Eclipse with the ESP32

California Plug Load Research Center
Winter 2018

Jerry Lee & Justin Le

March 20, 2018

Contents

1 Introduction 2

2 Download and Setup Toolchain 2
2.1 Download Toolchain 2
2.2 Extract Toolchain 2
2.3 Run Toolchain Environment 2

3 Download and Setup ESP-IDF 2
3.1 Download ESP-IDF 2
3.2 Setup Path to ESP-IDF 3

4 Establishing a Serial Connection with the ESP32 4
4.1 Connect 4
4.2 Serial 4

5 Install Eclipse 4

6 Create Workspace and Import Project 4
6.1 Workspace & Template 4
6.2 Configure Settings 5

7 Eclipse Setup 6

8 Building/Flashing from Eclipse 12

9 Results 14

1 Introduction

In this guide, you will be instructed on how to use Eclipse IDE with the ESP32, rather than using the Arduino IDE. This guide is based on having a Windows host so for Mac OS and Linux users, the steps may be different.

2 Download and Setup Toolchain

The toolchain being installed contains programs to compile and build the application.

2.1 Download Toolchain

Click here or use this link to directly download the toolchain:
`https://dl.espressif.com/dl/esp32_win32_msys2_environment_and_toolchain-20171123.zip`

2.2 Extract Toolchain

Once downloaded, unzip the file to C:\ and it should create a directory named **msys32** with a pre-prepared environment. To do this, right click the zip file, select "Extract files...", and select C: \ as the destination path to ensure it is extracted to the correct place. Make sure the file is extracted correctly and is not the unzipped zip which contains msys32 inside it.

2.3 Run Toolchain Environment

To check that the toolchain has been successfully downloaded and installed, run *C:\msys32\mingw32.exe*. The environment in the window that appears is a bash shell that we will be using to issue commands.

3 Download and Setup ESP-IDF

3.1 Download ESP-IDF

Now that the toolchain is downloaded, the next step is to get the ESP32 specific API/libraries that will be needed. They are provided in the ESP_IDF Github repository.
Link: `https://github.com/espressif/esp-idf`
To get the library, open the MsYS32 MINGW32 terminal by running *C:\msys32\mingw32.exe* that we had earlier. Create a folder by typing the following command:

```
mkdir esp
```

Then go into the directory and clone the ESP-IDF Repository

```
cd esp
git clone –recursive https://github.com/espressif/esp-idf.git
```

ESP-IDF will be downloaded into ~/esp/esp-idf after the command is issued.

Note: The esp folder should by default by saved in *C:\msys32\home\esp*
If your path is not this by default, manually move the esp folder to this directory.

3.2 Setup Path to ESP-IDF

The next step is to setup the path to ESP-IDF so the toolchain programs can access it. The toolcahin programs downloaded and installed access ESPIDF suing the IDF_PATH environment variable. If this variable is not set up on your pc, the projects will not build. The setting may be done manually each time the PC is rebooted or can be permanently done by defining the IDF_PATH in the user profile. The user profile scripts are contained in a folder located in *C:\msys32\etc \profile.d* \. Each time a MYSYS2 window is open, the scripts are executed.

First, you will need to create a new script file in *C:\msys32\etc \profile.d* directory and name it *export_idf_path.sh.* Do this by running *C:\msys32\mingw32.exe* which will open the MSYS MINGW32 terminal. There you will issue the following commands:

```
cd esp
cd espidf
cd /etc/profile.d
```

Use vim to create a file and edit it by issuing the following command:

```
vim export_idf_path.sh
```

When inside Vim within the terminal, type ”i” (without quotation marks) to enter INSERT Mode, which will allow you to write to the file. Here you will insert th e path to ESPIDF to the IDF_PATH environment variable by typing the following in the terminal text editor:

```
export IDF_PATH=”C:/msys32/home/yourusername/esp/espidf”
```

Be sure to replace yourusername with your computer user name. To save and exit the in-terminal editor, press ESC (escape key on keyboard) and type ”:wq” (colon, letter w, letter q, no quotation marks). Now close and reopen the MSYS32 terminal and issue the following command:

```
printenv IDF_PATH
```

When issue this command, the terminal should print out:

C:\msys32\home \yourusername\esp \espidf.

4 Establishing a Serial Connection with the ESP32

4.1 Connect

To establish a serial connection between the ESP and computer, connect the ESP32 devkit to the computer using a USB to micro-USB cable. Open device-manager by right-clicking on the windows taskbar. Scroll down through the manager and search for a label named Ports. You should be able to see the ESP connected. Note what port the ESP is connected to.

4.2 Serial

Open PuTTY. Under the Connection label, click on Serial. Type in the port the ESP is connected to and change the baud rate to 115200. Next, click on the Session label (first label from the top) and change the connection type to Serial. Click *Open* and you should see a log display by the ESP32. If what you see is legible, then it should be working properly.

5 Install Eclipse

If you don't already have Eclipse, download and install it from here.

Direct Link: http://www.eclipse.org/downloads/

In the Eclipse installer, choose the "Eclipse IDE for C/C++ Developers" option and continue to finish the installation.

6 Create Workspace and Import Project

All projects that are developed for ESP32 using Eclipse need to follow the ESP-IDF template. This must be done before using Eclipse.

6.1 Workspace & Template

The following steps use an ESP-IDF template to set up a "Hello World" program.

Open the MSYS32 terminal and issue the following command:

```
cd "C:\Users\your-user-name\Desktop"
mkdir workspace
```

This will make a folder "workspace" on your Desktop. The location of the folder can be changed as needed by changing the first command.] Next clone a ESP-IDF template in the "workspace" folder, in this case we will be cloning the "Hello World" template:

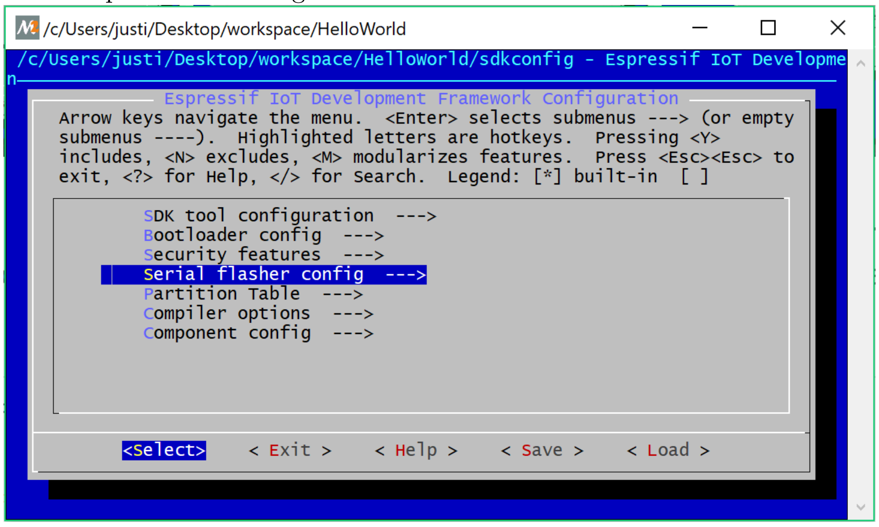
```
cd workspace
git clone https://github.com/espressif/esp-idf-template.git HelloWorld
```

6.2 Configure Settings

You now need to configure the SDK settings using the following command:

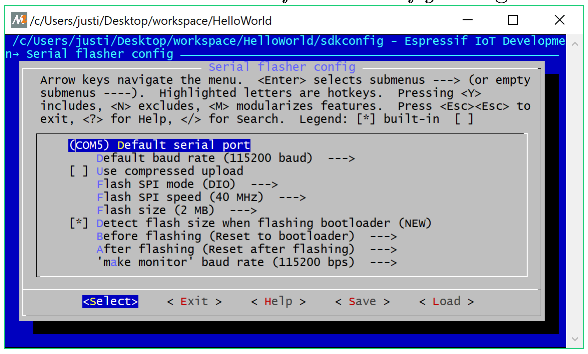
```
cd HelloWorld
make menuconfig
```

This will open the following menu:

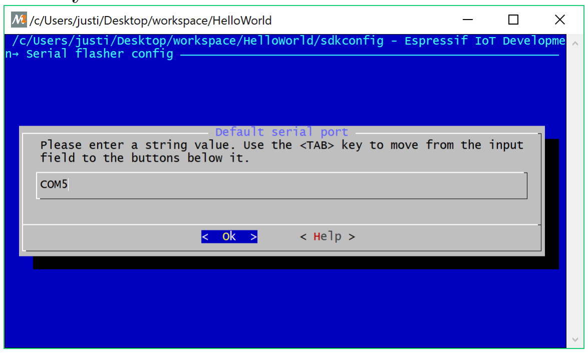


```
/c/Users/justi/Desktop/workspace/HelloWorld
 /c/Users/justi/Desktop/workspace/HelloWorld/sdkconfig - Espressif IoT Developme
n
               Espressif IoT Development Framework Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

          SDK tool configuration  --->
          Bootloader config  --->
          Security features  --->
          Serial flasher config  --->
          Partition Table  --->
          Compiler options  --->
          Component config  --->

      <Select>    < Exit >    < Help >    < Save >    < Load >
```

Select *Serial flasher config* using the Enter key.



```
/c/Users/justi/Desktop/workspace/HelloWorld
 /c/Users/justi/Desktop/workspace/HelloWorld/sdkconfig - Espressif IoT Developme
n→ Serial flasher config
               Serial flasher config
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

     (COM5) Default serial port
         Default baud rate (115200 baud)  --->
     [ ] Use compressed upload
         Flash SPI mode (DIO)  --->
         Flash SPI speed (40 MHz)  --->
         Flash size (2 MB)  --->
     [*] Detect flash size when flashing bootloader (NEW)
         Before flashing (Reset to bootloader)  --->
         After flashing (Reset after flashing)  --->
         'make monitor' baud rate (115200 bps)  --->

      <Select>    < Exit >    < Help >    < Save >    < Load >
```

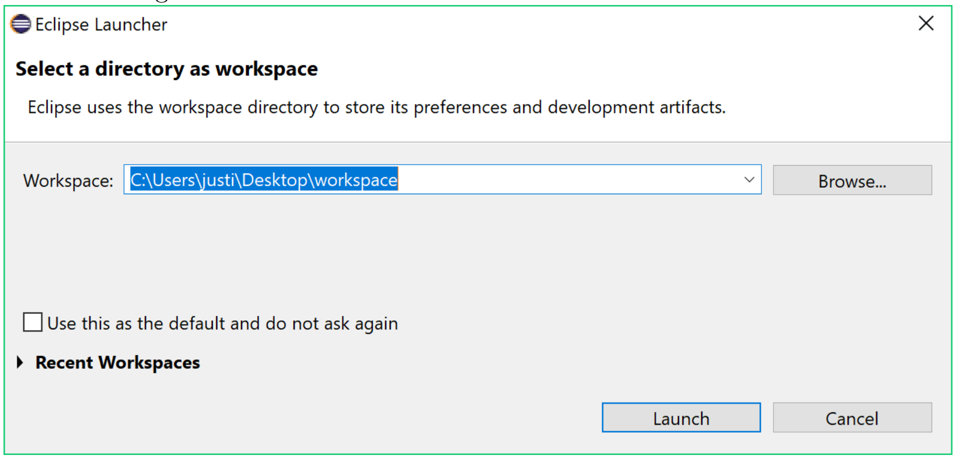


```
/c/Users/justi/Desktop/workspace/HelloWorld
 /c/Users/justi/Desktop/workspace/HelloWorld/sdkconfig - Espressif IoT Developme
n→ Serial flasher config

                     Default serial port
  Please enter a string value. Use the <TAB> key to move from the input
  field to the buttons below it.

  COM5

                     <  Ok  >      < Help >
```

Next select *Default serial port* and type in the port your ESP32 is connected to (Recall Section 4.1) and press the Enter key to select Ok. Use the arrow keys to select Save then Ok to save it to the default filename. You can now select Exit then Exit to quit the SDK menu.

7 Eclipse Setup

Now boot up eclipse by running eclipse.exe in your eclipse folder. You will see the following menu:



Eclipse Launcher

Select a directory as workspace

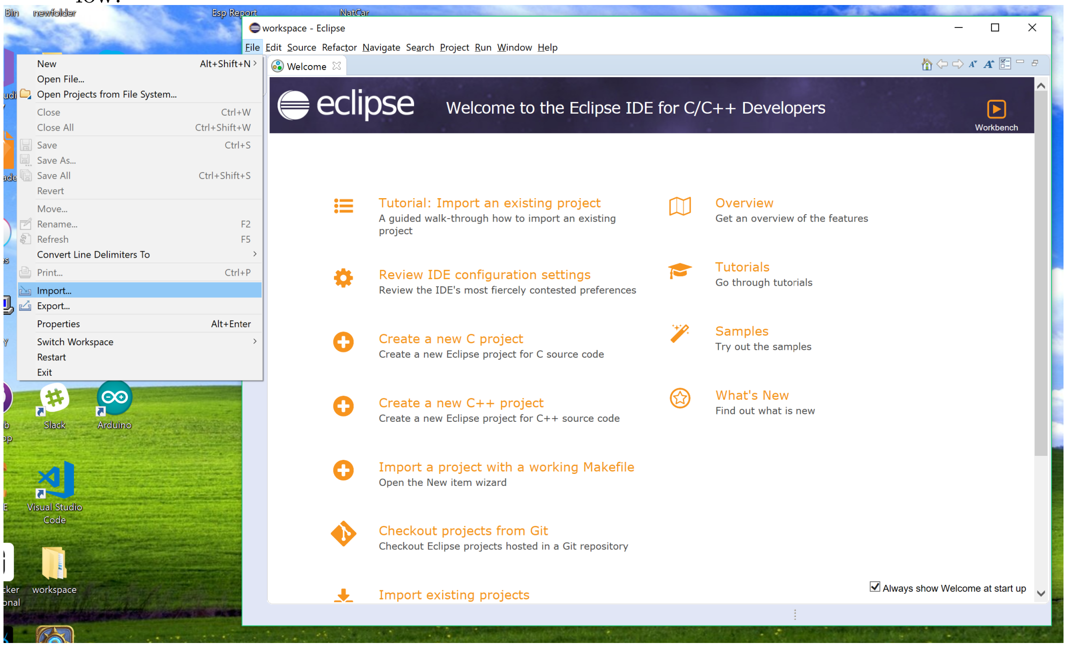
Eclipse uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Users\justi\Desktop\workspace

☐ Use this as the default and do not ask again

Recent Workspaces

Use Browse to select the workspace you created in Section 6.1 then select launch. Once eclipse opens, go to the top left and select File then Import as shown below:



workspace - Eclipse

Welcome to the Eclipse IDE for C/C++ Developers

Tutorial: Import an existing project
A guided walk-through how to import an existing project

Review IDE configuration settings
Review the IDE's most fiercely contested preferences

Create a new C project
Create a new Eclipse project for C source code

Create a new C++ project
Create a new Eclipse project for C++ source code

Import a project with a working Makefile
Open the New item wizard

Checkout projects from Git
Checkout Eclipse projects hosted in a Git repository

Import existing projects

Overview
Get an overview of the features

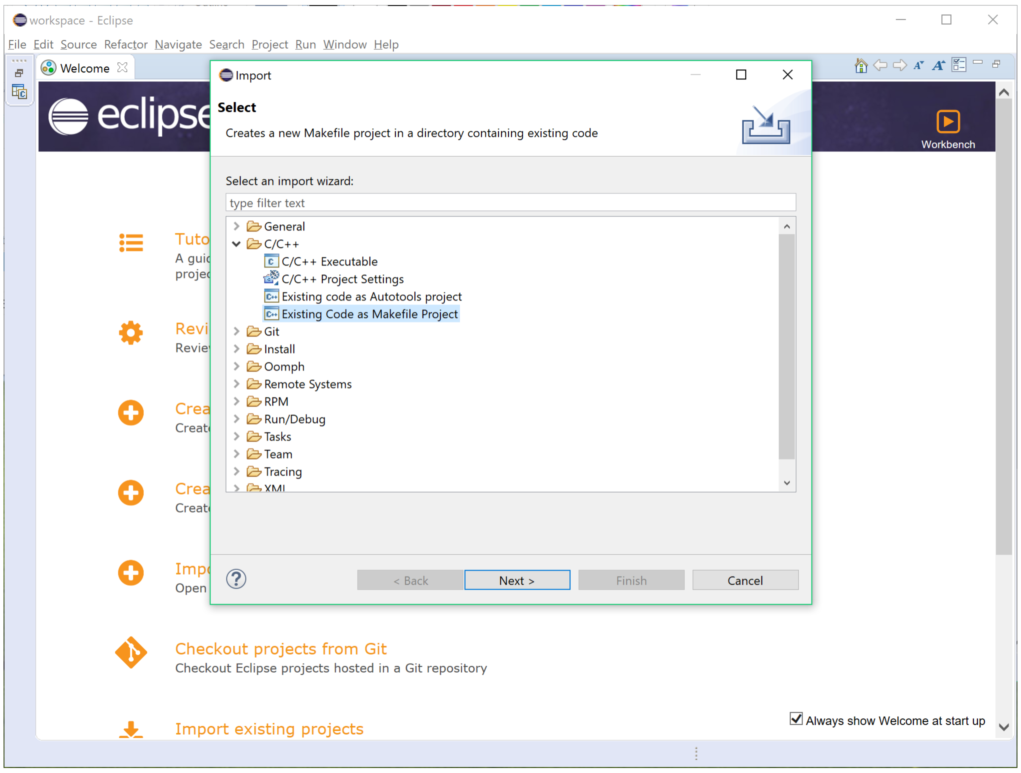
Tutorials
Go through tutorials

Samples
Try out the samples

What's New
Find out what is new

☑ Always show Welcome at start up

Open the directory labeled C/C++ and select *Existing Code as Makefile Project* as shown below:


workspace - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome
Workbench
< Back
Next >
Finish
Cancel
Always show Welcome at start up

Import

Select

Creates a new Makefile project in a directory containing existing code

Select an import wizard:

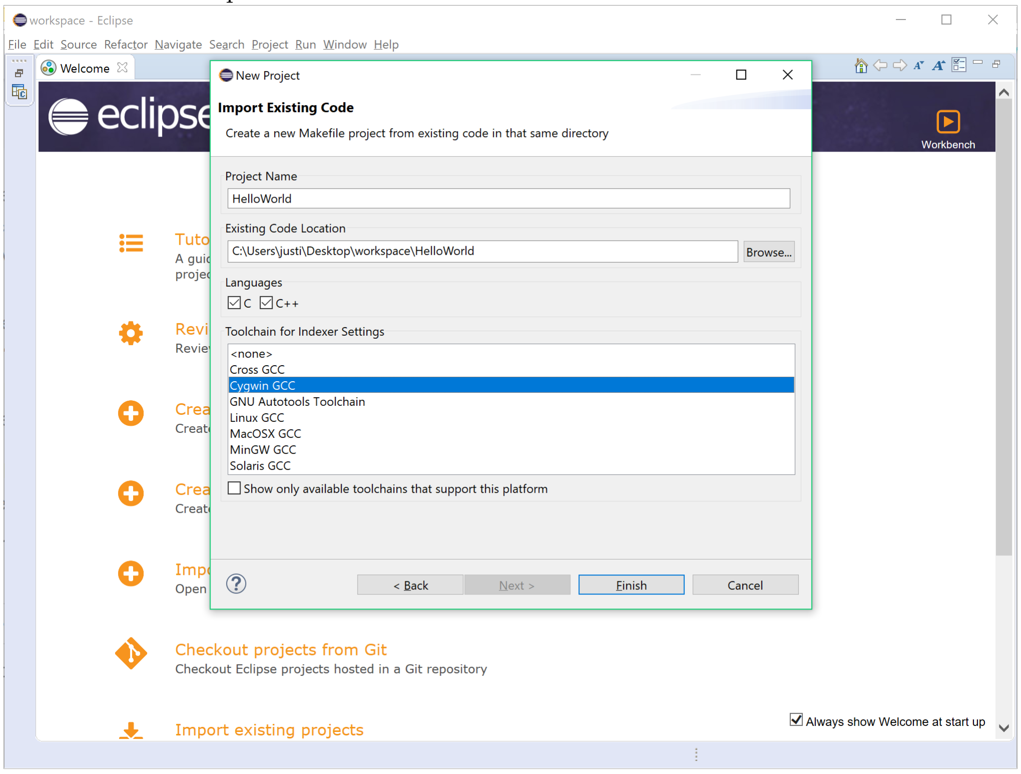
type filter text

- General
- C/C++
  - C/C++ Executable
  - C/C++ Project Settings
  - Existing code as Autotools project
  - Existing Code as Makefile Project
- Git
- Install
- Oomph
- Remote Systems
- RPM
- Run/Debug
- Tasks
- Team
- Tracing
- XML

Checkout projects from Git
Checkout Eclipse projects hosted in a Git repository

Import existing projects

Select Next to proceed to the next window.


workspace - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome
Workbench
Browse...
< Back
Next >
Finish
Cancel
Always show Welcome at start up

New Project

Import Existing Code

Create a new Makefile project from existing code in that same directory

Project Name: HelloWorld

Existing Code Location: C:\Users\justi\Desktop\workspace\HelloWorld

Languages: [✓ C] [✓ C++]

Toolchain for Indexer Settings:

- <none>
- Cross GCC
- [✓ Cygwin GCC]
- GNU Autotools Toolchain
- Linux GCC
- MacOSX GCC
- MinGW GCC
- Solaris GCC

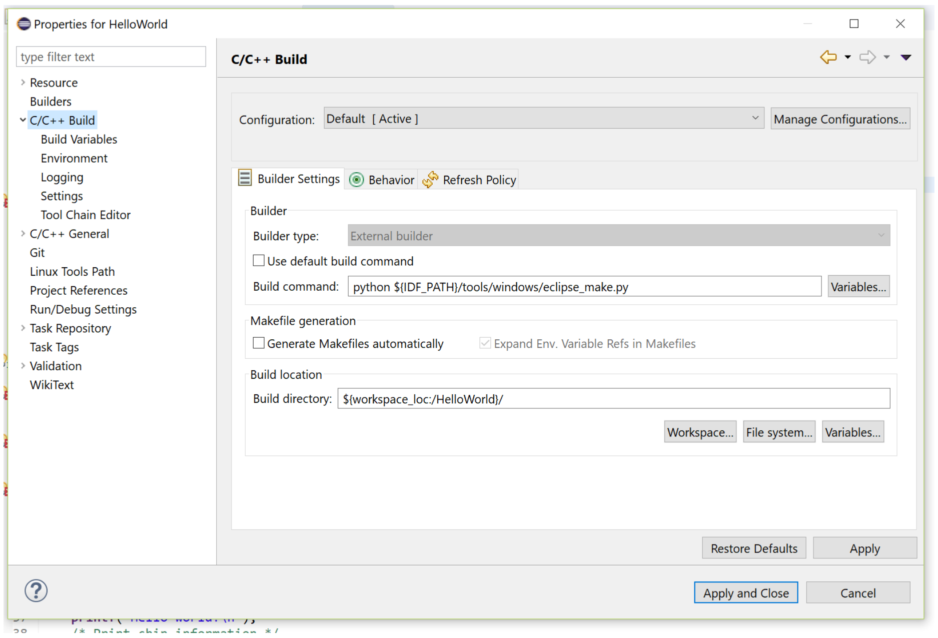
☐ Show only available toolchains that support this platform

Checkout projects from Git
Checkout Eclipse projects hosted in a Git repository

Import existing projects

Select Browse and select the ESP-IDF template folder that you cloned in the workspace folder, in this case is "HelloWorld". Next uncheck the box labeled *Show only available toolchains that support this platform* then select *Cygwin GCC* from the Toolchain for Indexer Settings dropdown menu. Now you can click Finish.

Note: Warnings can occur after selecting Finish but they do not affect the project so proceed as normal.


Properties for HelloWorld

type filter text

Resource
Builders
C/C++ Build
Build Variables
Environment
Logging
Settings
Tool Chain Editor
C/C++ General
Git
Linux Tools Path
Project References
Run/Debug Settings
Task Repository
Task Tags
Validation
WikiText

C/C++ Build

Configuration: Default [Active] Manage Configurations...

Builder Settings | Behavior | Refresh Policy

Builder
Builder type: External builder
☐ Use default build command
Build command: python ${IDF_PATH}/tools/windows/eclipse_make.py Variables...

Makefile generation
☐ Generate Makefiles automatically ✓ Expand Env. Variable Refs in Makefiles

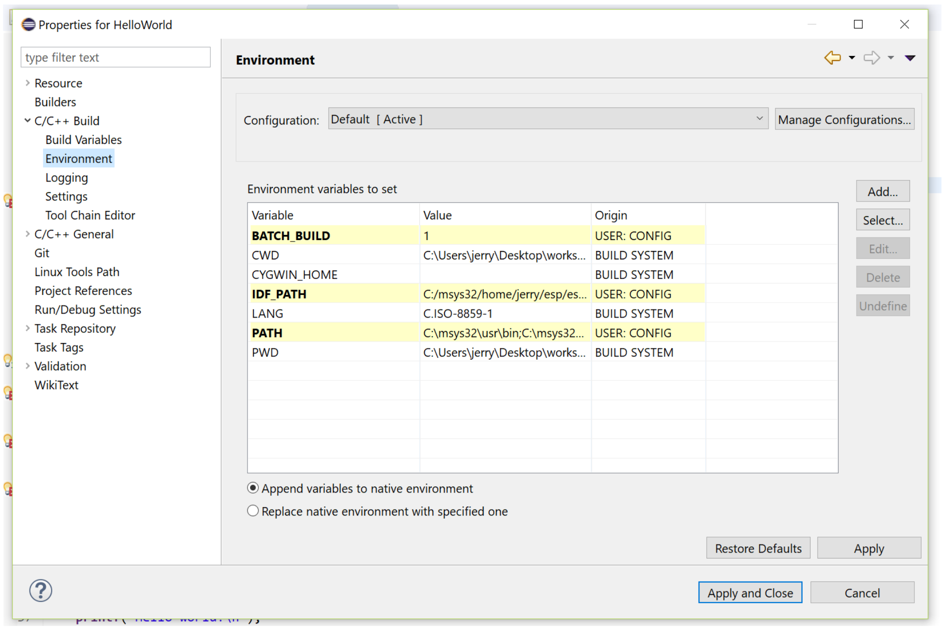
Build location
Build directory: ${workspace_loc:/HelloWorld}/
Workspace... File system... Variables...

Restore Defaults Apply

Apply and Close Cancel

The project should appear under You Project Explorer. Right click the project and select properties. Click on the properties page titled "C/C++ Build". (One of the Top-Level Options). Uncheck "Use default build command" and enter the following custom build command:

```
python ${IDF_PATH}/tools/windows/eclipse_make.py
```



Environment

Configuration: Default [Active]

Environment variables to set

Variable	Value	Origin
BATCH_BUILD	1	USER: CONFIG
CWD	C:\Users\jerry\Desktop\works...	BUILD SYSTEM
CYGWIN_HOME		BUILD SYSTEM
IDF_PATH	C:/msys32/home/jerry/esp/es...	USER: CONFIG
LANG	C.ISO-8859-1	BUILD SYSTEM
PATH	C:\msys32\usr\bin;C:\msys32...	USER: CONFIG
PWD	C:\Users\jerry\Desktop\works...	BUILD SYSTEM

[✓ Append variables to native environment]
Replace native environment with specified one

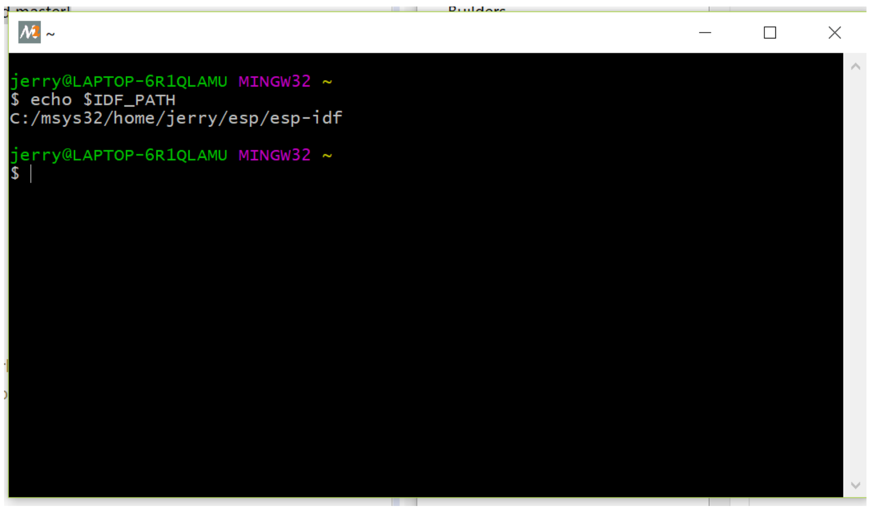
Click on the "Environment" properties page under "C/C++ BUILD". Click "Add" and enter the following:

Name: BATCH_BUILD Value: 1

You will need to add one more but before clicking "Add" again, you will need to obtain your IDF path. To do this, open you MSYS2 terminal and enter the following:

```
echo $IDF_PATH
```

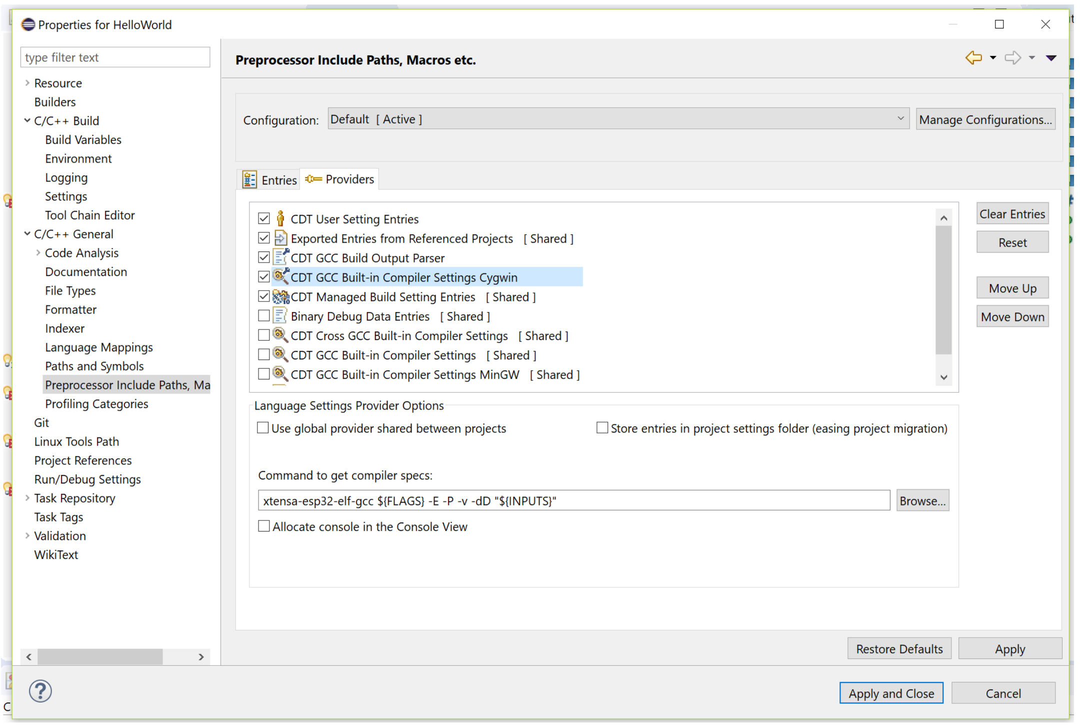
Now copy the output from the terminal, add with the name: IDF_PATH and paste the path into value.



```
jerry@LAPTOP-6R1QLAMU MINGW32 ~
$ echo $IDF_PATH
C:/msys32/home/jerry/esp/esp-idf

jerry@LAPTOP-6R1QLAMU MINGW32 ~
$ |
```

Next, you will need to edit the PATH environment. Delete the existing value and replace it with the following:

C:\msys32\usr\bin;C:\msys32\mingw32\bin;C:\msys32\opt\xtensa-esp32-elf\bin



Properties for HelloWorld

Preprocessor Include Paths, Macros etc.

Configuration: Default [Active]

- [✓] CDT User Setting Entries
- [✓] Exported Entries from Referenced Projects [Shared]
- [✓] CDT GCC Build Output Parser
- [✓] CDT GCC Built-in Compiler Settings Cygwin
- [✓] CDT Managed Build Setting Entries [Shared]
- [ ] Binary Debug Data Entries [Shared]
- [ ] CDT Cross GCC Built-in Compiler Settings [Shared]
- [ ] CDT GCC Built-in Compiler Settings [Shared]
- [ ] CDT GCC Built-in Compiler Settings MinGW [Shared]

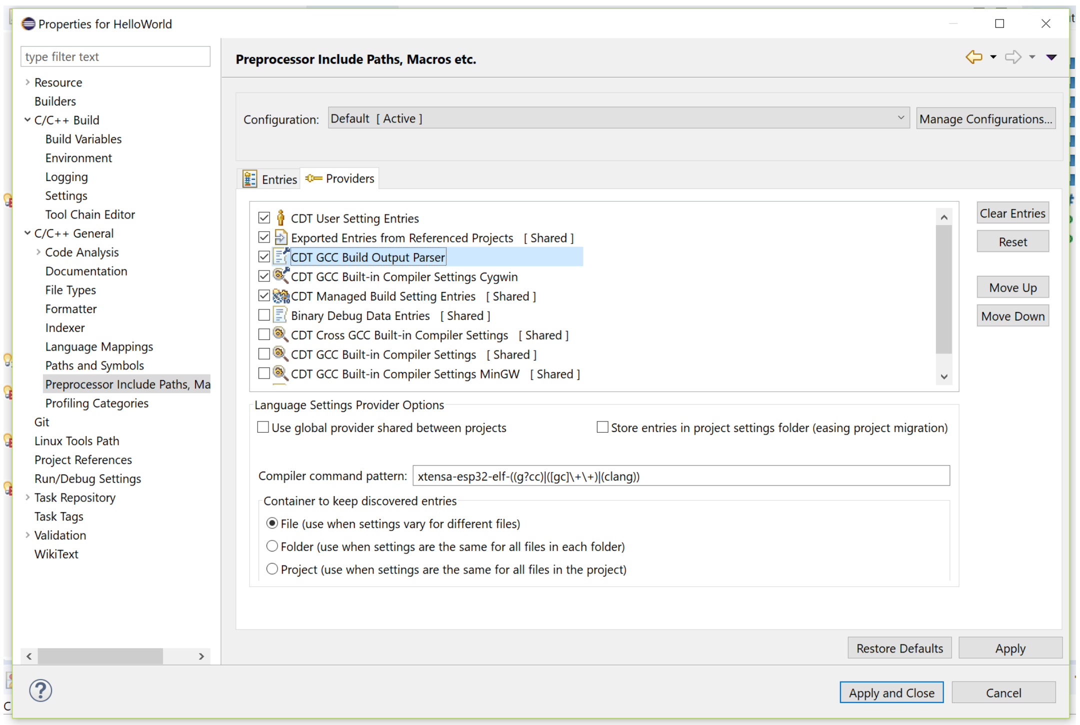
Language Settings Provider Options

- [ ] Use global provider shared between projects
- [ ] Store entries in project settings folder (easing project migration)

Command to get compiler specs:

```
xtensa-esp32-elf-gcc ${FLAGS} -E -P -v -dD "${INPUTS}"
```

- [ ] Allocate console in the Console View


Properties for HelloWorld

type filter text

- Resource
- Builders
- C/C++ Build
  - Build Variables
  - Environment
  - Logging
  - Settings
  - Tool Chain Editor
- C/C++ General
  - Code Analysis
  - Documentation
  - File Types
  - Formatter
  - Indexer
  - Language Mappings
  - Paths and Symbols
  - Preprocessor Include Paths, Ma
  - Profiling Categories
- Git
- Linux Tools Path
- Project References
- Run/Debug Settings
- Task Repository
- Task Tags
- Validation
- WikiText

Preprocessor Include Paths, Macros etc.

Configuration: Default [Active] Manage Configurations...

Entries | Providers

- [✓] CDT User Setting Entries
- [✓] Exported Entries from Referenced Projects [Shared]
- [✓] CDT GCC Build Output Parser
- [✓] CDT GCC Built-in Compiler Settings Cygwin
- [✓] CDT Managed Build Setting Entries [Shared]
- [ ] Binary Debug Data Entries [Shared]
- [ ] CDT Cross GCC Built-in Compiler Settings [Shared]
- [ ] CDT GCC Built-in Compiler Settings [Shared]
- [ ] CDT GCC Built-in Compiler Settings MinGW [Shared]

Clear Entries | Reset | Move Up | Move Down

Language Settings Provider Options

[ ] Use global provider shared between projects   [ ] Store entries in project settings folder (easing project migration)

Compiler command pattern: xtensa-esp32-elf-((g?cc)|([gc]\+\+)|(clang))

Container to keep discovered entries

- [✓ File (use when settings vary for different files)]
- Folder (use when settings are the same for all files in each folder)
- Project (use when settings are the same for all files in the project)

Restore Defaults | Apply

Apply and Close | Cancel

Click "Apply". Next, click on the properties page named "C/C++ General" and then "Preprocessor Include Paths, Macros, etc." Click the "Providers" tab, then in the list of providers, click "CDT GCC Built-in Compiler Settings Cygwin". Under "Command to get compiler specs", replace the text with the following:

xtensa-esp32-elf-gcc $FLAGS - E - P - v - dD$"INPUTS"

Also in the list of providers, click "CDT GCC Build Output Parser" and replaced compiler command pattern text with the following:

xtensa-esp32-elf-((g?cc)|([gc]\+\+)|(clang))

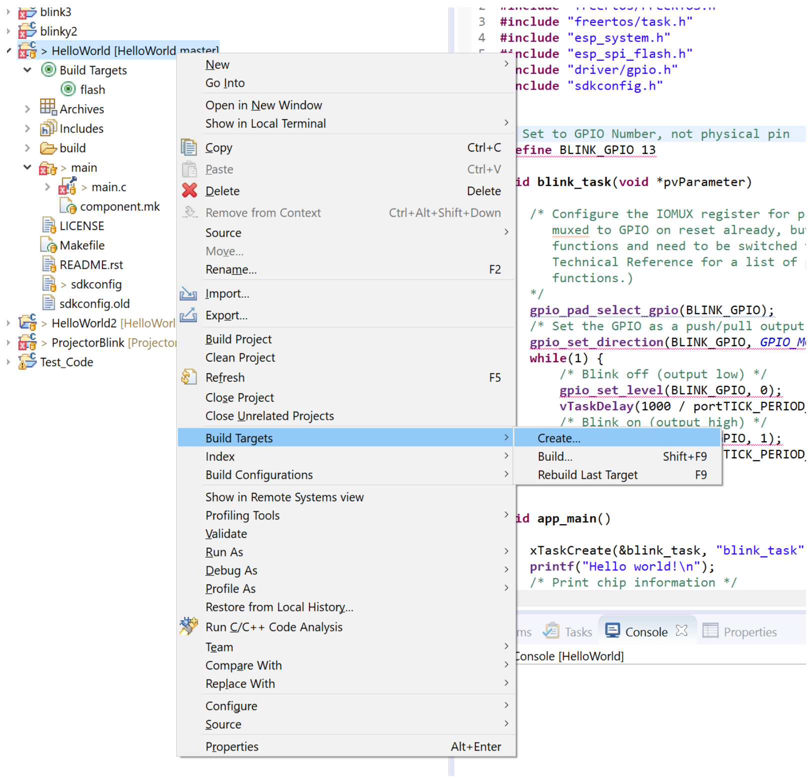
Now the Eclipse IDE is set for code and building from the IDE.

8 Building/Flashing from Eclipse

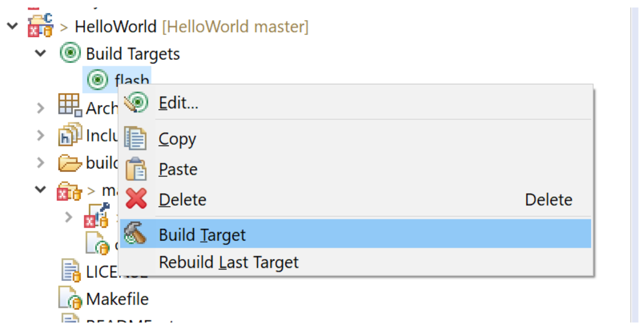
The final step before building and flashing the ESP32 is to copy this example code into the main.c of the project.

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_system.h"
#include "esp_spi_flash.h"

void app_main()
{
    printf("Hello world!\n");
    /* Print chip information */
    esp_chip_info_t chip_info;
    esp_chip_info(&chip_info);
    printf("This is ESP32 chip with %d CPU cores, WiFi%s%s, ",
            chip_info.cores,
            (chip_info.features & CHIP_FEATURE_BT) ? "/BT" : "",
            (chip_info.features & CHIP_FEATURE_BLE) ? "/BLE" : "");
    printf("silicon revision %d, ", chip_info.revision);
    printf("%dMB %s flash\n", spi_flash_get_chip_size() / (1024 * 1024),
            (chip_info.features & CHIP_FEATURE_EMB_FLASH) ?
            "embedded" : "external");
    for (int i = 10; i >= 0; i--) {
        printf("Restarting in %d seconds...\n", i);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
    printf("Restarting now.\n");
    fflush(stdout);
    esp_restart();
}
```

Hit Ctrl+S to save your code. Next, right click in the project explorer and slick on "Build Targets". Select "Create". In the pop-up window type "flash" as the Target name and click "ok". Click on "Build Targets" in the project explorer and right click "flash". Select "Build Target" and it should compile the code and load it into the ESP 32.



```
#include "freertos/task.h"
#include "esp_system.h"
#include "esp_spi_flash.h"
nclude "driver/gpio.h"
nclude "sdkconfig.h"

 Set to GPIO Number, not physical pin
efine BLINK_GPIO 13

id blink_task(void *pvParameter)

 /* Configure the IOMUX register for p
    muxed to GPIO on reset already, bu
    functions and need to be switched
    Technical Reference for a list of
    functions.)
 */
 gpio_pad_select_gpio(BLINK_GPIO);
 /* Set the GPIO as a push/pull output
 gpio_set_direction(BLINK_GPIO, GPIO_M
 while(1) {
     /* Blink off (output low) */
     gpio_set_level(BLINK_GPIO, 0);
     vTaskDelay(1000 / portTICK_PERIOD
     /* Blink on (output high) */
                   PIO, 1);
                   TICK_PERIOD

id app_main()

 xTaskCreate(&blink_task, "blink_task"
 printf("Hello world!\n");
 /* Print chip information */
```

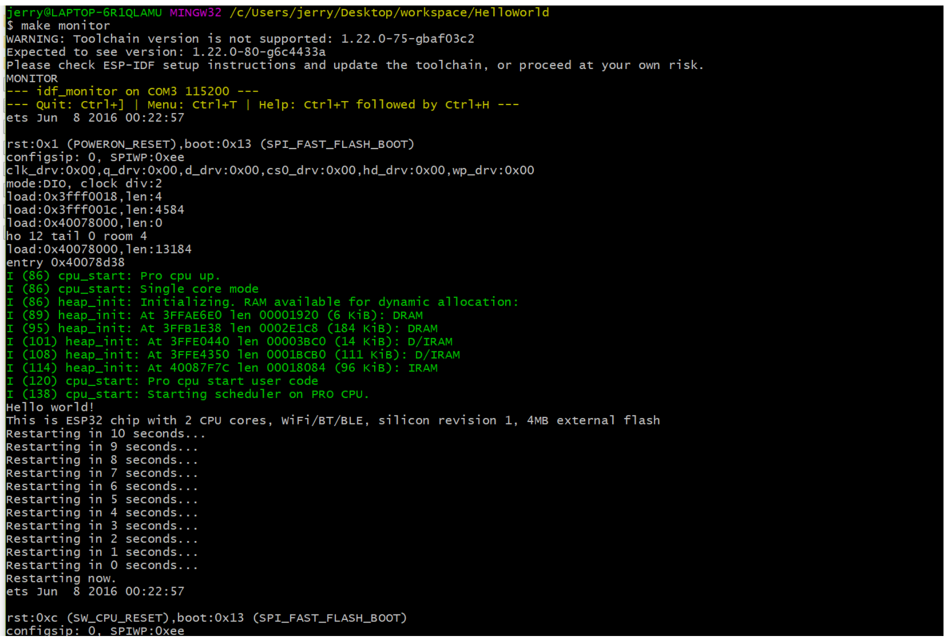


On the ESP 32 board, hold the RESET button first, then hold the BOOT button. Then release the RESET button followed by the BOOT button.

9 Results

To view the results and check that the code has been loaded and is running on the ESP 32 Devboard, open the MSYS2 terminal and issue the following commands:

```
cd "C:/Users/user-name/Desktop"

cd workspace

cd HelloWorld

make monitor
```



```
jerry@LAPTOP-6R1QLAMU MINGW32 /c/Users/jerry/Desktop/workspace/HelloWorld
$ make monitor
WARNING: Toolchain version is not supported: 1.22.0-75-gbaf03c2
Expected to see version: 1.22.0-80-g6c4433a
Please check ESP-IDF setup instructions and update the toolchain, or proceed at your own risk.
MONITOR
--- idf_monitor on COM3 115200 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:4584
load:0x40078000,len:0
ho 12 tail 0 room 4
load:0x40078000,len:13184
entry 0x40078d38
I (86) cpu_start: Pro cpu up.
I (86) cpu_start: Single core mode
I (86) heap_init: Initializing. RAM available for dynamic allocation:
I (89) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (95) heap_init: At 3FFB1E38 len 0002E1C8 (184 KiB): DRAM
I (101) heap_init: At 3FFE0440 len 00003BC0 (14 KiB): D/IRAM
I (108) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (114) heap_init: At 40087F7C len 00018084 (96 KiB): IRAM
I (120) cpu_start: Pro cpu start user code
I (138) cpu_start: Starting scheduler on PRO CPU.
Hello world!
This is ESP32 chip with 2 CPU cores, WiFi/BT/BLE, silicon revision 1, 4MB external flash
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
Restarting in 6 seconds...
Restarting in 5 seconds...
Restarting in 4 seconds...
Restarting in 3 seconds...
Restarting in 2 seconds...
Restarting in 1 seconds...
Restarting in 0 seconds...
Restarting now.
ets Jun  8 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
```

Now you'll be able to see the output of your board. It should say "Hello world!", some information about the ESP32, countdown from 10 and reset itself, repeating the same sequence again.