Modem driver

The Murata CMWX1ZZABZ is a LoRaWAN module which is made up of an stm32l072 microcontroller and an sx1276 LoRa transceiver. We'll call this module the modem.

The MVPI uses the LPUART1 to communicate with the modem. Between the MVPI and the modem, ALP operations are exchanged over the serial UART. To reference these operations, refer to the ALP_Specification_V1.2, available from the Dash7 Alliance here.

Serial Interface

The module for the serial interface to communicate with the modem is taken directly from the "Connect-BLE" firmware, which was also made using Zephyr. This module defines functions to initialize the interface on a specified communication bus (LPUART1 here), to transfer bytes over the serial interface and to add or remove handlers for received data over the interface. The serial interface uses the existing subsystems for UART and power management to set up the interface to drive the modem.

In the zephyr_mvpi, the serial interface gets initialized in modem.c and the handler for received data used, is alp_handler(). This handler will print out the received data and after a successful LoRa transmission, this handler will increment the counter of successful transmissions of the heartbeat file.

UART

The communication between MVPI and modem happens over UART. Zephyr has support for UART peripherals with 3 types of implementation API's:

- Polling API
- Interrupt-driven API
- Asynchronous API, using DMA

  Here, we'll make use of the Asynchronous API, using DMA to communicate over the LPUART1 bus. To use this, we will have to configure the lpuart1 and DMA nodes in the device tree. The DMA channels used for the LPUART need to be specified with the node identifiers <*dmas*> and <*dma-names*>, following the st,stm32-lpuart.yaml and st,stm32-dma-v2.yaml device tree binding.

```
// MVPI lpuart1 node
&lpuart1 {
    pinctrl-0 = <&lpuart1_tx_pb10 &lpuart1_rx_pb11>;
    pinctrl-names = "default";
    current-speed = <115200>;
    status = "okay";

    dmas = <&dma1 2 5 (STM32_DMA_PERIPH_TX | STM32_DMA_PRIORITY_HIGH)>,
           <&dma1 3 5 (STM32_DMA_PERIPH_RX | STM32_DMA_PRIORITY_HIGH)>;
    dma-names = "tx", "rx";
};
```

```
// stm32l0 lpuart1 node
lpuart1: serial@40004800 {
            compatible = "st,stm32-lpuart", "st,stm32-uart";
            reg = <0x40004800 0x400>;
            clocks = <&rcc STM32_CLOCK_BUS_APB1 0x00040000>;
            resets = <&rctl STM32_RESET(APB1, 18U)>;
            interrupts = <29 0>;
            status = "disabled";
        };
```

Now to use the async api, we need to enable the async API in Kconfig, which we'll do in prj.conf

```
CONFIG_UART_ASYNC_API=y
```

To hande the asynchronous UART communication, the tx and rx functions are used and these return events:

- UART_TX_DONE
- UART_TX_ABORTED
- UART_RX_RDY
- UART_RX_BUF_REQUEST
- UART_RX_BUF_RELEASED
- UART_RX_DISABLED
- UART_RX_STOPPED