Enabling GPIO and UART on BGM220P with Zephyr RTOS

This document summarizes the step-by-step process of enabling a GPIO (for LED) and UART (USART1 using PB1/PB2) for the BGM220P module using Zephyr RTOS, including troubleshooting errors encountered.

---

1. Prerequisites

- Zephyr RTOS installed and initialized in your working directory
- BGM220P-based board supported under custom sltb010a target
- CP2102 USB-to-TTL converter connected

---

2. GPIO (LED) Enablement

Modify sltb010a.dts

File: zephyr/boards/silabs/dev_kits/sltb010a/sltb010a.dts

```
user_gpio0: user_gpio0 {
   compatible = "gpio-leds";
   status = "okay";

   led1_user: led1_user {
      gpios = <&gpiob 1 GPIO_ACTIVE_HIGH>; // PB1
      label = "User LED 1 (PB1)";
   };
};
```

//

zephyr/boards/silabs/dev_kits/sltb010a/sltb010a.dts

```
/*
 * Copyright (c) 2021 Sateesh Kotapati
 *
 * SPDX-License-Identifier: Apache-2.0
 */

/dts-v1/;
#include <silabs/xg22/efr32bg22c224f512im40.dtsi>
#include "sltb010a-pinctrl.dtsi"
#include "thunderboard.dtsi"
#include <zephyr/dt-bindings/regulator/silabs_dcdc.h>
```

```
/ {

	model = "Silicon Labs EFR32BG22 Thunderboard (SLTB010A) using BRD4184B";
	compatible = "silabs,efr32bg22c224f512im40", "silabs,sltb010a",
		"silabs,efr32bg22";



	/* These aliases are provided for compatibility with samples */
	aliases {
		led0 = &led0;
	       gpio0 = &led1_user;
		uart1 = &usart1;
		spi0 = &usart0;
		sw0 = &button0;
		watchdog0 = &wdog0;

		/* If enabled, MCUboot uses this for recovery mode entrance */
		mcuboot-led0 = &led0;
		mcuboot-button0 = &button0;
	};

	chosen {
//		zephyr,bt-hci = &bt_hici_silabs;
		zephyr,code-partition = &slot0_partition;
	};

	user_gpio0: user_gpio0 {
	    compatible = "gpio-leds";
	    status = "okay";

	    led1_user: led1_user {
		gpios = <&gpiob 0 GPIO_ACTIVE_HIGH>;
		label = "User LED 1 (PA05)";
	    };
	};
};

&hfxo {
	ctune = <120>;
	precision = <50>;
	status = "okay";
};

&lfxo {
	ctune = <37>;
	precision = <50>;
	status = "okay";
};

&hfrcodpll {
```

```
        clock-frequency = <DT_FREQ_K(76800)>;
        clocks = <&hfxo>;
        dpll-autorecover;
        dpll-edge = "fall";
        dpll-lock = "phase";
        dpll-m = <1919>;
        dpll-n = <3839>;
};

&em23grpaclk {
        clocks = <&lfxo>;
};

&em4grpaclk {
        clocks = <&lfxo>;
};

&rtccclk {
        clocks = <&lfxo>;
};

&wdog0clk {
        clocks = <&lfxo>;
};

&dcdc {
        regulator-boot-on;
        regulator-initial-mode = <SILABS_DCDC_MODE_BUCK>;
        status = "okay";
};

&flash0 {
        partitions {
                /* Reserve 48 KiB for the bootloader */
                boot_partition: partition@0 {
                        reg = <0x00000000 0x0000c000>;
                        label = "mcuboot";
                        read-only;
                };

                /* Reserve 224 KiB for the application in slot 0 */
                slot0_partition: partition@c000 {
                        reg = <0x0000c000 0x00038000>;
                        label = "image-0";
                };

                /* Reserve 224 KiB for the application in slot 1 */
                slot1_partition: partition@44000 {
                        reg = <0x00044000 0x00038000>;
                        label = "image-1";
                };
```

```
                /* Set 16 KiB of storage at the end of the 512 KiB of flash */
                storage_partition: partition@7c000 {
                        reg = <0x0007c000 0x00004000>;
                        label = "storage";
                };
        };
};

&sw_imu_enable {
        enable-gpios = <&gpiob 4 GPIO_ACTIVE_HIGH>;
};

&radio {
        pa-voltage-mv = <1800>;
};

&bt_hci_silabs {
        status = "okay";
};
&gpiob {
   status = "okay";
};
&usart1 {
   status = "okay";
   current-speed = <115200>;
   pinctrl-0 = <&usart1_default>;
   pinctrl-names = "default";
};
```
//

Enable Port B

```
&gpiob {
   status = "okay";
};
```

Aliases (optional for compatibility)

```
aliases {
   gpio0 = &led1_user;
};
```

Common Error Faced

- Device tree node 'led1_user' is not available or enabled
  Fix: Ensure status = "okay"; and alias is declared.

---

3. UART Enablement (USART1 on PB1/TX and PB2/RX)

Modify sltb010a-pinctrl.dtsi

File: zephyr/boards/silabs/dev_kits/sltb010a/sltb010a-pinctrl.dtsi

```
usart1_default: usart1_default {
   group0 {
      pins = <USART1_TX_PB1>;
      drive-push-pull;
      output-high;
   };
   group1 {
      pins = <USART1_RX_PB2>;
      input-enable;
      silabs,input-filter;
   };
};
```

Ensure USART1 Node in efr32bg22*.dtsi is enabled:

```
&usart1 {
   status = "okay";
   current-speed = <9600>;
   pinctrl-0 = <&usart1_default>;
   pinctrl-names = "default";
};
```

Common Errors Faced

- Nothing printed in Minicom: **Fix**: Match UART baud rate (9600), check TX/RX wiring
- Only RX LED blinking: **Fix**: Check correct mapping PB1 (TX), PB2 (RX)
- Cursor moving but no data: **Fix**: Disable CONFIG_UART_CONSOLE and CONFIG_CONSOLE

---

4. Application Code

File: gpio_app/src/main.c

GPIO Blink Example

```
#include <zephyr/device.h>
#include <zephyr/drivers/gpio.h>
#include <zephyr/kernel.h>
#include <zephyr/sys/printk.h>

#define LED_NODE DT_NODELABEL(led1_user)

void main(void) {
   const struct device *gpio_dev = DEVICE_DT_GET(DT_GPIO_CTLR(LED_NODE, gpios));
   gpio_pin_t pin = DT_GPIO_PIN(LED_NODE, gpios);
   gpio_flags_t flags = DT_GPIO_FLAGS(LED_NODE, gpios);

   if (!device_is_ready(gpio_dev)) return;
   gpio_pin_configure(gpio_dev, pin, GPIO_OUTPUT_ACTIVE | flags);

   while (1) {
      gpio_pin_toggle(gpio_dev, pin);
      k_msleep(500);
   }
}
```

UART Print Example

```
#include <zephyr/device.h>
#include <zephyr/drivers/uart.h>
#include <zephyr/kernel.h>

#define UART_NODE DT_NODELABEL(usart1)

void main(void) {
   const struct device *uart_dev = DEVICE_DT_GET(UART_NODE);
   if (!device_is_ready(uart_dev)) return;

   const char *msg = "Hello from BGM220P via USART1\r\n";
   while (1) {
      for (int i = 0; msg[i]; i++) uart_poll_out(uart_dev, msg[i]);
      k_sleep(K_SECONDS(1));
   }
}
```

---

5. prj.conf Configuration

```
CONFIG_SERIAL=y
CONFIG_UART_CONSOLE=n
CONFIG_CONSOLE=n
CONFIG_PRINTK=y
CONFIG_UART_1_BAUD_RATE=9600
```

---

6. Build and Flash Commands

```
# Build application
west build -b sltb010a gpio_app -p

# Flash
west flash
```

---

7. Serial Communication Test (Host)

```
# Start Minicom (adjust /dev/ttyUSB0 if needed)
sudo minicom -b 9600 -D /dev/ttyUSB0
```

You should now see "Hello from BGM220P via USART1" printed every second.

---

8. Wiring for CP2102

CP2102 Pin	BGM220P Pin
TX	PB2 (RX)
RX	PB1 (TX)

CP2102 Pin	BGM220P Pin
GND	GND

Summary of Issues & Fixes

Issue	Cause / Fix
LED node not found	Alias or node not declared properly in .dts
Nothing printed in UART (Minicom)	Wrong baud rate / Wrong TX-RX pins / Missing CONFIG_PRINTK=y
RX LED blinks but no TX	TX line not correctly configured or not connected
Cursor moves but no print	Console not disabled (CONFIG_UART_CONSOLE=n)
Baud rate mismatch	Set current-speed in DTS and CONFIG_UART_1_BAUD_RATE in .conf

---

Yes, you previously encountered an **application function-related error** during the build stage, which was tied to **incorrect or missing node compatibility and references** in the device tree.

Here's a summary of that error and its fix:

Here’s a detailed list of **common errors and their solutions** you encountered while enabling GPIO and UART on the **BGM220P board** with Zephyr. You can include this as a new section at the end of your documentation:

Common Errors and Solutions

Error 1: Device tree node 'led1_user' is not available or enabled

- **Cause**: The LED node led1_user was either not defined or not marked as status = "okay" in the .dts file.
- **Fix**:
    - Add the following under / node:

```
user_gpio0: user_gpio0 {
    compatible = "gpio-leds";
    status = "okay";

    led1_user: led1_user {
        gpios = <&gpiob 1 GPIO_ACTIVE_HIGH>;
        label = "User LED 1 (PB1)";
    };
};
```

    - Ensure the GPIO port (&gpiob) is enabled:

```
&gpiob {
    status = "okay";
};
```

Error 2: USART1 not enabled in device tree

- **Cause**: &usart1 node missing or not set as status = "okay" in efr32bg22*.dtsi or board overlay.
- **Fix**: Add or update this node:

```
&usart1 {
   status = "okay";
   current-speed = <9600>;
   pinctrl-0 = <&usart1_default>;
   pinctrl-names = "default";
};
```

Error 3: Minicom shows blank screen or moving cursor

- **Cause**: Baud rate mismatch, incorrect TX/RX wiring, or UART0 still enabled.
- **Fix**:
    - Match baud rate:
        - minicom: sudo minicom -b 9600 -D /dev/ttyUSB0
        - prj.conf:

          CONFIG_UART_1_BAUD_RATE=9600
    - Disable default console:

      CONFIG_UART_CONSOLE=n
      CONFIG_CONSOLE=n
      CONFIG_PRINTK=y
      CONFIG_SERIAL=y
    - Correct wiring:

CP2102	BGM220P
TX	PB2 (RX)
RX	PB1 (TX)
GND	GND

Error 4: UART output only shows repeated Minicom2.7.1 or junk characters

- **Cause**: TX and RX lines reversed.
- **Fix**: Swap TX ↔ RX wires:
    - CP2102 TX → BGM220P RX (PB2)
    - CP2102 RX → BGM220P TX (PB1)

Error 5: Application builds but no LED blinks or UART output

- **Cause**: Wrong pin defined or device_is_ready() returns false.
- **Fix**:
    - Confirm correct GPIO controller and pin using DT_GPIO_CTLR() and DT_GPIO_PIN().
    - Use printk() for debug logs.

Error 6: Baud rate mismatch with CP2102 default (9600)

- **Fix**: Manually set baud rate in device tree:

  current-speed = <9600>;

  And also match in prj.conf:

  CONFIG_UART_1_BAUD_RATE=9600

Would you like me to insert this section directly into your documentation now?

Error 7: undefined node label, incompatible node or devicetree node not available

Example Error Messages:

error: ‘led1_user’ is not available or enabled
devicetree error: /chosen: undefined node label 'bt_hici_silabs'

Cause:

- The node (e.g., led1_user or bt_hici_silabs) was:
    - Not **declared** using a label or node label
    - Not given a status = "okay" to enable it
    - Or the referenced label didn’t match the one in your main.c or .overlay

Fix:

1. **Declare the node with a valid label** (either as a node label or label = "...") inside user_gpio0:

```
user_gpio0: user_gpio0 {
   compatible = "gpio-leds";
   status = "okay";

   led1_user: led1_user {
      gpios = <&gpiob 1 GPIO_ACTIVE_HIGH>;
      label = "User LED 1 (PB1)";
   };
};
```

2. **Ensure it's referenced correctly in application code**:

```
#define LED_NODE DT_NODELABEL(led1_user)
```

3. **Enable the parent GPIO port**:

```
&gpiob {
   status = "okay";
};
```

4. **Ensure aliases {} block does not conflict**, or is updated correctly:

```
aliases {
   gpio0 = &led1_user;
};
```

This kind of issue often arises when you:

- Use DT_NODELABEL() or DT_ALIAS() for a label that **isn’t defined** in the .dts file.
- Call a node like &bt_hici_silabs in /chosen without enabling or defining it elsewhere.