Start a Project

Copy Sample

Now you are ready to prepare your application for ESP32. You can start with get-started/hello_world project from examples directory in IDF. Copy get-started/hello_world to ~/esp directory: Linux and macOS

```
cd ~/development/esp32/projects
cp -r $IDF_PATH/examples/get-started/hello_world .
```

Configure

This is a one time initial setup for a new project. Navigate to the project directory. Set ESP32 chip as the target and run the project configuration utility menuconfig.

```
cd ~/development/esp32/projects/hello_world
idf.py set-target esp32
idf.py menuconfig
```

Setting the target with idf.py set-target {IDF_TARGET} should be done once, after opening a new project. If the project contains some existing builds and configuration, they will be cleared and initialized. The target may be saved in environment variable to skip this step at all. If the previous steps have been done correctly, the Espressif IOT menu appears. You are using this menu to set up project specific variables, e.g. Wi-Fi network name and password, the processor speed, etc. Setting up the project with menuconfig may be skipped for “hello_word”. This example will run with default configuration.

Build

Build the project by running:

```
idf.py build
```

This command will compile the application and all ESP-IDF components, then it will generate the bootloader, partition table, and application binaries.

```
$ idf.py build
Running cmake in directory /path/to/hello_world/build
Executing "cmake -G Ninja --warn-uninitialized /path/to/hello_world"...
Warn about uninitialized values.
-- Found Git: /usr/bin/git (found version "2.17.0")
-- Building empty aws_iot component due to configuration
-- Component names: ...
```

```
-- Component paths: ...

... (more lines of build system output)

[527/527] Generating hello-world.bin
esptool.py v2.3.1

Project build complete. To flash, run this command:
../../../components/esptool_py/esptool/esptool.py -p (PORT) -b 921600
write_flash --flash_mode dio --flash_size detect --flash_freq 40m 0x10000
build/hello-world.bin  build 0x1000 build/bootloader/bootloader.bin 0x8000
build/partition_table/partition-table.bin
or run 'idf.py -p PORT flash'
```

If there are no errors, the build will finish by generating the firmware binary .bin file.

Flash

Flash the binaries that you just built onto your ESP32 board by running:

```
idf.py -p PORT [-b BAUD] flash
```

Replace PORT with your ESP32 board’s serial port name from Step 6. Connect Your Device.

You can also change the flasher baud rate by replacing BAUD with the baud rate you need. The default baud rate is 460800.

For more information on idf.py arguments, see idf.py.