Last updated: 30 May 2024

**Instructions**

For instructional video see: https://www.youtube.com/watch?v=v3C9WHLpoeY&t=425s

1. Open special ESP-IDF 5.2 PowerShell window 1 for OpenOCD the Open On-Chip Debugger
2. Create copy of blink example
   1. Create folder: "C:\ESP-IDF\esp-idf-v5.2.1\mycode"
   2. Copy from: "C:\ESP-IDF\esp-idf-v5.2.1\examples\get-started\blink"
   3. Paste into: "C:\ESP-IDF\esp-idf-v5.2.1\mycode\blink"
   4. Edit main.c to match the code listed in Appendix A, or just copy & paste
   5. Create "C:\ESP-IDF\esp-idf-v5.2.1\mycode\blink\gdbinit" with following code
      1. set remotetimeout 100
      2. target extended-remote :3333
      3. set remote hardware-watchpoint-limit 2
      4. mon reset halt
      5. maintenance flush register-cachethb app\_main
      6. c
3. PS C:\ESP-IDF\esp-idf-v5.2.1> cd mycode
4. PS C:\ESP-IDF\esp-idf-v5.2.1\mycode> cd blink
5. PS C:\ESP-IDF\esp-idf-v5.2.1\mycode\blink> idf.py set-target esp32
6. PS C:\ESP-IDF\esp-idf-v5.2.1\mycode\blink> idf.py build
7. PS C:\ESP-IDF\esp-idf-v5.2.1\mycode\blink> idf.py -p COM3 flash
8. PS C:\ESP-IDF\esp-idf-v5.2.1\mycode\blink> openocd -f board/*esp32-wrover-kit-3.3v.cfg*
9. Config file located at: C:\Espressif\tools\openocd-esp32\v0.12.0-esp32-20230921\openocd-esp32\share\openocd\scripts\board\*esp32-wrover-kit-3.3v.cfg*
10. Open special ESP-IDF 5.2 PowerShell window 2 for GDB, the GNU Project debugger
11. PS C:\ESP-IDF\esp-idf-v5.2.1> cd mycode
12. PS C:\ESP-IDF\esp-idf-v5.2.1\mycode> cd blink
13. PS C:\ESP-IDF\esp-idf-v5.2.1\mycode\blink> *xtensa-esp32-elf-gdb* build/blink.elf -x gdbinit
14. Executable app located at: C:\Espressif\tools\xtensa-esp-elf-gdb\12.1\_20231023\xtensa-esp-elf-gdb\bin\*xtensa-esp32-elf-gdb.exe*
    1. Thread 2 "main" hit Temporary breakpoint 1, app\_main () at C:/ESP-IDF/esp-idf-v5.2.1/mycode/blink/main/blink\_example\_main.c:31
    2. 31 configure\_led();
    3. (gdb) next
    4. Note: automatically using hardware breakpoints for read-only addresses.
    5. [esp32.cpu0] Target halted, PC=0x400D53A6, debug\_reason=00000001
    6. Set GDB target to 'esp32.cpu0'
    7. [esp32.cpu1] Target halted, PC=0x400845E2, debug\_reason=00000000
    8. 34 blink\_led();
15. The gdb command “next” or just “n” steps through the source code

**Explanation**

1. Special PowerShell shortcuts are created when ESP-IDF the IoT Development Framework for ESP32 is installed from: <https://dl.espressif.com/dl/esp-idf> Opening a special PowerShell Window from the shortcut sets up all the Windows paths for correct operation of idf.py command-line tool, in this instance for OpenOCD the Open On-Chip Debugger
2. Make a copy of the BLINK example supplied with the ESP-IDF and add to a new folder mycode. Edit the source code to make simpler and create an initialisation file for GDB, the GNU Project debugger. Note the port used for listening by OpenOCD is port 3333.
3. Change directory to mycode
4. Change directory to blink
5. Run the command to set the target hardware to ESP32
6. Run the command to build the source code to an ESP32 binary image
7. Flash, or upload, the built image to the target hardware, COM port could be different
8. Start OpenOCD using the hardware configuration file for the esp32-wrover
9. This is where the configuration file is located along with other targets
10. Open a second instance of the special ESP-IDF PowerShell window
11. Change directory to mycode
12. Change directory to blink
13. Start the GDB debugger
14. Location of gdb executables for various target hardware and example output from GDB
15. Other GDB commands are listed at: https://www.tutorialspoint.com/gnu\_debugger/gdb\_commands.htm

**Appendix A - Code in main.c**

/\* Blink Example

\*/

#include <stdio.h>

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "driver/gpio.h"

#include "sdkconfig.h"

int delay\_mS = 1000;

#define BLINK\_GPIO 2 // CONFIG\_BLINK\_GPIO

static uint8\_t s\_led\_state = 0;

static void blink\_led(void)

{

/\* Set the GPIO level according to the state (LOW or HIGH)\*/

gpio\_set\_level(BLINK\_GPIO, s\_led\_state);

}

static void configure\_led(void)

{

gpio\_reset\_pin(BLINK\_GPIO);

/\* Set the GPIO as a push/pull output \*/

gpio\_set\_direction(BLINK\_GPIO, GPIO\_MODE\_OUTPUT);

}

void app\_main(void)

{

configure\_led();

while (1) {

blink\_led();

/\* Toggle the LED state \*/

s\_led\_state = !s\_led\_state;

vTaskDelay(delay\_mS / portTICK\_PERIOD\_MS); //Delay 200mS

}

}