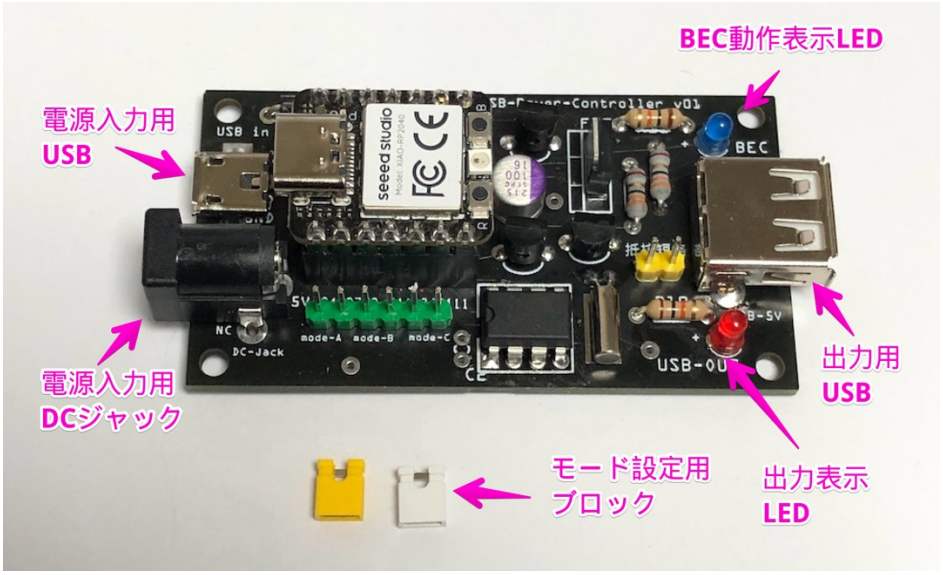
UPC 仕様説明書　V2.3
(Usb Power Controller)

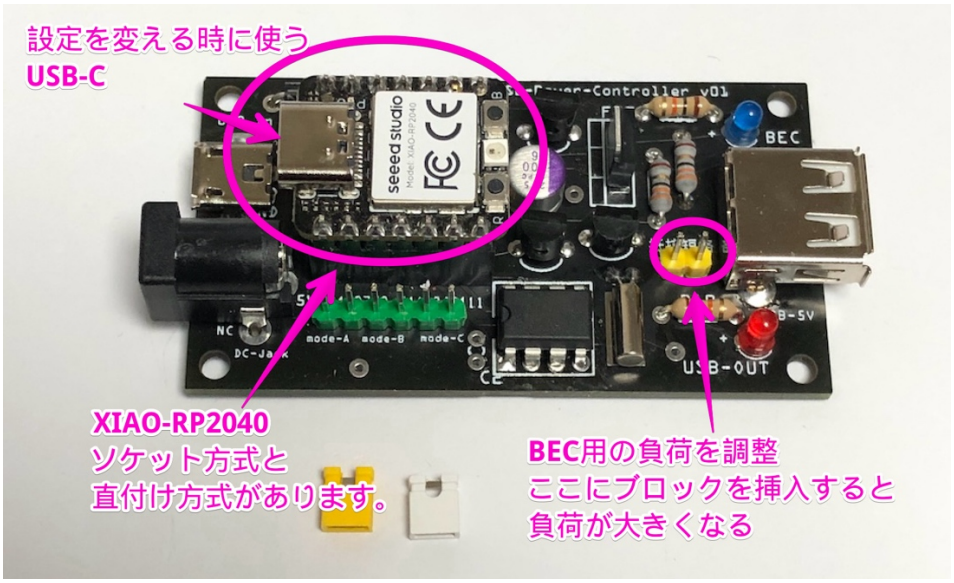
2023/11/16
2023/11/17
2023/11/21
2024/1/18
2024/5/8
2025/2/1

1. 基板について

（ア）


BEC動作表示LED
電源入力用
USB
出力用
USB
電源入力用
DCジャック
モード設定用
ブロック
出力表示
LED

（イ）

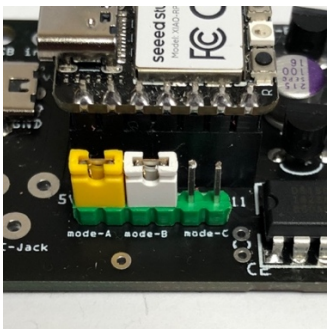

設定を変える時に使う
USB-C
XIAO-RP2040
ソケット方式と
直付け方式があります。
BEC用の負荷を調整
ここにブロックを挿入すると
負荷が大きくなる

2. 設定ブロックについて

（ア）ターミナルブロックを挿入、非挿入でモードを設定する。

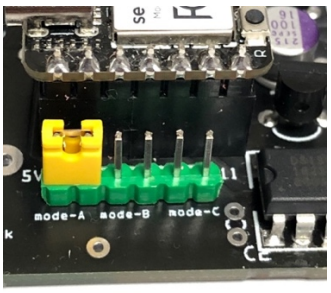
（イ）modeA,modeB のみ使用するモード

① modeA、modeB の挿入で USB-A 電源 on して、BEC 動作を続ける

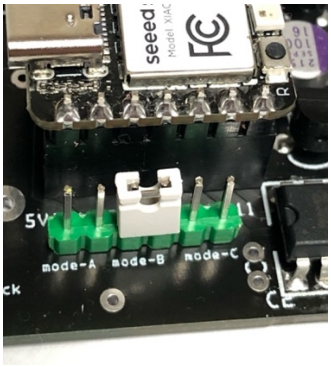


- BEC: バッテリーエコ機能キャンセル

② modeA のみ挿入で電源投入後 USB-A 電源 off でスタートし、
timer_off タイマーカウント後に USB-A 電源 on



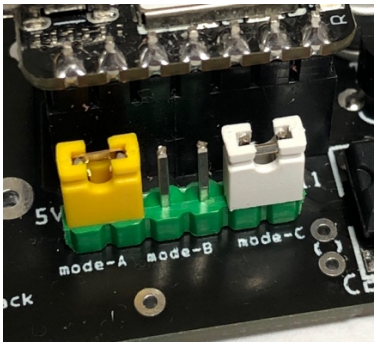
③ modeB のみ挿入で電源投入後 USB-A 電源 on でスタートし、
timer_on タイマーカウント後に USB-A 電源 off



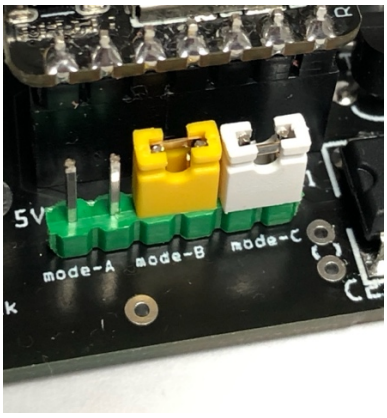
④ modeA、modeB の非挿入で、何もしない

（ウ）modeC

① modeA、modeC の挿入で modeC



② modeB、modeC の挿入で modeC



modeC は A,B どちらでも同じ動作です。

③ 電源投入後 timer_offset オフセットタイマーカウント後に USB-A 電源 on
④ timer_on オンタイマーカウント中 USB-A 電源 on
⑤ timer_on オンタイマーカウント後に USB-A 電源 off
⑥ timer_off オフタイマーカウント中 USB-A 電源 off
⑦ timer_off オフタイマーカウント後に USB-A 電源 on
⑧ ④～⑦を繰り返す。

本機は実時間時計を持っていないので、timer_offset カウント後に午前 0 時になるということを念頭においてください。実時間との差を考えながらセッティングする必要があります。
ただし、電源を入れる時間からちょうど午前 0 時になるように timer_offset をセッテイングすることで、実時間と一致させることはできます。

（エ）抵抗短絡ブロック

① このブロックにすると 33Ω抵抗ひとつが短絡され、BEC 負荷が大きくなります。

② 33Ω直列 2 個で約 75mA、33Ω1 個で約 150mA の負荷がモバイルバッテリーにかかり、エコ機能をキャンセルします。バッテリーにより設定が違うので、動作を確認してください。

3. 設定ファイル

（ア）タイマー設定値

① BEC タイマー（使用するモバイルバッテリーにより設定）

1. 無負荷間隔 BEC_timer
2. 負荷をかける時間 Load_time

② modeA、modeC が使うタイマー値 timer_off

③ modeB、modeC が使うタイマー値 timer_on

④ modeC が使う timer_offset

⑤ 誤差補正設定 ADJ

4. 動作イメージ

（ア）電源側 USB を繋ぐとスタート

① ただし modeC のみターミナルブロックを 3 つとも装着した状態で USB を繋ぎ、スタートのタイミングで modeA、modeB どちらかのプロックを抜くことでタイマーカウントをはじめます。これは、modeC では動作時刻をより正確にしたいと考えてこのようにしています。

（イ）UserLED、RGBLED を消灯する。

（ウ）Config.py を読み込む

（エ）Config_pin.py を読み込み GPIO 設定

① 2,3,27 を出力ピン

② 7,28,29 を入力ピン

③ ModeA,B,C の GPIO を入力にして、A、B、C についてはソフトプルダウンとする。

（オ）モードの確認方法

① modeA ブロック挿入で 100

② modeB ブロック挿入で 10

③ modeC ブロック挿入で 1

④ として、モードブロックを確認して、足し算し mode とする。

⑤ よって、modeC と modeA ブロック挿入で 101、modeC と modeB ブロック挿入で 11 となる

（カ） mode が 110 であれば BEC 動作をする。

① USB-A 電源 on とする。

② Load_time の間、負荷をかけるとともに UserLED-blue を点灯。

③ BEC_timer の間 DeepSleep に入る。

④ DeepSleep から起動して、繰り返し。

（キ）mode が 0、1、111 であれば、プログラムを終了する。

① このとき UserLED 緑を 3 回点滅させる。

② FET を off とする。

③ プログラム終了

（ク）mode を確認して 100 であれば modeA、timer_off カウント後 USB-A 電源 on とする。

① このときカウント中は青点灯、カウント後はみどり点灯とする。

② その後プログラム終了する。

③ この時 BEC 機能稼働するので、モバイルバッテリーでの使用可能

（ケ）mode を確認して 10 であれば modeB、USB-A 電源 on として timer_on カウント後に USB-A 電源 off とする。

① このときカウント中はみどり点灯、カウント後は青点灯とする。

② その後プログラム終了する。その後 BEC も中止する。

③ USB-A 電源 on 時 BEC 機能稼働するので、モバイルバッテリーでの使用可能

（コ）mode を確認して

① 101 か 11 であれば、modeC

② timer_offset オフセットタイマーカウント後に USB-A 電源 on

③ timer_on オンタイマーカウント中 USB-A 電源 on

④ timer_on オンタイマーカウント後に USB-A 電源 off

⑤ timer_off オフタイマーカウント中 USB-A 電源 off

⑥ timer_off オフタイマーカウント後に USB-A 電源 on

⑦ ④～⑦を繰り返す。

⑧ この時 BEC 機能稼働するので、モバイルバッテリーでの使用可能

5. RTC 動作について

（ア）本装置は RTC を用いて正確な時間をカウントしますが、時刻設定機能や外部との時刻合わせ機能はありません。また、時刻保持用のバッテリーも持たないので、電源がなければリセットされます。電源を入れ timer_offset

がカウントアップ後が 0 時 0 分となります。

（イ）精度については、個体により違いますが、概ね日差±10 秒以内だと思われますが、周囲温度によっても変化します。

（ウ）精度補正機能もあり、ある程度微調整ができますが、温度補償機能はないため、季節により精度が変わることになります。

（エ）上記のような状況のため、使用にあたっては十分なマージンを持って運用してください。

6. 拡張機能 （UPC_RTC_xx.py）

（ア）3の項目の動作が基本であるが、modeC の拡張機能として以下を追加しました。

（イ）通常 on と off をずっと繰り返すが、これを回数制限できる機能を拡張機能として追加する。

（ウ）action_setting.py に設定値をセットする

（エ）on_count_enable この機能を生かす設定

（オ）on_count 何回 on 動作を行うかの設定

（カ）on_count_daily 日替わり処理時にリセットするかどうかの設定

（キ）この機能と timer_offset を使うと、毎日朝 7 時から 10 回毎正時に 10 分 on 動作を行うなどの設定が可能となる。

① timer_on 10 分
② timer_off 50 分
③ on_count_enable 1
④ on_count 10
⑤ on_count_daily 1
⑥ などと設定し、timer_offset カウントアップ時に朝 7 時になるように起動します。前日夜 10 時に電源を入れるとしたら 9 時間後にカウントアップするように timer_offset を 540 とセットします。

以上