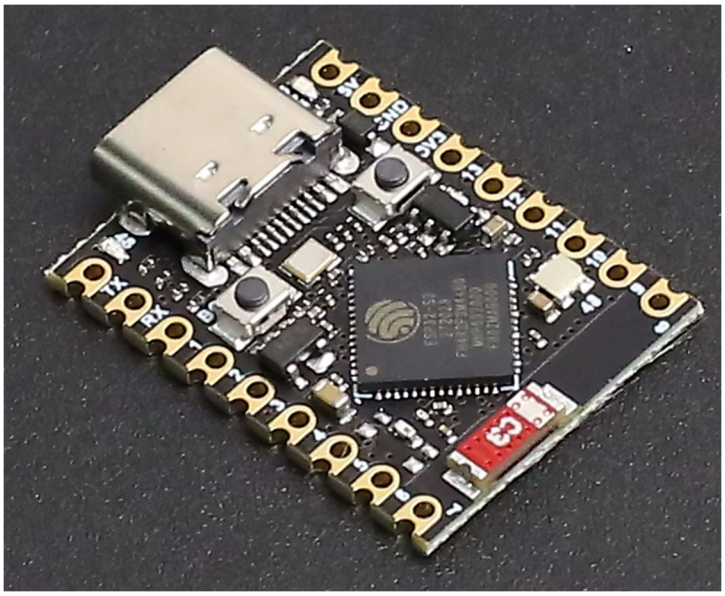
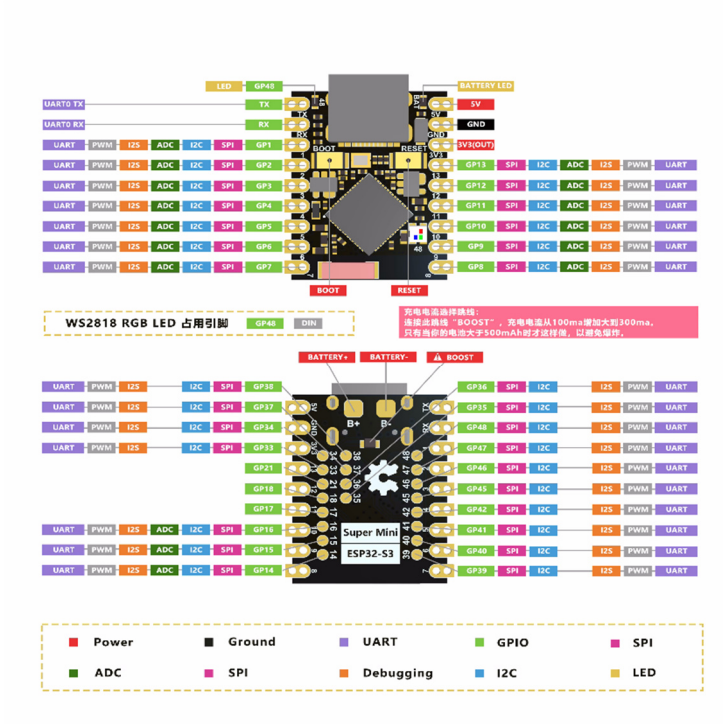
Debug ESP32S3 using PowerShell

Target hardware

1. ESP32-S3 Super Mini Black Gold
2. Clone of a ZERO board
3. Has 18 pins, boot and reset buttons and a USB-C connector
4. MCU is ESP32-S3 FH4R2
5. Datasheet https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf
6. CPU is Xtensa® dual-core 32-bit LX7 microprocessor
7. Clock speed: up to 240 MHz
8. Flash is 4Mb, PSRAM 2Mb and SRAM 512 Kb
9. Assume USB JTAG/debug unit (interface 0) set as COM3
10. Assume USB JTAG/debug unit (interface 2) configured as a Universal Serial Bus Device called USB JTAG/Serial debug unit



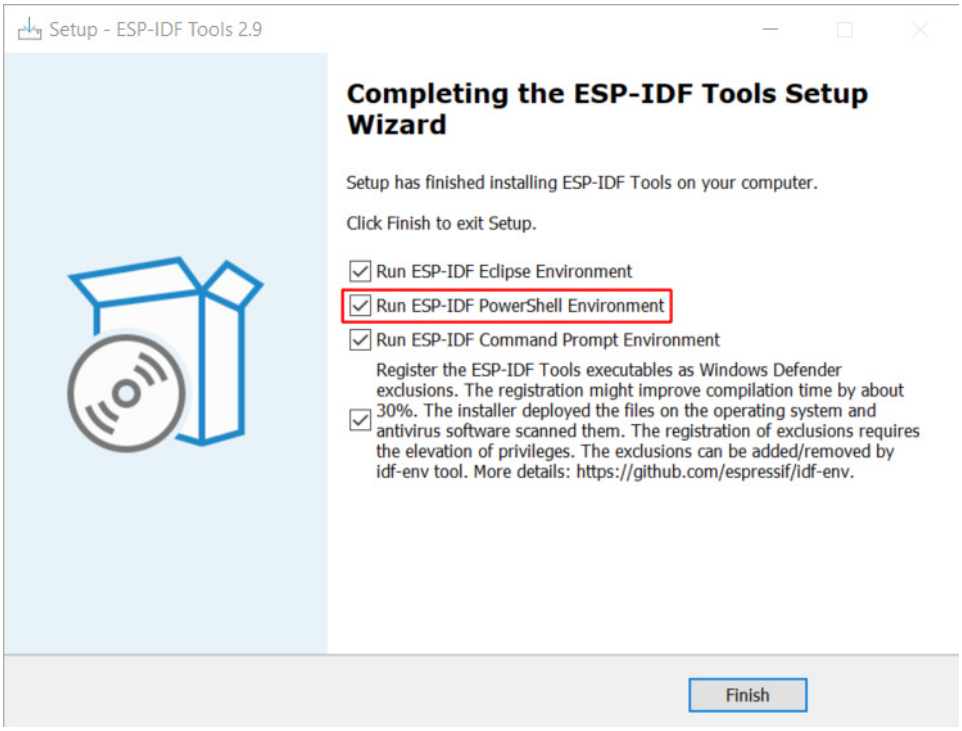


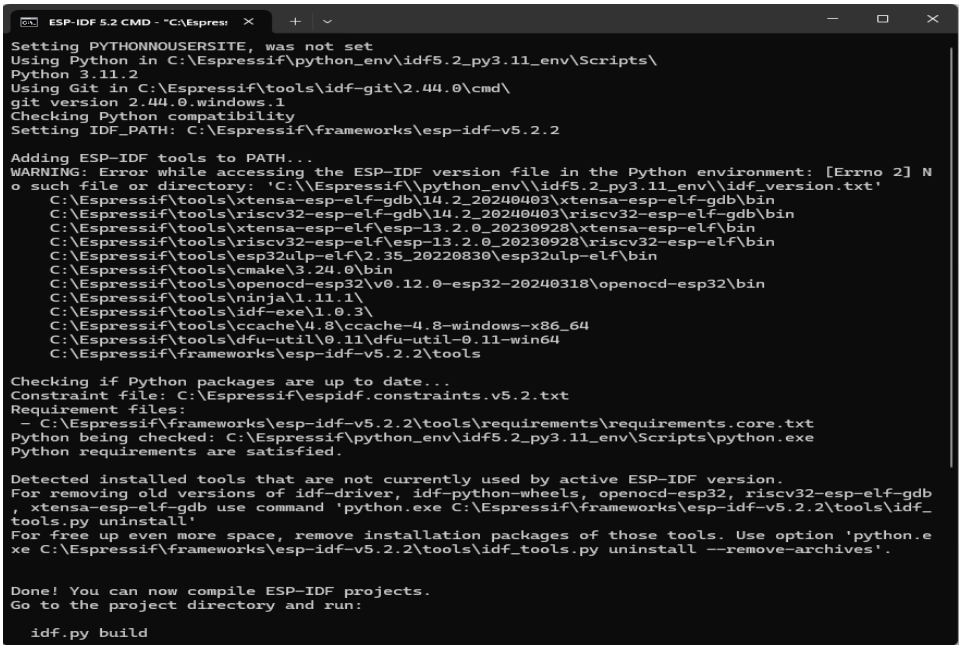
Configuring the USB ports

1. Use Zadig to configure USB JTAG/debug unit (interface 0) as usbser
2. Use Zadig to configure USB JTAG/debug unit (interface 2) as WinUSB

Install ESP-IDF

1. Download the offline tools installer from https://dl.espressif.com/dl/esp-idf/?idf=4.4
    a. Either a specific version, or use the universal online installer and choose the version
    b. Idf-env.exe runs so OK it
2. During the installation two special shortcuts (CMD & PowerShell) will be added to the desktop
    a. Both shortcuts setup the Python environment ready for ESP-IDF commands
    b. I prefer PowerShell as can cd to a new path in one step





3. Updating ESP-IDF tools and adding ESP-IDF tools to PATH using an export script
    a. Open the PowerShell shortcut and run install.ps1 and then export.ps1
    b. Open the Command shortcut and run install.bat and then export.bat
    c. Look at the output and run any other suggestions for missing items

Building the code

1. Copy the blink example from E:\Users\Steven\Downloads\esp-idf-v5.2.2\esp-idf-v5.2.2\examples\get-started\blink to E:\Users\Steven\Documents\GitHub\ESP-IDF\blink
2. Open a special ESP-IDF 5.2 PowerShell window, do not use the CMD version as less flexible.
3. Change to blink directory using cd E:\Users\Steven\Documents\GitHub\ESP-IDF\blink
4. Set the target idf.py set-target esp32s3
5. Open Menuconfig idf.py menuconfig
6. Enter Serial flasher config
7. Change flash size from 2MB to 4MB
8. Change any other settings as needed
9. Build the code using idf.py build

Uploading the code

1. Upload the built binaries to the ESP32S3 idf.py -p COM3 flash
2. JTAG upload can be used as an alternative to serial via a COM port using either of the following:
    a. openocd -s scripts -f interface/esp_usb_jtag.cfg -f board/esp32s3-builtin.cfg -c "program_esp E:/Users/Steven/Documents/GitHub/ESP-IDF/blink/build/blink.bin 0x10000 verify reset exit"
    b. openocd -s scripts -f board/esp32s3-builtin.cfg -c "program_esp E:/Users/Steven/Documents/GitHub/ESP-IDF/blink/build/blink.bin 0x10000 verify reset exit"

Starting OpenOCD the Open On-Chip Debugger server

1. Create a file called gdbinit in the blink folder at E:\Users\Steven\Documents\GitHub\ESP-IDF\blink
2. Edit gdbinit to include the following text:

```
target remote :3333
set remote hardware-watchpoint-limit 2
mon reset halt
maintenance flush register-cache
thb app_main
c
```

3. Open a special ESP-IDF 5.2 PowerShell window, do not use the CMD version as less flexible.
4. Check OpenOCD working using openocd --version
5. Change to blink directory using cd E:\Users\Steven\Documents\GitHub\ESP-IDF\blink
6. Start OpenOCD using openocd -f board/esp32s3-builtin.cfg
7. OpenOCD Responds with:

```
Open On-Chip Debugger v0.12.0-esp32-20240318 (2024-03-18-18:26)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
Info : only one transport option; autoselecting 'jtag'
Info : esp_usb_jtag: VID set to 0x303a and PID to 0x1001
Info : esp_usb_jtag: capabilities descriptor set to 0x2000
Info: Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : esp_usb_jtag: serial (24:58:7C:DC:25:C0)
Info : esp_usb_jtag: Device found. Base speed 40000KHz, div range 1 to 255
Info : clock speed 40000 kHz
Info : JTAG tap: esp32s3.cpu0 tap/device found: 0x120034e5 (mfg: 0x272 (Tensilica), part: 0x2003, ver: 0x1)
Info : JTAG tap: esp32s3.cpu1 tap/device found: 0x120034e5 (mfg: 0x272 (Tensilica), part: 0x2003, ver: 0x1)
Info : [esp32s3.cpu0] Examination succeed
Info : [esp32s3.cpu1] Examination succeed
Info : starting gdb server for esp32s3.cpu0 on 3333
Info : Listening on port 3333 for gdb connections
Info : [esp32s3.cpu0] Debug controller was reset.
Info : [esp32s3.cpu0] Core was reset.
```

```
Info : [esp32s3.cpu1] Debug controller was reset.
Info : [esp32s3.cpu1] Core was reset.
Info : [esp32s3.cpu0] Target halted, PC=0x40379532, debug_reason=00000000
Info : [esp32s3.cpu0] Reset cause (1) - (Power on reset)
Info : [esp32s3.cpu1] Target halted, PC=0x40379532, debug_reason=00000000
Info : [esp32s3.cpu1] Reset cause (1) - (Power on reset)
```

Starting GDB the GNU Debugger

1. Open a second special ESP-IDF 5.2 PowerShell window
2. Change to blink directory using cd E:\Users\Steven\Documents\GitHub\ESP-IDF\blink
3. Start GDB using xtensa-esp32s3-elf-gdb.exe build/blink.elf -x gdbinit

Using GDB commands

1. For a variety of different layouts within the shell run layout next
    a. To step through assembly code use nexti or stepi
    b. To step through lines of code use next or n or step or s
    c. With the code layout the stepping or nexting can be seen line by line
2. Stepping one line of code, including stepping into subroutine s or s1
3. Stepping multiple lines of code, including stepping into subroutine s7
4. Set a breakpoint break blink_example_main.c:106 or just break 106
5. To list current breakpoints and times hit info break
6. To delete a breakpoint del 3 where 3 is the breakpoint, not line, number
7. To delete a breakpoint dis 3 where 3 is the breakpoint, not line, number
8. To enable a breakpoint en 3 where 3 is the breakpoint, not line, number
9. Continue execution to the next breakpoint c or continue
10. To step to the next line of code, and stepover subroutines n or next
11. To view the source code lines around the current line list or l
12. To examine a variable print s_led_state or print count
13. To set the value of a variable set s_led_state=1 or count=146
14. To exit GDB, use the quit command q

Viewing serial output

1. The blink example prints Turning the LED ON! and Turning the LED OFF!
2. To view open a serial terminal e.g. SmarTTY and connect to COM3 at 115200 baud

Stopping GDB the GNU Debugger

1. To exit GDB, use the quit command q

Stopping OpenOCD the Open On-Chip Debugger server

1. To shut down the OpenOCD server Ctrl C

For more info about using GDB see:

- https://docs.espressif.com/projects/esp-idf/en/v5.2.2/esp32/api-guides/jtag-debugging/index.html
- https://docs.espressif.com/projects/esp-idf/en/v5.2.2/esp32/api-guides/jtag-debugging/debugging-examples.html
- https://ftp.gnu.org/old-gnu/Manuals/gdb/html_node/gdb_toc.html
- https://web.eecs.umich.edu/~sugih/pointers/gdbQS.html
- https://www.youtube.com/playlist?list=PL9IEJIKnBJjHGWPN_S9NS_Ky1-tC8ZrUI