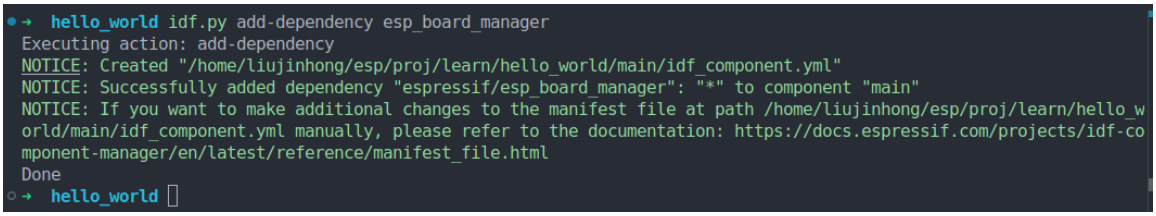
ESP_BOARD_MANAGER Quick Start Guide

Date	Version	Release Notes	Component Version
2025-11-10	v1.0	Initial release of ESP_BOARD_MANAGER Quick Start Guide	0.4.x
2025-12-18	v1.1	Adapted to new board path scanning and selection mechanism	0.5.x

Table of Contents

Step 1: Add the Component to Your Project

Execute the command: `idf.py add-dependency esp_board_manager` to add the `esp_board_manager` component to your project. You will see the following output:



After execution, the component dependency will be automatically added to the `main/idf_component.yml` file:

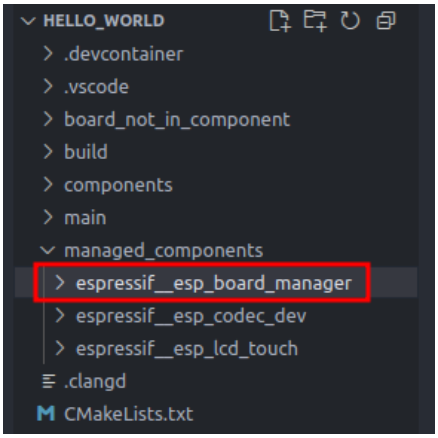
idf_component.yml

```yaml
## IDF Component Manager Manifest File
dependencies:
  ## Required IDF version
  idf:
    version: '>=4.1.0'
  # # Put list of dependencies here
  # # For components maintained by Espressif:
  # component: "~1.0.0"
  # # For 3rd party components:
  # username/component: ">=1.0.0,<2.0.0"
  # username2/component2:
  #   version: "~1.0.0"
  #   # For transient dependencies `public` flag can be set.
  #   # `public` flag doesn't have an effect dependencies of the `main` component.
  #   # All dependencies of `main` are public by default.
  #   public: true
  espressif/esp_board_manager: '*'
```

Step 2: Pull the Component Locally

Execute the command `idf.py set-target esp32s3` to pull the `esp_board_manager` component locally. After execution, you can see the `espressif__esp_board_manager` component in the `managed_components` directory.

- The target chip selected here can be modified according to the actual chip used in the project.



Step 3: Set Board Manager Component Path and Verify

Execute `export IDF_EXTRA_ACTIONS_PATH=managed_components/espressif__esp_board_manager` to set the component path to the locally pulled `esp_board_manager` component.

- The `IDF_EXTRA_ACTIONS_PATH` path configured via the export command is a temporary path in the current terminal. If you open another terminal, you need to reconfigure it.

Execute `idf.py gen-bmgr-config --help` to verify if the component path is set successfully. If the following output appears, it indicates successful configuration:

```
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
```

If the following error appears, you can use the command `echo $IDF_EXTRA_ACTIONS_PATH` to print the path and check if it is configured correctly, or check the `managed_components` path to ensure the component has been successfully pulled locally:

```
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 extension auto-discovery is available from ESP-IDF v6.0. Starting from IDF v6.0, there is no need to set `IDF_EXTRA_ACTIONS_PATH` as it will automatically discover IDF action extensions.

Step 4: Select Development Board

Execute `idf.py gen-bmgr-config -l` to check available development boards:

```
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
```

- The printed development boards are those that have been partially adapted with devices in the current repository. For specific support details, refer to the **Supported Boards** section in `managed_components/espressif__esp_board_manager/README_CN.md`.
- Check `managed_components/espressif__esp_board_manager/boards/xxx/board_devices.yml` and `managed_components/espressif__esp_board_manager/boards/xxx/board_peripherals.yml` to get detailed device and peripheral configurations for the board.

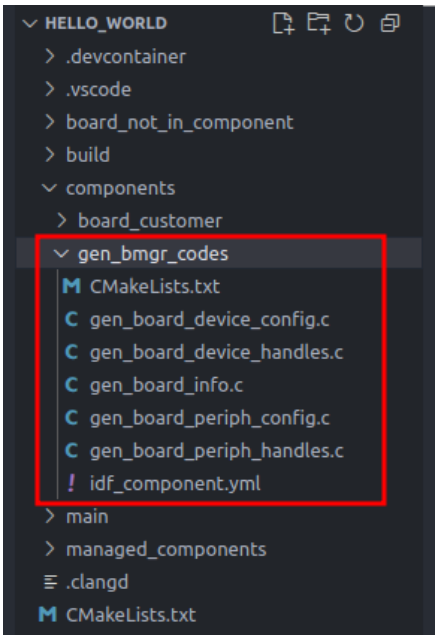
This command only prints available development boards. After selecting a development board, you can execute `idf.py gen-bmgr-config -b xxx` to generate the board's configuration files. After executing the command, the output will be as follows:

> **Note:** In versions after `esp_board_manager` v0.5, selecting development boards by index is supported. For example, `idf.py gen-bmgr-config -b echoear_core_board_v1_2` and `idf.py gen-bmgr-config -b 3` select the same development board.

```
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
```

After executing the command, you can see the generated configuration files in the `components/gen_bmgr_codes` path:



At this step, the required development board-related configuration code has been generated in the local project path. For specific usage methods in code, you can refer to:

```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)
{
    // Initialize board manager, which will automatically initialize all peripherals and de
    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;
    }
    // Get device handle, based on device naming in esp_board_manager/boards/YOUR_TARGET_BO
    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;
    }
    // Get device configuration, based on device naming in esp_board_manager/boards/YOUR_TA
    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;
    }
    // Print board information
    esp_board_manager_print_board_info();
    // Print board manager status
    esp_board_manager_print();
    // Use handle...
}
```

- For specific usage methods of peripherals and devices, refer to the test applications in the `managed_components/espressif__esp_board_manager/test_apps/main/` directory.

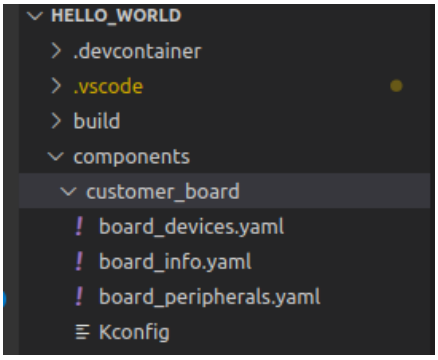
How to Customize a Development Board

The board_manager component supports custom development boards. Related configurations can be placed in the default path `{YOUR_PROJECT_PATH}/components/` or in other paths. For specific usage methods, refer to the following steps:

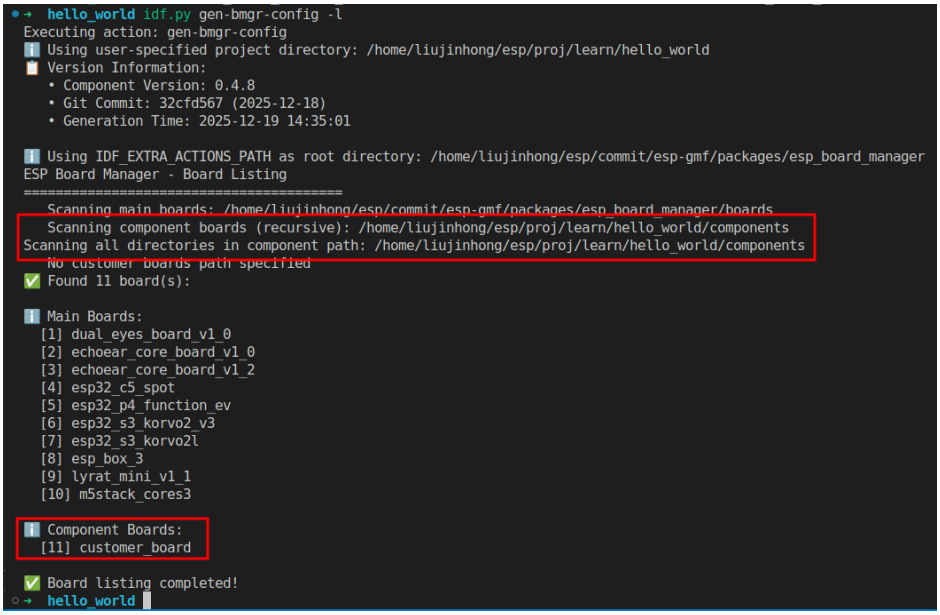
Assuming the custom board is named `customer_board`

Default Path

Place custom development board configurations in `{YOUR_PROJECT_PATH}/components/`:

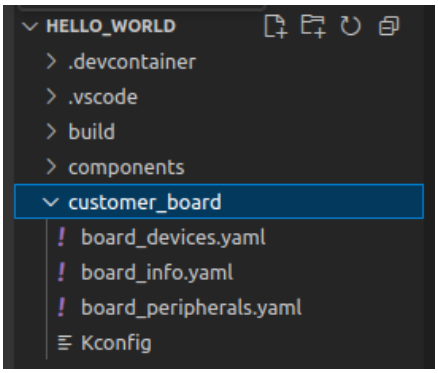


Now execute the script `idf.py gen-bmgr-config -l` to see your added development board:

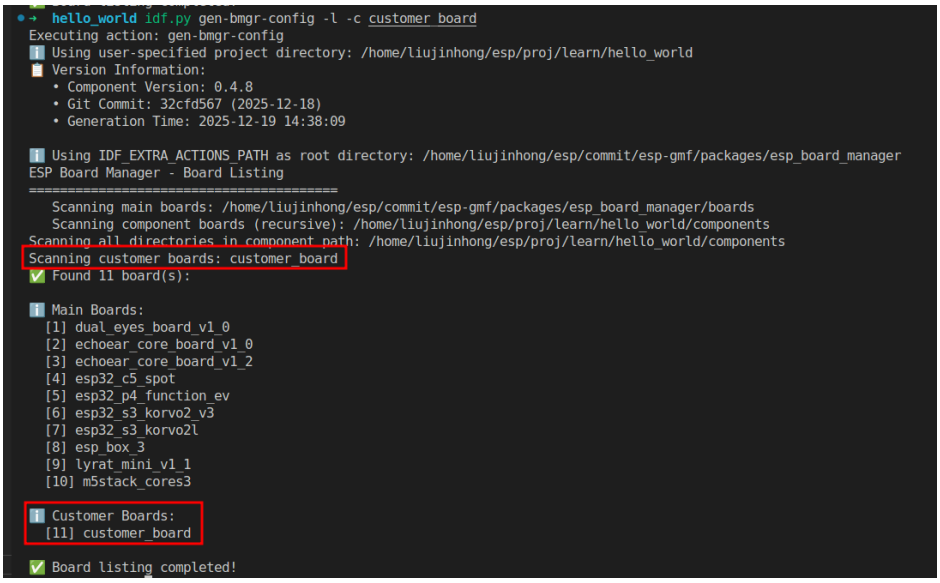


Other Paths

If custom development board configuration files are placed in other directories, you need to **add `-c xxx` when executing the command to specify the path**. Assuming the development board configuration files are placed in `{YOUR_PROJECT_PATH}/xxx`:



Then execute the command `idf.py gen-bmgr-config -l -c customer_board`. Now you can see your added `customer_board` development board:



Custom Devices

esp_board_manager supports adding custom devices. For specific usage methods, refer to the following steps:

- Place the device's `init` and `deinit` function code in the development board path, and register the device with the board manager using `CUSTOM_DEVICE_IMPLEMENT("device_name", init_func, deinit_func)`.
- Add the configuration for the custom device in the development board path `board_devices.yaml`, where the `type` needs to be configured as `custom`.
- After executing the command `idf.py gen-bmgr-config -b xxx` to generate the development board's configuration code, the `gen_board_device_custom.h` header file will be generated in the `components/gen_bmgr_codes` path for use by the application.
- You can refer to the implementation method in `esp_board_manager/boards/esp32_s3_korvo2l`.

Usage Tips

- Users can copy a similar repository-built development board to their project directory, then modify the YML file content according to the pins and drivers used on their development board.
- `components/gen_bmgr_codes/gen_board_device_config.c` and `components/gen_bmgr_codes/gen_board_periph_config.c` are configuration codes generated based on the `yml` files in the `boards/xxx` path. If you need to modify some device or peripheral configurations for debugging, you can directly modify the code here. However, please note that **changes here are only temporarily effective. After testing the functionality, please apply the changes to the corresponding `yml` files, because configuration code generation is based on `yml` files. Each time you execute the `idf.py gen-bmgr-config -b xxx` command, new configuration code will be generated based on the `yml` file content and overwrite all files under `components/gen_bmgr_codes`.**

- The board's `YAML` configuration files use the driver's original parameters as much as possible. For example: configuration items, enumeration values, and parameter value ranges are referenced from the driver. For detailed content, refer to the YAML files in `esp_board_manager/devices` and `esp_board_manager/peripherals`.

Important Notes

- Development board initialization is related to the configuration files generated in the `components/gen_bmgr_codes` path. Currently, selecting development boards or specific `periph` and `device` through `menuconfig` is not supported.
- Initialization of some `device`s depends on `factory` functions, which need to be implemented by users in the development board's `setup_device.c`. Refer to the code `esp_board_manager/boards/esp32_s3_korvo2_v3/setup_device.c`.
- This document is a simple `quick start` aimed at helping users quickly get started with `esp_board_manager`. For detailed usage methods, it is recommended to read esp_board_manager/README.md.