2017/11/16

# Maxell インテリジェントバッテリー

# 並列接続アダプタ仕様書

Ver 1.0

空撮サービス株式会社

# 1　はじめに

## 1.1　インテリジェントバッテリーの必要性

ドローン用のバッテリーはラジコン用と異なり、より広範囲に使われることから非常に高い安全性が求められる。しかし DJI を除きドローン業界で安全なバッテリーへの採用はなかなか進んできていない。実際に墜落事故から発火事故に至っている例もあり、安全な運用にはインテリジェントバッテリーが必要である。マクセル社のリチウムイオン二次電池バッテリー（以下　Maxell バッテリー）はドローン用としては初めてインテリジェント機能を備え過充電・過放電・短絡に対する保護機能を備えて安全性を高めている。

## 1.2　ドローンでのバッテリー管理

ドローンでのバッテリー管理は主にバッテリー電圧で残容量を判断しているが、一般的な LIPO バッテリーと Maxell バッテリーは電圧が異なり、LIPO の常識で判断すると誤判断することになる。この様にバッテリーの電圧だけで残容量を判断する方式は誤解を生みやすく、一般人には理解不能で有り容量不足による墜落の元にもなりやすい。Maxell バッテリーは I2C による通信機能を持っており、バッテリーの電圧、電流、温度、セル毎の電圧、残電流量、最大電流量、エラーステータスなどを見ることができ、オートパイロットからログに残すことができる。この機能により正確な残容量を操縦者が常に確認することができるようになり、容量不足による墜落や電圧低下による不意の強制帰還中の衝突事故を防ぐことができる。またトラブルの場合でもログを調べることにより迅速に原因の究明ができる。

## 1.3　並列接続

この様に Maxell バッテリーは画期的なバッテリーであるが、残念ながら標準で 5200mAh の容量では enroute QC730 など中型機体では容量不足である事も否めない、そこで並列接続して 10400mAh の容量が確保できれば QC730 クラスで 15 分程度のフライト時間を確保することができ実用性が高まる。また並列にすることにより冗長性が高まり、どちらかに問題が起きたときでも安全に着陸でき、安全性が高まる。しかしそのまま並列直結で接続した場合、様々な問題が発生する。また I2C 接続もバッテリーの ID が共通で固定されているため、そのままでは片方の情報は読めますが両方の情報は読めません。

これらの問題を解決したのが、当社の並列アダプタです。

## 2　並列処理アダプター

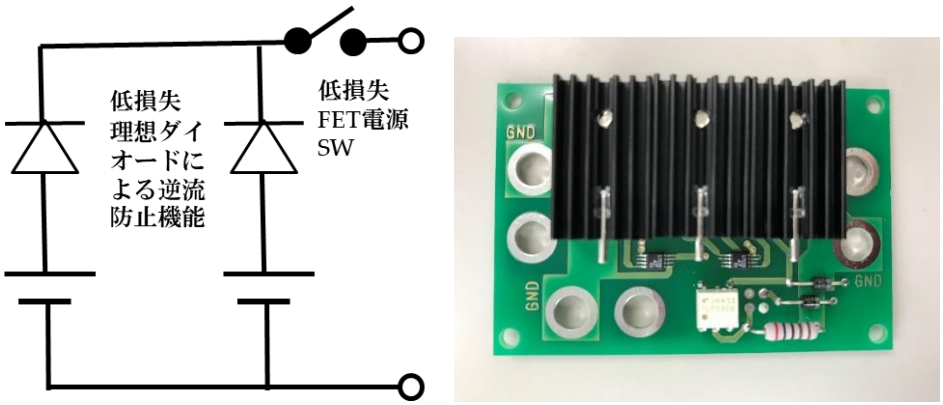
これらの様々な問題を解決し、使いやすく安全な運用を可能とする並列アダプタを開発しました。

### 2.1 オン・オフボタン

機体の調整などしていると、頻繁に電源のオン・オフが必要になる。一般にドローンには電源のオン・オフ　スイッチは付いていないのが普通である。そのため電池のコネクターを直接入れたり出したりしているのが普通である。しかしこの作業は間違ってけがをしたり、ドローンを傷つけてしまうこともあり得る。本アダプタには前面にオンスイッチとオフスイッチが付いており、電源の入り切りが簡単にできます。低損失の FET を用いており、電圧降下はほとんどありません。オン状態は電気的に保持されているので機械接点による誤切断もありません。

## 2.2　逆流防止機能

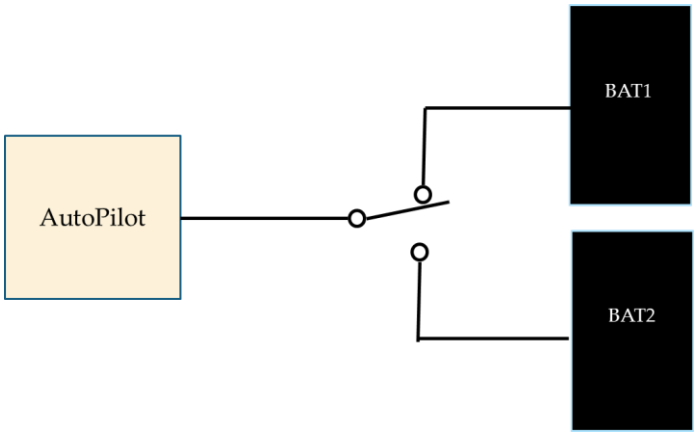
充電状態が異なるバッテリーを並列直結にすると、充電量の大きいバッテリーから充電量の少ないバッテリーに大量の充電電流が流れ過充電保護機能によりバッテリーのヒューズが溶断し使用不能になる。確かに RC 業界の一般常識では並列接続する場合は満充電のバッテリー同士を接続するというのが、常識ではあるが一般人にはなかなか徹底することは難しい。つい充電中のバッテリーを使ってしまうことも考えられる。この様なときにも安全にバッテリーを保護するにはダイオードによる逆流防止機能が必要であるが、ドローンのように大量の電流が流れる場合は大きな損失が発生して莫大な発熱がおき、実用的では無い。下図のように低損失 FET により理想ダイオードを実現しました。




FET と機能 IC を組み合わせた理想ダイオード回路により低損失で確実な逆流防止機能を実現した。

## 2.3　I2C 読み取り

Maxell バッテリーは I2C 接続によりドローンのフライトコントローラから内部の情報を読み取れるようになっています。この機能は Ardupilot 3.2 から実装され接続すると電圧・電流・温度などの情報が読めるようになります。ただ残念ながら現バージョン Ver 3.5 では残量は正しく読めないようです。また I2C ID が同一のため二つのバッテリーを pihawk の I2C に接続しても正しく読めません。そこで下図のように I2C Switch により二つのバッテリーを切り替えて読める様にします。




## 2.4 Ardupilot ドライバー

I2C スイッチを通じて、各バッテリーの状態を読み取りログにするためには専用のドライバが必要になります。

### 2.4.1 Ardupilot ドライバー

I2C の情報は下記になります。

| I2C | 名称 | バイト数 | 単位 | 備考 |
|---|---|---|---|---|
| 0x08 | Temperature | 2 | 0.1° K | |
| 0x09 | Voltage | 2 | mV | |
| 0x0A | Current | 2 | +-mA | 2倍にする |
| 0x0D | State of Charge（SOC) | 1 | % | 残電流割合（低温補正有） |
| 0x0F | REMAINING_CAPACITY | 2 | mAh | 2倍にする |
| 0x10 | FULL_CHARGE_CAPACITY | 2 | mAh | 2倍にする |
| 0x4F | State of Health(SOH) | 1 | % | |
| 0X50 | Safety Alert | 4 | | |
| 0X51 | Safety Status | 4 | | |
| 0X52 | PF Alert | 4 | | |
| 0X53 | PF Status | 4 | | Bit0 セル過放電<br>Bit1 セル過充電 ヒューズ断<br>Bit2 充電過電流 |

### 2.4.2 Mavlink 出力

Mavlink の出力は下記になります。（バッテリー関連のみ抜粋）

| Field Name | Type | 仕様 | 出力内容 |
|---|---|---|---|
| voltage_battery | uint16_t | Battery voltage, in millivolts (1 = 1 millivolt) | スロット毎の Voltage の最大値 |
| current_battery | int16_t | Battery current, in 10*milliamperes (1 = 10 milliampere), -1: autopilot does not measure the current | Current の全スロット分の合計(10mA 単位) |
| battery_remaining | int8_t | Remaining battery energy: (0%: 0, 100%: 100), -1: autopilot estimate the remaining battery | RemainingCapacity の全スロット合計 /（スロット毎の DesignCapacity 最大値 x2) * 100 |
| | | | |
| BATTERY_STATUS ( #147 ) | | | |
| | | | |
| **Field Name** | **Type** | **仕様** | **出力内容** |
| Id | uint8_t | Battery ID | 0 固定 |
| battery_function | uint8_t | Function of the battery | 0(MAV_BATTERY_FUNCTION_UNKNOWN)固定 |
| Type | uint8_t | Type (chemistry) of the battery | 0(MAV_BATTERY_TYPE_UNKNOWN)固定 |
| temperature | int16_t | Temperature of the battery in centi-degrees celsius. INT16_MAX for unknown temperature. | Temperature の最大値(1/100℃単位) |
| voltages | uint16_t[10] | Battery voltage of cells, in millivolts (1 = 1 millivolt). Cells above the valid cell count for this battery should have the UINT16_MAX value. | スロット 1 または 2 に接続されている電池の Cell Voltage 1～6(mV) |
| current_battery | int16_t | Battery current, in 10*milliamperes (1 = 10 milliampere), -1: autopilot does not measure the current | Current の全スロット分の合計(10mA 単位) |
| current_consumed | int32_t | Consumed charge, in milliampere hours (1 = 1 mAh), -1: autopilot does not provide mAh consumption estimate | FC 起動時からの消費容量の合計(mAh)<br>全スロットについて RemainingCapacity に変化があった場合の変化値の合計 |

| | | | |
|---|---|---|---|
| energy_consumed | int32_t | Consumed energy, in HectoJoules (intergrated U*I*dt) (1 = 100 Joule), -1: autopilot does not provide energy consumption estimate | -1 固定 |
| battery_remaining | int8_t | Remaining battery energy: (0%: 0, 100%: 100), -1: autopilot does not estimate the remaining battery | RemainingCapacity の全スロット合計 /（スロット毎の DesignCapacity 最大値 x2) * 100 |

### 2.4.3 DataFlash ログ出力

バッテリー１情報　右側のバッテリの情報

| Field Name | Type | 出力内容 |
|---|---|---|
| TimeUS | uint64_t | 記録時時刻(FC 起動からのマイクロ秒) |
| Volt | float | Voltage(V 単位) |
| Curr | float | Current(A 単位) |
| CurrTot | float | FC 起動時からの消費容量の合計(mAh 単位)<br>RemainingCapacity の変化量 |
| Temp | int16_t | Temperature(1/100℃単位) |
| V1 | uint16_t | Cell Voltage 1(mV) |
| V2 | uint16_t | Cell Voltage 2(mV) |
| V3 | uint16_t | Cell Voltage 3(mV) |
| V4 | uint16_t | Cell Voltage 4(mV) |
| V5 | uint16_t | Cell Voltage 5(mV) |
| V6 | uint16_t | Cell Voltage 6(mV) |
| V7 | uint16_t | 0 固定 |
| V8 | uint16_t | 0 固定 |
| V9 | uint16_t | 0 固定 |
| V10 | uint16_t | 0 固定 |

バッテリー2 も同様

## 2.5　並列統合情報

ArduPilot ではバッテリー1 本が搭載されていることしか想定していないため、並列接続された各バッテリーの情報を統合する必要がある。二つのバッテリの統合した情報は下記で計算される。

注）満充電状態のバッテリーを一つだけ接続した場合、残容量は５０％以下になります。

1.残容量(%)

$$\text{統合残容量}=\frac{\text{REMAINING\_CAPACITY1} + \text{REMAINING\_CAPACITY2}}{\text{FULLCHARGE\_CAPACITY} \times 2} \times 100$$

２．バッテリー電圧

Battery 1 または Battery2 のどちらか高い方の電圧

（これは並列誤挿入防止回路があるときの条件、通常は同じとなる）

３．電流値

Battery 1 と Battery2 の電流値を 2 倍して加算

４．その他パラメータは、それぞれのバッテリーログに格納される。

## 2.6　バッテリーエラー情報

バッテリーから読み出せるエラー情報には 0X53 PF Status がある、Bit1 セル過充電ヒューズ断はセルが過充電になったため、保護回路が働きメインのヒューズを破断して最悪でも発火などが起きないようにする。これは充電状態が異なるバッテリーを直結したばあいなど逆流によっても発生する。ヒューズが破断するのでそのままでは使用できない。その他のエラーは、必ずしも検出できるとは限らない。またショートなどの場合、１００A 以上の電流でヒューズが溶断して出力電圧は０V となるが、この場合でも I2C を通じて読み出される電圧値は内部状態を表示するので全て正常である。並列保護回路が入っているので、二つのバッテリーの内一方がヒューズ破断している

場合、電流が流れないので使用中にもかかわらず何時までも電圧は高いままである、出力電圧は二つのバッテリーの内高い方の電圧が表示されるので、ドローン供給電圧表示は高い状態で固定となり正しい電圧が表示されない。そこでバッテリー外部出力電圧が供給されているかどうかを確認する回路を追加した、出ていなければヒューズ破断と見なし、I2C データを使用しない。

## 3　Ardupilot ドライバー機能

実際にマクセル並列処理ドライバーを導入した ArduPilot　Ver 3.5 でのミッションプランナーでの表示例を各種紹介する。

### 3.1　正常並列接続

50%と 30%程度残量があるバッテリーを並列に挿入しました。直結した場合はバッテリーが壊れますが、保護回路あるので問題ありません、Bat1


ミッションプランナーでの表示は下記内容になります。統合残容量は各バッテリーの残容量を合算し 2 台分の全容量で割ったものになります。

1）Bat1 Voltage(V) 各バッテリーの電圧を読み取って処理した電圧値

2）Bat2 Voltage(V) Pixhawk がアナログ端子から読み取った電圧値

3）Bat Remaining(%) ドライバーが計算した統合バッテリー残量、100%では 10400mmAh となる。

4）Bat Current(A) ドライバーが各バッテリーの電流値を加算した値


### 3.2　正常単体接続

正常なバッテリーを片方に挿入したとき、バッテリー残量は２台分を１００％としているので、半分の値となる。



### 3.3　出力ショートでヒューズ破断

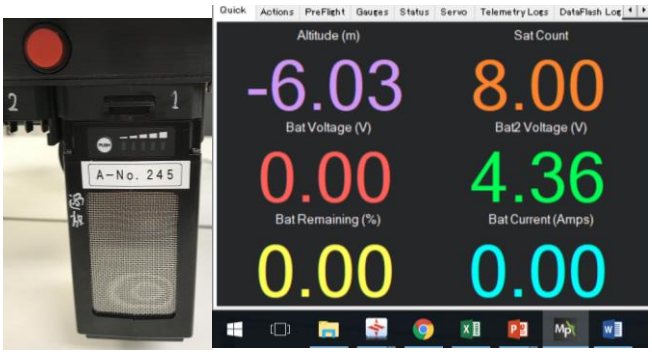
出力がショートしてヒューズが破断（１００A 以上の電流が流れた）した状態のバッテリーを接続、出力電圧は０V だが I2C 情報では電圧・残容量は正常に表示される。出力検出回路によりドライバーが電圧、残容量を０と表示している。


### 3.4　充電過電流でヒューズは断

異なる残容量のバッテリーを直結などで過充電したときは内部回路でヒューズが溶断される。この状態では出力は 0 V であり、I2C の過充電エラーフラグがオンとなっている。I2C で電圧は読めるがドライバーは電圧を 0 V にしている。

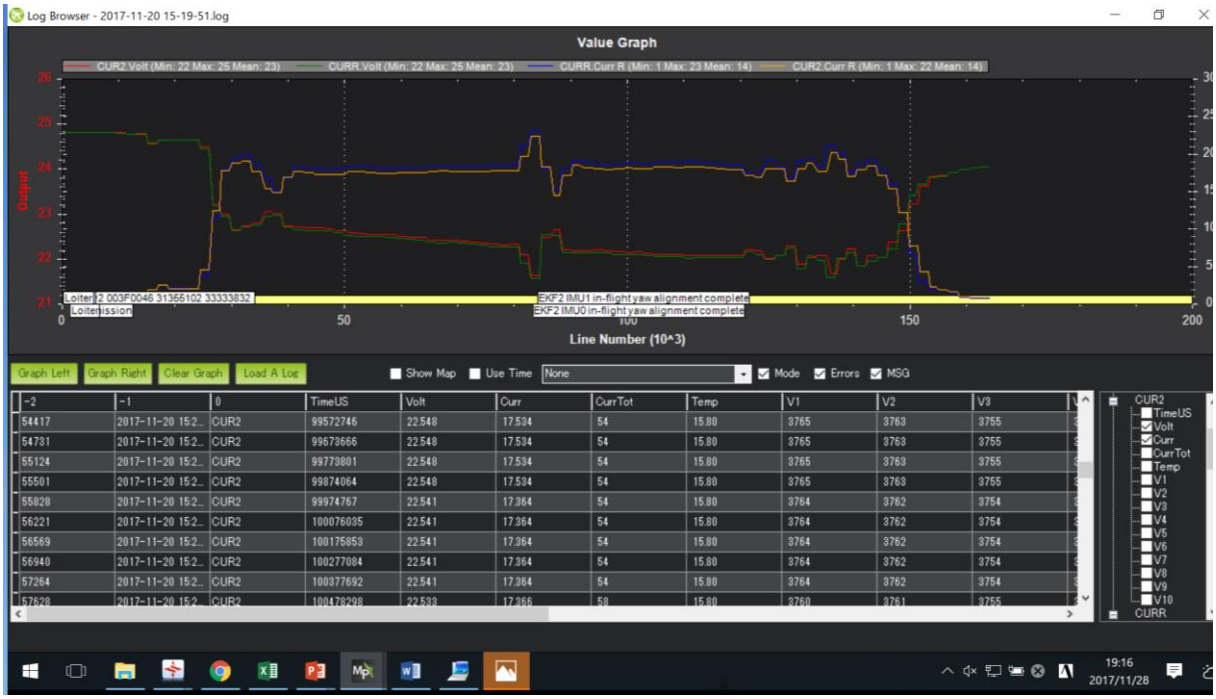

### 3.5　過放電状態

バッテリーを放電し続けると過放電状態となり充電はできなくなる。この場合でも出力電圧は若干あるので、ユニットの電源をオンすることはできる。I2C では過放電エラーとなっているので、ドライバーは残量、出力電圧ともに 0 V としている。下記の例では PixHawk では 4.36V の出力を検出している。



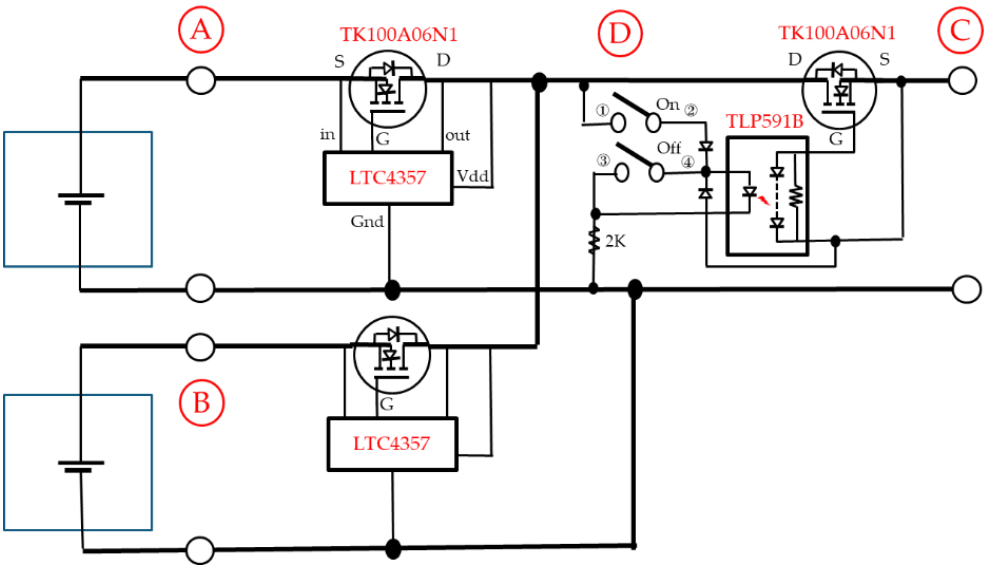
## 3.6　ログ情報

Ver3.5 では Bat1 の情報しかログできなかったが、並列処理ドライバーでは Bat2 の情報も正しく記録されています。



# 4　回路図

## 4.1　並列接続回路




## 4.2　I2C　スイッチ