# Adjusting i.MX8 Yocto configuration

The aim in to deliver a Pull Request to our existing Yocto build setup that build firmware for Compulab UCM i.MX8M Plus. The source for the build are GitHub repositories. You must fork and modify the existing build to configure the new functionality.

- Test and Document USB Gadget setup with kernel module for combined: Serial, ECM Ethernet, MSG over USB OTG connector
- Test and Document Configuration for Console over USB OTG connector
- Configure Kernel modules to keep support, but get kernel below 10MBytes
- Adjust default U-Boot environment variables
- Bundle base OS (bin,lib,etc) with SquashFS on boot drive
- Configure U-Boot fallback images for Kernel + FDT

## U-Boot Process

Adjust the U-Boot environment to have a slightly different boot logic. If it fails find anything to load it should expose USB Mass Storage Gadget for the boot partition over USB OTG connector.

The boot order should be:

- USB
- SD Card (SD2)
- `kernel-x8.img` on eMMC
- `fallback-x8.img` on eMMC
- U-Boot Mass Storage Gadget for eMMC boot partition

Add in U-Boot build config:

- SquashFS sqfsls/sqfsload
- EROFS if available
- Console over USB1 OTG port

Make sure to document the steps needed in hush to:

- boot from USB
- boot from SD Card
- boot from NFS
- load USB Mass Storage Gadget

Compulab already has some documentation for network booting, but is isn't detailed enough. Your delivery should also include instructions for setting up the TFTP/NFS server up on a Debian developent box such as Raspberry Pi. I would like to be able to follow your instructions to connect a development board and a Raspberry Pi and boot a Yocto image on the development board.

### U-Boot Additionally

If possible describe how the environment partition on the eMMC can be exanded to 8K. If possible describe how the boot partition on the eMMC can be expanded to 2GB.

In order to verify the work the existing mmcblk2p1 needs to be expanded from 83.2M to 2G.

## Device Tree Adjustment

The sound pins have been reassigned from the default development config. The pins are allocated as SAI5 TX and RX. What reviewing the Device Tree Source it seemed to me that UART4 isn't mapped to a device driver. Is that due to it being used by the Cortex M7 as console?

### GPIO Pin Allocations

### P1 function allocation

| Misc | PX pin | Pad connected | Functionality |
|---|---|---|---|
| P20.11 | P1.19 | UART1_RXD | |
| P20.4 | P1.21 | UART3_RXD | |
| P21.11 | P1.26 | SAI3_TXD | SAI5_RX_DATA3 |
| P21.17 | P1.28 | SAI3_RXD | SAI5_RX_DATA0 |
| P21.15 | P1.30 | SAI3_MCLK | SAI5_MCLK |
| P21.19 | P1.32 | SAI3_RXC | SAI5_RXC |
| P21.8 | P1.33 | CAN2_TX | |
| P21.23 | P1.34 | SAI3_RXFS | SAI5_RX_SYNC |
| P21.13 | P1.36 | SAI3_TXC | SAI5_RX_DATA2 |
| P21.21 | P1.38 | SAI3_TXFS | SAI5_RX_DATA1 |
| P21.10 | P1.49 | CAN2_RX | |
| P21.12 | P1.51 | CAN1_RX | |
| P21.14 | P1.53 | CAN1_TX | |
| P20.12 | P1.59 | GPIO1_IO00 | EX_OH_nINT |
| | P1.60 | GPIO4_IO19 | EX0_nINT / PD_USBC_INT |
| P20.2 | P1.61 | UART3_TXD | |
| P21.31 | P1.63 | HDMI_DDC_SDA | |
| P21.29 | P1.70 | HDMI_DDC_SCL | |
| P20.9 | P1.72 | UART1_TXD | |
| P20.1 | P1.74 | UART2_TXD | |
| P20.3 | P1.76 | UART2_RXD | |

| Misc | PX pin | Pad connected | Functionality |
|---|---|---|---|
| P20.5 | P1.77 | PWM1_OUT | |
| | P1.79 | PWM2_OUT | |
| | P1.81 | PWM3_OUT | |
| P20.10 | P1.84 | UART4_RXD | |
| | P1.85 | HDMI_CEC | |
| P20.8 | P1.86 | UART4_TXD | |
| P21.2 | P1.87 | I2C6_SCL | |
| P21.4 | P1.89 | I2C6_SDA | |
| | P1.91 | I2C3_SDA | |
| | P1.85 | HDMI_HPD | |
| P20.33 | P1.94 | I2C3_SCL | |
| P21.1 | P1.96 | I2C5_SDA | |
| P20.14 | P1.98 | GPIO1_01 | EX_T_nINT |
| P21.3 | P1.100 | I2C5_SCL | |

P2 function allocation

| Misc | PX pin | Pad connected | Functionality |
|---|---|---|---|
| P20.21 | P2.41 | ~~ENET1_RD0~~ | SD1_DATA2 |
| P20.23 | P2.43 | ~~ENET1_RD1~~ | SD1_DATA3 |
| P20.25 | P2.45 | ~~ENET1_RD2~~ | GPIO2_IO8/OE_SOUND |
| P20.27 | P2.47 | ~~ENET1_RD3~~ | GPIO2_IO9/OE_CAM |
| | P2.49 | GPIO2_IO20 | |
| P21.16 | P2.51 | GPIO2_IO19 | |
| P21.32 | P2.52 | GPIO4_IO20 | PCIE_WAKE_B |
| P20.19 | P2.53 | ~~ENET1_RX_CTL~~ | SAI5_TXFS |
| P20.29 | P2.55 | ~~ENET1_RXC~~ | SAI5_TXC |
| P20.22 | P2.59 | ~~ENET1_TD0~~ | SD1_DATA1 |
| P21.34 | P2.60 | GPIO4_12 | SAI5_TXD0 |
| P20.24 | P2.61 | ~~ENET1_TD1~~ | SD1_DATA0 |
| P20.13 | P2.62 | GPIO2_IO10 | |

| Misc | PX pin | Pad connected | Functionality |
|---|---|---|---|
| P20.26 | P2.63 | ~~ENET1_TD2~~ | SAI5_TXD2 |
| P20.28 | P2.65 | ~~ENET1_TD3~~ | SAI5_TXD3 |
| P20.20 | P2.67 | ~~ENET1_TX_CTL~~ | GPIO4_16 |
| P20.15 | P2.68 | ~~ENET1_MDC~~ | SD1_CLK |
| P20.30 | P2.69 | ~~ENET1_TXC~~ | GPIO4_17 |
| P20.17 | P2.70 | ~~ENET1_MDIO~~ | SD1_CMD |
| P20.16 | P2.76 | ~~ENET1_nRST~~ | SAI5_TXD1 |
| P21.27 | P2.77 | ENET TD2_BYPASS | |
| P20.18 | P2.88 | ~~ENET1_INT~~ | GPIO4_IO21 |
| P21.22 | P2.89 | ECSPI2_MISO | |
| P21.20 | P2.90 | PCIE_CLKREQ_B | |
| P21.24 | P2.91 | ECSPI2_SS0 | |
| | P2.92 | SD2_nCD | |
| P21.26 | P2.93 | ECSPI2_SCLK | |
| | P2.94 | SD2_DATA2 | |
| P21.28 | P2.95 | ECSPI2_MOSI | |
| | P2.96 | SD2_CLK | |
| | P2.97 | SD2_DATA0 | |
| | P2.98 | SD2_DATA3 | |
| | P2.99 | SD2_DATA1 | |
| | P2.100 | SD2_CMD | |

## Linux Installation

The existing Yocto build should be adjusted.

Break out features as kernel modules to get Kernel below 10Mbyte.

Expose the `boot` partition as a Mass Storage Gadget over USB OTG.

- Add libcamera API in addition to V4L2.
- Add kernel support for F2FS and SquashFS.
- EROFS - Enhanced Read-Only File System if available
- Console over USB1 OTG port
- Core linux commands are bundled in a mounted SquashFS image
- Bundle Kernel, Device Tree and Core Linux Commands SquashFS for single file to load

Refs

- Yocto UUU & SDP
- How to update U-Boot on the board
- How to recover from corrupted eMMC U-Boot

## Partition Structure

## Boot Partition

The boot partition can be found on the internal eMMC, SD Card, USB Stick or NFS server. The format of boot is FAT and must be at least 256MB, preferably 1GB+.

The boot partition contains

- `kernel-x8.img` FIT image of Linux Kernel
- `kernel-x8.fdt` containing: Linux Kernel, Default Device Tree
- `base-x8.squashfs.img` containing essential executables and default configuration files
- `sdk-x8.squashfs.img` containing SDK executables and default configuration files
- `Assets` directory with Device Tree, Overlays and Kernel Modules
- `Packages` directory with installable packages
-
-

A U-Boot environment variable defines the `image` name `kernel-x8.img` that will be loaded. Another variable determines the `fdt_file` name `ucm-imx8m-plus-usbdev.dtb` that will be loaded.

Flattened device trees should ideally be placed in `/Assets` if U-Boot can load it from there. EROFS should be reviewed as an alternative to SquashFS. Support for EROFS should be added to our U-Boot/Kernel config asap.

- What is EROFS file system that Google will reportedly implement in Android 13?
- EROFS - Enhanced Read-Only File System

### Existing eMMC boot partition

```
root@ucm-imx8m-plus:~> lsblk
NAME         MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk2      179:0    0 29.1G  0 disk
|-mmcblk2p1  179:1    0 83.2M  0 part /run/media/mmcblk2p1
`-mmcblk2p2  179:2    0   29G  0 part /
mmcblk2boot0 179:32   0    4M  1 disk
mmcblk2boot1 179:64   0    4M  1 disk
root@ucm-imx8m-plus:~> ls /run/media/mmcblk2p1/
Image                        tee.bin
ucm-imx8m-plus_mipi-csi1.dtb
imx8mp_m7_TCM_hello_world.bin
ucm-imx8m-plus-usbdev.dtb  ucm-imx8m-plus_mipi-csi2.dtb
imx8mp_m7_TCM_rpmsg_lite_str_echo_rtos.bin  ucm-imx8m-plus.dtb
```

### Kernel Image

The kernel image is saved as a Flattened Device Tree file containing

- Linux Image
- Device Tree
- Kernel Configuration

TODO Device Tree. Sound SAI5. OTG.

### Base Image

The base image contains `/bin`, `/lib` and `/etc` with binaries and default configs for ,

- Networking if/dropbear/nfs/resolv/avahi/bluetooth/..
- Busybox
- Fonts
- Systemd
- X11/Weston

So largely the output of the bitbake target `imx-image-core`. As that is quite large some modules will have to be excluded from the image to get it down to a good size. Since `/sbin` is a legacy name, the files should be merged in `/bin`.

Parts not included in base image, but instead in the SDK image,

- gnu compiler collection
- gdb/gdbserver
- cmake
- Docker
- Containerd
- /usr/include

Bitbake can also build an SDK, perhaps that output should also be included in the SDK SquashFS image.

## Linux Logical Directory Structure

OverlayFS is used to combine `/media/base` and `/media/apps`.

`/media/base` mounts `base-x8.squashfs.img` on i.MX8 systems. `/media/apps` mounts the Ext2 partition that would traditionally be called root. `/media/sdk` mounts the `sdk-x8.squashfs.img`. `/boot` mounts the Boot FAT partition. `/` is a combination of sys or apps. `/home` is on the Ext2 partition. `/sys` mounts the Linux sysfs. `/proc` mounts the Linux procfs. `/dev` has available devices. Contains `/dev/console`, `/dev/null`, and `/dev/tty`. `/tmp` is an in-memory drive.