek-sd:~# ifup eth0
root@sama5d27-som1-ek-sd:~# ifconfig
```

Now test that you can communicate with your Linux host machine from your SAMA5D27_SOM1_EK board. Exit the ping command by hitting **ctrl-C**.

```
root@sama5d27-som1-ek-sd:~# ping 10.0.0.1
```

A network connection is now established between the Linux host and the SAMA5D27_SOM1_EK target.

Working Outside of Yocto

In this section we will describe the instructions to build the Yocto SDK for the ATSAMA5D27-SOM1 device:

```
~/sama5d2_sumo/poky/build$ bitbake -c populate_sdk atmel-qt5-demo-image
~/sama5d2_sumo/poky/build$ cd tmp/deploy/sdk/
~/sama5d2_sumo/poky/build/tmp/deploy/sdk$ ls
~/sama5d2_sumo/poky/build/tmp/deploy/sdk$ ./poky-atmel-glibc-x86_64-atmel-qt5-
demo-image-cortexa5hf-neon-toolchain-2.5.1.sh

Enter target directory for SDK (default: /opt/poky-atmel/2.5.1):
You are about to install the SDK to "/opt/poky-atmel/2.5.1". Proceed[Y/n]?
Extracting
SDK...................................................................
......done
Setting it up...done
SDK has been successfully set up and is ready to be used.
```

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

```
 ~$ . /opt/poky-atmel/2.5.1/environment-setup-cortexa5hf-neon-poky-linux-gnueabi
```

Building the Linux Kernel

In this section we will describe the instructions to build the Linux kernel for the ATSAMA5D27-SOM1 device::

Download the kernel sources from the Microchip git:

```
~$ git clone git://github.com/linux4sam/linux-at91.git
~$ cd linux-at91/
~/linux-at91$ git branch -r
~/linux-at91$ git checkout origin/linux-4.14-at91 -b linux-4.14-at91
```

Compile the kernel image, modules, and all the device tree files:

```
~/linux-at91$ make mrproper
~/linux-at91$ make ARCH=arm sama5_defconfig
~/linux-at91$ make ARCH=arm menuconfig
```

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

```
Device drivers >
   [*] SPI support  --->
          <*>   User mode SPI device driver support

Device drivers >
   [*] LED Support  --->
           <*>   LED Class Support
           -*-   LED Trigger support  --->
                        <*>   LED Timer Trigger
                        <*>   LED Heartbeat Trigger

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
          <*>   Userspace platform driver with generic irq and dynamic memory

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

Save the configuration and exit from menuconfig.

Source the toolchain script and compile kernel, device tree files and modules in a single step:

```
~/linux-at91$ source /opt/poky-atmel/2.5.1/environment-setup-cortexa5hf-neon-poky-
linux-gnueabi
~/linux-at91$ make -j4
```

Once the Linux kernel, dtb files and modules have been compiled they must be installed. In the case of the kernel image this can be installed by copying the zImage file to the location where it will be read from. There are two partitions on the SD card. The smaller one, formatted as a FAT file system, holds boot.bin, u-boot.bin, uboot.env, and the FIT image **sama5d27_som1_ek.itb**. The larger partition, formatted as ETX4, contains a root file system. The SD card as whole is represented among the devices as /dev/mmcblk0, its partitions as /dev/mmcblk0p1 and /dev/mmcblk0p2 respectively. The second partition is mounted as a root on the target. The boot partition is not mounted. We must mount it first before we can access it.

Starting from **U-boot 2018.07** released in **Linux4SAM6.0**, we can use the feature of patching the Device Tree Blob (DTB) with additional Device Tree Overlays (DTBO). A device tree overlay is a file that can be used at runtime (by the bootloader in our case) to dynamically modify the device tree, adding nodes to the tree and making changes to properties in the existing tree.
You can easily download DT Overlay source code from Linux4SAM GitHub DT Overlays repository. Clone the Linux4sam GitHub DT Overlay repository:

```
~$ git clone git://github.com/linux4sam/dt-overlay-at91.git
   Cloning into 'dt-overlay-at91'...
   remote: Enumerating objects: 760, done.
   remote: Counting objects: 100% (760/760), done.
   remote: Compressing objects: 100% (428/428), done.
   remote: Total 760 (delta 340), reused 735 (delta 315), pack-reused 0
   Receiving objects: 100% (760/760), 369.55 KiB | 1.23 MiB/s, done.
   Resolving deltas: 100% (340/340), done.
```

```
~$ cd dt-overlay-at91/
```

The source code has been taken from the master branch which is pointing to the latest branch we use.

Now, build the DT-Overlay and the FIT image. The FIT image is a placeholder that has the zImage and the base Device Tree, plus additional overlays that can be selected at boot time. To build the overlays and the FIT image with overlays for a board make sure the following steps are done:

- the environment variables ARCH and CROSS_COMPILE are set correctly
- (optional) the environment variable KERNEL_DIR points to Linux kernel and the kernel was already built for the board. This is needed because the DT Overlay repository uses the Device Tree Compiler (dtc) from the kernel source tree. By default, KERNEL_DIR is set to a linux directory that would be under the parent directory in the directory tree: ../linux
- (optional) the environment variable KERNEL_BUILD_DIR that points to where the Linux kernel binary and Device Tree blob, resulting of your compilation of the kernel,

are located. By default, KERNEL_BUILD_DIR is set to the same directory as KERNEL_DIR. It shouldn't be changed if you have the habit of compiling your kernel within the Linux source tree

- mkimage is installed on the development machine
- the Device Tree Compiler from Linux kernel is in the PATH environment variable

```
~/dt-overlay-at91$ source /opt/poky-atmel/2.5.1/environment-setup-cortexa5hf-neon-
poky-linux-gnueabi
```

```
~/dt-overlay-at91$ gedit Makefile
#KERNEL_DIR?=../linux
KERNEL_DIR?=/home/alberto/linux-at91
```

```
~/dt-overlay-at91$ make sama5d27_som1_ek_dtbos
~/dt-overlay-at91$ make sama5d27_som1_ek.itb
```

Now, on the target mount /dev/mmcblk0p1 on /mnt. The /mnt is an empty directory that is frequently used to mount external file systems.

```
root@sama5d27-som1-ek-sd:~# ls -lF /mnt
total 0
root@sama5d27-som1-ek-sd:~# mount /dev/mmcblk0p1 /mnt
root@sama5d27-som1-ek-sd:~# ls -lF /mnt/
total 4278
-rwxr-xr-x 1 root root   19141 Oct 12  2018 BOOT.BIN
-rwxr-xr-x 1 root root 3901236 Oct 12  2018 sama5d27_som1_ek.itb
-rwxr-xr-x 1 root root  442415 Oct 12  2018 u-boot.bin
-rwxr-xr-x 1 root root   16384 Oct 12  2018 uboot.env
root@sama5d27-som1-ek-sd:~# umount /mnt
root@sama5d27-som1-ek-sd:~# reboot
```

Copy the FIT from the host to target's /mnt directory. We use **scp** (secure copy) program. We copy the file as root, because we logged in to the target as root. Enter the commands on the host terminal. When a secure copy program runs for the first time, it will notice it is communicating with an unknown machine. It will report the fact. Accept it. The host PC will add the target to the list of known machines.

```
~/dt-overlay-at91$ scp sama5d27_som1_ek.itb root@10.0.0.10:/mnt
```

For the SAMA5Dx based boards, Microchip implements a common dtsi files, which define peripherals present on a board, a SoM and a SoC. Each variant of SAMA5Dx-EK or Xplained board has its own dts file. For example, the at91-sama5d27_som1_ek.dts, which defines a plain board, includes the at91-sama5d27_som1_ek_common.dtsi that enables the board's common peripherals. It includes the at91-sama5d27_som1.dtsi file that defines the peripherals present on the SoM. This file in turn includes the sama5d2.dtsi file, which defines all peripherals present on the SoC's, but sets their status to "disabled" except for the three crypto devices. The top dts files

provide the final touch enabling extra devices and creating through it different flavors of the board. We will be using this technique to make a device tree for the board with a click board plugged in. The device tree source files (.dts and .dtsi) for ARM core based devices in Linux kernel are stored together with Linux kernel sources in linux-at91/arch/arm/boot/dts directory. They can be seen also on Linux4SAM github at
https://github.com/linux4sam/linux-at91/tree/master/arch/arm/boot/dts

The at91-sama5d27_som1_ek.dts is the Device Tree file for the SAMA5D27-SOM1-EK board. Every time you add a new node or modify something in this DT source file you have to execute the next steps to make the new dtb file available on your target device:

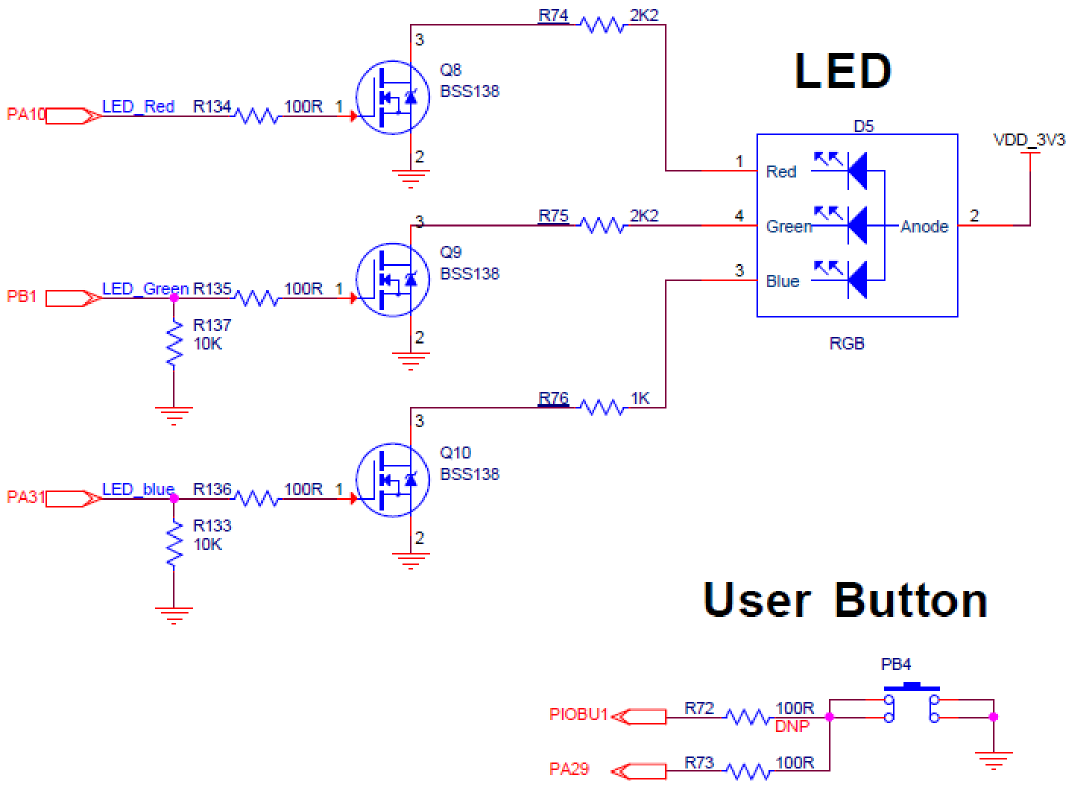
```
~/linux-at91$ make dtbs
~/dt-overlay-at91$ make sama5d27_som1_ek.itb
root@sama5d27-som1-ek-sd:~# mount /dev/mmcblk0p1 /mnt
~/dt-overlay-at91$ scp sama5d27_som1_ek.itb root@10.0.0.10:/mnt
root@sama5d27-som1-ek-sd:~# umount /mnt
root@sama5d27-som1-ek-sd:~# reboot
```

Hardware Descriptions for the Microchip SAMA5D27-SOM1 System-On-Module Labs

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

LAB 5.2, 5.3 and 5.4 Hardware Descriptions

The SAMA5D27-SOM1-EK board integrates a RGB LED. Go to the pag.8 of the schematic to see it. The tricolor LED (D5) contains three RGB LEDs in the same housing. Each LED is individually controlled by a processor pin programmed as GPIO output. The pins are PA10, PA31, PB1. Currently these pins are used by the “leds” driver, therefore you’ll have to disable it.

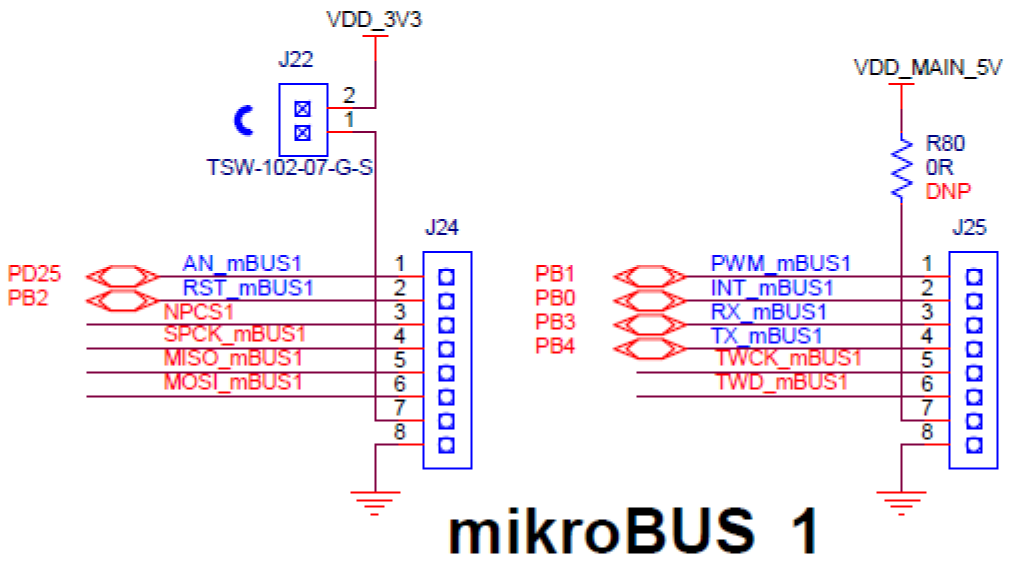


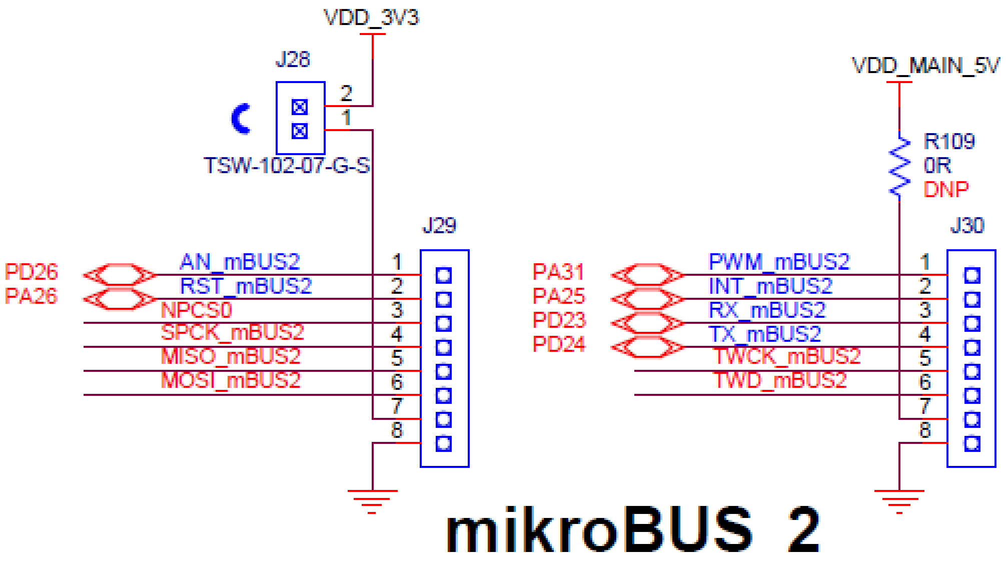
LAB 6.1 Hardware Description

In this lab the driver will manage several PCF8574 I/O expander devices connected to the I2C bus. You can use one 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.

The baseboard features a wide range of peripherals, as well as an user interface and expansion options, including two mikroBUS™ click interface headers. The mikroBUS standard defines the main board sockets and add-on boards (a.k.a. "click boards") used for interfacing microprocessors with integrated modules with proprietary pin configuration and silkscreen markings. The pinout consists of three groups of communication pins (SPI, UART and TWI), four additional pins (PWM, interrupt, analog input and reset) and two power groups (+3.3V and

GND on the left, and 5V and GND on the right 1x8 header). Go to the pag.9 of the schematic to see both connectors.





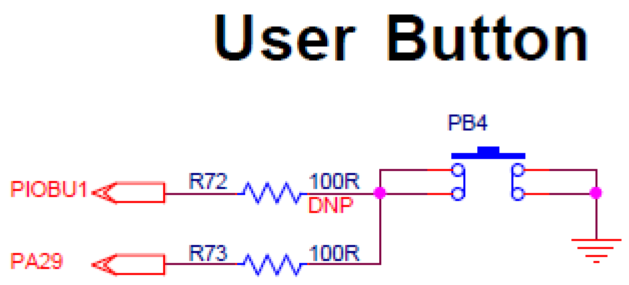
You can use two PCF8574 boards, one connected to the TWCK_mBUS1, and TWD_mBUS1 I2C signals and the other one to the the TWCK_mBUS2, and TWD_mBUS2 I2C signals . Don´t forget to connect 3V3 and GND signals between each PCF8574 board and its respective mikroBUS connector.

LAB 6.2 Hardware Description

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 I2C pins of the SAMA5D27-SOM1-EK mikroBUS 1 to connect to the DC749A - Demo Board. Go to the pag.9 of the schematic to see the mikroBUS 1 connector and look for the TWCK_mBUS1 and TWD_mBUS1 pins. Connect the mikroBUS TWD pin to the pin 7 (SDA) of the DC749A J1 connector and mikroBUS TWCK pin to the pin 4 (SCL) of the DC749A J1 connector. Connect the 3.3V mikroBUS pin to the DC749A Vin J2 pin and to the DC749A J20 DVCC connector. Connect the mikroBUS AN_mBUS1 pin (PD25 pad) to the pin 6 (ENRGB/S) of the DC749A J1 connector. Do not forget to connect GND between the two boards.

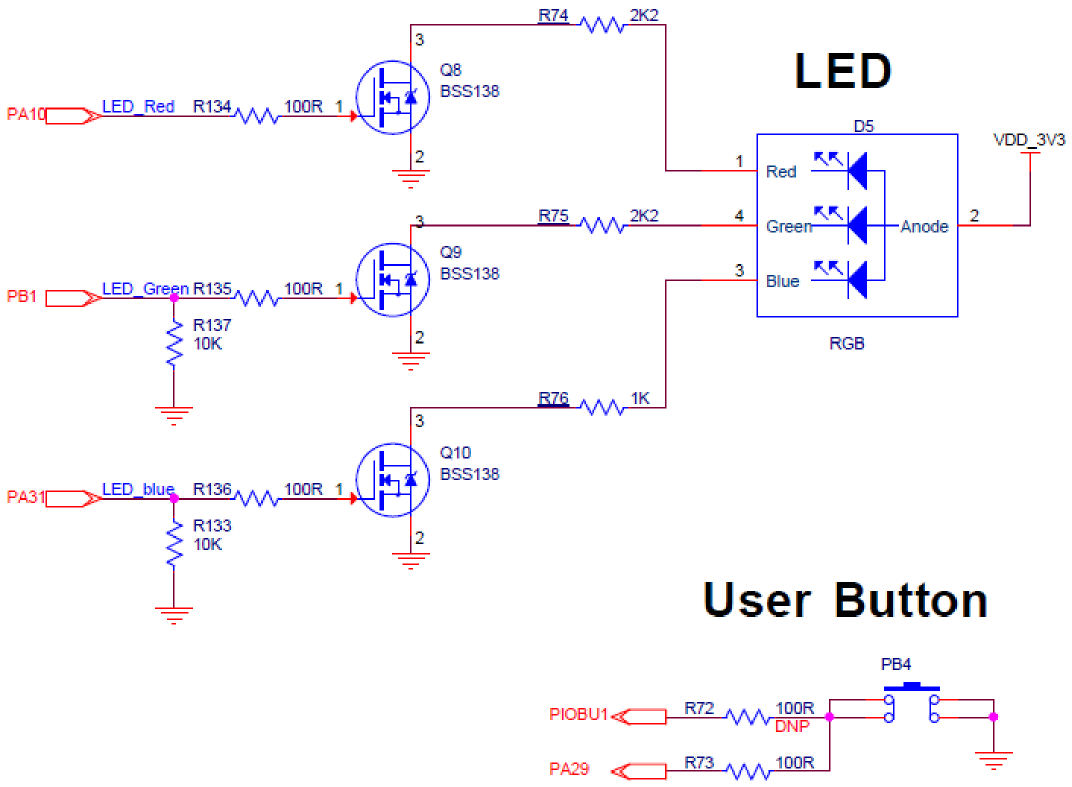
LAB 7.1 and 7.2 Hardware Descriptions

In these two labs you will use the "USER" button (PB4). The button is connected to PA29 pin. The pin is used by "gpio_keys" driver, so it must be disabled. The pin will be programmed as an input generating an interrupt. You will also have to ensure the mechanical key is debounced. Open the schematic and find the button PB4 in pag.8.



LAB 7.3 Hardware Description

The SAMA5D27-SOM1-EK board integrates a RGB LED. Go to the pag.8 of the schematic to see it. The tricolor LED (D5) contains three RGB LEDs in the same housing. Each LED is individually controlled by a processor pin programmed as GPIO output. The pins are PA10, PA31, PB1. Currently these pins are used by the "leds" driver, therefore you'll have to disable it. In this lab you will also use the "USER" button (PB4). The button is connected to PA29 pin. The pin is used by "gpio_keys" driver, so it must be disabled. The pin will be programmed as an input generating an interrupt. You will also have to ensure the mechanical key is debounced.
Open the schematic and find the PB4 button in pag.8.



A second interrupt will be generated using the button of the **MikroElektronika Button R click** board. See the board at https://www.mikroe.com/button-r-click. You can download the schematic from that link or from the GitHub repository of this book. Connect the board to the SAMA5D27-SOM1-EK mikroBUS 1 connector. The pin INT_mBUS1 (pad PB0) of the mikroBUS 1 will be programmed as an input generating a second interrupt.

LAB 10.1,10.2 and 12.1 Hardware 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 acces to the schematic of the board at http://www.mikroe.com/click/accel/. Connect the board to the SAMA5D27-SOM1-EK mikroBUS 2 connector. The INT_mBUS2 pin

(PA25 pad) of the mikroBUS 2 will be programmed as an input generating an interrupt in the LAB 10.2 and LAB 12.1

LAB 11.1 Hardware Description

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
You will connect the TWCK_mBUS2 and TWD_mBUS2 pins of the SAMA5D27-SOM1-EK mikroBUS 2 connector to the SDA and SCL pins of the LTC2607 DC934A evaluation board. You are going to power the LTC2607 with the microBUS 3.3V pin 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 mikroBUS 2 GND pin of the SAMA5D27-SOM1-EK board.

LAB 11.2 and LAB 11.3 Hardware Descriptions

In these two labs you will reuse the hardware description of the LAB 11.1 and will use the SPI pins of the SAMA5D27-SOM1-EK mikroBUS 2 connector to connect to the LTC2422 dual ADC SPI device that is included in the DC934A board.

Open the SAMA5D27-SOM1-EK schematic to see the mikroBUS 2 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 microBUS 2 pins to the LTC2422 SPI ones obtained from the DC934A board J1 connector:

- Connect the SAMA5D27-SOM1 **NPCS0** (CS) to LTC2422 **CS**
- Connect the SAMA5D27-SOM1 **SPCK_mBUS2** (SCK) to LTC2422 **SCK**
- Connect the SAMA5D27-SOM1 **MISO_mBUS2** (MISO) to LTC2422 **MISO**

In the lab 11.3 you will also use the “USER” button (PB4). The button is connected to PA29 pin. The pin is used by “gpio_keys” driver, so it must be disabled. The pin will be programmed as an input generating an interrupt.

The kernel 4.14 modules developed for SAMA5D27-SOM1 are included in the linux_4.14_sama5d27-SOM_drivers.zip file and can be accessed through the GitHub repository at https://github.com/ALIBERA/linux_book_2nd_edition.