HOW TO SETUP JABIT SOLO MINER (BITAXE V2.2)

STEPS:

1. Install and Setup standard ESP-IDF Toolchain – It's Free
2. Install Microsoft Visual Studio Code (VS Code) and ESP-IDF Plugin - It's Free
3. Establish Serial Connection with ESP32-S3 on JaBIT Solo Miner and setup up your configurations
4. Flash the Production Firmware to JaBIT Solo Miner (Bitaxe V2.2)
5. Start Mining.
6. For any issues of questions please join our discord bitaxe community. https://discord.gg/nRbcVjXm

1. *Installing and setting up standard ESP-IDF V4.4.4*

NOTE:
ESP-IDF requires some prerequisite tools to be installed so you can build firmware for supported chips. The prerequisite tools include Python, Git, cross-compilers, CMake and Ninja build tools.

Components of the installation

The installer deploys the following components:

- Embedded Python
- Cross-compilers
- OpenOCD
- CMake and Ninja build tools
- ESP-IDF

Follow the detailed instructions at :

For Windows:

https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/get-started/windows-setup.html

The easiest way to install ESP-IDF's prerequisites for windows is to download one of ESP-IDF Tools Installers at:

https://dl.espressif.com/dl/esp-idf/?idf=4.4

For Linux :

https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/get-started/linux-macos-setup.html

2. ***Installing Microsoft Visual Studio Code:***

   For Windows , Linux and Mac OS :

   For starters Visual Studio code provides an easier way of using ESP-IDF Toolchain.

   Follow instructions to install VSCode at :

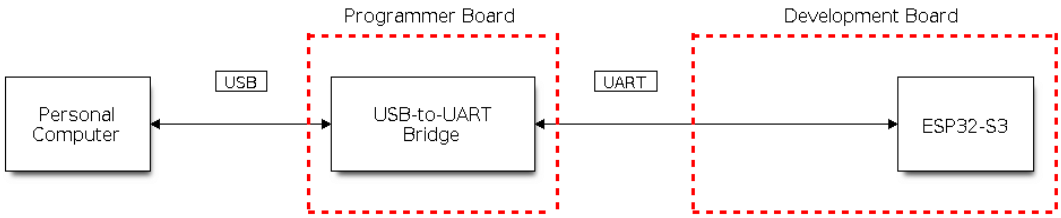
   https://github.com/espressif/vscode-esp-idf-extension/blob/master/docs/tutorial/install.md

3. ***Establish Serial Connection with ESP32-S3 on JaBIT Solo Miner***

   Currently JaBIT solo miner supports two ways of connecting to JaBIT solo Miner and flashing the firmware.

   1. Using ESP-PROG (USB-to_UART Bridge) and TAG-Connect TC2030-IDC-NL cable
   2. Using ESP-PROG (USB-to_UART Bridge) and Mini JTAG (J10) on JaBIT Solo Minier

External USB-to-UART Bridge



For detailed instructions:
https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/get-started/establish-serial-connection.html

4. ***Flash the Production Firmware (Nonce_checking Branch of ESP-Miner) to JaBIT Solo Miner (Bitaxe V2.2) .***

Note: Applies to both Windows and Linux OS

(a). After installing ESP-IDF V4.4 , VS Code and connecting your ESP-PROG to JaBIT Solo Miner you're now ready to Flash the production firmware to the Miner.

(b). Download the latest production firmware from the bitaxe community github repository and extract to folder.

- https://github.com/skot/ESP-Miner/tree/nonce_checking

OR

- https://github.com/developeralgo8888/ESP-Miner/tree/nonce_checking

(c). In VSCode select File → Open Folder → Select the Folder you extracted the nonce_checking files into.

(d). You use VS Code Terminal or command line to configure your project and flash the firmware .

1. Click on View --> Command Palette --> ESP-IDF: Device Configuration --> Device Target --> Select ESP-Miner-Nonce_Checking Folder -->esp32s3 --> ESP32-S3 chip(Via ESP-PROG)
2. Click on View --> Command Palette --> ESP-IDF: Device Configuration --> Device Port --> Enter your USB Port i.e COMxxx (Windows) or /dev/ttyUSBxxx (Linux)
3. Click on View --> Command Palette --> ESP-IDF: Device Configuration --> Flash baud rate --> 460800
4. Click on View --> Command Palette --> ESP-IDF: Device Configuration --> OpenOCD Config Files --> interface/ftdi/esp32_devkitj_v1.cfg,target/esp32s3.cfg
5. Click on View --> Command Palette --> ESP-IDF: Add VS Code configuration folder
6. Click on View --> Command Palette --> ESP-IDF: Open ESP-IDF Terminal --> it will open a terminal window in VS Code window and use the ESP-IDF Terminal to configure and flash the JaBIT Solo Miner. Type the following commands in the ESP-IDF terminal

Set the target:

```
idf.py set-target esp32s3
```

Use menuconfig to set the stratum server address/port and WiFi SSID/Password:

```
idf.py menuconfig
```

Set following parameters under **Stratum Configuration** **Options**, these will define the stratum server you connect to:

- Set `Stratum Address` to the stratum pool domain name. example "solo.ckpool.org"
- Set `Stratum Port` to the stratum pool port. example "3333"
- Set `Stratum username` to the stratum pool username
- Set `Stratum password` to the stratum pool password

Set following parameters under **Example Connection Configuration** Options:

- Set `WiFi SSID` to your target wifi network SSID.
- Set `Wifi Password` to the password for your target SSID

Build the project and flash it to the board, then run monitor tool to view serial output:

For Windows: (where xxx is your USB port number)

```
idf.py -p COMxxx flash monitor
```

For Linux:

```
idf.py -p /dev/ttyUSBxxx flash monitor
```

(To exit the serial monitor, type `Ctrl-]`.)

If the flashing is successful you should see something similar to the output below and your small OLED screen should start displaying mining information.

developer@Testbox:~/Downloads/JABIT_BITAXE_FIRMWARE/ESP-Miner-i2c_test$ `idf.py -p /dev/ttyUSB2 flash monitor`

```
Executing action: flash
Running ninja in directory /home/developer/Downloads/JABIT_BITAXE_FIRMWARE/ESP-Miner-
i2c_test/build
Executing "ninja flash"...
[1/5] cd /home/developer/Downloads/JABIT_BITAXE_FIRMWARE/ESP-Miner-
i2c_test/build/es...home/developer/Downloads/JABIT_BITAXE_FIRMWARE/ESP-Miner-
i2c_test/build/I2c_test.bin
I2c_test.bin binary size 0x37810 bytes. Smallest app partition is 0x100000 bytes. 0xc87f0 bytes
(78%) free.
[2/5] Performing build step for 'bootloader'
[1/1] cd /home/developer/Downloads/JABIT_BITAXE_FIRMWARE/ESP-Miner-i2c_test/build/bootloader/esp-
idf/esptool_py && /home/developer/.espressif/python_env/idf4.4_py3.10_env/bin/python
/home/developer/esp/esp-idf/components/partition_table/check_sizes.py --offset 0x8000 bootloader
0x0 /home/developer/Downloads/JABIT_BITAXE_FIRMWARE/ESP-Miner-
i2c_test/build/bootloader/bootloader.bin
Bootloader binary size 0x5240 bytes. 0x2dc0 bytes (36%) free.
[2/3] cd /home/developer/esp/esp-idf/components/esptool_py && /usr/bin/cmake -D IDF_...st/build" -
```

```
P /home/developer/esp/esp-idf/components/esptool_py/run_serial_tool.cmake
esptool.py esp32s3 -p /dev/ttyUSB2 -b 460800 --before=default_reset --after=hard_reset write_flash
--flash_mode dio --flash_freq 80m --flash_size 2MB 0x0 bootloader/bootloader.bin 0x10000
I2c_test.bin 0x8000 partition_table/partition-table.bin
esptool.py v3.3.2
Serial port /dev/ttyUSB2
Connecting....
Chip is ESP32-S3
Features: WiFi, BLE
Crystal is 40MHz
MAC: 34:85:18:9c:54:18
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00000000 to 0x00005fff...
Flash will be erased from 0x00010000 to 0x00047fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Compressed 21056 bytes to 13153...
Writing at 0x00000000... (100 %)
Wrote 21056 bytes (13153 compressed) at 0x00000000 in 0.6 seconds (effective 292.5 kbit/s)...
Hash of data verified.
Compressed 227344 bytes to 119999...
Writing at 0x00010000... (12 %)
Writing at 0x0001c59b... (25 %)
Writing at 0x0002267e... (37 %)
Writing at 0x000287aa... (50 %)
Writing at 0x0002e9fd... (62 %)
Writing at 0x00037959... (75 %)
Writing at 0x0003f4ba... (87 %)
Writing at 0x000454be... (100 %)
Wrote 227344 bytes (119999 compressed) at 0x00010000 in 2.8 seconds (effective 638.6 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 103...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.0 seconds (effective 510.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
Executing action: monitor
```