Overview Hardware Code (https://os.mbed.com/code/) Documentation Case studies (https://os.mbed.com/built-with-m

v6.15
(https://os.mbed.com)

Mistake on this page?
Report an issue in GitHub (https://github.com/ARMmbed/mbed-os-5-docs/issues) or email us (mailto:docs-mbed@arm.com)

≡

Docs (..) › Porting (index.html) › Porting the HAL APIs (porting-the-hal-apis.html) › Static pin map extension (static-pinmap-port.html)

Static pin map extension

The **static pin map extension** allows you to statically specify the peripheral configuration (pin, peripheral or function) in the HAL API function.

In modern MCUs, you can often map peripherals to different pins, and each pin can have multiple functions. Mbed supports dynamic pin mapping, meaning you can reconfigure pins at run time for different drivers to use. That provides great flexibility, but it's not free. There's a nontrivial ROM cost to maintain the pin map tables and infrastructure to parse it. In some use cases, this flexibility is worth the cost. Often, pin configuration is frozen at the hardware design stage and doesn't require run time modification. Shifting this configuration to compile time allows free memory associated with the dynamic approach.

HAL APIs using pins take these pins in their constructor and use those pins to look up which peripheral or function to use. The process of looking up the peripheral or function requires there to be a pin map table that maps pins to peripherals or functions. This pin map table takes up ROM. The static pin map extension provides additional HAL API constructors, which take pin map as a parameter where the pin, peripheral or function is specified statically, and there is no need to use the pin map tables.

Supported peripherals:

- `PWM`.
- `AnalogIn`.
- `AnalogOut`.
- `SPI`.
- `I2C`.
- `UART`.
- `QSPI`.
- `CAN`.

Implementing static pin map extension

If you want to make static pin map available on your platform please perform the following steps:

1. Provide implementation of `xxx_init_direct(xxx_t *obj, static_pinmap_t *)` function and update implementation of `xxx_init()`.
   - `xxx_init()` uses pin map tables to determine the associated peripheral or function with the given pins, populates the pin map structure and calls void `xxx_init_direct()`.
   - `xxx_init_direct()` performs peripheral initialization using given static pin map structure.

   Example implementation:

```
void xxx_init_direct(xxx_t *obj, const PinMap *pinmap)
{
    obj->spi.instance = pinmap->peripheral;

    // pin out the xxx pins
    pin_function(pinmap->pin, pinmap->function);
    pin_mode(pinmap->pin, PullNone);

    // Some additional init code
}

void xxx_init(xxx_t *obj, PinName pin)
{
    int peripheral = (int)pinmap_peripheral(pin, PinMap_xxx);
    int function = (int)pinmap_find_function(pin, PinMap_xxx);

    const PinMap static_pinmap = {pin, peripheral, function};

    xxx_init_direct(obj, &static_pinmap);
}
```

2. Provide `constexpr` pin-map tables in the header file.

   Move pin map tables from `PeripheralPins.c` to `PeripheralPinMaps.h` (create new file), and add `constexpr` specifier in the pin map table declarations.

   The tables are required in the header file, so constant expression utility functions can include and use them to find and return mapping without pulling the pin map table into the image.

Note: Please include the `<mstd_cstddef>` module, and use the `MSTD_CONSTEXPR_OBJ_11` macro instead of the `constexpr` specifier. When `PeripheralPinMaps.h` is included from the Mbed OS C++ code, we need to see it as `constexpr`, but if the target code includes it from C, it has to have it as `const`.

Example pin map table:

```
#include <mstd_cstddef>

MSTD_CONSTEXPR_OBJ_11 PinMap PinMap_ADC[] = {
    {P0_23, ADC0_SE0,    0},
    {P0_10, ADC0_SE1,    0},
    {P0_31, ADC0_SE3,    0},
    {P1_8,  ADC0_SE4,    0},
    {P2_0,  ADC0_SE5,    0},
    {P2_13, ADC0_SE6,    0},
    {P2_11, ADC0_SE7,    0},
    {NC   , NC      ,    0}
};
```

1. Provide macros for pin map tables.

   Because pin map table names are not common across all targets, the following macros for available pin map tables are required in the `PeripheralPinMaps.h` file:

```
#define PINMAP_ANALOGIN [PinMap ADC]
#define PINMAP_ANALOGOUT [PinMap DAC]
#define PINMAP_I2C_SDA [PinMap I2C SDA]
#define PINMAP_I2C_SCL [PinMap I2C SCL]
#define PINMAP_UART_TX [PinMap UART TX]
#define PINMAP_UART_RX [PinMap UART RX]
#define PINMAP_UART_CTS [PinMap UART CTS]
#define PINMAP_UART_RTS [PinMap UART RTS]
#define PINMAP_SPI_SCLK [PinMap SPI SCLK]
#define PINMAP_SPI_MOSI [PinMap SPI MOSI]
#define PINMAP_SPI_MISO [PinMap SPI MISO]
#define PINMAP_SPI_SSEL [PinMap SPI SSEL]
#define PINMAP_PWM [PinMap PWM]
#define PINMAP_QSPI_DATA0 [PinMap QSPI DATA0]
#define PINMAP_QSPI_DATA1 [PinMap QSPI DATA1]
#define PINMAP_QSPI_DATA2 [PinMap QSPI DATA2]
#define PINMAP_QSPI_DATA3 [PinMap QSPI DATA3]
#define PINMAP_QSPI_SCLK [PinMap QSPI SCLK]
#define PINMAP_QSPI_SSEL [PinMap QSPI SSEL]
#define PINMAP_CAN_RD [PinMap CAN RD]
#define PINMAP_CAN_TD [PinMap CAN RD]
```

2. Provide `STATIC_PINMAP_READY` macro in `PinNames.h`

   Adding this macro enables the static pin map support for the target.

```
/* If this macro is defined, you can use constexpr utility functions for pin map search. */
#define STATIC_PINMAP_READY 1
```

Testing

Use the code below to test the static pin map extension:

```
int main()
{
    /* Regular use */
    SPI spi(D1, D2, D3, D4);

    /* Static pinmap */
    const spi_pinmap_t static_spi_pinmap = {SPI_1, D1, 2, D2, 2, D3, 2, D4, 2};
    SPI spi(static_spi_pinmap);

    /* Static pinmap with constexpr utility function */
    constexpr spi_pinmap_t static_spi_pinmap = get_spi_pinmap(D1, D2, D3, D4);
    SPI spi(static_spi_pinmap);

    return 0;
}
```

When you use the static pin map extension, you save on ROM:

```
| Module                          |        .text |   .data |       .bss |
|---------------------------------|--------------|---------|------------|
| [lib]\c_w.l                     |    11175(+0) |  16(+0) |    348(+0) |
| [lib]\fz_wm.l                   |       34(+0) |   0(+0) |      0(+0) |
| [lib]\m_wm.l                    |       48(+0) |   0(+0) |      0(+0) |
| anon$$obj.o                     |       32(+0) |   0(+0) | 197888(+0) |
| drivers\source                  |      192(+0) |   0(+0) |      0(+0) |
| features\netsocket              |      143(+0) |   0(+0) |      0(+0) |
| hal\mbed_critical_section_api.o |      154(+0) |   0(+0) |      2(+0) |
| hal\mbed_gpio.o                 |       96(+0) |   0(+0) |      0(+0) |
| hal\mbed_pinmap_common.o        |      0(-272) |   0(+0) |      0(+0) | // removed pinmap lib (this is common for all peripherals)
| hal\mbed_pinmap_common.o        |      0(-272) |   0(+0) |      0(+0) | // remove pinmap lib (this is common for all peripherals)
| hal\mbed_ticker_api.o           |      978(+0) |   0(+0) |      0(+0) |
| hal\mbed_us_ticker_api.o        |      114(+0) |   4(+0) |     65(+0) |
| main.o                          |      70(+32) |   0(+0) |      0(+0) | // extra space for static pinmap structure in application
| platform\source                 |    5683(+46) |  64(+0) |    249(+0) | // extra space for UART static pinmap structure to initialize the console
| main.o                          |      70(+32) |   0(+0) |      0(+0) | // extra space for static pin map structure in application
| platform\source                 |    5683(+46) |  64(+0) |    249(+0) | // extra space for UART static pin map structure to initialize the console
| rtos\source                     |     8990(+0) | 168(+0) |   6626(+0) |
| targets\TARGET_Freescale        |  16581(-816) |  12(+0) |    340(+0) | // removed pinmaps + driver code reduction
| targets\TARGET_Freescale        |  16581(-816) |  12(+0) |    340(+0) | // remove pin maps and driver code reduction
| Subtotals                       | 44290(-1010) | 264(+0) | 205518(+0) |
Total Static RAM memory (data + bss): 205782(+0) bytes
Total Flash memory (text + data): 44554(-1010) bytes
Total static RAM memory (data + bss): 205782(+0) bytes
Total flash memory (text + data): 44554(-1010) bytes
```

Run FPGA tests to check whether your implementation is valid:

```
mbed test -n "hal-tests-tests*fpga*" --app-config TESTS/configs/fpga.json
```

The `FPGA_FORCE_ALL_PORTS` macro can be defined to force all pinouts of all peripherals to be tested. Some FPGA tests only test one pinout of one peripheral by default, to save time.

```
mbed test -n "hal-tests-tests*fpga*" --app-config TESTS/configs/fpga.json -DFPGA_FORCE_ALL_PORTS
```

Note: Your target must be ready to run FPGA Test Shield tests.