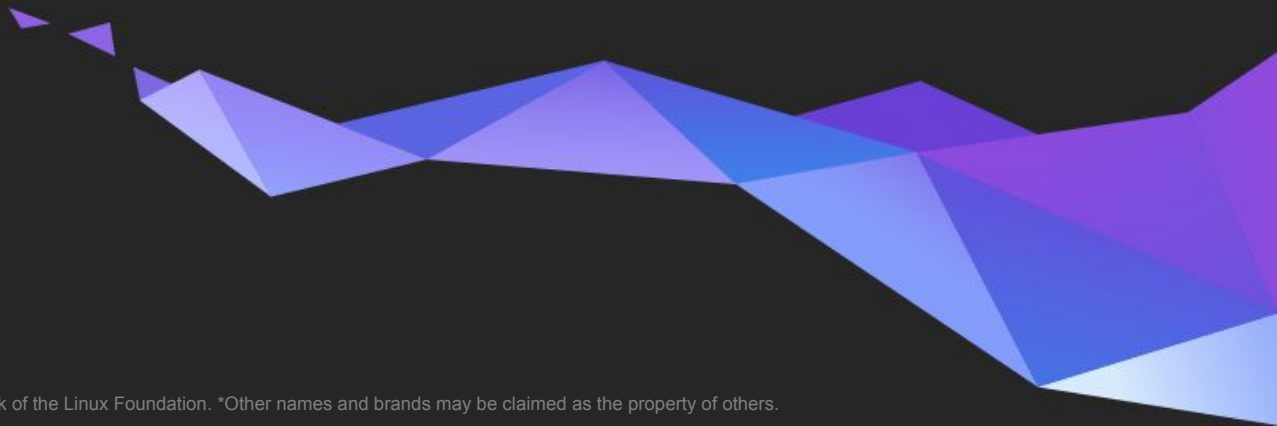
Zephyr™ OS Configuration via Device Tree

Andy Gross - Linaro IoT



Configuration in Zephyr today

- Configuration is spread out across the system.
- Most configuration is hardcoded.
- Difficult to deal with device multiples.
- Definitions come from multiple file sources, (CMSIS, vendor includes, etc)
- Not extensible for similar boards or SoCs.

Board and driver initialization

- Initialization predominately uses Kconfig
- #ifdefs are used in some initialization code to accommodate differences in boards or options
- Lots of #ifdef usage to deal with specific configuration and device multiples
- Hardcoded init structures to define platform data

Using device tree for Zephyr OS configuration

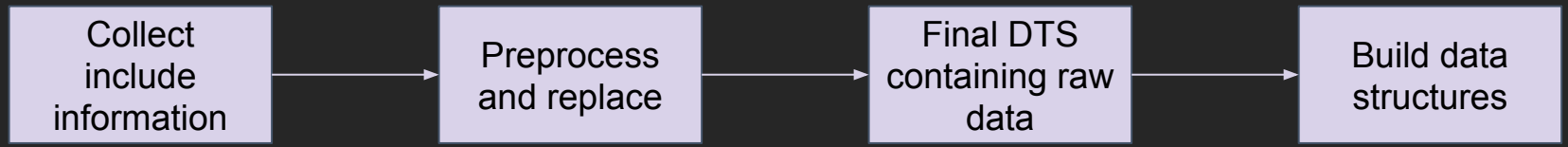
- Device tree is architecturally neutral
- Less need for Kconfig options as specific config comes from DTS
- Device tree can describe any device node
- Device description is extensible
- Other layers could use device tree information (apps, hal, etc)
- Adding new boards/SoCs is easier

To blob or not to blob

- Don’t want to use runtime flattened device tree blob
- Only interested in defining configuration and board initialization information
- Space usage is at a premium. Don’t want to consume more space than required (memory or flash)
- Current parsers target the flattened device tree blob.

Required tooling for device tree usage

- Use the available configuration sources where applicable (CMSIS, vendor files, etc)
- Use the C preprocessor to leverage those configuration sources
- Build the target configuration from the processed device tree information



Work for the near term

- Define DTS files for a few platforms, not just the NXP* FRDM. Note: Some SoC vendors already have some device tree implementations (STMicroelectronics).
- Get the tooling in place for creating a set of include files and data structures
- Generate the config from the dts files and work this into the Makefiles
- Cleanup the configuration directories for the boards as the required existing config and board files are retired. This will most likely involve complete removal of the board/ directories.
- Leverage the generated files and use this information to initialize drivers.

Example Device Tree


```
/dts-v1/;

/ {
        compatible = "nxp,k64f", "nxp,mk64f12";
        #address-cells = <1>;
        #size-cells = <1>;


        cpus {
                cpu@0 {
                        compatible = "arm,cortex-m4f";
                };
        };

        memory {
                compatible = "mmio-sram";
                reg = <0x20000000 0x30000>;
        };

        soc {
                #address-cells = <1>;
                #size-cells = <1>;

                interrupt-controller@e000e100 {
                        compatible = "arm,cortex-m4-nvic";
                        reg = <0xe000e100 0x3ef>;
                        num-irq-prio-bits = <4>;
                        num-irqs = <86>;
                };

                timer@e00e010 {
                        compatible = "arm,cortex-m4-systick";
                        reg = <0xe000e010 0x10>;

                        clk-source = <0>;               /* AHB or AHB/8 */
                };

                mpu@4000d000 {
                        compatible = "nxp,k64f-mpu";
                        reg = <0x4000d000 0x824>;
                        status = "disabled";
                };

                clock-controller@40064000 {
                        compatible = "nxp,k64f-mcg";
                        reg = <0x40064000 0xd>;
                        system-clock-frequency = <120000000>;
                };

                clock-controller@40065000 {
                        compatible = "nxp,k64f-osc";
                        reg = <0x40065000 0x4>;

                        enable-external-reference;
                };

                rtc@4003d000 {
                        compatible = "nxp,k64f-rtc";
                        reg = <0x4003d000 0x808>;
                        clock-frequency = <32768>;              /* fixed 32kHz clk */
                };
```

Example Device Tree - Continued


```
sim: sim@40047000 {
        compatible = "nxp,k64f-sim";
        reg = <0x40047000 0x1060>;

        /* also encode active/sleep/deep sleep */
        clk-divider-core = <1>;
        clk-divider-bus = <2>;
        clk-divider-flexbus = <3>;
        clk-divider-flash = <5>;
};

uart0: uart@4006a000 {
        compatible = "nxp,k64f-uart";
        reg = <0x4006a000 0x1000>;
        zephyr,label = "UART_0";
        interrupts = <31 0 0>;          /* irq 31 - no flags - prio 0 */
        baud-rate = <115200>;
        pinctrl-0 = <&uart0_default>;
        pinctrl-1 = <&uart0_lpm>;
        pinctrl-names = "default", "lpm";
};

uart1: uart@4006b000 {
        compatible = "nxp,k64f-uart";
        reg = <0x4006b000 0x1000>;
        zephyr,label = "UART_1";
        interrupts = <33 0 0>;
        baud-rate = <115200>;
};
```

```
pinmux@40049000 {
        compatible = "nxp,k64f-pinmux";
        reg = <0x40049000 0x40ca>;
        zephyr,label = "PINMUX";

        uart0_default: uart0_default {
                rx-tx {
                        port = <&gpiob>;
                        pins = <16>, <17>;
                        function = <3>;
                };
        };

        uart0_lpm: uart0_lpm {
                rx-tx {
                        port = <&gpiob>;
                        pins = <16>, <17>;
                        function = <0>;
                };
        };

        spi0_default: spi0_default {
                miso-mosi-clk {
                        port = <&gpiob>;
                        pins = <11>, <10>, <9>;
                        function = <2>;
                };
        };
};
```

Example Device Tree - Continued


```
gpioa: gpio@400ff000 {
        compatible = "nxp,k64f-gpio";
        reg = <0x400ff000 0x40>;
        zephyr,label = "GPIO_0";
        interrupts = <59 0 3>;
};

gpiob: gpio@400ff040 {
        compatible = "nxp,k64f-gpio";
        reg = <0x400ff040 0x40>;
        zephyr,label = "GPIO_1";
        interrupts = <60 0 3>;
};
```

```
spi0: spi@4002c000 {
        compatible = "nxp,k64f-spi";
        reg = <0x4002c000 0x88>;
        zephyr,label = "SPI_0";
        interrupts = <26 0 3>;
        clocks = <&sim 0x103C 12>;          /* clk gate */

        cs = <&gpiob 10>, <&gpiob 9>;       /* cs0 = PTB10, cs1 = PTB9 */
        pinctrl-0 = <&spi0_default>;
        pinctrl-names = "default";
};

spi1: spi@4002d000 {
        compatible = "nxp,k64f-spi";
        reg = <0x4002d000 0x88>;
        zephyr,label = "SPI_1";
        interrupts = <27 0 3>;
        clocks = <&sim 0x103C 13>;          /* clk gate */
};
```

Generated Configuration from DTS


```
/*****************************************************
 * Generated include file for nxp,mk64f12
 *              DO NOT MODIFY
 */
#ifndef _GENERATED_BOARD_H
#define _GENERATED_BOARD_H

/* Memory Definitions */
#define SRAM_BASE_ADDRESS              0x20000000
#define SRAM_SIZE                      0x30000
#define SRAM_0_BASE_ADDRESS            0x20000000
#define SRAM_0_SIZE                    0x30000

/* Flash Definitions */
#define FLASH_CONTROLLER_0_BASE_ADDRESS 0x4001f000
#define FLASH_CONTROLLER_0_IRQ          18
#define FLASH_CONTROLLER_0_IRQ_PRIO     3
#define FLASH_CONTROLLER_0_IRQ_FLAGS    0

/* UART Definitions */
#define NUM_UARTS                      1
#define UART_PORT_0_BASE_ADDRESS       0x4006a000
#define UART_PORT_0_IRQ                31
#define UART_PORT_0_IRQ_PRIO           3
#define UART_PORT_0_IRQ_FLAGS          0
#define UART_PORT_0_DEV_NAME           "UART_0"
#define UART_PORT_0_BAUD_RATE          115200

/* IRQ Definitions */
#define NUM_IRQS                       86
#define NUM_IRQ_PRIO_LEVELS            4

#endif
```

Zephyr™

Questions?