Ras p-On

Ras p-On Datasheet

電源管理／RTC（リアルタイムクロック）

特徴

■タクトスイッチによる電源オン
■シャットダウンの完了を確認してからの安全な電源供給カット
■電源供給カット時、GPIO の 5V、3.3V、USB ポートへの給電も完全にカット
■タクトスイッチの押下によるシャットダウン要求通知
■タクトスイッチの長押しによる強制終了
■AC アダプタへの通電により自動的に電源をオンにするモードをサポート
集中電源による一斉起動に対応
■電源供給カット時の待機電流 60μA（6V 入力時）
■広い入力電源電圧をサポートしたレギュレータを搭載（6V ～ 25V）
※巻末の「電源の取り扱い上の注意」を参照下さい。
■ノイズの少ないリニアレギュレータを採用
■電圧 5.1V、最大出力電流 3A で Raspberry Pi に余裕を持って電力を供給

■DC 入力用ジャック（外径 5.5mm、内径 2.1mm）
■外部電源出力ポートを用意
LCD パネル等の周辺装置に電源供給可能（合計 3A まで）
■RTC（リアルタイムクロック）を搭載（DS1307）
コイン型リチウム電池（CR1220）でバックアップ
■タクトスイッチの耐 ESD 電圧（人体モデル）±25kV
■Raspberry Pi の USB ポートからの同時給電による逆電流保護回路
■スタック可能な GPIO ポート
■専有 GPIO の変更が可能
■ケースへの組み込みを考慮し、2つのタイプを用意

概要

「Ras p-On」は、Raspberry Pi 4B/3B/3B+/2B に、電源スイッチによる ON/OFF 機能と、使用可能な AC アダプタの選択肢を広げるレギュレータ、現在時刻を刻み続ける RTC（リアルタイムクロック）の3つの機能を提供するアドオンボードです。

Raspberry Pi は、教育や個人利用の枠を超え、今や IoT 機器の試作、開発にはなくてはならない存在に成長しました。ところが、本格的に利用しようとすると、次のような3つの不便を感じることがあります。

① 電源スイッチがない。
② 電源アダプタに自由度がない。
③ RTC が搭載されていない。

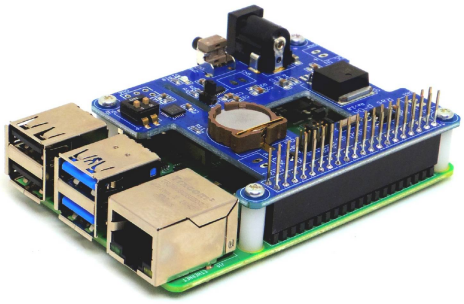
電源スイッチがない為、Raspberry Pi の起動は、AC アダプタのプラグの挿抜で行わなければなりません。

また、シャットダウンしても、CPU が HALT しただけなので、GPIO ポートへの給電は続き、周辺回路などをオフすることができません。そこで、スイッチ操作で電源の給電オン、給電カットを安全に行える回路を用意しました。

また、RaspberryPi の推奨 AC アダプタは 5.1V/2.5A (Raspberry Pi 4B は 3A)で、しかも接続が Micro-USB (Raspberry Pi 4B は USB Type-C)です。実質、この条件に適合する AC アダプタは、ほぼ純正品のみとなります。そこで、レギュレータ回路を搭載し、15W を供給できる電源であれば、市販の様々な AC アダプタやバッテリが利用できるようにしました。壊れやすい USB コネクタを排し、扱いやすい DC ジャックを用意しました。

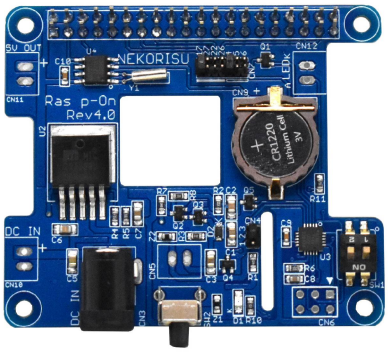
実用運用において致命的な問題となる RTC もアドオンボードに搭載しました。

外観



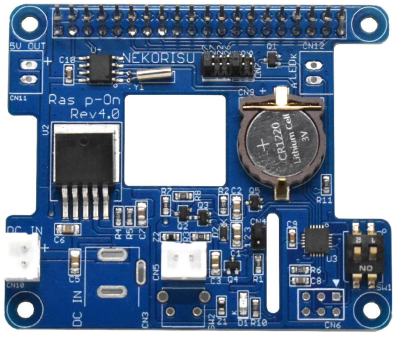
Raspberry Pi4B に接続した状態

Type-A



DC ジャック、タクトスイッチ付き

Type-B



電源入力用コネクタ、スイッチ用コネクタ付き

機能詳細

①電源スイッチ回路

RaspberryPi の電源を完全に ON/OFF できるスイッチ回路を用意しました。スイッチの押下の際に生じるチャタリングを除去し、確実に電源を ON/OFF します。人間の手が触れるスイッチですので、静電気等の高電圧への保護として、+/-25kV の ESD 対策も施されています。
電源 OFF の時、スイッチの押下を監視するための待機電流は 6V の AC アダプタ使用時で 60μA の低消費電流を実現しました。

電源スイッチ回路は、Raspberry Pi の OS に組み込んだ専用のソフトウェアと協調して動作します。専用のソフトウェアはインストールスクリプトと共に提供されます。組み込まれたソフトウェアは Service としてバックグランドで動作します。

②電源 ON のシーケンス

タクトスイッチの押下を検出すると、約 32msec のチャタリング除去の後、レギュレーターを ON し、Raspberry Pi への給電を開始します。(タイミングチャート 1)

③ 電源 OFF のシーケンス

OS のシャットダウンを検出し、安全に電源を OFF します。
シャットダウンの検出には、GPIO を利用します。
シャットダウン検出に使用する GPIO は、GPIO17, GPIO22, GPIO26, GPIO27 の中から1つを選択することができます。(アドオンボード上のジャンパーピンで選択します。)

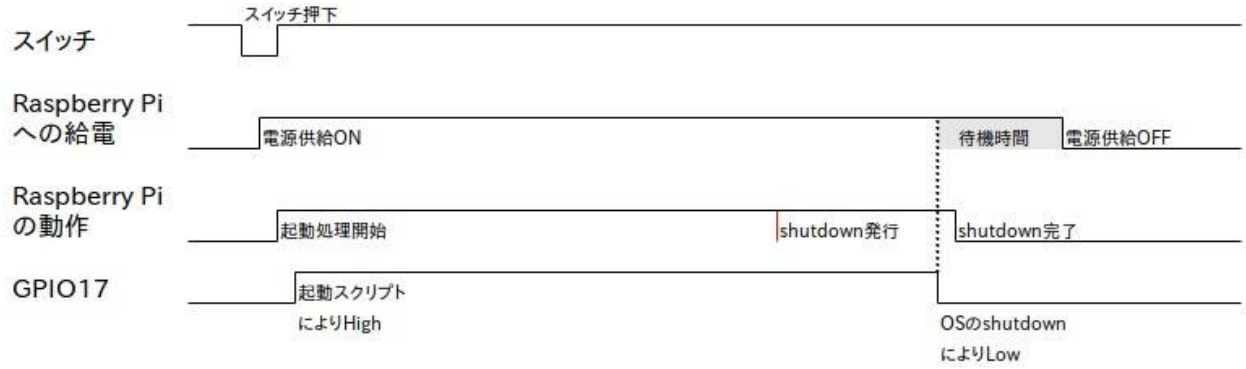
RaspberryPi 起動時に、専用ソフトウェアのサービスがシャットダウン検出用の GPIO ピンを速やかに High ステートにます。

アドオンボードの回路は、この GPIO ピンが Low ステートになるのを検出すると、シャットダウンと判断します。
GPIO17, GPIO22, GPIO26, GPIO27 は、RaspberryPi の仕様上、プルアップされていないため、OS がシャットダウンすると、必ず Low ステートになります。(終了時のスクリプトなどで、GPIO ピンを Low ステートにする必要はありません。)従って、この GPIO ピンを監視していれば、OS のシャットダウンが検出できます。

本モジュールは、当該 GPIO ピンが Low ステートになったことを検出した場合、指定された時間、Low ステートを維持していることを確認した後、レギュレータを OFF して電源の供給をカットします。(タイミングチャート2)
この待機時間は、シャットダウン処理中の SD カードへのアクセスが完全に終了するのを待ち、安全に電源を切るためのものです。
リブートをサポートするためには、シャットダウンで GPIO ピンが一度 Low ステートに移行してから、次の起動で当該 GPIO ピンが High ステートに戻るまでの十分な時間を待機時間として指定する必要があります。待機時間は、アドオンボード上の DIP スイッチで 15 秒～25 秒の間で指定できます。

SW2	SW1	待機時間
OFF	OFF	15 秒(出荷時設定)
OFF	ON	20 秒
ON	OFF	25 秒
ON	ON	電源カット機能 Disable

タイミングチャート 1 電源 ON のシーケンス



タイミングチャート 2 電源 OFF のシーケンス

DIP スイッチの両方が ON（電源カット機能 Disable）のモードは、OS のインストール作業や、本アドオンボード用の専用ソフトウェアをセットアップする前で、シャットダウン検出用の GPIO ピンを High ステートにできない状態に備えて、シャットダウン検出用の GPIO ピンの状態に関係なく、電源供給をカットしないモードです。

④ スイッチによる電源 OFF シーケンス

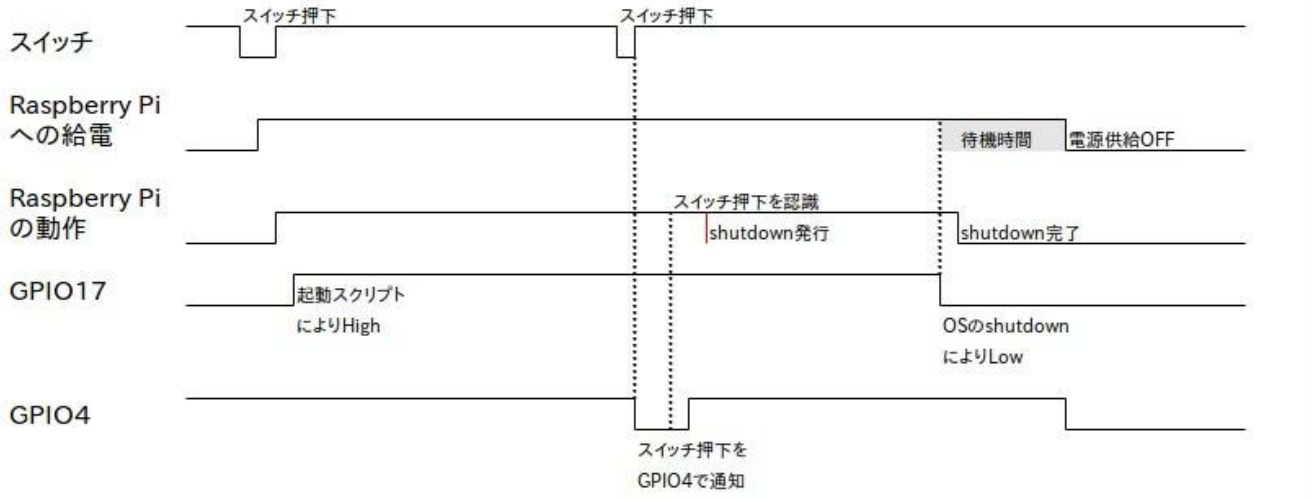
OS の稼働中にスイッチを押下すると、GPIO を介して、スイッチの押下を通知します。通知に使用する GPIO ピンは、GPIO4, GPIO5, GPIO6 から1つを選択することができます。（アドオンボード上のジャンパーピンで選択します。）

これらの GPIO ピンは、RaspberryPi の仕様上、ハードウェアでプルアップされています。通常、これらのピンを他のファンクションに割り当てていない場合、電源 ON と同時に、High ステートに保たれます。アドオンボードは、ボタンの押下を検出すると、当該 GPIO ピンを Low ステートに落とします。

専用のソフトウェアで当該 GPIO ピンを監視し、Low ステートに変化したことを検出したら、shutdown コマンドを発行して、安全にシャットダウンします。

これにより、サーバ用途や組み込み IoT 機器などのヘッドレス構成（モニターやマウス、キーボードなどがない構成）でも、スイッチ操作で安全にシャットダウンすることができます。

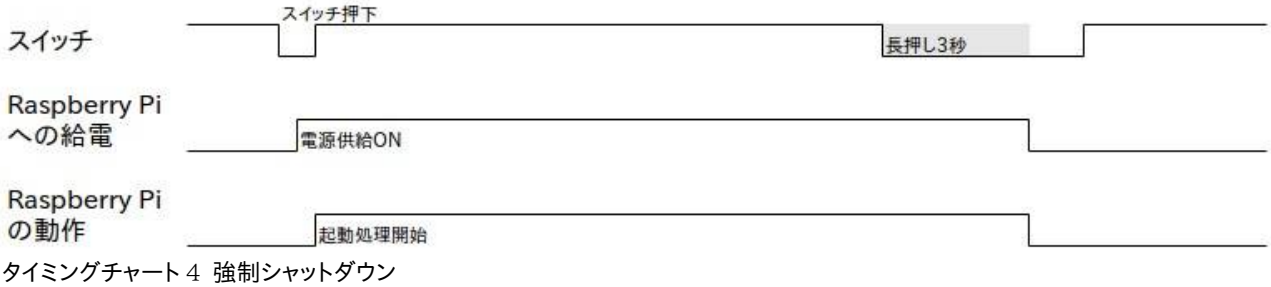
シャットダウンが実行されれば、③の電源 OFF シーケンスにより、安全に、そして完全に、電源を切ることができます。（タイミングチャート 3）



タイミングチャート 3 スイッチによる電源 OFF シーケンス

⑤ 強制シャットダウン
スイッチを 3 秒以上押し続けると、強制的に電源供給をカットします。OS がフリーズして、シャットダウン処理が実行できない場合、強制的に電源を切ることができます。
（タイミングチャート 4）

⑥ 30 秒のブランキング時間
誤操作を防ぐために、電源 ON から 30 秒間はスイッチ操作を受け付けないブランキング時間を設けています。



タイミングチャート 4 強制シャットダウン

⑦ 集中電源に対応
アドオンボード上のショートピンの切り替えにより、AC アダプタから電力が供給されると、電源スイッチを押さなくても、自動的に電源供給をオンにすることが可能です。
集中電源により、各装置を一斉に起動したいシステムに利用できます。
（本モードでも、スイッチによる電源 OFF、次回以降の電源 ON などの機能は全て有効です。）

⑧ LED インジケーター
電源 ON の状態を示す緑色の LED インジケーターを実装しています。OS のシャットダウンを検出したあと、待機時間の間、LED が点滅します。
ケースに組み込む際、LED を外部に引き出せるよう、外部接続ポートも用意しました。
180Ω の電流制限抵抗が入っていますので、殆どの LED をそのまま接続できます。（5V）

⑨ 電源レギュレーター
電源回路には、5.1V/3A の電力を供給できるリニアレギュレータが搭載されています。
6V～25V の広範囲の DC 電源を入力として利用できるため、様々な市販の AC アダプタが使用出来ます。
Raspberry Pi 4B/3B+の性能を充分に発揮するには、3A 以上の電流を供給できる AC アダプタをご使用ください。周辺回路の消費電流も考慮し、余裕のある電源を用意してください。
6V 以上の電源を使用する場合は、放熱対策を充分に検討してください。詳しくは巻末の「電源の取り扱い上の注意」をご参照ください。

⑩ 2つのタイプを用意
Raspberry Pi を実験などの用途でそのまま使う場合と、ケースなどに組み込んで使う場合を考慮し、コネクタ等の違いによる2つのタイプを用意しました。
Type-A
　外径 5.5mm、内径 2.1mm、センタープラスの DC ジャックと、電源スイッチとしてのタクトスイッチを搭載したモデル
Type-B
　ケースなどへの組込みを考慮し、電源供給と電源スイッチへの接続の XH 2P コネクタを搭載したモデル

⑪ RTC（リアルタイムクロック）
コイン型リチウム電池でバックアップされた RTC（リアルタイムクロック）を搭載しています。RaspberryPi の電源が OFF でも時刻を刻み続け、ネットワークのない環境でも正しい時刻を取得できます。
RTC には、RaspberryPi で定番の DS1307 を採用。Raspbian などの OS 標準のドライバで、すぐに利用可能です。
専用のソフトウェアにより、Raspberry Pi の起動時に自動的にシステム時刻を設定します。

⑫ ソフトウェアによる待機時間変更機能

「② 電源 OFF のシーケンス」で前述のとおり、シャットダウン検出用の GPIO（GPIO17, GPIO22, GPIO26, GPIO27）が Low ステートになってから、実際に電源の供給をカットするまでには、待機時間が設けられています。この待機時間は、リブートをサポートするために必要で、シャットダウンで GPIO ピンが一度 Low ステートに移行してから、次の起動で当該 GPIO ピンが High ステートに戻るまでの十分な時間を確保する必要があります。

DIP スイッチの設定により、この待機時間を変更できますが、最近の OS では、リブートに必要な時間が長くなる傾向にあります。そこで、DIP スイッチにプリセットされている時間では足りない場合に対処するために、ソフトウェアで任意の時間を設定する事ができます。

待機時間の設定は、I2C を介して行います。
設定した待機時間は、電源を切ると消えてしまいますので、起動のたびに再設定する必要があります。
（.bashrc などに設定コマンドを記述して、起動時に自動設定することができます。）
何度でも設定できるため、アプリの中から状況に合わせて待機時間を変更できます。例えば、終了時は、待機時間を短めに設定し、再起動時は、待機時間を十分な長さに設定してからリブートするといった使い方ができます。

設定できる時間は、DIP スイッチの位置に依らず1つです。
（DIP スイッチが両方 ON の場合は、電源カット機能は Disable）
ソフトウェアで設定した待機時間が、DIP スイッチによるプリセット時間よりも優先します。

設定は、i2cset コマンドを使用します。
Raspberry Pi OS には、標準でインストールされています。
i2cset コマンドが見つからない場合は、i2c-tools をインストールしてください。

i2cset コマンドでの設定方法は次のとおりです。

```
sudo i2cset -y 1 [Addr] 0x00 [Time]
```

[Addr]には、I2C デバイスアドレスを指定します。
デバイスアドレスは、DIP スイッチの位置で変更できます。

SW2	SW1	Addr
OFF	OFF	0x6A
OFF	ON	0x6B
ON	OFF	0x6C
ON	ON	電源カット機能 Disable

他のセンサー機器とアドレスが重複しないよう選択してください。

[Time]には、待機時間を秒で指定します。
15～200（秒）が指定できます。

例）DIP スイッチが両方 OFF の場合で 30 秒に設定

```
sudo i2cset -y 1 0x6A 0x00 30
```

例）DIP スイッチの SW1 のみ ON で 1 分に設定

```
sudo i2cset -y 1 0x6B 0x00 60
```

参考

i2cset コマンドがない場合は、下記のコマンドでインストールします。

```
sudo apt install i2c-tools
```

I2C が「有効」になっていない場合、下記のいずれかの方法で有効にします。

GUI で設定する場合

「設定」ー「 Raspberry Pi の設定」ー「インターフェイス」ー「 I2C 」 ”有効”にチェック。

または、

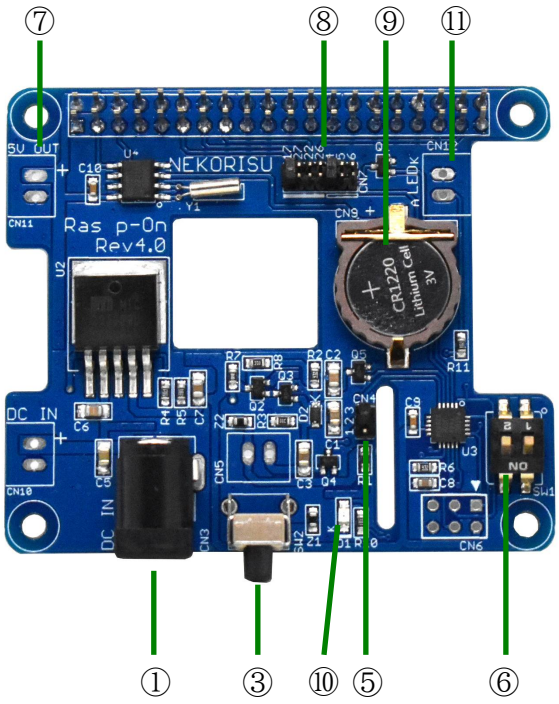
sudo raspi-config で、「 Interface Options 」-「 I2C 」 で、” Enable”に設定。

コマンドで設定する場合

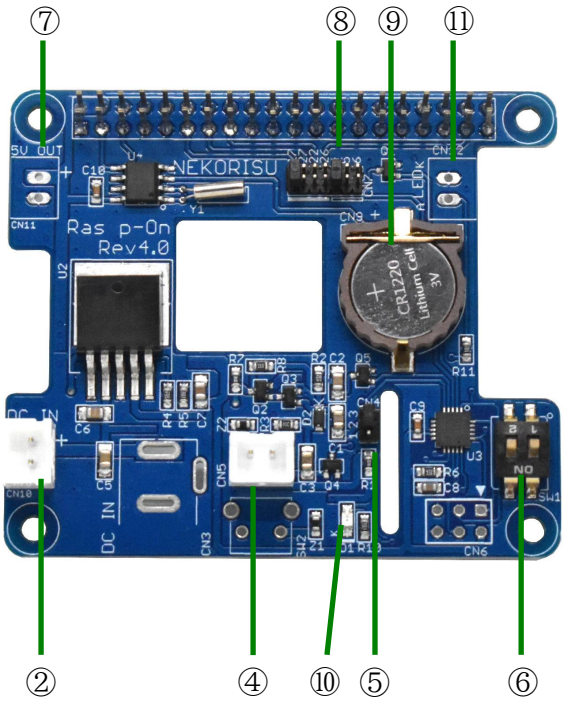
```
sudo raspi-config nonint do_i2c 0
```

各部の名称と機能

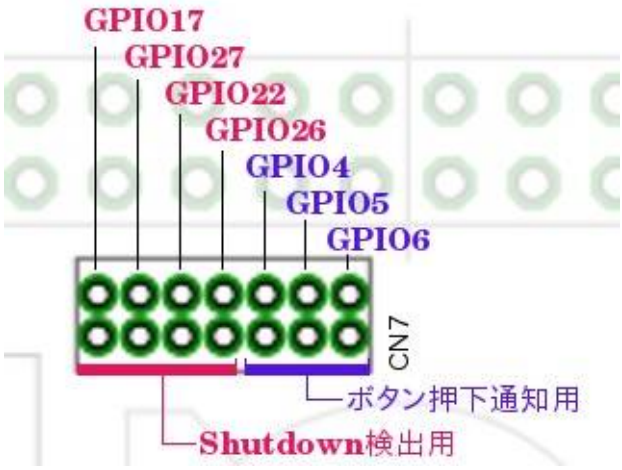
Type-A



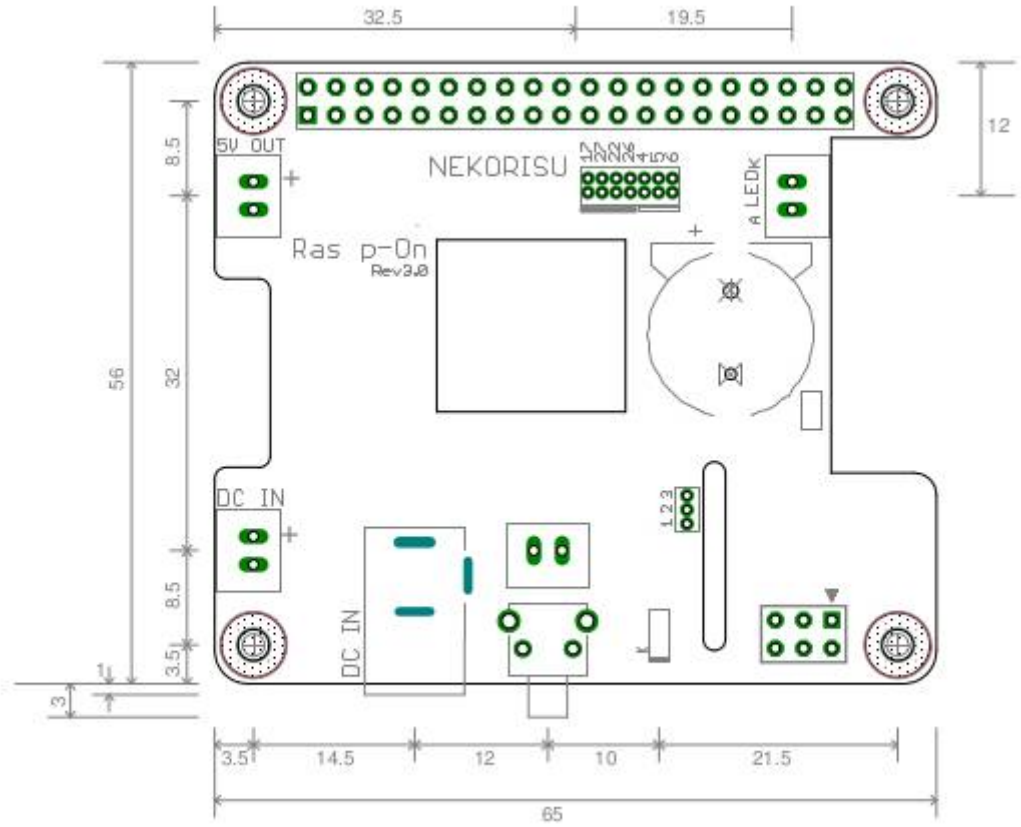
Type-B



No.	端子名	機能
1	DC ジャック	外径 5.5mm、内径 2.1mm、センタープラスの DC ジャック （TypeA にのみ実装）
2	DC IN コネクタ	DC 入力用 XH 2P コネクタ （TypeB にのみ実装）
3	タクトスイッチ	電源 ON/OFF 用のスイッチ （TypeA にのみ実装）
4	外部スイッチ用コネクタ	電源 ON/OFF 用のスイッチを外部に引き出すコネクタ。 プッシュスイッチ（モーメンタリ ノーマルオープン）を接続可能。 XH 2P（TypeB にのみ実装）
5	電源 ON モードの切り替え	1-2 をショートした場合、AC アダプタから給電されると、自動的に電源が ON になります。 2-3 をショートすると、スイッチ操作でのみ電源を ON できます。
6	待機時間設定用 DIP スイッチ	スイッチの組み合わせで待機時間を設定します。 ソフトウェアで待機時間を設定する場合は、I2C のデバイスアドレスをスイッチの組み合わせで指定します。

7	電源外部出力ポート	LCD など周辺回路に電力を供給するポートです。 5.1V 出力。Raspberry Pi の消費電流と合わせて 3A まで供給可能。 XH ２P などのコネクタが接続できます。(2.5mm ピッチ) コネクタは付属しません。組み込まれる筐体の形状等に合わせて適切な方法で接続してください。
8	GPIO 選択	下図のように、ショートピンの位置で、使用する GPIO を選択できます。他の用途に使わない GPIO を選択してください。 
9	RTC バックアップ電池	RTC バックアップ用にコイン型リチウム電池(CR1220)をセットします。
10	LED インジケータ	電源状態を示すインジケータ
11	LED 外部接続ポート	LED インジケータをケース等の外部に引き出す際に使用できるポートです。2.5mm ピッチ。(XH コネクタが実装できます。) 出力電圧 5V。 180Ω の電流制限抵抗が実装されています。 アノード、カソードの極性に注意してください。(基板にシルクで表示されています。)

外型寸法



定格･性能

項目	最小	標準	最大	単位
入力電圧	6		25	V
出力電圧	5.03	5.1	5.17	V
出力電流			3	A
待機消費電流	60			μA
スイッチ押下通知パルス時間	2.8	3	3.2	sec
強制終了スイッチ押下時間	2.8	3	3.2	sec

Raspberry Pi 対応バージョン	Raspberry Pi 4B/3B/3B+/2B
重さ	22g
環境基準	RoHS 準拠

ソフトウェアについて

Ras p-On の電源スイッチ回路、RTC 回路は、Raspberry Pi にインストールされた専用ソフトウェアと協調して機能します。これらのソフトウェアは、シェルスクリプトで記述され、起動時にサービスとしてバックグランドで動作します。

① 電源スイッチ回路用のソフトウェア
電源スイッチ回路には、/usr/local/bin/raspon/pwrctl.sh というスクリプトが用意されています。
スクリプトは、Raspberry Pi OS のバージョン（Bullseye 以前と、Bookworm 以降）で異なります。

※ Raspberry Pi OS のバージョンが Bullseye 以前の場合

```
#!/bin/bash

#################################
# PIN DEFINITION
#################################
GPIO_SHUTDOWN_NOTIFY_PIN=17
GPIO_REQUEST_DETECT_PIN=4

#################################
# Rise up shutdown notify pin
#################################
if [ ! -e /sys/class/gpio/gpio$GPIO_SHUTDOWN_NOTIFY_PIN ]; then
   echo $GPIO_SHUTDOWN_NOTIFY_PIN > /sys/class/gpio/export
   if [ $? -ne 0 ]; then
      exit 1
   fi
   sleep 1
   while [ ! -e /sys/class/gpio/gpio$GPIO_SHUTDOWN_NOTIFY_PIN/direction ]
   do
      sleep 0.2
   done
   echo out > /sys/class/gpio/gpio$GPIO_SHUTDOWN_NOTIFY_PIN/direction
   if [ $? -ne 0 ]; then
      exit 1
   fi
   echo 1 > /sys/class/gpio/gpio$GPIO_SHUTDOWN_NOTIFY_PIN/value
fi


#################################
# Wait shutdown request and shutdown
#################################
if [ ! -e /sys/class/gpio/gpio$GPIO_REQUEST_DETECT_PIN ]; then
    echo $GPIO_REQUEST_DETECT_PIN > /sys/class/gpio/export
    if [ $? -ne 0 ]; then
        exit 1
    fi
    sleep 1
    while [ ! -e /sys/class/gpio/gpio$GPIO_REQUEST_DETECT_PIN/direction ]
    do
      sleep 0.2
   done
   echo in > /sys/class/gpio/gpio$GPIO_REQUEST_DETECT_PIN/direction
   if [ $? -ne 0 ]; then
        exit 1
    fi
fi

while :
do
   pin_state=`cat /sys/class/gpio/gpio$GPIO_REQUEST_DETECT_PIN/value`
   if [ $pin_state -eq 0 ]; then
       shutdown -h now
       exit 1
   fi
   sleep 0.5
done
```

A

B

C

※ Raspberry Pi OS のバージョンが Bookworm 以降の場合

```
#!/bin/bash
```

A

```
###################################
# PIN DEFINITION
###################################
GPIO_SHUTDOWN_NOTIFY_PIN=17
GPIO_REQUEST_DETECT_PIN=4
```

B

```
###################################
# Rise up shutdown notify pin
###################################
pinctrl set $GPIO_SHUTDOWN_NOTIFY_PIN op
pinctrl set $GPIO_SHUTDOWN_NOTIFY_PIN dh
```

C

```
###################################
# Wait shutdown request and shutdown
###################################
while :
do
   pin_state=`pinctrl get $GPIO_REQUEST_DETECT_PIN | awk -F'[|]' '{print $2}' | awk -F'[ ]' '{print $2}'`
    if [ "$pin_state" = "lo" ]; then
        shutdown -h now
        exit 1
    fi
    sleep 0.5
done
```

スクリプトの詳説

スクリプトは、A～C の3つのブロックで構成されます。

ブロック A

使用する GPIO を定義します。

デフォルトでは、シャットダウンを検出する GPIO を GPIO17、電源スイッチの押下を通知し、シャットダウンを要求する GPIO を GPIO4 としています。これらのピンは、アドオンボード上の、GPIO 選択用ショートピンの設定と一致している必要があります。

ブロック B

シャットダウン検出用の GPIO ピン（GPIO_SHUTDOWN_NOTIFY_PIN）を High ステートに設定します。

Ras p-On のアドオンボードは、シャットダウン検出用の GPIO ピンが Low ステートになった時、シャットダウンが完了したと判断します。

そこで、Raspberry Pi の起動直後、可能な限り速やかに、当該 GPIO ピンを High ステートに設定する必要があります。

※ 当該 GPIO ピンは、Raspberry Pi のハードウェア仕様上、プルアップされていないため、Raspberry Pi の CPU が HALT すると、自動的に Low ステートになります。シャットダウン時にソフトウェアで当該ピンを Low ステートにする必要はありません。

ブロック C

Ras p-On の電源スイッチが押下され、シャットダウン要求された事を監視するループです。

シャットダウン要求を受ける GPIO ピン（GPIO_REQUEST_DETECT_PIN）を入力モードに設定したあと、0.5 秒間隔の無限ループでピンの状態を監視します。

当該ピンが Low ステートに変化した場合、Ras p-On からのシャットダウン要求があったと判断し、shutdown コマンドを実行します。

※ 当該 GPIO ピンは、Raspberry Pi のハードウェア仕様上、プルアップされています。Raspberry Pi の起動後、自動的に High ステートになっています。

サービスとして登録

本スクリプトは、Raspberry Pi の起動時に、可能な限り速やかに自動起動する必要があります。
そこで、下記のようなサービス定義ファイルを用意し登録します。(/etc/systemd/system/pwrctl.service)

```
[Unit]
Description=Power Contrlo Script
DefaultDependencies=no
After=slices.target
Before=local-fs-pre.target
ConditionPathIsDirectory=/usr/local/bin/raspon

[Service]
ExecStart=/usr/local/bin/raspon/pwrctl.sh
Restart=no
Type=simple

[Install]
WantedBy=multi-user.target
```

② RTC（リアルタイムクロック）用のソフトウェア

RTC 回路には、/usr/local/bin/raspon/rtcsetup.sh というスクリプトが用意されています。

```
#!/bin/bash

RTC_LOCAL=0

echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device

if [ ${RTC_LOCAL} -eq 0 ]; then
   ntpdate ntp.nict.jp
   if [ $? -eq 0 ]; then
      hwclock -w
   else
      hwclock -s
   fi
else
   hwclock -s
fi

exit 0
```

スクリプトの詳説

本スクリプトは、起動直後に自動実行されます。

・ RTC ドライバ DS1307 に、使用する I2C アドレスを設定します。

・ RTC の時刻を、OS の時刻として設定します。
この時、ネットワークに接続可能で、NTP サーバーにアクセスできると判断した場合、NTP サーバーから ntpdate で得られた時刻をシステム時刻とし、同時に RTC の時計も補正します。
NTP サーバーから時刻を得られない場合は、RTC の時刻をシステム時刻とします。

サービスとして登録

本スクリプトは、サービスとして登録され、起動時に自動実行されます。
そこで、下記のようなサービス定義ファイルを用意し登録します。（/etc/systemd/system/rtcsetup.service）

```
[Unit]
Description=RTC setup script
Wants=network.target
After=network.target
ConditionPathIsDirectory=/usr/local/bin/raspon

[Service]
ExecStart=/usr/local/bin/raspon/rtcsetup.sh
Restart=no
Type=simple

[Install]
WantedBy=multi-user.target
```

ソフトウェアのインストール

必要なソフトウェアは、専用のインストールスクリプトでインストールする事ができます。
インストール手順の詳細は、「ユーザーズマニュアル」を参照してください。

動作確認済みOS

Ras p-On の動作確認済みOSは、下記の通りです。

Raspberry Pi OS Bookworm 32bit Desktop

Raspberry Pi OS Bookworm 32bit Lite

Raspberry Pi OS Bookworm 64bit Desktop

Raspberry Pi OS Bookworm 64bit Lite

Raspberry Pi OS Bullseye 32bit Desktop

Raspberry Pi OS Bullseye 32bit Lite

Raspberry Pi OS Buster 32bit Desktop

Raspberry Pi OS Buster 32bit Lite

FAQ

Q1　電源を入れても、すぐに電源が勝手に切れます。
A1　「Ras p-On」用のソフトウェアが正しくインストールされていません。
ユーザーズマニュアルのセットアップの手順に従って、ソフトウェアをインストールしてください。

Q2　OS をバージョンアップしたいのに、インストール作業中に電源が 切れてしまいます。
A2　OS のインストール中は、「Ras p-On」のソフトウェアがまだ入っていないため、Raspberry Pi が稼働中であることを「Ras p-On」が認識できず、電源を切ってしまいます。
OS のインストール中、または、「Ras p-On」用のソフトウェアをインストールするまでは、DIP スイッチを両方とも ON に設定してください。

Q3　起動直後に電源スイッチを押しても、シャットダウンできません。
A3　起動直後の30秒間は、誤操作を防ぐために、電源スイッチの操作を受け付けません。

Q4　OS をシャットダウンしたのに、電源が切れません。
A4　DIP スイッチが両方とも ON になっています。
DIP スイッチを両方とも OFF にしてください。

Q5　リブート中に電源が切れて、リブートできません。
A5　OS のシャットダウン処理、再起動処理に、時間がかかる環境では、リブート中に電源が切れる可能性があります。 このような時は、「Ras p-On」の待機時間を DIP スイッチにより変更してください。
NOOBS でインストールされた環境では、再起動時に非常に長い時間を必要とします。
Raspberry Pi OS (もしくは Raspbian)を直接インストールされることをお勧めします。

DIP スイッチを変更しても、まだリブート中に電源が切れる場合は、ソフトウェアから待機時間を変更できます。最長、2分まで延長可能です。
詳しくは、P.5 の「⑫ ソフトウェアによる待機時間変更機能」を参照してください。

Q6　GPIO 選択ショートピンの設定を変更すると、電源が勝手に切れてしまい、正しく動作しません。
A6　GPIO 選択ショートピンの設定を変更した場合、ソフトウェアも修正する必要があります。
スクリプトの詳細の頁を参照し、GPIO の定義を適切に修正してください。

Q7　どのような AC アダプタが使用できますか？
A7　出力電圧と、最大出力電流、プラグの形状をご確認ください。
・出力電圧が、6V 以上、25 V 以下
・最大出力電流が、2.5A 以上
・プラグの形状が、外径 5.5mm 内径 2.1mm
Raspberry Pi 4B / 3B+の性能を発揮するには、3A以上の AC アダプタを推奨します。
6V 以上の AC アダプタをご使用になる際は、熱対策を充分に考慮して下さい。
巻末の「電源の取り扱い上の注意」をご参照下さい。

Q8 「Ras p-On」の電源回路が非常に熱くなります。

A8 ご使用になる AC アダプタの電圧が高い場合、その損失が熱となり、電源回路周辺が熱くなます。
高い電圧の電源を使用される場合は、、ヒートシンクなどの熱対策をご検討ください。
85℃まで温度上昇した場合、サーマルシャットダウンが機能します。火傷などにご注意ください。
巻末の「電源の取り扱い上の注意」をご参照下さい。

Q9 「電源 ON モードの切り替え」ショートピンで、給電と同時に自動起動のモードに設定しました。集中電源を入れて自動起動させた時、一度目は起動しますが、2度目以降に起動しない事があります。

A9 自動起動モードは、AC アダプタからの給電の立ち上がりエッジを検出して電源を ON にしています。
AC アダプタの中には、電源供給を切ってから 0V になるまで放電するのに時間がかかるものがあります。
集中電源等で電源を ON/OFF する場合は、OFF した後、AC アダプタが充分に放電する時間を待ってから ON してください。 AC アダプタによっては、1 分以上の間隔を必要とする物があります。

Q10 コイン電池は必要ですか？

A10 「Ras p-On」には、電源が OFF の時でも、リアルタイムクロックの時刻を刻み続けるためにコイン電池が載っています。
リアルタイムクロックの機能が必要ない場合は、電池がなくても構いません。

Q11 コイン電池は交換できますか？

A11 はい。
市販の「コイン型リチウム電池 CR1220」をお買い求め頂き交換してください。

Q12 起動時に NTP との同期ができません。

A12 RTC のセットアップスクリプトが実行される前に、ネットワークが確立している必要があります。
raspi-config により、「Wait for Network at Boot」を設定すると改善する場合があります。

Q13 シャットダウン時の待機時間をソフトウェアで延長したのに、再起動すると短くなります。

A13 i2cset コマンドで設定した待機時間は、電源が切れると消えてしまいます。
従って、起動時に毎回設定する必要があります。
.bashrc に、i2cset コマンドを追記し、起動時に自動的に設定することができます。

Q14 シャットダウン時の待機時間を延長したいのですが、エラーとなり設定できません。

A14 「Error: Write failed」が表示される場合は、デバイスアドレスが一致していません。
DIP スイッチの位置で、デバイスアドレスを 0x6A, 0x6B, 0x6C から選択できます。
DIP スイッチで示されているアドレスと一致しているかご確認下さい。
「Error: Could not open file '/dev/i2c-1' or '/dev/i2c/1' : No such file or directory」が表示される場合は、Raspberry Pi の I2C 機能が有効になっていません。
詳しくは、P.5 の「⑫ ソフトウェアによる待機時間変更機能」を参照してください。

Q15 インターネットのない環境でインストールできますか？

A15 「Ras p-On」のセットアップには、基本的にインターネットへの接続が必要です。
インターネット接続が必要な理由は、
- インストールファイルのダウンロード
- ntpdate コマンドのインストール

の2つです。

どうしてもインターネットに接続することが困難な場合、次の手順でセットアップすることができます。

① 予めセットアップに必要なインストールファイルをダウンロードしておきます。
インターネット接続が可能な PC を使用して、セットアップに必要なファイルをダウンロードしておきます。
ブラウザを使用して、弊社の Web サイト
http://www.nekorisu-embd.com/ras_p-on_products.html
からダウンロードするか、
wget http://www.nekorisu-embd.com/download/raspon-installer.tar.gz
でダウンロードできます。

② ダウンロードしたファイルを USB メモリに入れ、Raspberry Pi と接続します。

③ Raspberry Pi で、次のような手順でセットアップします。
```
sudo tar xzpvf raspon-installer.tar.gz
sudo ./install.sh  -local
```

注）上記手順でセットアップした場合は、NTP サーバーを使って時刻補正を行う機能は動作しません。

Q16 ソフトウェアのアンインストールの方法を教えてください。

A16 次のコマンドで、完全にアンインストール可能です。
```
sudo systemctl stop pwrctl.service
sudo systemctl disable pwrctl.service
sudo systemctl stop rtcsetup.service
sudo systemctl disable rtcsetup.service
sudo rm -r /usr/local/bin/raspon
```

Q17 「Ras p-On」で占有している GPIO はありますか？

A17 「Ras p-On」は、デフォルトの設定で、下記の GPIO を使用します。

GPIO17	シャットダウン検出用
GPIO4	シャットダウン通知用

これらの GPIO は変更可能です。

電源の取り扱い上の注意

① 本アドオンボードで電源を供給する場合、Raspberry Pi 上の Micro-USB/USB Type-C からの電源供給は行わないように注意してください。
本アドオンボードの電源回路には、逆電流防止回路が含まれているため、故障の原因とはなりませんが、Raspberry Pi 4B / 3B+は、USB Type-C / Micro-USB からの電源入力時、逆電流防止用のダイオードが省略されたため、同時に電源が供給されると故障の原因となります。
（Raspberry Pi 3 model B, Raspberry Pi 2 model B には、保護回路が実装されています。）

② アドオンボード Type-B において、電源をコネクタから供給する場合、使用する線材は定格電流 3A、15W 以上のものを使用してください。使用する線材、ジャック、コネクタによっては、Raspberry Pi 及び周辺回路に必要な電力を充分に供給できない場合があります。
DC IN コネクタに適合するハウジングは、JST の XHP-2 です。
極性を確認して、正しく配線してください。

③ 本ボードに供給する電源は、6V／3Aを強く推奨致します。
本ボードのレギュレータは、リニアレギュレータを採用しているため、電源の損失分は全て熱として放出されます。
例えば、24Vの電源を接続された場合、

(24V - 6V) x 3A = 54W

となり、最大54W分の電力が熱になります。これは数十秒で 100℃に達する熱量です。
これを抑えるためには、相応の放熱対策が必要となりますが、かなり大きなヒートシンクと、強力なファンが必要となります。
組み込まれる他の機器との兼ね合いで、どうしても 6V 以上の電源を使用する必要がある場合は、本ボードへの入力の前に、DC/DC コンバーターで 6V 程度に降圧して供給するのが現実的です。