

ESP32の概要

ESP32 D0WDQ6チップを搭載した ESP-WROOM-32 モジュール。

製造元 Espressif Systems
種別 マイクロコントローラ
発売日 2016年9月6日
CPU Tensilica Xtensa LX6 マイクロ
 プロセッサ @ 160 or 240 MHz
メモリ 520 KByte SRAM
電源 3.3 V DC
前世代ハード ESP8266

ESP32シリーズは、[Wi-Fi](#)と [Bluetooth](#)を内蔵する低コスト、低消費電力な [SoC](#)の マイクロコントローラである。 Tensilicaの Xtensa LX6 マイクロプロセッサを 採用し デュアルコアとシングルコア版の バリエーションがある。

ESP32は、上海に拠点を置くEspressif Systemsが開発し、台湾積体電路製造(TSMC)の40nm工程で製造されている。
ESP8266の 後継製品にあたる。

[Wikipedia](#)から引用しました。
[Wikipediaさん、ありがとうございます。](#)

以上の説明において使用されている専門用語に関して、一部説明しておきます。

[Wi-Fi](#) :

よく聞く言葉ですが、無線LANの事だろうと思っていましたが、無線LANに関する登録商標でした。アメリカの IEEE 802.11規格を使用したデバイス間の相互接続が、認められた事を示す名称。
([これも Wikipediaに 書いてありました。](#))

Bluetooth:

これもよく聞く言葉ですが、デジタル機器用の近距離無線通信規格の1つである。

Bluetooth Basic Rate/Enhanced Data(BR/EDRと、Bluetooth Low Energy(LE)から、構成される。数メートルから数十メートル程度の距離の情報機器間で、電波を使い簡易な情報のやり取りを行うのに使用される。

(これも Wikipediaからです。)

非常に大雑把な表現をすると、Wi-Fiは 通常無線で、インターネットに接続するための物で、

Bluetoothは、無線の キーボード、マウス、プリンターを接続する 無線規格と解釈出来ます。

で、ESP32は、この Wi-Fi と Bluetooth の両方の機能を搭載した マイコンです。ESP32は、IoT で、使用して下さい。と言わんばかりのマイコンですね。

前ページにて、ESP32は ESP8266の後継製品にあたる。と、ありましたが、何が違うかということ ESP8266は、Wi-Fi にのみ対応してます。ESP32は、Wi-Fi と Bluetoothの両方に対応しています。ということです。これが大きな違いで、後は CPUも同じで、さほど大差はないと思います。

価格的には、ESP8266の方が やや安いです。パッと見た目は、殆ど同じなので、ESP32の つもりで買ったら、ESP8266だった。という事もあり得ます。私も、1回やらかしました。型式をよく確認してから買いましょう。

一応 ESP8266と ESP32の画像を 表示しておきます。 上が ESP8266 です。

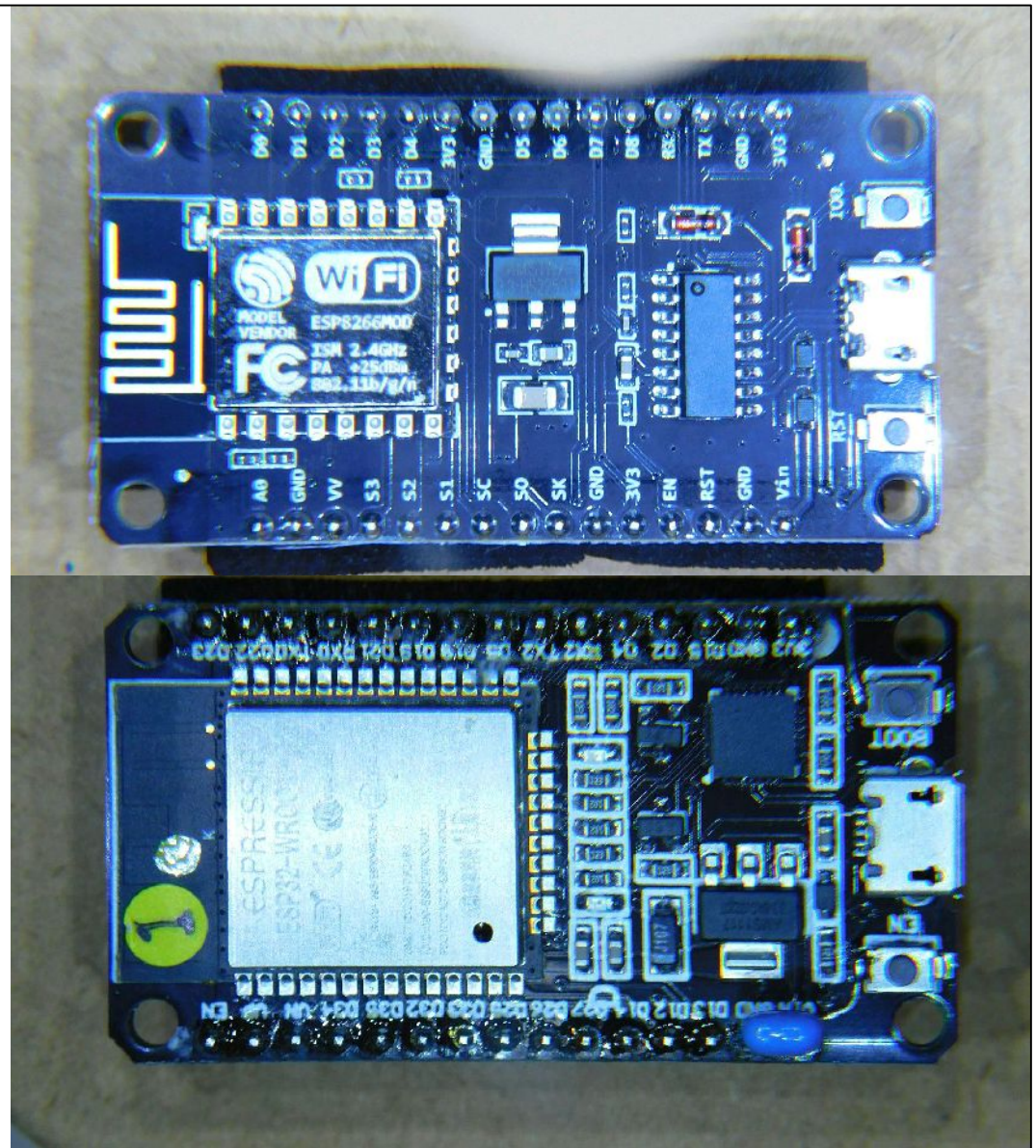
下が、ESP32(ESP-WROOM-32)です。

左下に 黄色の丸い 1と書かれたシールは私が、貼ってます。 因みに、今私が持っている物は、ESP8266と ESP32の両方とも、基板から 30ピンの足が出ています。 よって片側の列は 15ピンです。

ESP32の基板は 基板から出ているピンは 30ピンのものと、38ピンの 物があります。ピン数が多い物の方が 価格が高いようです。

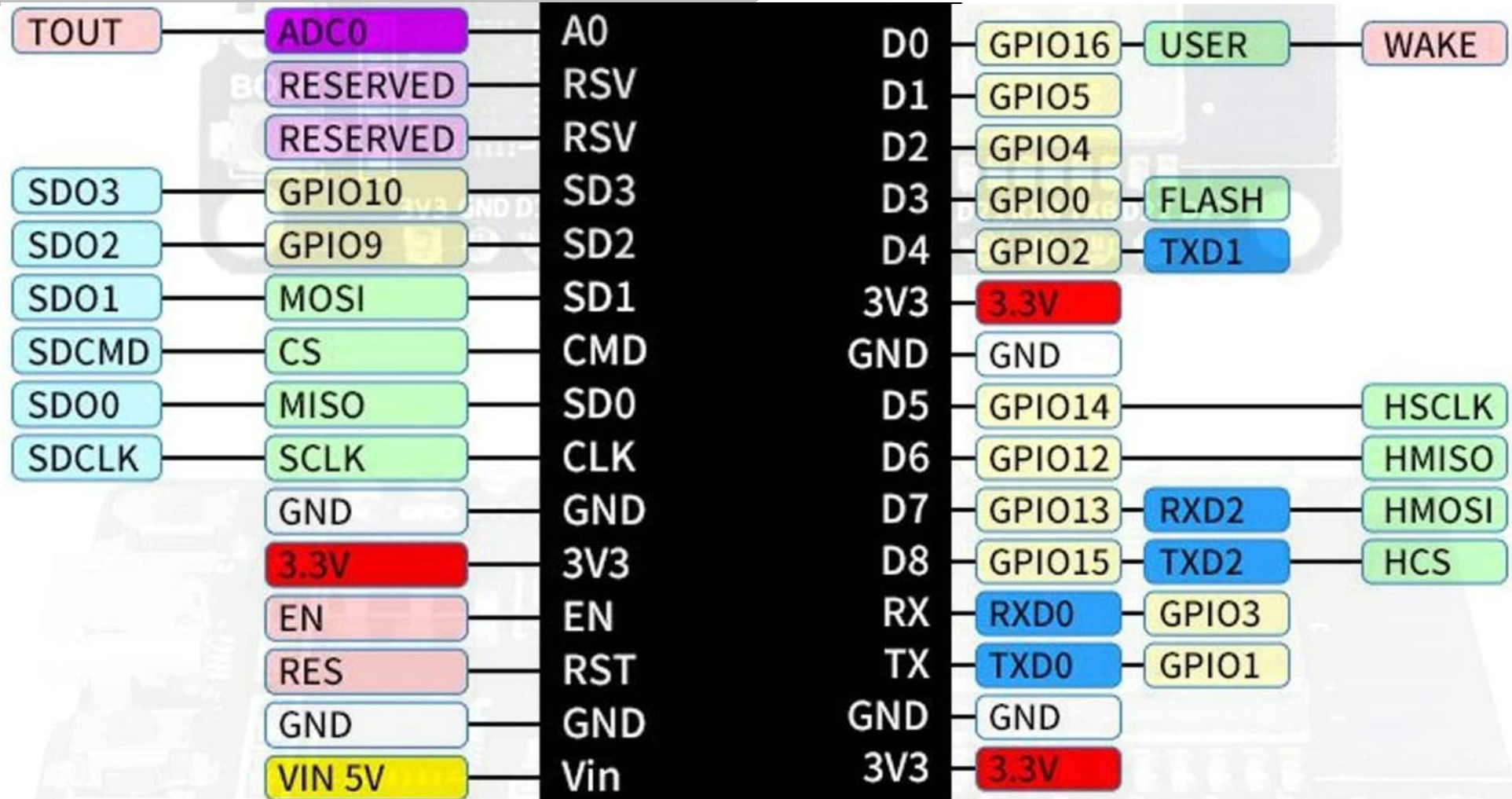
右の2つは、両方とも Amazonで購入しました。 ESP8266が 約 700円で、ESP32の方が 約 850円ぐらいだったと思います。

ESP32の 38ピンの物を 秋月電子に注文しましたが、これは まだ届いていません。

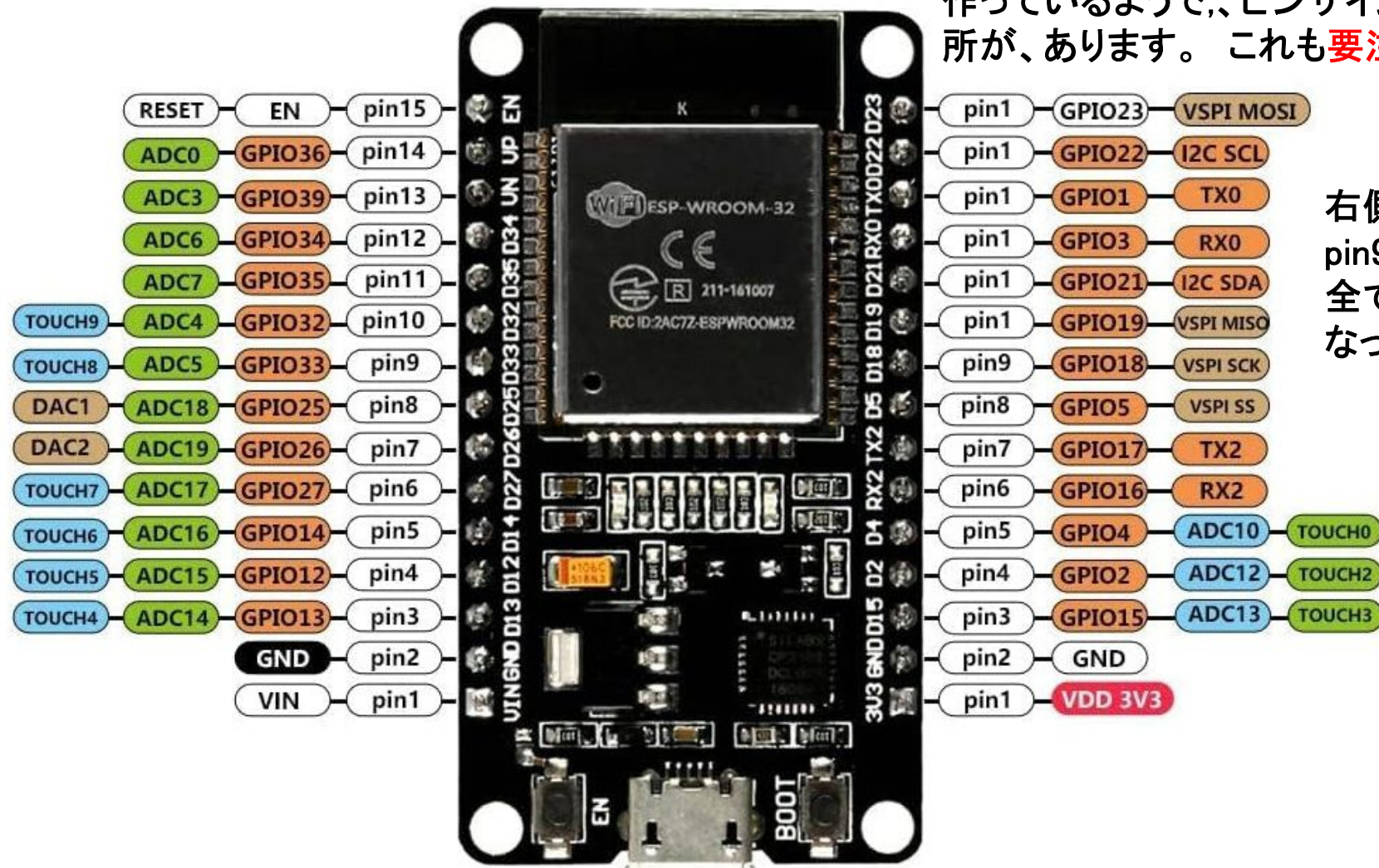


ESP32の基板 30ピン仕様.1

この 30ピン仕様.1 と 次ページの 30ピン仕様.2 とでは、ピンアサインが食い違うところがあります。



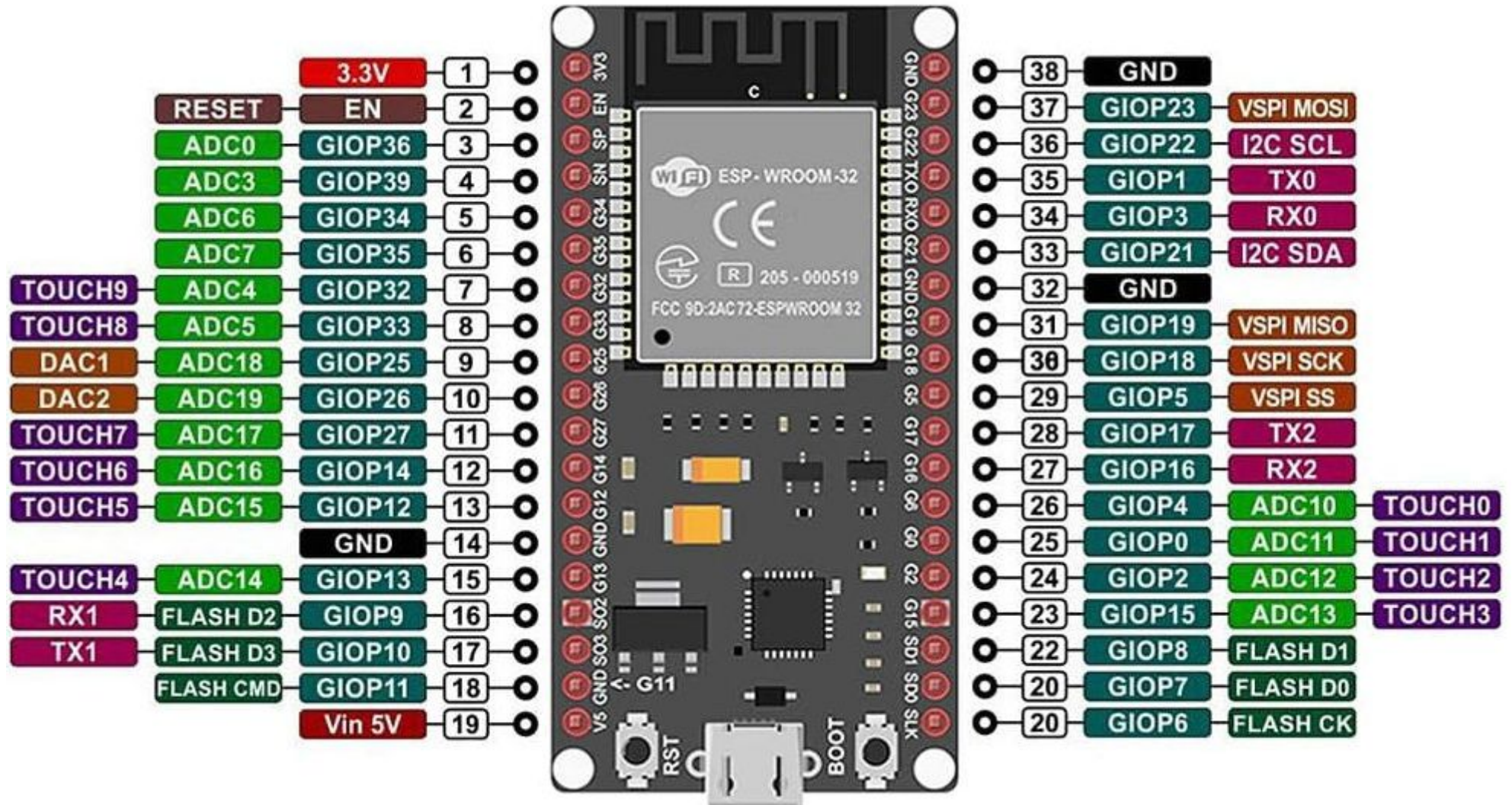
ESP32の基板 30ピン仕様.2



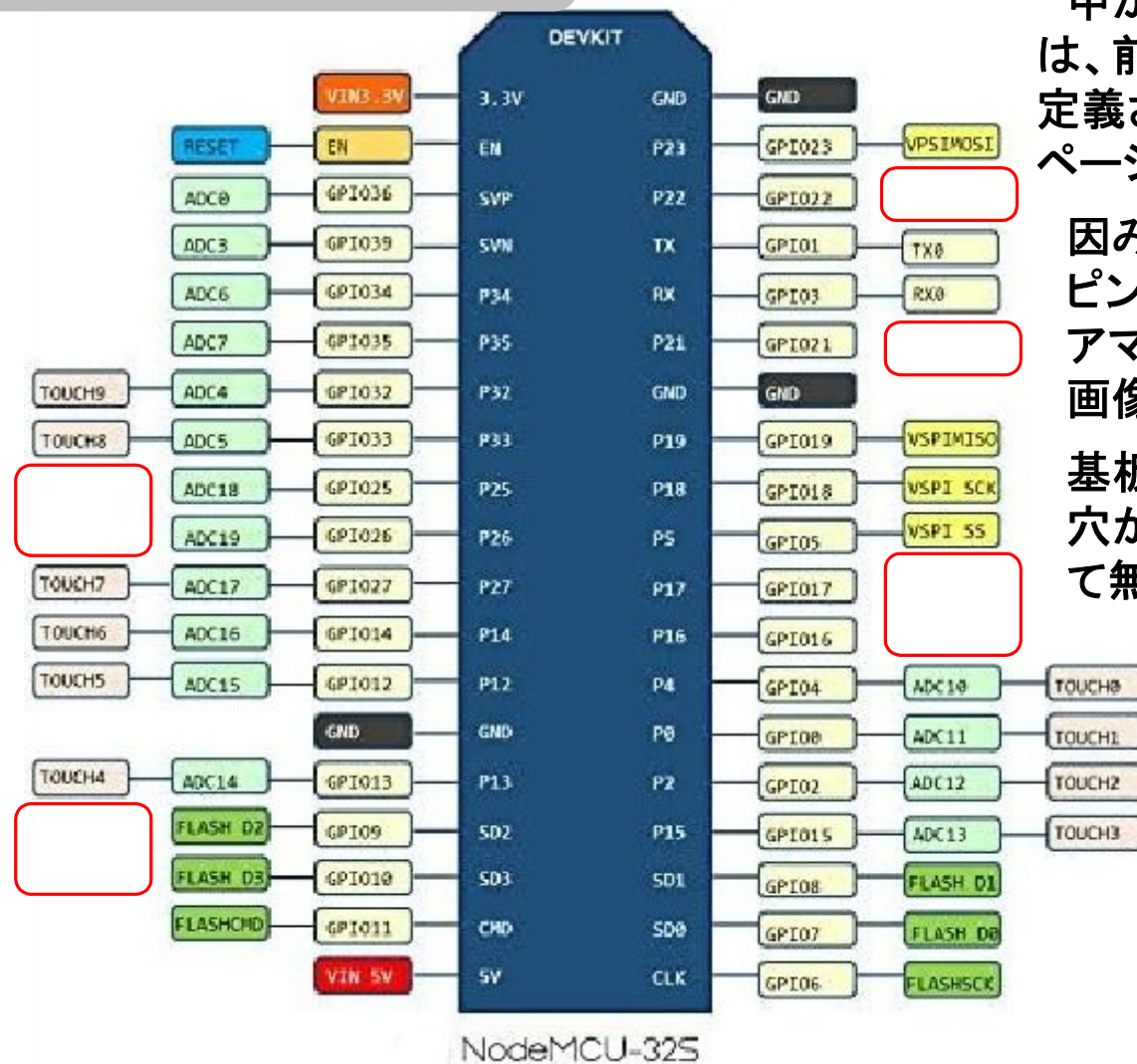
安い基板は、サードパーティが 基板を作っているようで、ピンサインが食い違う所が、あります。これも**要注意**です。

右側のピン、pin9 から上は全て **pin1** になってます。

ESP32の基板 38ピン仕様.1



ESP32の基板 38ピン仕様.2



中が空の 角が丸い赤四角は、前のページでは信号名が定義されています。が、このページではありません。？

因みに、今までの 4つのピンアサイン表は、アマゾンの商品サイトで画像コピーしました。

基板によって、四隅に ネジ穴が、開いている物と、開いて無い物があります。

30ピンの物は、穴が開いている物が、多いです。

38ピンの物は、基板が長くなるので、穴を開けてないのかもかもしれませんね。

ついでに、秋月電子サイトの ESP32 の紹介をします。他にも まだありましたが、この2品種に絞って紹介します。

上の、ESP32-WROOM-32 は 38ピンの 標準的な仕様の物です。1,600円で Amazon より高いですが メーカー純正品と 思います。

私は、下の ESP32-WROVER-E を 1個注文しました。RAMの容量が、8Mbyteと大きいです。これについては、後で説明します。

ESP32-DevKitC-32E ESP32-WROOM-32E開発ボード 4MB

秋月電子サイト
1,600円



在庫グレード: AAA

マイコンボード本体: ヘストセラー

販売コード: 115673

型番: ESP32-DevKitC-32E

発売日: 2020/11/02

メーカーカテゴリ: [Espressif Systems \(Shanghai\) Pte. Ltd.](#)

[よくある質問\(Q&A\)](#)

商品選定・製作の参考にしてください。

1個

- CPU: ESP32-WROOM-32E
- ROM: 4MB
- RAM: 520kB
- GPIO: 19
- 電源電圧min.: 3.3V・5V
- 電源電圧max: 5V
- IO電圧min.: 3.3V
- IO電圧max.: 3.3V
- IDE: ArduinoIDE・ESP-IDF
- 実行言語: C/C++
- 長辺: 48.2mm
- 短辺: 27.9mm

ESP32-DevKitC-VE ESP32-WROVER-E開発ボード 8MB

秋月電子サイト
1,770円



在庫グレード: AAA

販売コード: 115674

型番: ESP32-DevKitC-VE

発売日: 2020/11/04

メーカーカテゴリ: [Espressif Systems \(Shanghai\) Pte. Ltd.](#)

履歴

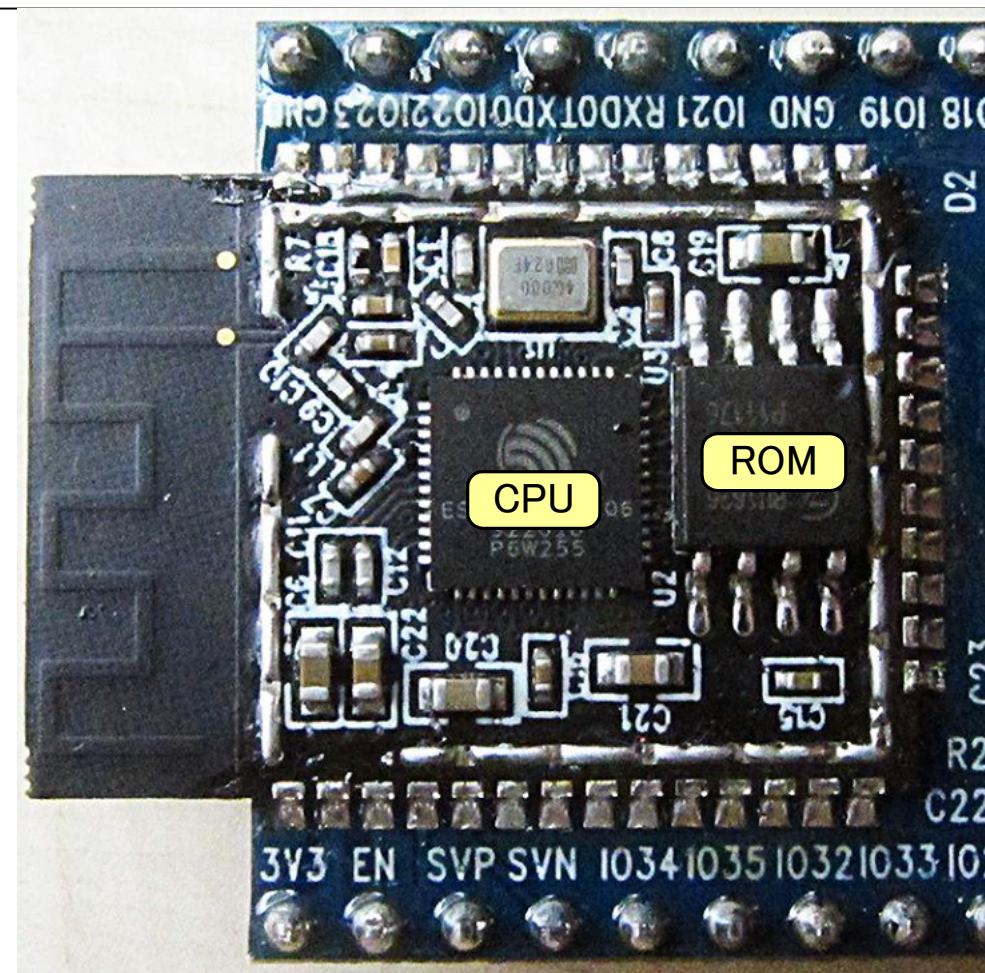
[よくある質問\(Q&A\)](#)

商品選定・製作の参考にしてください。

1個

- CPU: ESP32-WROVER-E
- ROM: 8MB
- RAM: 8MB
- GPIO: 17
- 電源電圧min.: 3.3V・5V
- 電源電圧max: 5V
- IO電圧min.: 3.3V
- IO電圧max.: 3.3V
- 書き込みインターフェイス: USB
- IDE: ArduinoIDE・ESP-IDF
- 実行言語: C/C++
- 長辺: 48.2mm
- 短辺: 27.9mm

まず ROMから説明しますが、基本フラッシュROMです。但し、半導体製造工程の都合か、ROMは、別チップになっています。そしてCPUチップとROMチップの間は、何と、SPIインタフェースで接続されています。因みに、右の画像は、Wikipediaからコピーした シールドを外した画像ですが、左側の黒いチップがCPUです。一辺に12ピン、計48ピンと、チップの真下にアースパッドが有り、計49ピンです。その右にある、8ピンのチップがROMと思われます。8ピンだから、やはりSPIのROMですね。



SPIという事は、当然 bit シリアル通信です。
アクセス速度は遅いです。

何故、遅い **SPIインタフェース**の**ROM**を使っているのか。？ これは一つには **コストダウン**ですね。で、高速をうたい文句にしている **ESP32**で、**ROMアクセス**で **CPU**は **足引っ張られるじゃないか**。？ という事になるのですが **CPU**内の **RAM**の一部の区画を **多分 標準 64Kbyte**だったと思いますが、**キャッシュメモリ**として動作させています。

これで謎が解けた方もいるでしょうけど、**キャッシュメモリ**が、分からない方にはどうしよう。簡単に説明しておきます。

今のパソコンは、大容量の **D-RAM**を搭載しています。 **D-RAM**は **大容量メモリを構築するには、有利**ですが、アドレス変換やリフレッシュ動作が必要で、どうしても**アクセスが遅くなります**。あと、アクセス速度には関係ないですが**消費電力もやや大きい**です。

で、**CPU内部の 高速アクセス可能な、S-RAMの一部を キャッシュメモリ**(高速にデータやプログラムを読み出せるメモリ)として、利用します。例えば、プログラム内で 狭い範囲で、ずっとループし続ける処理があったとして、その**ループ部分が、すっぽりキャッシュメモリに入るならば**最初の1回は、**ROM**から **SPI**経由で読み出す事になりますが、読み出すと同時に、**キャッシュメモリに、読み出した部分を、書き込んでおく処理を**します。よって最初の**1回目**は **時間がかかります**が、**2回目以降は、ROMでは無くて、キャッシュメモリから、高速にプログラムを読み出せる**事になります。キャッシュから、外れると またROMを 読み出す事になるので遅くなります。

何度かやっていると、容量が足りればという条件はありますが、プログラムの殆どがキャッシュに入ってくると考えられます。

キャッシュの容量が やや不足する場合は SPIの ROMをアクセスする頻度が 上がって来ると思われます。

それと、ESP32-WROVER-Eの 内部RAMが 8Mbyteある件ですが、実は 通常のS-RAMでは有りません。PSRAMと呼ばれる物で、外見上は、S-RAMとして振舞い、中身のメモリセルは D-RAMであるメモリです。ローコストで大容量というのが、特徴で、アクセスも通常の SRAMのようにアクセスできます。あと、D-RAMに起因する複雑なメモリコントローラも 不要と書いてあります。

本質的には D-RAMであるため、メモリのリフレッシュ動作が必要で、メモリコントローラも持っているので、消費電力が大きく、速度的には、純粋な D-RAMより劣る。と書いてあります。

用途としては、安価に大容量メモリを実装したい場合に使用する。との事です。

因みに ESP32の中で ROVER という文字が入っているモジュールが PSRAMを 採用しています。それと 買う人は まずいないと思いますが ESP32-SOLO-1 というのも有り、これは シングルコアCPU です。省電力向け以外では利用しない。と書いてあります。

ESP32の 入門用は どれがいいか。？

初めて、ESP32を購入する方には、お勧めは ESP32-WROOM-32 を お勧めします。

それと 電波を出す用途で使用する場合は、**技術適合の認証番号を 正しく取得した製品**を使用して下さい。aの 安いやつは 危ないかも…。秋月電子は 大丈夫と思います。



技術基準適合証明等を受けた機器の検索

[HOME](#) > [基準認証制度](#) > [技術基準適合証明等を受けた機器の検索](#) > 技術基準適合証明等を受けた機器の検索

| 相互承認(MRA)による工事設計認証に関する詳細情報 | |
|-----------------------------|--|
| 工事設計認証番号 | |
| 工事設計認証をした年月日 | 平成30年3月13日 |
| 工事設計認証を受けた者の氏名 又は名称 | Espressif Systems (Shanghai) PTE Ltd. |
| 工事設計認証を受けた特定無線 設備の種別 | 第2条第19号の2に規定する特定無線設備 |
| 工事設計認証を受けた特定無線 設備の型式又は名称 | ESP-WROOM-32, ESP32-WROOM-32 |
| 電波の型式、周波数及び空中線 電力 | G1D 2484MHz 1.0mW/MHz |
| スプリアス規定 | 新スプリアス規定 |
| 周波数等を維持する機能 | 無 |
| BODY SAR | — |
| 備考 | |
| 登録外国適合性評価機関名 | Bay Area Compliance Laboratories Corp |

注：「氏名又は名称」、「型式又は名称」、「電波の型式、周波数及び空中線電力」について変更があった場合は、備考欄に変更履歴として表示します。

担当：総合通信基盤局電波部電波環境課認証推進室

私が、持っている **ESP8266**は、技適マークが付いていませんでした。 **電波を出す用途では使えません。**

私が 持っている **ESP32-WROOM-32**は 認証番号を検索して出てきました。

左ですが、公的な場で 認証番号を出してい
いか分からなかったので、認証番号の欄だけ
消しました。

a で、**ESP32**の 安い製品を買われた方の
レビューで 技適マークが Yのようなマークで
認証番号が、ぼやけて読み取れない状態だっ
たそうです。 別途 FCCIDで調べたところ、登
録されてない偽の番号だったそうです。

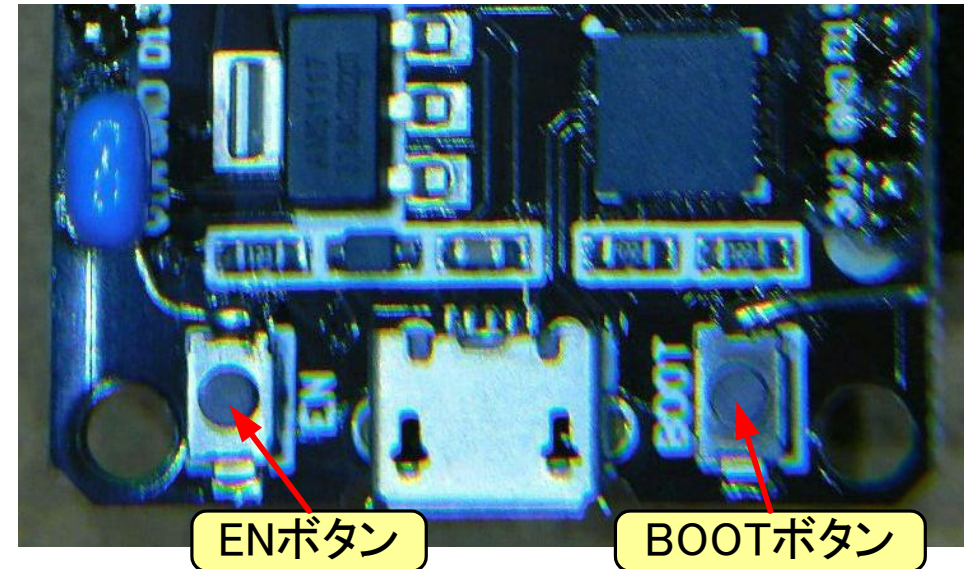
という事で、**電波を出す機器を扱う場合は
注意しましょう。** という事でした。

ESP32に付いてる2つの 押しボタンスイッチは 何？

ESP32には、マイクロUSBコネクタの左右に押しボタンスイッチが 付いています。右の画像ですが、左側の **EN**ボタンは **RESET** スイッチです。押せば即座に **CPUリセット** 信号が 出ます。右側の **BOOT**ボタンですがこれは、CPUを **プログラム書き込みモードに変更**するための押しボタンスイッチです。

しかし、**BOOTボタン単独では機能しません**。手動で、**BOOTモード**というか **プログラム書き込みモード**にするには、**BOOTボタン**を **押しながら、ENボタンを チョンと押す**事で、書き込みモードに変更できます。でも手動で **BOOTモード**にする事は、まずないと思います。

Arduinoを 使っておられる方なら分かりますが、プログラムを **ビルド**したら、自動的に

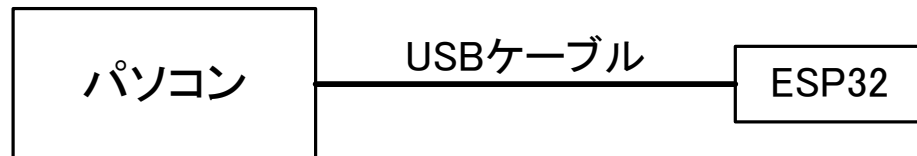


プログラム書き込みモードに 移ります。

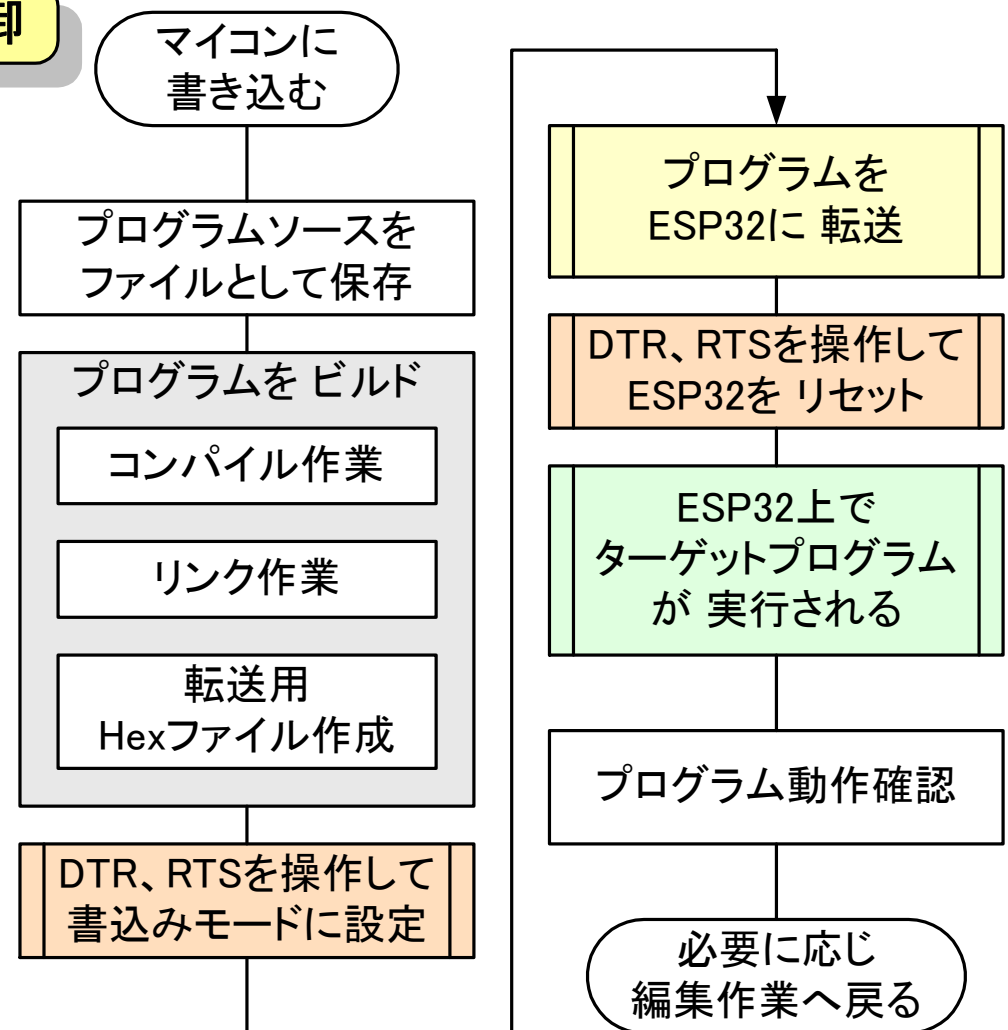
これは、**USBシリアル変換IC**にて、モデムの制御線である、**DTR**(**Data Terminal Ready**)と **RTS**(**Request to Send**)を利用して、**ENボタン**と **BOOTボタン**を リモートコントロールしているのです。そして**プログラム書き込み後**、次は **ENボタン**だけ押した状態を作り **プログラムを実行**します。えっ、よく分からない。？

DTRと RTSによる EN、BOOTの リモート制御

まず、全体の構成ですが プログラム開発用のパソコンが あります。そして ターゲット CPUの ESP32が、あります。 その間を USB ケーブルで、接続します。



パソコン側では、開発環境の **Arduino IDE**を起動します。 **Arduino IDE**の場合、プログラムを 切りのいいところまで入力して、プログラムを動かしてみたい場合、**丸の中に →**のボタンをクリックします。吹き出しには、「**マイコンに書き込む**」と表示されます。このボタンをクリックすると、**Arduino IDE**内で 何が行われているのか説明します。右のフローで、薄い赤の項目が リモートでESP32側の ENや BOOTボタンを 操作しているところです。



DTR信号と RTS信号は、元々 シリアル通信の RS-232C (今は EIA-232Cと呼ぶのが正しいらしいです。)の モデム制御線でした。

モデム制御線は 他にもありますが、ここでは扱わない事にします。

今回の場合 DTR、RTSを 操作するのは、パソコン内の Arduino IDEです。Windows環境の場合 Arduino IDEのようなアプリは COMポートをオープンします。

通常 WindowsのようなOSでは、COMポートも 通常の ファイルアクセスと同じように、Read、Writeで データのやり取りを行います。これは、Writeが COMポートの向こう側(相手側)の 端末に向かい、送信することになり、Readが、相手側から データを受信する事になります。

しかし、通信のような I/O処理は 一元的にファイルとして扱えない場合があります。

その関係で、通信を行う上で 必要となる制御線的な 信号を扱うため IOCTLという概念が デバイスドライバに 用意されました。

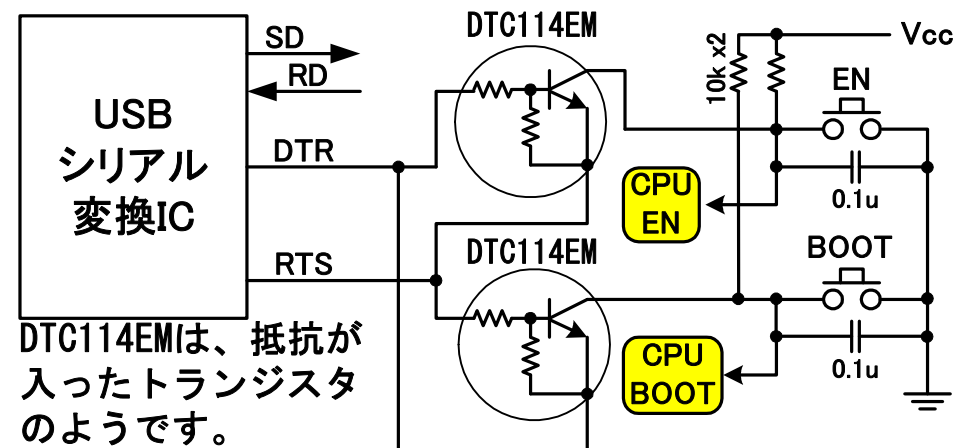
よって、IOCTLの中に DTR、RTSの制御機能が 実装されました。Windows側のアプリとしては、COMポート オープン後 API関数で、DTR、RTSの ON、OFF (プログラム上では True、Falseで扱う場合が多いです)を 行う事になります。昔の RS-232Cケーブルは ケーブル内に DTR、RTSの信号線が 物理的に通っていましたが、今は USB接続が主流なので、USBケーブル内には、DTR、RTSの信号線は通っていません。USBケーブル内は GND、5Vの 電源線と一組のツイストされた 高速な並行伝送路が入ってます。よって半二重通信で、USBの通信規格で定められたパケットが 流れます。そのパケット内に 情報として DTR、RTSのフラグが

入ってます。そして USBインタフェースは Windowsが 約 10msぐらいのインターバルで 各 USBデバイスに対しポーリングを 行っています。その関係で 伝送する信号タイミング等は、リアルタイム性が 損なわれます。

Windowsパソコン側の話が続いたので、今度は ESP32の USB-シリアル変換ICと 先ほどの ENと BOOTのボタンの リモート操作に関わる話をします。

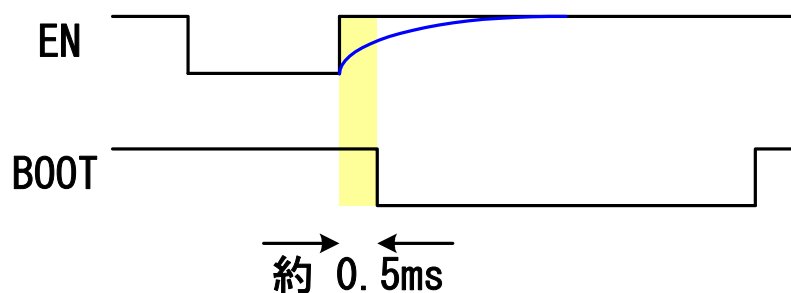
やっと ここに たどり着いた。
USB-シリアル変換ICからは、種類にもよりますが、必ずあるものは SD(送信データ)と RD(受信データ)が、あります。DTR、RTSは 有る場合と 無い場合があります。ESP32では、DTR、RTSの両方が ある物を使用しています。通常この USB-シリアル変換ICは 信号が Lowアクティブで出てくるものが 殆どです。

その DTR、RTSで、どのようにして ENボタン BOOTボタンをリモート制御しているかですが 押しボタンの接点は 信号線を GNDに落とすように 作られています。DTR、RTS信号出力を 小信号トランジスタの ベース、エミッタに 接続し オープンコレクタの形で 押しボタンの接点に パラって接続されています。よって、押しボタンを 指で押さえて 接点の信号線を Lowに落とすか、リモートで DTR、RTSでトランジスタを駆動して、接点の信号線を Lowに落とす事になります。

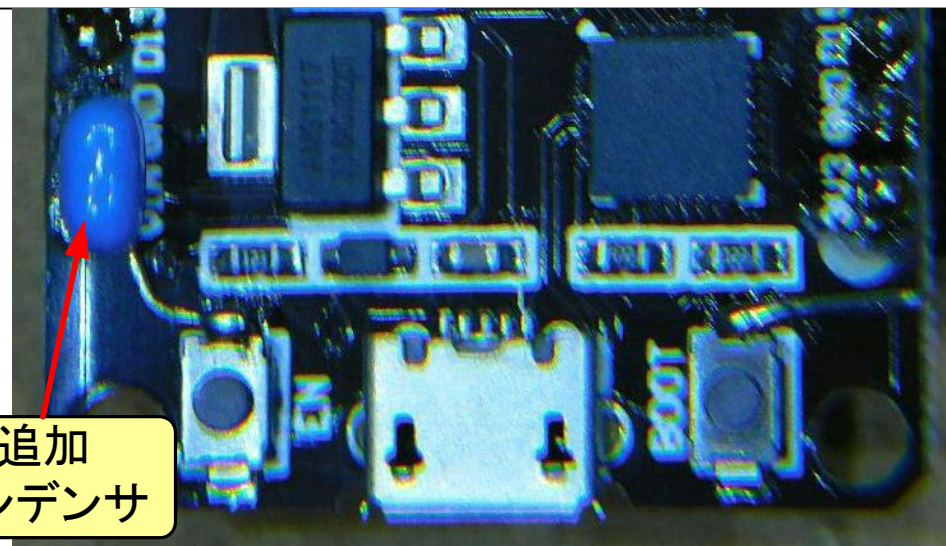


Arduino IDE にて 書き込みに失敗する場合

過去(2019年頃 一時期 ESP32を扱っていた時期があります)にこのようなトラブルが、あったので、トラブルの原因を切り分けるためにも、この ENと BOOTボタンと DTR、RTS の仕組みを、説明しておいた方がいいと思った次第です。要は、間に USBインタフェースが入る事により、DTRと RTS間の 微妙なタイミングが崩れてしまうようです。そのタイミングのずれを、測定した方がおられて図に示しておられました。以下のような図です。

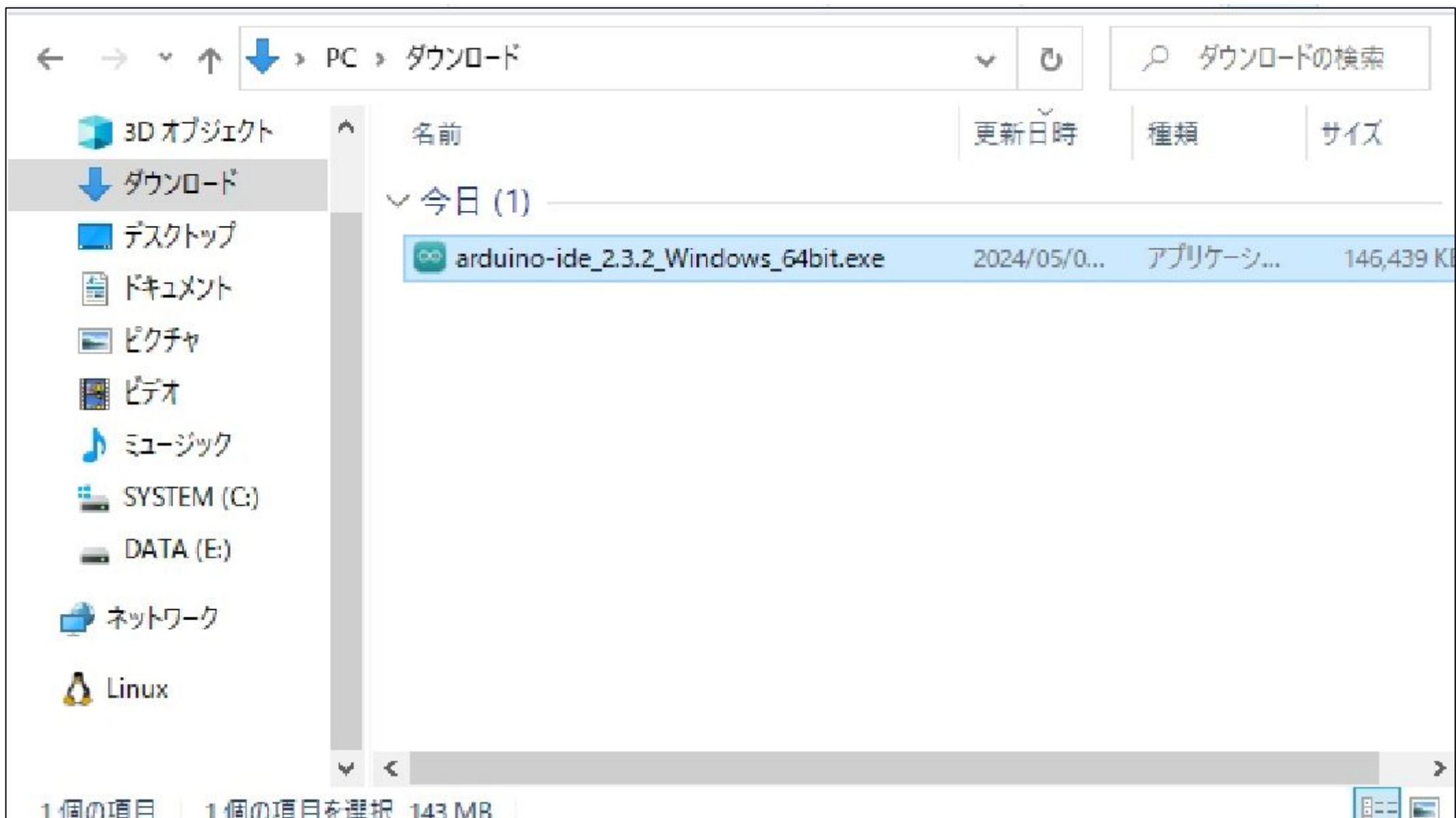


追加
コンデンサ



左のタイミングチャートでは 本来 2つの Lowレベルの状態が、部分的に重なる必要があります。BOOTが、右側に長いのは好ましいのですが、ENが Highに上がるタイミングが、早過ぎて、書き込みに失敗するという事です。

よって、書き込みに失敗するトラブルが出る場合は ENのスイッチと GND間に $0.3 \sim 1\mu\text{F}$ の 積層セラコン等を 追加して下さい。青の積分カーブになると思われます。





```
1 ESP32 ボードマネージャの URL↓
2 ↓
3 https://dl.espressif.com/dl/package\_esp32\_index.json↓
4 ↓
5 |[EOF]
```