ESP_BOARD_MANAGER 快速入门指南

日期	版本	发布说明	组件版本
2025-11-10	v1.0	发布 ESP_BOARD_MANAGER 快速入门指南	0.4.x
2025-12-18	v1.1	适配新的 board 路径扫描和选择机制	0.5.x

目录

第一步：将组件添加到你的工程

执行命令： `idf.py add-dependency esp_board_manager` 将 `esp_board_manager` 组件添加到你的工程中，会看到以下打印

```
→  hello_world idf.py add-dependency esp_board_manager
Executing action: add-dependency
NOTICE: Created "/home/liujinhong/esp/proj/learn/hello_world/main/idf_component.yml"
NOTICE: Successfully added dependency "espressif/esp_board_manager": "*" to component "main"
NOTICE: If you want to make additional changes to the manifest file at path /home/liujinhong/esp/proj/learn/hello_w
orld/main/idf_component.yml manually, please refer to the documentation: https://docs.espressif.com/projects/idf-co
mponent-manager/en/latest/reference/manifest_file.html
Done
→  hello_world
```

执行完后组件依赖会被自动添加到 `main/idf_component.yml` 文件中

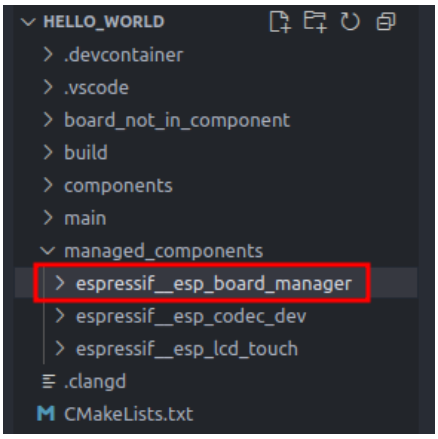
idf_component.yml

```
main > ! idf_component.yml
1  ## IDF Component Manager Manifest File
2  dependencies:
3    ## Required IDF version
4    idf:
5      version: '>=4.1.0'
6    # # Put list of dependencies here
7    # # For components maintained by Espressif:
8    # component: "~1.0.0"
9    # # For 3rd party components:
10   # username/component: ">=1.0.0,<2.0.0"
11   # username2/component2:
12   #   version: "~1.0.0"
13   #   # For transient dependencies `public` flag can be set.
14   #   # `public` flag doesn't have an effect dependencies of the `main` component.
15   #   # All dependencies of `main` are public by default.
16   #   public: true
17   espressif/esp_board_manager: '*'
18
```

第二步：将组件拉到本地

执行命令 `idf.py set-target esp32s3` 将 `esp_board_manager` 组件拉到本地，执行完之后可以在 `managed_components` 目录下看到 `espressif__esp_board_manager` 组件

- 此处选择的目标芯片可以根据项目实际使用的芯片进行修改



第三步：设置 board_manager 组件路径并验证

执行 `export IDF_EXTRA_ACTIONS_PATH=managed_components/espressif__esp_board_manager` 设置组件路径为刚才拉到本地的 `esp_board_manager` 组件

- 通过 export 指令配置的 `IDF_EXTRA_ACTIONS_PATH` 路径为当前终端下的临时路径，如果重新启用另一个终端需要重新配置

执行 `idf.py gen-bmgr-config --help` 验证组件路径是否设置成功，如果出现以下打印则表示组件路径配置成功

```
hello_world idf.py gen-bmgr-config --help
Usage: idf.py gen-bmgr-config [OPTIONS]

  Generate ESP Board Manager configuration files for board peripherals and devices.

  This command generates C configuration files based on YAML configuration files in the board directories. It can process peripherals,
  devices, generate Kconfig menus, and update SDK configuration automatically.

  Usage:     idf.py gen-bmgr-config -b <board_name>      # Specify board by name

      idf.py gen-bmgr-config -b <board_index>     # Specify board by index number

      idf.py gen-bmgr-config --list-boards        # List all available boards

      idf.py gen-bmgr-config -x                   # Clean generated files created by gen-bmgr-config

      idf.py gen-bmgr-config --clean              # Clean generated files created by gen-bmgr-config (same as -x)

  Note: When using idf.py, you must use the -b option to specify the board. For positional argument support, run the script directly:
  python gen_bmgr_config_codes.py <board>

  For more examples, see the README.md file.

Options:
  -C, --project-dir PATH    Project directory.
  -l, --list-boards         List all available boards and exit
  -b, --board TEXT          Specify board name or index number (bypasses sdkconfig reading)
  -c, --customer-path TEXT  Path to customer boards directory (use "NONE" to skip)
  -x, --clean               Clean generated .c and .h files, and reset CMakeLists.txt and idf_component.yml
  --peripherals-only        Only process peripherals (skip devices)
  --devices-only            Only process devices (skip peripherals)
  --kconfig-only            Generate Kconfig menu system for board and component selection (default enabled)
  --log-level TEXT          Set the log level (DEBUG, INFO, WARNING, ERROR)
  --help                    Show this message and exit.
hello_world
```

如果出现以下报错，可以使用指令 `echo $IDF_EXTRA_ACTIONS_PATH` 打印路径检查是否配置正确，或是查看路径 `managed_components` 确保组件是否已被成功拉到本地

```
hello_world idf.py gen-bmgr-config --help
Usage: idf.py gen-bmgr-config [OPTIONS]

  Execute targets that are not explicitly known to idf.py

Options:
  --help  Show this message and exit.
hello_world
```

- IDF action 扩展自动发现功能从 ESP-IDF v6.0 开始可用，从 IDF v6.0 开始无需设置 `IDF_EXTRA_ACTIONS_PATH`，因为它会自动发现 IDF action 扩展

第四步：选择开发板

执行 `idf.py gen-bmgr-config -l` 检查可用的开发板

```
hello_world idf.py gen-bmgr-config -l
Executing action: gen-bmgr-config
ℹ️ Using user-specified project directory: /home/liujinhong/esp/proj/learn/hello_world
📋 Version Information:
   • Component Version: 0.4.8
   • Git Commit: 32cfd567 (2025-12-18)
   • Generation Time: 2025-12-18 20:45:52

ℹ️ Using IDF_EXTRA_ACTIONS_PATH as root directory: /home/liujinhong/esp/commit/esp-gmf/packages/esp_board_manager
ESP Board Manager - Board Listing
======================================
   Scanning main boards: /home/liujinhong/esp/commit/esp-gmf/packages/esp_board_manager/boards
   Scanning component boards (recursive): /home/liujinhong/esp/proj/learn/hello_world/components
Scanning all directories in component path: /home/liujinhong/esp/proj/learn/hello_world/components
   No customer boards path specified
✅ Found 10 board(s):

ℹ️ Main Boards:
  [1] dual_eyes_board_v1_0
  [2] echoear_core_board_v1_0
  [3] echoear_core_board_v1_2
  [4] esp32_c5_spot
  [5] esp32_p4_function_ev
  [6] esp32_s3_korvo2_v3
  [7] esp32_s3_korvo2l
  [8] esp_box_3
  [9] lyrat_mini_v1_1
  [10] m5stack_cores3

✅ Board listing completed!
hello_world
```

- 打印出的开发板为当前仓库已经适配了部分设备的开发板，具体的支持情况可以查看 `managed_components/espressif__esp_board_manager/README_CN.md` 中的 **支持的板级** 一节的描述;
- 查看 `managed_components/espressif__esp_board_manager/boards/xxx/board_devices.yml` 和 `managed_components/espressif__esp_board_manager/boards/xxx/board_peripherals.yml` 可以获取板子详细的设备和外设配置

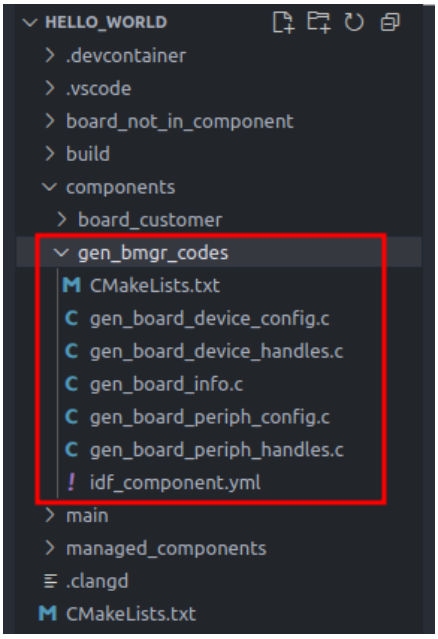
这条命令只是打印可用的开发板，选中开发板之后可以执行 `idf.py gen-bmgr-config -b xxx` 来生成开发板的配置文件，执行命令后打印如下。

> **注意:** 在 `esp_board_manager` v0.5 之后的版本中，支持通过索引来选择开发板，例如 `idf.py gen-bmgr-config -b echoear_core_board_v1_2` 和 `idf.py gen-bmgr-config -b 3` 选中的开发板是一样的。

```
hello_world idf.py gen-bmgr-config -b esp32_s3_korvo2_v3
Executing action: gen-bmgr-config
ℹ️ Using user-specified project directory: /home/liujinhong/esp/proj/learn/hello_world
📋 Version Information:
   • Component Version: 0.4.2
   • Git Commit: Unknown (Unknown)
   • Generation Time: 2025-11-10 13:37:51

ℹ️ Using IDF_EXTRA_ACTIONS_PATH as root directory: managed_components/espressif__esp_board_manager
=== Board Manager Configuration Generator ===
⚙️ Step 1/8: Scanning board directories...
   Scanning main boards: managed_components/espressif__esp_board_manager/boards
   No customer boards path specified
✅ Found 9 boards: ['echoear_core_board_v1_0', 'echoear_core_board_v1_2', 'esp_box_3', 'esp32_s3_korvo2_v3', 'esp32_c5_spot', 'lyrat_
mini_v1_1', 'esp32_s3_korvo2l', 'esp32_p4_function_ev', 'dual_eyes_board_v1_0']
⚙️ Step 2/8: Reading board selection...
✅ Using board from command line: esp32_s3_korvo2_v3
✅ Board path: managed_components/espressif__esp_board_manager/boards/esp32_s3_korvo2_v3
⚙️ Step 3/8: Finding board configuration files...
   Using board configuration files for 'esp32_s3_korvo2_v3':
   Peripherals: managed_components/espressif__esp_board_manager/boards/esp32_s3_korvo2_v3/board_peripherals.yaml
   Devices: managed_components/espressif__esp_board_manager/boards/esp32_s3_korvo2_v3/board_devices.yaml
✅ Configuration files found:
⚙️ Step 4/8: Processing peripherals...
   Successfully parsed 6 peripherals from managed_components/espressif__esp_board_manager/boards/esp32_s3_korvo2_v3/board_peripherals
.yaml
✅ Peripheral processing completed: 5 types found
⚙️ Step 5/8: Processing devices and dependencies...
   Scanning for source files in board directory: managed_components/espressif__esp_board_manager/boards/esp32_s3_korvo2_v3
   Parsing device YAML file...
   Loaded 7 devices from managed_components/espressif__esp_board_manager/boards/esp32_s3_korvo2_v3/board_devices.yaml
✅ Successfully validated 7 devices
✅ Device processing completed: 6 types found
⚙️ Step 6/8: Generating Kconfig menu system...
✅ Generated Kconfig for 5 peripherals: ['gpio', 'i2c', 'i2s', 'ledc', 'spi']
✅ Generated Kconfig for 11 devices: ['audio_codec', 'camera', 'custom', 'display_lcd_spi', 'fatfs_sdcard', 'fatfs_sdcard_spi', 'fs_s
piffs', 'gpio_ctrl', 'gpio_expander', 'lcd_touch_i2c', 'ledc_ctrl']
✅ Writing Kconfig file to: managed_components/espressif__esp_board_manager/gen_codes/Kconfig.in
✅ Kconfig generation completed successfully
⚙️ Step 7/8: Updating SDK configuration...
   Auto-config enabled via sdkconfig (or not set)
   Successfully updated sdkconfig with 7 changes
✅ Updated 7 sdkconfig features
⚙️ Step 8/8: Writing board information and setting up components...
   Added board source directory: ../../managed_components/espressif__esp_board_manager/boards/esp32_s3_korvo2_v3
   Created CMakeLists.txt: /home/liujinhong/esp/proj/learn/hello_world/components/gen_bmgr_codes/CMakeLists.txt
   Created idf_component.yml: /home/liujinhong/esp/proj/learn/hello_world/components/gen_bmgr_codes/idf_component.yml
✅ Board source files will be compiled from: managed_components/espressif__esp_board_manager/boards/esp32_s3_korvo2_v3
✅ Generated files directly to components/gen_bmgr_codes completed successfully!
✅ === Board configuration generation completed successfully for board: esp32_s3_korvo2_v3 ===
✅ ESP Board Manager configuration generation completed successfully!
hello_world
```

执行命令后可以在 `components/gen_bmgr_codes` 路径看到生成的配置文件



执行到这一步，已经将所需的开发板相关的配置代码生成到本地工程路径下，在代码中的具体使用方法可以参考

```c
#include <stdio.h>
#include "esp_log.h"
#include "esp_err.h"
#include "esp_board_manager_includes.h"

static const char *TAG = "MAIN";

void app_main(void)
{
    // 初始化板级管理器，这将自动初始化所有外设和设备
    ESP_LOGI(TAG, "Initializing board manager...");
    int ret = esp_board_manager_init();
    if (ret != ESP_OK) {
        ESP_LOGE(TAG, "Failed to initialize board manager");
        return;
    }
    // 获取设备句柄，根据 esp_board_manager/boards/YOUR_TARGET_BOARD/board_devices.yaml 中的设备
    dev_display_lcd_spi_handles_t *lcd_handle;
    ret = esp_board_manager_get_device_handle("display_lcd", &lcd_handle);
    if (ret != ESP_OK) {
        ESP_LOGE(TAG, "Failed to get LCD device");
        return;
    }
    // 获取设备配置，根据 esp_board_manager/boards/YOUR_TARGET_BOARD/board_devices.yaml 中的设备
    dev_audio_codec_config_t *device_config;
    ret = esp_board_manager_get_device_config("audio_dac", &device_config);
    if (ret != ESP_OK) {
        ESP_LOGE(TAG, "Failed to get device config");
        return;
    }
    // 打印板子信息
    esp_board_manager_print_board_info();
    // 打印板级管理器状态
    esp_board_manager_print();
    // 使用句柄...
}
```

- 有关具体的外设和设备的使用方法可以参考 `managed_components/espressif__esp_board_manager/test_apps/main/` 目录中的测试应用程序

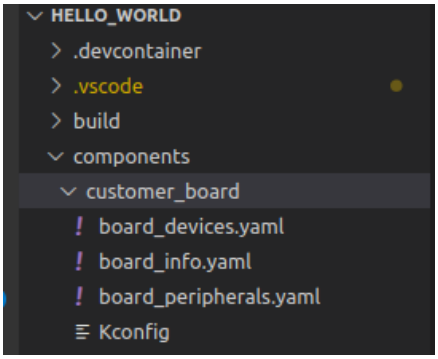
如何自定义开发板

board_manager 组件支持自定义开发板，相关配置可以放到默认路径 `{YOUR_PROJECT_PATH}/components/` 下，也可以放到其他路径下，具体使用方法可以参考以下步骤：

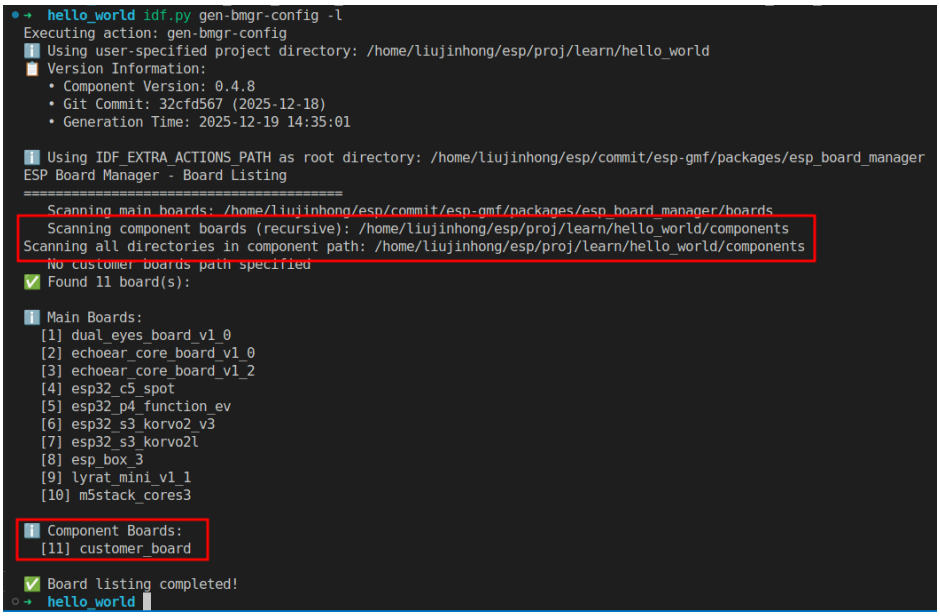
假设自定义板子的命名为 `customer_board`

默认路径

将自定义开发板的相关配置放到 `{YOUR_PROJECT_PATH}/components/` 下

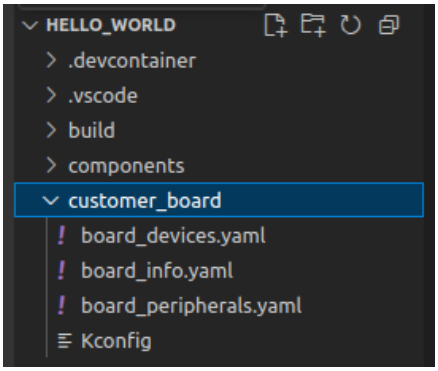


此时执行脚本 `idf.py gen-bmgr-config -l` 就可以看到自己添加的开发板

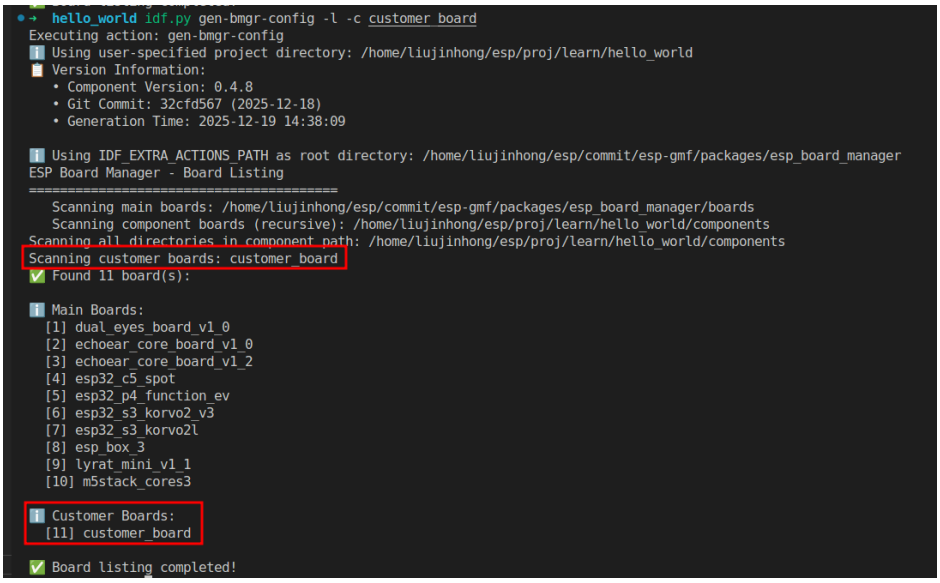


其他路径

如果将自定义开发板相关的配置文件放到别的目录，则需要**在执行命令时添加 `-c xxx` 来指定路径**，假设开发板配置文件被放到 `{YOUR_PROJECT_PATH}/xxx` 中



则需要执行命令 `idf.py gen-bmgr-config -l -c customer_board` ，此时可以看到自己添加的 `customer_board` 开发板



自定义设备

esp_board_manager 支持添加自定义设备，具体使用方法可以参考以下步骤：

- 将设备的 `init` 和 `deinit` 函数代码放到开发板路径下，通过 `CUSTOM_DEVICE_IMPLEMENT("device_name", init_func, deinit_func)` 将设备注册到板级管理器中
- 在开发板路径 `board_devices.yaml` 中添加自定义设备的配置，其中 `type` 需要配置成 `custom`
- 执行命令 `idf.py gen-bmgr-config -b xxx` 生成开发板的配置代码后，在 `components/gen_bmgr_codes` 路径下会生成 `gen_board_device_custom.h` 头文件，供应用程序使用
- 可以参考 `esp_board_manager/boards/esp32_s3_korvo2l` 中的实现方法

使用技巧

- 用户可以复制一个相似的仓库内置开发板到自己的工程目录，然后根据自己所使用开发板的pin和驱动来修改 YML 文件内容
- `components/gen_bmgr_codes/gen_board_device_config.c` 和 `components/gen_bmgr_codes/gen_board_periph_config.c` 为根据 `boards/xxx` 路径下的 `yml` 文件生成的配置代码，如果需要修改一些设备或外设的配置进行调试可以直接修改此处的代码，不过请注意，**此处改动仅为临时生效，测试功能正常后请将改动应用到相应的 `yml` 文件，因为配置代码的生成是基于 `yml` 文件，每次执行 `idf.py gen-bmgr-config -b xxx` 命令会根据 `yml` 文件的内容生成新的配置代码并覆盖 `components/gen_bmgr_codes` 下的所有文件**
- 板子的 `YAML` 配置文件尽可能使用了驱动的原始参数，比如：有哪些配置项，枚举值和参数的取值范围都参考驱动而来，详细的内容请参考 `esp_board_manager/devices` 和 `esp_board_manager/peripherals` 中的YAML文件

注意事项

- 开发板的初始化与 `components/gen_bmgr_codes` 路径下生成的配置文件有关，目前暂不支持通过 `menuconfig` 来选择开发板或是具体的 `periph` 和 `device`

- 部分 `device` 的初始化依赖 `factory` 函数，需要用户自行在开发板的 `setup_device.c` 中进行实现，可以参考代码 `esp_board_manager/boards/esp32_s3_korvo2_v3/setup_device.c`
- 本文为简单的 `quick start` ，旨在帮助用户快速上手使用 `esp_board_manager` ，对于详细的使用方法，建议阅读 esp_board_manager/README_CN.md