Internship Report

NXP IMX8MM-EVK

Evaluation kit

- NXP LPDDR4 iMX8MM-EVK 2018 version
- EVK

Getting started with the EVK

Installing Linux

- Download the Linux binary demo.
- Download UUU.

```
~$ chmod a+x UUU

~$ sudo ./uuu <linux_images_MX8MMEVK.zip>
```

- **NOTE:** Latest Linux distro will not run on this board.
- The error message returned will booting Linux:

> *"Can't find PMIC:PCA9450"*

- PMIC for our board is:

> ***ROHM BD71847MWV***

- To use the newest BSP, we need to do the PMIC porting for our board.
- Refer to boot switch setup in the startup guide.
- Linux User Guide.
- Reference Manual.

PMIC Porting

- Clone repository and checkout the branch for the Linux version.

```
~$ git clone https://source.codeaurora.org/external/imx/linux-imx.git

~$ git fetch --all

~$ git checkout -b FAH-imx origin/lf-5.15.y
```

- Disable **PMIC** in **imx8mm-evk.dts**:

```
&pmic_nxp {
status= "disabled";
};
```

- Disable **PCIe** in **imx8mm-evk.dtsi**:

```
&pcie0{
status = "disabled";
};
```

- Disable **wdog** driver in **imx8mm-evk.dtsi**:

```
&wdog1 {
status = "disabled";
};
```

- Generate dtb:

```
~$ export ARCH=arm64

~$ make dtbs
```

- **NOTE:** Do not forget to plug the board with a 12V power supply. The board will shut down if powered with a USB from a PC.

Load Device Tree File

- copy **.dtb** to TFTP server:

```
sudo cp <DIR>arch/arm64/boot/dts/freescale/imx8mm-evk.dtb ~/tftp/<filename.dtb>
```

- Edit U-boot environment:

```
setenv loadfdt "setenv autoload no; dhcp; setenv serverip <IP_Address>; tftp 0x43000000 <filename.dtb>"

setenv fdtfile <filename.dtb>

saveenv
```

MCUXpresso SDK

- Download SDK.
- We used MCUXpresso version **2.11.1**.
- **NOTE:** Initialization of pins in **pinmux.c** of a few examples is missing (fixed in new version 2.12.0).

```
void BOARD_InitBootPins(void){

        BOARD_InitPins();//missing function added

}
```

- PinMux Tool.
- Set environment:

```
export ARMGCC_DIR=<DIRECTORY>/gcc-arm-11.2-2022.02-x86_64-arm-none-eabi
```

- To build an example run:

```
<DIR>/armgcc/build_.sh
```

- Running the application on Cortex-4:
  - Setup TFTP on PC:

> *~$ sudo apt update*
>
> *~$ sudo apt install tftpd-hpa*
>
> *~$ sudo systemctl status tftpd-hpa*
>
> *Change username and directory:*
>
> *~$ sudo nano /etc/default/tftpd-hpa*
>
> *~$ sudo mkdir /tftp*
>
> *~$ sudo chown tftp:tftp /tftp*
>
> *~$ sudo systemctl restart tftpd-hpa*
>
> *~$ sudo systemctl status tftpd-hpa*

- Copy .bin to TFTP/:

> *~$ sudo cp release/app.bin ~<DIR>/tftp/*
>
> *~$ sudo chown tftp:tftp ~<DIR>/tftp/app.bin*

- Open terminal on PC with the following config:

> *~$ sudo picocom -b 112500 /dev/ttyUSB**

- Load u-boot on IMX8MM-EVK.
- To run TCM builds from u-boot:

> *=> setenv <env_name> "dhcp; setenv serverip <ip_addr>; tftp 0x48000000 <application>.bin; cp.b 0x48000000 0x7e0000 0x20000; bootaux 0x7e0000"*
>
> *=> saveenv*
>
> *=> run <env_name>*

- Open terminal for Cortex-M4 to see serial output from the app running on it.

Testing Examples

- **ECSPI Driver**
  - Interrupt B2B Transfer master as loopback using a single board.
  - Connect MOSI and MISO pins physically on board (J1003-PIN_19-ECSPI2_MOSI & J1003-PIN_21-ECSPI2_MISO).
  - Edition in driver code:

    Assign **masterTxData** and **masterRxData** buffer to escpi_transfer_t object.
- **GPIO Driver**
  - Used led_output example.
  - Digital output -> Digital input (rising edge triggered).
  - Changing pin from **ECSPI2_MOSI** to **SAI5_RXD0** in pinmux.c.

    Refer to PinMux Tool for more information.
  - Interrupt Enabled on GPIO pin J1003-38 (originally configured to J1003-19).
  - Implement **GPIO3_Combined_16_31_IRQHandler()** function.

Extras

- Logic Analyzer
- ARM GCC Toolchain

Spark SDK & UWB

Historical Background of UWB: https://semiwiki.com/5g/283112-the-story-of-ultra-wideband-part-1-the-genesis/

Tools and Software

- SDK & Documentation: https://www.sparkmicro.com/products/#SDK.
- The version we used: was **v1.0.0.**
- CubeIDE for debugging.
- STM32 ST-Link V2.
- STM32CubeProgrammer to flash .bin files.

Testing Examples

- Add SDK to the workspace in CubeIDE.
- Connect the EVK board via STM32 ST-Link V2.
- Build the project and start debugging.

- **Audio Streaming**
  - Added necessary functions and header files for serial output as implemented in the Hello World example.

```
void iface_board_init(void)
{
    evk_init();
    usb_detect_callback_init();
    usb_connection_init();
}

void evk_usb_device_cdc_send_buf(uint8_t *buf, uint16_t buf_len);
```

  - Serial output:

```
~$ sudo picocom -b 112500 /dev/ttyACM*
```

  - Generating a custom sine wave and sending it through the wireless core.
  - Codec writes data into DMA then DMA transfer the data to the wireless core.
  - Implementing our sine wave where the codec generates buffer and starts the DMA; "iface_audio_evk.c".
  - Amplitude = 32767.0.
  - Sample Rate = 48kHz.

```
uint8_t dummy[256];
static uint16_t ep_max98091_action_produce(void *instance, uint8_t *data, uint16_t size)
{
    (void)instance;

    int16_t *samples = (int16_t *)data;
    uint16_t samples_count = size / 2;
    for (int n = 0; n < samples_count / 2; n = n + 2)
    {
        samples[n] = wave[sample_count];
        samples[n + 1] = samples[n];
        sample_count += 1;
        if (sample_count >= 48000)
        {
            sample_count = 0;
        }
    }

    evk_audio_sai_read_non_blocking(dummy, size);

    return 0;
}

static void ep_max98091_start_produce(void *instance)
{
    (void)instance;
    for (int n = 0; n < 48000; n++)
    {
        wave[n] = (double)(AMP * sinf((M_PI * 2 / SAMPLE_RATE) * FREQUENCY * n));
    }

    evk_audio_sai_start_read_non_blocking();
}
```

Implementation of Spark Stack on MCUXpresso SDK

- Spark stack porting guide.
- Add Spark Wireless Core API to hello_world example in XpressoSDK
- Add executables and header files in armgcc/CMakeList.txt.
- Wireless Core API is contained in hello_world/wireless directory
- Adapt BSP interface. Implemented in hello_world/**SR1020.h**

Pin configuration on IMX8-EVK:

J1003 Expansion Connector		
Pin Number	**Net Name**	**Description**
1	VEXT_3V3	Power Output, 3.3V
19	ECSPI2_MOSI	SPI2 data signal, master output slave input
21	ECSPI2_MISO	SPI2 data signal, master input slave output
23	ECSPI2_SCLK	SPI2 clock signal
24	GPIO5_IO13	GPIO5 chip select signal
36	GPIO3_IO21	Radio shutdown
37	GPIO3_IO22	Radio reset
38	GPIO3_IO23	External radio interrupt
39	GND	Ground

Pin configuration on SR1020 chip:

Pin Number	**Net Name**
1	Radio shutdown
2	IRQ
3	CS
4	SCK
5	MOSI
6	MISO
7	RSTN
8	3V3
9	GND