Chronos Utility

Chronos is an utility software for testing FTM (Fine Time Measurement) on Espressif ESP32 devices. It can be built and installed on any ESP32 device with FTM capability (S2, S3 and C3 families).

The source code is available from the following github repository:

https://github.com/cezmen/chronos

How to use the Chronos Utility

[1] Set Target

idf.py set-target esp32s2

[2] Configure the Project

idf.py menuconfig

[2.1] Set Configuration Parameters in the following menus:

- Example Configuration ➤ SoftAP
- Example Configuration ➤ TCP Server
- Example Configuration ➤ FTM

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>DESCRIPTION</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP_WIFI_SSID</td>
<td>WiFi SSID</td>
<td>SoftAP</td>
</tr>
<tr>
<td>ESP_WIFI_PASSWORD</td>
<td>WiFi Password</td>
<td></td>
</tr>
<tr>
<td>ESP_WIFI_CHANNEL</td>
<td>WiFi Channel</td>
<td></td>
</tr>
<tr>
<td>ESP_MAX_STA_CONN</td>
<td>Maximal STA connections</td>
<td></td>
</tr>
<tr>
<td>ESP_INTERFACE_IP</td>
<td>IPv4 Address</td>
<td>TCP Server</td>
</tr>
<tr>
<td>ESP_INTERFACE_GW</td>
<td>Gateway IPv4 Address</td>
<td></td>
</tr>
<tr>
<td>ESP_INTERFACE_NETMASK</td>
<td>Netmask</td>
<td></td>
</tr>
<tr>
<td>ESP_IPV4</td>
<td>IPv4 (y/n)</td>
<td>TCP Server</td>
</tr>
<tr>
<td>ESP_IPV6</td>
<td>IPV6 (y/n)</td>
<td></td>
</tr>
<tr>
<td>ESP_PORT</td>
<td>Port</td>
<td></td>
</tr>
<tr>
<td>ESP_KEEPALIVE_IDLE</td>
<td>TCP keep-alive idle time(s)</td>
<td></td>
</tr>
<tr>
<td>ESP_KEEPALIVE_INTERVAL</td>
<td>TCP keep-alive interval time(s)</td>
<td></td>
</tr>
<tr>
<td>ESP_KEEPALIVE_COUNT</td>
<td>TCP keep-alive packet retry send counts</td>
<td></td>
</tr>
<tr>
<td>ESP_FTM_REPORT_LOG_ENABLE</td>
<td>FTM Report logging (y/n)</td>
<td>FTM</td>
</tr>
<tr>
<td>ESP_FTM_REPORT_SHOW_DIAG</td>
<td>Show dialog tokens (y/n)</td>
<td></td>
</tr>
<tr>
<td>ESP_FTM_REPORT_SHOW_RTT</td>
<td>Show RTT values (y/n)</td>
<td></td>
</tr>
<tr>
<td>ESP_FTM_REPORT_SHOW_T1T2T3T4</td>
<td>Show T1 to T4 (y/n)</td>
<td></td>
</tr>
<tr>
<td>ESP_FTM_REPORT_SHOW_RSSI</td>
<td>Show RSSI levels (y/n)</td>
<td></td>
</tr>
</tbody>
</table>
[2.2] Additional Parameters Setup

Component Config ► WiFi ►
- WiFi FTM : y
- FTM Initiator Support : y
- FTM Responder Support : y

Serial Flasher Config ►
- Flash Size : 4MB
- After Flashing : Stay In Bootloader

Component Config ► Common ESP Related ►
- Channel for Console Output :
  - USB CDC (if using Franzininho WiFi)
  - UART0 (if using ESP32-S2-Devkit-C)

[3] Build
Build the project :

idf.py build

[4] Flash
Put the board in DFU mode (by pressing BOOT and RESET keys in the following sequence : press BOOT, press RESET, release RESET, release BOOT).

Flash the firmware to the board :

```
idf.py -p <device name> flash
```
(Note: Use 'ls /dev/tty*' to discover the exact <device name> in your environment)

Reset the board (by pressing and releasing the RESET key).

[5.1] Monitor the Franzininho WiFi board (through USB CDC)
This Demo doesn't work well with "idf.py monitor" when the Console Output is using USB CDC port.

In this case, use a serial terminal emulator (such as screen) instead.

```
screen <device name> 115200,cs8
```
(Note: Use 'ls /dev/tty*' to discover the exact <device name>)
(To exit screen, type 'Ctrl-A with k', pressing 'y' right after to kill the window).
Wi-Fi FTM RTT Based Positioning System

[5.2] Monitor the ESP32-S2-Devkit-C board (through UART0)
Run the ESP-IDF monitor

```
idf.py -p <device name> monitor
```

(Note: Use `ls /dev/tty*` to discover the exact `<device name>` in your environment)
(To exit the serial monitor, type `Ctrl-]`.)

Refer to the `README.md` file (from the github repository) for further instructions.