# リチウムイオンバッテリーの経時特性測定用プログラムの開発

Development of a Program for Measuring Lithium Ion Batteries Time Elapsed Characteristics

袴田 吉朗* 恩田 一*

**Yoshiro HAKAMATA** **Hajime ONDA**

Abstract: The paper describes the design of a program for measuring Lithium Ion batteries management system time elapsed characteristics. The system consists of series connected fourteen batteries that are assumed to be charged from or discharged to a solar system constructed in the SIST campus. The program is made with Microsoft VC++6.0, and a Windows XP machine. It measures Lithium Ion batteries voltages and temperature of the batteries every two minutes. It displays data on a PC screen, and it saves data on a hard disk. These operations are also performed every two minutes. Also alarms are displayed on a screen, and based on these, relays connected to batteries are controlled to discharge batteries through relays and 4.7Ω resistors. The program confirmed to work well using the prototype system constructed on a desk. The system field test is hoped to be done in near future using the program.

## 1. はじめに

一昨年の3月11日における東日本大震災以降，原子力に代わる代替エネルギーに関する議論や実験，実用的なメガソーラーシステムの構築などが盛んに行われてきている．本学においても開学20周年記念事業の一環として太陽光発電を主体とした蓄電型太陽光発電式電気自動車（ＰＶ／ＥＶ）充電システムの構築が行われ，一部稼働を始めたところである［1］．

本検討は，このシステムの一翼を担うものでありその主眼は，ハイブリッドカーなどに使用して不要になったリチウムイオンバッテリーを，太陽光発電システムの蓄電器として再利用することにある．

本検討において筆者らが考えているシステムは，ハイブリッドカーに使用した中古のリチウムイオンバッテリーを14個直列接続し，これらを太陽光発電システムを用いて充放電する蓄電器として再利用するシステムである．このシステムを実現するためには直列に接続した各セルの電圧値を連続的に測定し，異常な（高）電圧を示すセルの有無を監視する必要がある．もしそのようなセルを検出した場合には，そのセルを強制的に放電させ，全セル電圧の平準化を図るいわゆる「バランシング」を行う必要がある．

そのために以下のようなシステムを構築し，データを取得してその状況を把握し，このようなシステムが実用に耐えられるか否かを検証することにした．

① 各セルの端子電圧，セルの電流およびセルの温度を連続的に測定し，データを収集する
② 異常に高い電圧を示すセルがある場合には，リレーを制御して抵抗を介して放電させる
③ システムを制御するプログラムを開発する

本資料では，開発した制御プログラムの設計および動作，システムの使用方法などを述べる．なお袴田がプログラムの設計，開発およびプログラムの修正を行い，恩田がシステム構成の検討および机上システムを用いた実機によるデバッグを担当した．

2013年2月19日受理

*理工学部 電気電子工学科

## 2. バッテリーマネジメントシステムのハードウェア構成

### (1) 設計目標

以下を設計目標とした．

- 太陽光発電システムから充放電される 14 個のリチウムイオンバッテリーの電圧を連続的に監視し測定する．
- セルの温度（7 カ所）も同様に連続的に監視測定する．
- 測定結果を画面に表示し，ハードディスクにセーブする．
- 測定結果を評価し，異常に高い電圧を表示するセルのアラームを発出する．
- まだデータがない状況であり，本システムでは放電の必要なセルのアラームを発出させるのみとする．実際にリレーおよび抵抗を介してセルを強制的に放電させるバランシングを行うか否かは保守者が判断するものとする．
- 2 台以上の複数のセルが同時に異常に高い電圧になった場合にも正しくアラームを発出できるようにするが，同時に放電を行うセルの数は唯一つであるとする．

### (2) システムの概要

図 2.1 に試作したシステムの構成図を示す．

- 被測定対象は直列接続された 14 個のリチウムイオンバッテリーである．これらのバッテリーは双方向性 DC/DC コンバータを介して HVDC に接続されており，太陽光発電システムから充放電される．
- この回路に放電用のリレー14 個，メインリレーおよび電流検出用の 0.1Ω抵抗を直列接続して制御に用いる．

制御部分は，データロガーGL820 (AD／DA 変換器)，8 ポートミニリレー（2 台），放電用の 4.7Ω抵抗（14 個），熱電対（7 個），3 ポートハブ，端子台，回転灯および PC などからなる．

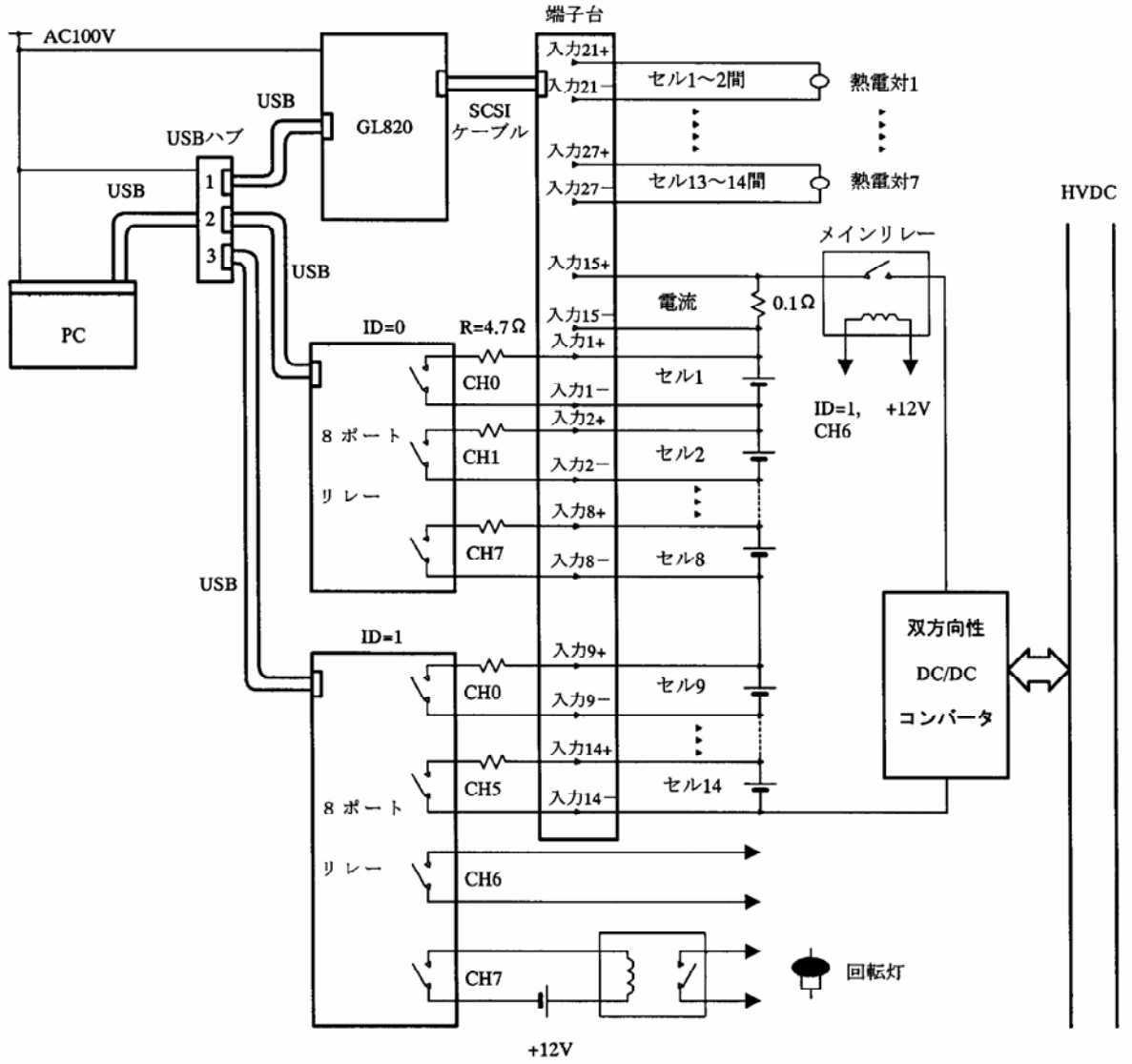


図2.1　バッテリーマネジメントシステムのハードウェア構成

### 3．開発環境の構築

リレーおよびデータロガーGL820が入荷された時点で開発環境についての調査を行った．その結果

PC：　WindowsXP

開発ソフトウェア：　VC++6.0

を用いることにした．調査した結果を表3.1に示す．

筆者らはUSB接続に関しては今までにプログラム経験がなく，そのためUSBに関する部分はGraphtec社のサンプルプログラムを利用し，それ以外の部分を新規に設計・試作することにした．

データロガーGL820に付属しているVC++2005サンプルプログラムでは，MFCを使用する必要がある．2012年5〜6月に予備的な検討を行いこの時にはフリーのVC++2010 Express Editionを使用した．しかし，VC++2010 Express EditionはMFCには対応していない．正規版のVC++2010を購入すればMFCを使用できるので予備的な検討結果をそのまま活かせたが，正規版は10万円を超える高額な商品であり，予算の都合上使用を断念した．代わりに従来から使用してきた手持のちVC++6.0を用いて開発を進めることにした．必然的にPCはWindowsXPマシンを使用することになった．

表3.1　PCおよび開発環境とUSBドライバに関する調査結果

| OS | Graphtec社 USBドライバ | Turtle社 USBドライバ | VC++6.0 | VC++2010 Express Edition |
|---|---|---|---|---|
| WindowsXP | ○ | ○ | ○ | △（MFC ×） |
| Windows7 | ○ | ○ | × | △（MFC ×） |

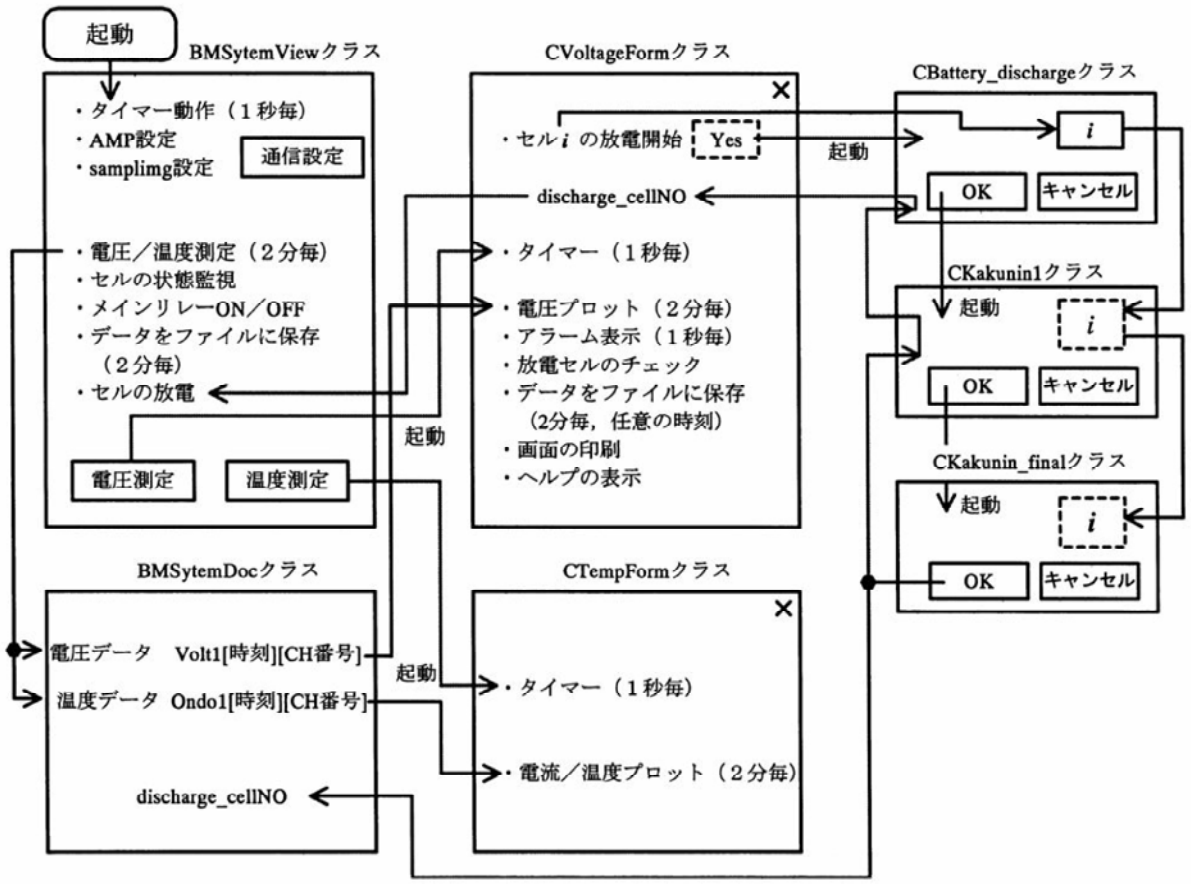


図4.1 バッテリーマネジメントシステムプログラムにおける処理の主な流れ

## 4． バッテリマネジメントシステムプログラムの動作

図 4.1 にバッテリマネジメントシステムプログラムにおける処理の主な流れを示す.

### 4.1 データロガーGL820 の初期設定

スタートボタンをクリックするとプログラム（BMSystemView クラス）が起動する．このクラス中にある StartButtonThread において，データロガーに関する以下の 2 項目の初期設定を行っている.

① 電圧測定レンジ（フルスケール）10V，フィルタ OFF

② 温度測定 K 熱電対（TCK），フィルタ OFF

また OnInitialUpdate メンバ関数においてタイマー割り込み（OnTimer メンバ関数）の周期を 1s に設定し，測定周期（サンプリング間隔）を設定している.

ドキュメントクラスに設定した変数 sampling_clock を OnTimer メンバ関数においてカウントし，データを表示する 2 分毎の周期を作成して CVoltageForm クラスと同期させている.

### 4.2 リレードライバの初期設定

Turtle 社のリレーをプログラムにより動作させるためには，

TUSBKRL.h ヘッダファイル

TUSBKRL.lib ライブラリファイル

をプロジェクトに追加する必要がある.

その上で「USB インタフェース付き接点出力ユニット取り扱い説明書 [2]」の p.12 に記されているように，まず

TUSBKRL_Load（）；

を実行してドライバを呼び出す必要がある．このときリレーの準備ができるまでに時間がかかるので，待ち時間を入れなければならない．待ち時間を入れないと次の動作を実行できない.

この処理は BMSytem クラスの OninitialUpdate メンバ関数において実行している．ドライバをロードした後，500ms の待ち時間を複数個入れてリレーのステータスが OK となるまで待つプログラムとしているが，その結果から判断して単に sleep(1000)；を入れるだけでも良さそうである.

### 4.3 電圧および電流／温度の測定

図 4.2 に測定のフローチャートを示す.

BMSystemView クラスにおいて 2 分毎にコールされる voltage_temp_measure メンバ関数において，データロガーから一連のデータを取得する以下のコマンドを実行し，電圧および電流／温度のデータを取得している.

```
if(pDev->SendCommand(":MEAS:OUTP:ONE?"))
    {ErrGraphtec(3);}
```

このコマンドの実行により図 4.3 に示すバイナリデータが複数個からなるデータが取得される[3]．これを以下の命令を実行してヘッダおよびデータを char 配列に読み込んでいる.

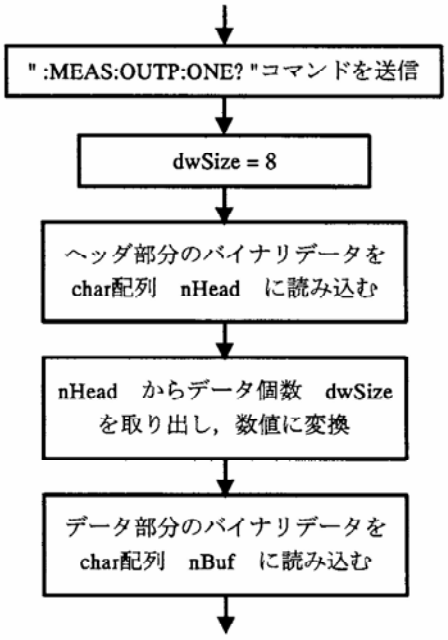


図4.2 データ測定のフローチャート

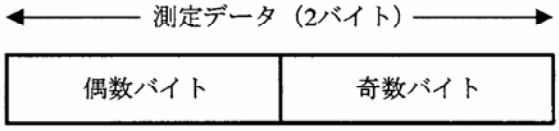


図4.3 GL820の測定データの構成

```
if(pDev->ReadBinary(nHead,dwSize,R_TIMOUT))
        {ErrGraphtec(3);}
if(pDev->ReadBinary(nBuf,dwSize,R_TIMOUT))
        {ErrGraphtec(3);}
```

ヘッダ長は8バイトであり，#6xxxxxx の形式をしている．なおバッファ nBuf は char 型の配列として定義しなければエラーになる．電圧および温度のデータは，2 バイト整数としてデータロガーからビッグエンディアン形式で送信されてくる．これを以下のように BYTE 型（unsigned char 型）にキャストして偶数バイトが上位に，奇数バイトが下位となるように組み立てている．BYTE 型にキャストしないと下位バイトのビット 7 が 1 になる数値の場合に，下位バイトが負数と判断されて計算されるため正しく変換できない．

```
ADCVal_WORD=(BYTE)nBuf[i*2]<<8+(BYTE)nBuf[i*2+1]; (1)
```

i はセルの番号である．

なお，標準電圧発生器を使用して GL820 本体で測定した電圧値と PC に取り込んだ電圧値の関係を調べたところ，10mV の精度を達成できていることを確認した．

### 4.4 負数の処理

電流は負になることもあるので，式(1)の ADCVal_WORD の最上位ビットが 1 になった場合には負数であると判断し以下の式を用いて実際の値に変換している．

```
if( (unsigned)ADCVal_WORD >32767)
        ADCValue = ADCVal_WORD − 65536;
else    ADCValue = ADCVal_WORD;                    (2)
```

### 4.5 2バイト整数の実際の電圧，電流および温度への変換

電圧および電流はフルスケール10V の設定である．このため，仕様により以下の処理を行って実際の値に変換している．

```
ADCValue >>1;    ADCValue = ADCValue / 1000;        (3)
```

また温度については，仕様に従って 2 バイト整数値を 10 で除して実際の温度を得ている．

## 5．リレーの制御

4 章で述べたように，電圧および電流／温度の測定は 2 分毎に実施される．その測定直後に実行する cell_status_check2 メンバ関数においてリレーの状態を表すフラグ rclayFlag[ ]およびアラームの表示に使用するフラグ statasFlag[ ]の設定を図 5.1 に示すように行っている．

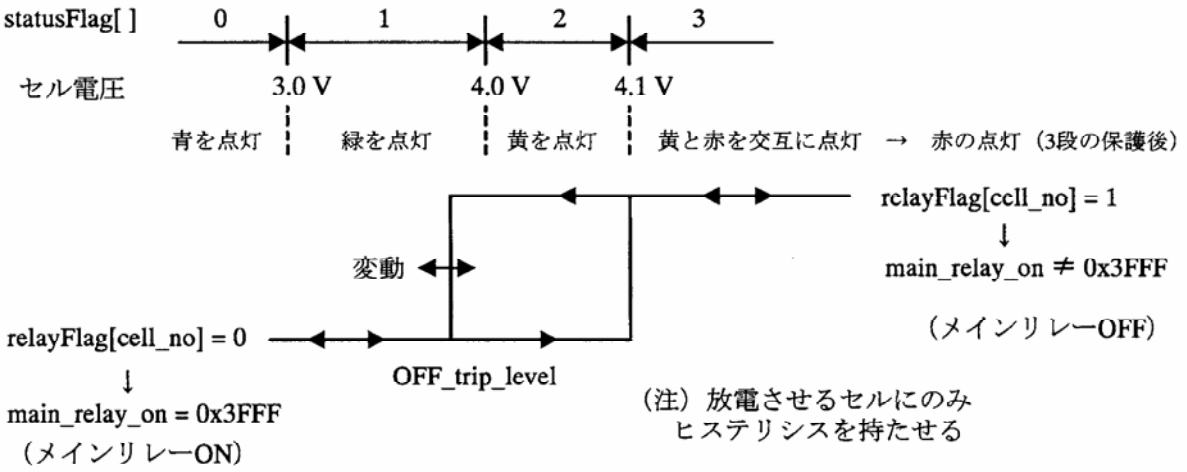


図5.1 リレーおよびアラームの制御論理

### 5.1 リレーフラグ relayFlag[ ] の制御

一旦放電を始めたセルに対しては図 5.1 に示すようにヒステリシス特性を持たせて放電を終了させる．すなわち

セル電圧 ≧ 4.1V・・・・・・・relayFlag[ ]=1

< OFF_trip_level・・・relayFlag[ ]=0

とする．ここに OFF_trip_level は，放電を行っているセルを除く残り 13 個のセル電圧の平均値であり，その都度値が変動する．セルの放電が終了した場合には，全セルがほぼ同一の電圧になることが期待される．

放電を行っていないセルはヒステリスを持たせることなく，電圧が 4.1V 未満になると即時に relayFlag[ ]=0 とする．

### 5.2 ステータスフラグ statusFlag[ ]の制御

セルの状態を画面上に示すために statusFlag[ ]を用いており，各セルの電圧によって以下のように設定している．

電圧 < 3.0V・・・・低電圧アラーム statusFlag[ ]=0
青色を点灯

≧ 3.0V・・・・正常状態 statusFlag[ ]=1
緑色を点灯

≧ 4.0V・・・・アラーム発出 statusFlag[ ]=2
黄色を点灯

≧ 4.1V・・・・放電開始電圧 statusFlag[ ]=3
3 段の保護動作中は黄色と赤色を 1s 毎に点滅，その後赤色の点灯に移行

### 5.3 リレーの制御論理

リレーの制御論理を図 5.2～図 5.4 に示す．

#### 5.3.1 relayFlag[ ]の 1，0 を検査

全てのセル電圧が 4.1V 未満の場合には図 5.2 における最初の処理において変数 discharge_cellNo=0 となり，メインリレーの状態を示すラジオボタンを ON に保ちつつ，Ⓐを通る処理が行われる．Ⓐ以降の処理は図 5.4 に示す BMSystemView クラスの OnTimer メンバ関数において 1s 毎に実行され，この部分において relay_off メンバ関数が実行され全リレーが OFF になる．

その後ドキュメントクラスの変数を cell_relay_ON=0 として，リレーの OFF 動作を変化の生じた場合だけに留め，いたずらにリレーの信頼性を損なわないようにしている．

#### 5.3.2 relayFlag[ ]の差分を検出

一般的に複数個のセル電圧が 4.1V を超える状況が考えられる．そのため relayFlag[ ]の 1，0 の検査に続いて新旧 relay_Flag[ ] の差分を取り，一つ前のサンプリング点におけるセル電圧と比べて変化があったか否かを検出し変数 res[ ]に保存する．単に relay_Flag[ ]=1 を検出するだけでは複数個のセルが同時に 4.1V 以上になったか否かを検出できない．

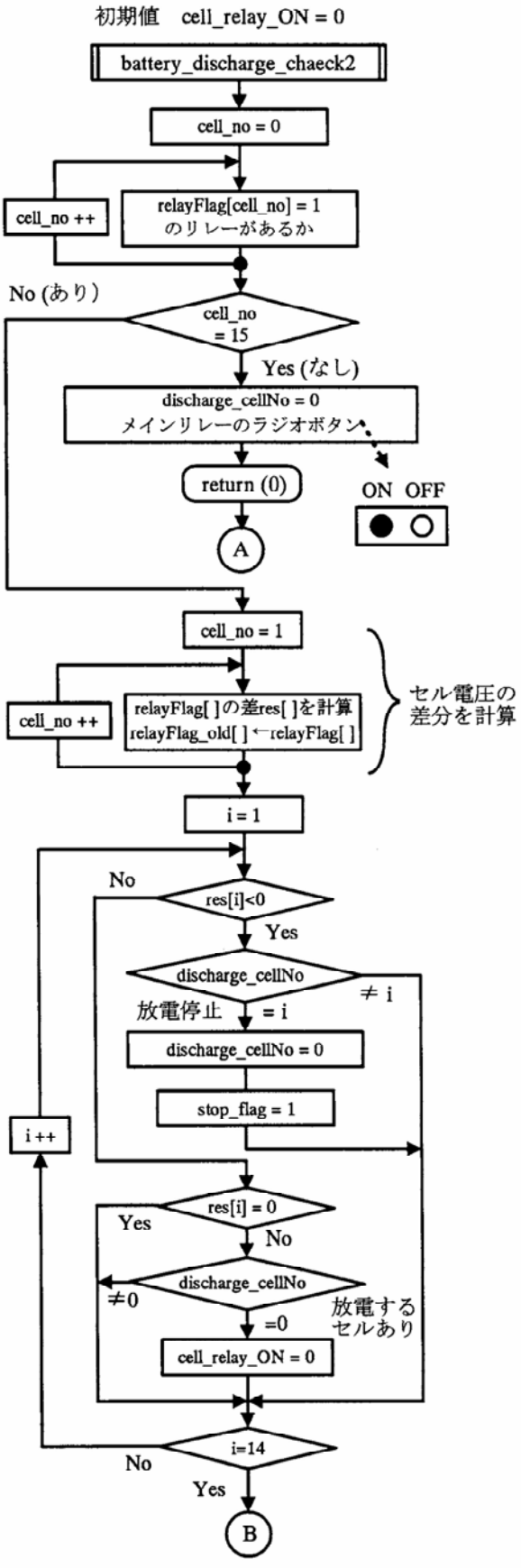


図5.2 リレーの制御論理（その 1）

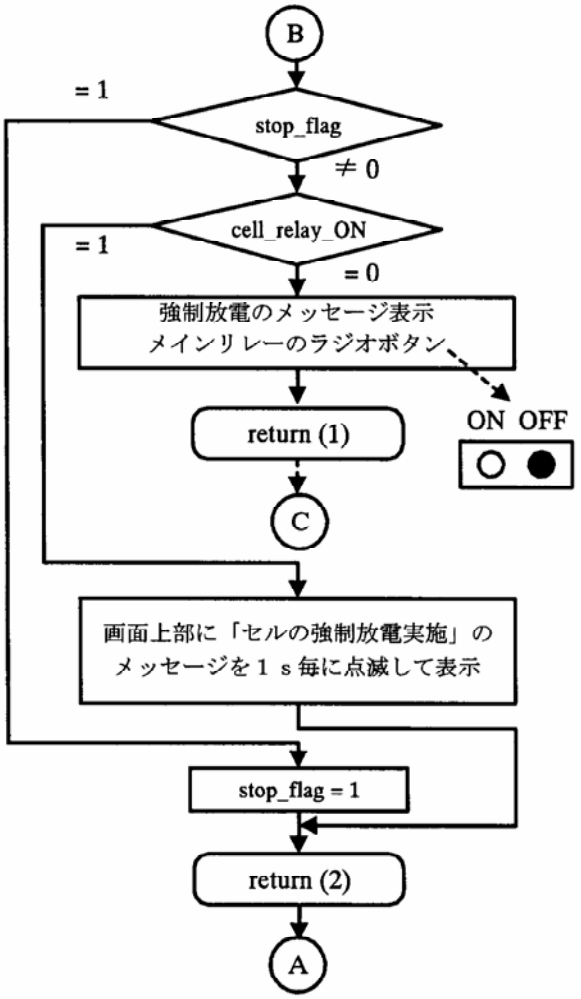


図5.3　リレーの制御論理（その2）

**5.3.3　リレーの制御**

次ぎに検出した差分 res[ ]に基づき以下の３つに分類する.

① res[ ]=0・・・状況に変化がない

② res[ ]>0・・・電圧が 4.1V 以上のセルがある.

③ res[ ]<0・・・電圧がしきい値未満になったセル $i$ がある

なお, ドキュメントクラスの変数 discharge_cellNo は以下の意味を表している.

・discharge_cellNo≠0・・・当該番号の唯一つのセルが放電中

・discharge_cellNo=0・・・全セルが放電停止中

以上を考慮して以下のようにリレーのＯＮ／ＯＦＦの制御を行っている.

①の場合　処理を継続する

②の場合

discharge_cellNo=0　→　リレーＯＮ

discharge_cellNo≠0　→　すでに放電中のセルがあるので, 処理を継続（後続セルの放電を見合わせる）

③の場合

discharge_cellNo＝$i$（$i$はセル番号）　→　リレーＯＦＦ

discharge_cellNo≠$i$　→　放電中のセルではない, 処理継続

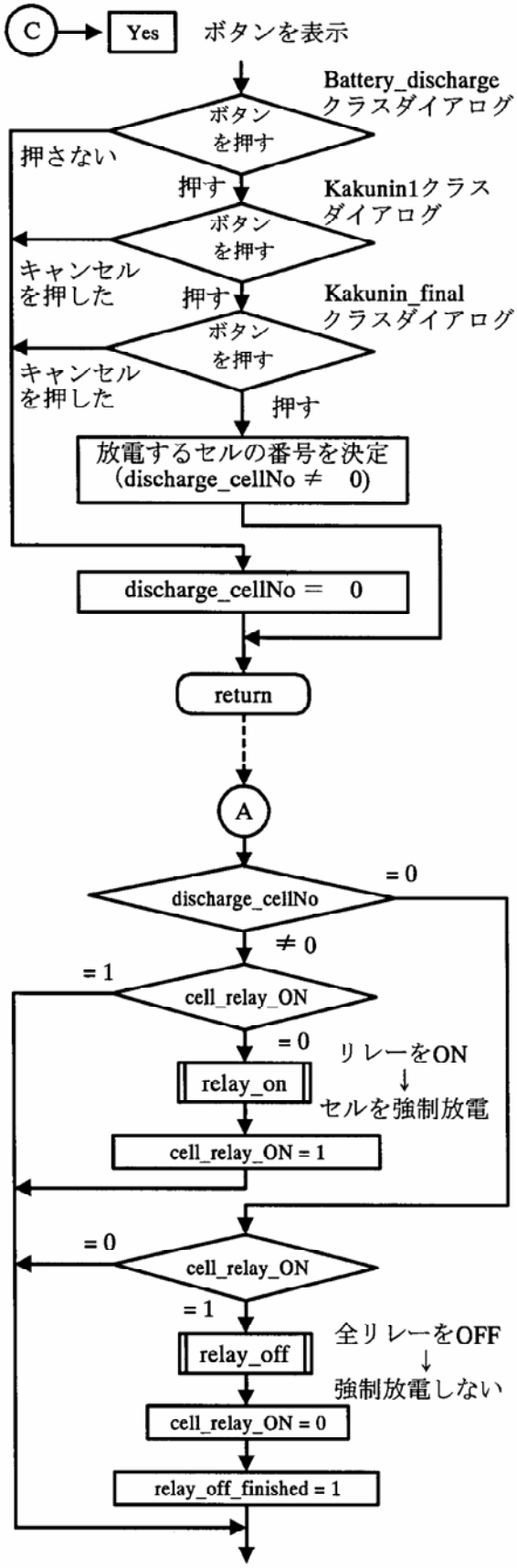


図5.4　リレーの制御論理（その3）

### 5.4 メッセージの処理

セル電圧の値に応じて画面上に以下を表示する.

① 「セル $i$ の強制放電を行いますか?」($i$ はセル番号)
② 「セル $i$ の強制放電を行っています」
③ メインリレーOFF時の時刻
④ メインリレーON/OFFを示すラジオボタン

(1) res[ ]>0 となった場合の処理

フローに従ってⒷを通る. この結果メインリレーのOFFをラジオボタンに表示する. また「セル $i$ の放電を行いますか?」のメッセージを表示する. このメッセージには同時に年月日および時刻も表示する. しかしセルの放電を自動的に行う方法ではなく, 保守者が判断して行う方法を取っている. このため一般的には放電開始のメッセージが表示された時点で直ぐ放電開始することはなく, メッセージは時間が経過してから見られる可能性が高い. この時にメッセージが発出された時刻が画面を見て分かるように表示を保持するようにした. なお図 7.1 に示すログにおいて main_relay_OFF_time はこの時刻を示す.

メッセージの表示と共に Yes のボタンを表示する. このボタンをクリックすると放電すべきセルの番号 $i$ をダイアログに表示する. 念のため 3 段の保護をかけて放電するセルの番号 discharge_cellNo を確定する. その結果が BMSystemView クラスの OnTimer メンバ関数において 1s 毎に実行される図5.4のⒶを通る処理に渡され, この部分において relay_on メンバ関数が実行されてセルの強制放電が行われることになる.

この過程で変数 cell_relayON=1 となるので, Ⓑを通過して「セル $i$ の強制放電実施中」のメッセージを 1s 毎に点滅して表示する. またリレーの ON は最初の 1 回だけに限定される.

(2) rse[ ]<0 になった場合の処理

図 5.2 において return(2)を抜ける処理である. 当該セルが放電中のセルであるか否かを検査して, 放電中のセルである場合にはセルの放電を停止するために discharge_cellNo=0 とする. この結果図 5.4 においてⒶ以降の処理が行われ, 全リレーが OFF となり強制放電が終了する.

一方放電中のセルでない場合には, そのまま処理を継続する.

(3) リングバッファの適用

メッセージキューを格納するためにバッファ数 15 (セル数+1) のリングバッファを適用した. 書き込み制御は%演算子を適用した通常のリングバッファを用いている. しかしキューから抜けるのは単に放電停止によるだけでない. 例えばセル電圧が 4.1V 以上となったが放電はしておらず, 時間経過後何らかの原因により 4.1V 未満になるセルも想定されるので, この状況を考慮して変則的な制御を行っている.

### 5.5 メインリレーの制御

cell_status_check2 メンバ関数の最後の部分において relay_Flag[ ]の否定を, セル 1 をビット 1 に, セル 14 をビット 14 に組み立てて変数 main_relay_on を作成している. この結果全セルが正常 (電圧が 4.1V 未満) の場合には

main_relay_on = 0x3FFF

になる. この変数の値を検査しこの値が 0x3FFF になったときにメインリレーを ON にし, 双方向 DC/DC コンバータからの充放電を再開する. なおこの条件は各セルリレーを OFF とする条件でもあるが, プログラムにおいて関数を実行する順序は

メインリレーの ON → 各セルリレーの OFF

になっている. この順序ではまだセルの放電実施中にメインリレーを ON することになり不都合である. そのため図 5.4 における relay_off メンバ関数の直後に変数 relray_off_finishcd=1 を挿入し, メインリレーの ON は relray_off_finished=1 を検出してから実行するようにした. これより

各セルリレーの OFF → メインリレーの ON

の順序を保証している. なおこの結果メインリレーの ON は, セルリレーの OFF 後 2 分間遅延してから行われる.

一方, main_relay_on≠0x3FFF の場合になるとメインリレーを OFF にセットし, また回転灯用のリレーを ON にして回転灯を点灯させる.

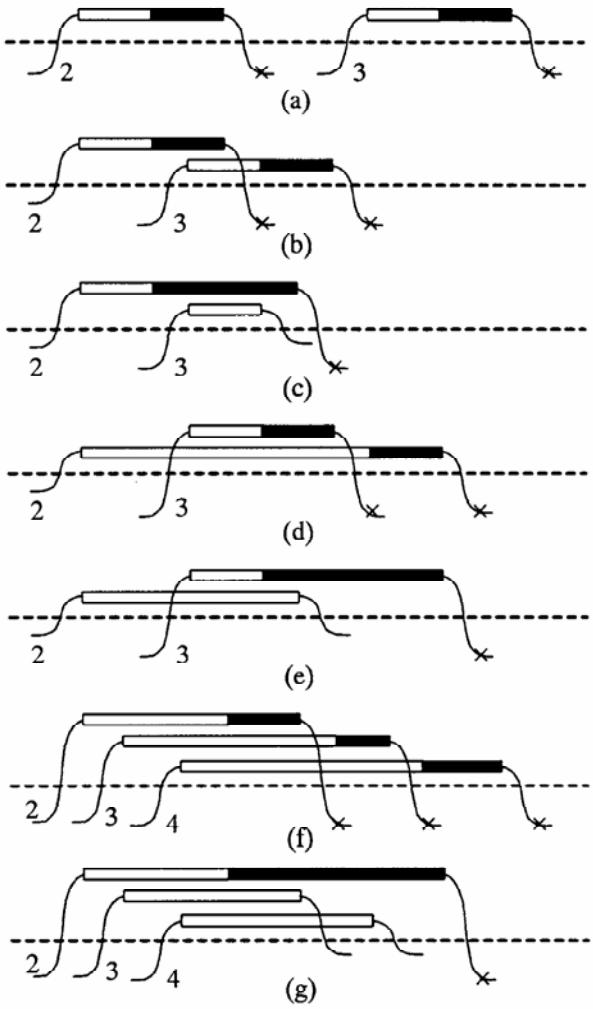


図 5.5 正常動作を確認したセル電圧経時特性のシナリオ

### 5.6 セル電圧の経時特性における想定されるシナリオ

筆者らはリチウムイオンバッテリーの経時特性のデータを現時点で何も取得しておらず，その挙動についての知見がない．そこで種々の状況にプログラムが対応できるか否かをシミュレーションおよび机上システムを用いて検証した．その結果検討した全シナリオにおいてセルリレーの動作，メインリレーの動作およびメッセージの表示が期待通りであることを確認できた．

検討したシナリオを図 5.5 に示す．横軸は時間，縦軸は電圧である．図5.5 において以下の記号を使用している．

- ▭ セル電圧が 4.1V 以上，放電を放置
- ▬ 放電中
- × 放電を停止
- ------ 4.1V の電圧を表す

なお数字はセル番号を表す．

図 5.5 において(a)は最も基本的な動作である．(b)および(c)はセル 2 が放電中に，セル 3 が 4.1V を越えた場合である．(d)～(g)は最も可能性の高いと考えられるシナリオである．保守者のいない夜間に複数のセルが 4.1V 以上となり，朝方に保守者がそれを認識したような場合を想定している．実際に放電させるセルは，電圧の高さ，電圧の上昇具合などを尺度として判断することになると考えており，画面に表示されている情報を見てセルの番号を選択できるようにしている．

なお，図 5.5 ではセル電圧が 4.1V 以上になる順番が若番→老番になっているが，プログラムの動作はこの順番に関係なく正しく動作することを確認している．

## 6． 測定データおよびアラームの画面への表示

### (1) 電圧の表示

電圧の表示枠を図 6.1(a)に示す．測定電圧を赤色の実線で，平均電圧を黒色の実線でプロットしている．2 分毎に測定を行っているので時間軸は 720 データ／日 である．したがって 2 分／ピクセルである．一方電圧の表示範囲は 2.5V～4.5V であり，これを 40 ピクセルで表示している．したがって電圧の分解能は 0.5V／ピクセルである．

補助線として 3.7V の横線と，6 時，12 時，18 時の縦線をプロットしている．

### (2) 電流の表示

電流の表示枠を図 6.1(b)に示す．測定電流を黒色の実線でプロットしている．時間軸は電圧と同様に 2 分／ピクセルである．電流の表示範囲は 50A～ −50A である．これを 100 ピクセルで表示しているので電流の分解能は 1A／ピクセルである．

補助線として 6 時，12 時，18 時の縦線をプロットしている．

### (3) 温度の表示

温度の表示枠を図 6.1(c)に示す．測定温度を黒色の実線でプロットしている．時間軸は電圧と同様に 2 分／ピクセルである．温度の表示範囲は 0℃～ 50℃である．これを 50 ピクセルで表示しているので温度の分解能は 1℃／ピクセルである．

補助線として 6 時，12 時，18 時の縦線をプロットしている．

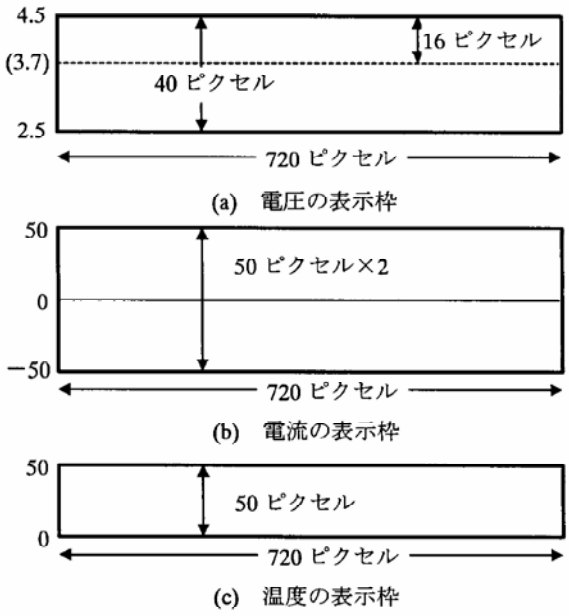


図6.1 電圧，電流および温度の表示枠

### (4) 電圧のエディットボックスへの表示

各セル電圧の値および平均電圧（放電セルのない場合には 14 個の平均値，放電セルのある場合には放電セルを除く 13 個の平均値）を表示している．測定電圧を float 型で処理しており，この値を CString 型に変換するために float_2_EditBox メンバ関数 float_2_EditBox(float source, CEdit *m_ed, CFont *font)を使用している．このメンバ関数の処理を以下に示す．

① 入力の float 型変数を実数に変換

```
buf = _fcvt(source, 3, &decimal, &sign)
```

decimal および sign は int 型の変数，buf は char*変数である．

② CString 型に組み立て ss2 を得る

```
ss1 = buf;
ss2 = ss1.Left(decimal)+"."+ss1.Mid(decimal,2);
```

③ sign= −1 のときは負号（マイナス符号）を付ける

```
ss2 = "−"+ss2;
```

④ SetFont(font,TRUE)関数により表示フォントを指定し，SetWindowText(ss2)関数により表示

⑤ フォントを設定する

```
Font.CreatePointFont(120,"MS ゴシック");
```

## 7． ツールバーにおける処理

CVoltageForm クラスのダイアログ上に 3 個のツールバーを設定し，データのファイルへのセーブ，画面の pdf ファイルへの印刷およびヘルプの表示ができるようにしている．

### 7.1 データのファイルへのセーブ

BMSytemView クラスにおいて自動的に 2 分毎の電圧，電流／温度データをファイルにセーブしている．また CVoltageForm クラスにフロッピーディスクを模したツールバーのアイコンを配置しており，これをダブルクリックするとクリックした任意時刻までのデータをファイルにセーブすることができる．両者の仕様は全く同様である．

例えば 2013 年 3 月 2 日のデータであれば，作成されるファイルのホルダーは以下のようになる．

LOG¥2013_03

またファイル名は以下のようになる．

BMS2013_03_02.csv

2 分毎に 0 時からその時刻までのデータが上書きされていき，23 時 58 分に測定したデータが書き込まれると次の日時に移る．

LOG ホルダーはあらかじめ実行ファイルの存在するディレクトリに作成しておく必要があるが，配下の 年_月 毎のホルダーは自動的に作成される．これは

_mkdir(path);

の命令により行っている．この命令は，既にホルダーが存在する場合にはエラーとなるが，これを無視することで対処している．なお direct.h をインクルードする必要がある．

C の関数 fopen(filename,"w") の実行によりファイルが自動作成される．C の関数を使用したのは単に C++の関数をうまく使用できなかったためである．

図 7.1 に CSV ファイルに出力したデータの一例を示す．

### 7.2 画面の pdf ファイルへの印刷

CVoltageForm クラスのダイアログのツールバーアイコンにおいてプリンタのアイコンをダブルクリックすると，現在表示している画面を pdf ファイルにセーブできる．この実装は Win32API を使用した参考書である［4］に記述されているプログラムを使用させて頂いた．手順の概略を以下に示す．

① 画面にコンパチブルなメモリデバイスコンテキストを作成する．

② 表示されている画面を，BitBlt 関数を使用してメモリデバイスコンテキストにコピーする．

③ CreateDC 関数を用いて，出力先を Acrobat Distiller とするデバイスコンテキストを作成する．

CreateDC("WINSPOOL","Acrobat Distiller",NULL,NULL);

④ GetDIBits 関数を用いてメモリデバイスコンテキストからビットマップ情報を取り出す．

⑤ StrechDIBits 関数を用いて画像を拡大してプリンタに送信する．

図 7.2 は電圧の経時特性を表示した画面を，プリントスクリーン機能を用いてビットマップに変換したものである．丁度セル 3 が 4.1V を越え，セルの強制放電のメッセージが出ている状況である．pdf ファイルの場合には，ビットマップよりはかなり解像度が低下する．

### 7.3 ヘルプの組み込み

新規作成時にヘルプを組み込むためには，状況依存のヘルプにチェックを入れておけば良い．しかし今回使用したサンプルプログラムにはヘルプが組み込まれていなかった．そこで以下のようにしてヘルプを組み込んだ．

① 参考書［5］に基づき winhlp32 をプロジェクト名として必要な状況依存のヘルプを作成した．

② CVoltageForm クラスにヘルプを示すツールバー（？）を組み込む．

③ ツールバーから以下のように ShellExecute 関数を実行して①で作成したヘルプを組み込んだ．

ShellExecute(NULL,NULL,"WINHLP32.HLP",NULL,
"¥¥EV_20121128¥¥BMSytem¥¥hlp",SW_SHOWNORMAL);

ポイントは，第 5 引数である．この第 5 引数によりヘルプの存在するディレクトリをフルパスで設定すると，ファイル選択用のダイアログを表示させることなく動作させることができる．

なお第 3 引数に単に WINHLP32 と記述すると，ファイル選択用のダイアログが開くので処理が幾分面倒になる．

図 7.1 データの CSV ファイルへの出力例

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 保存年月日 時刻 2013/01/30 18:32:46 | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | |
| 3 | リチウムイオンバッテリーにおける各セル電圧(単位V)X | | | | | | | | | | | | | | | | | | |
| 4 | main_relay= 1 =0･･･OFF | | | | | | | | | | | | | | | | | | |
| 5 | セル番号 ･･･放電を行っているセルの番号を意味 | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | |
| 7 | 回数 | 時刻 | 平均 | セル1 | セル2 | セル3 | セル4 | セル9 | セル10 | セル11 | セル12 | セル13 | セル14 | 電流 | main-re | セル番 | main_relay | メッセ | cell_relay_ON |
| 483 | 475 | 15:50 | 3.96 | 3.96 | 3.96 | 3.96 | 4.02 | 4.01 | 3.96 | 3.96 | 3.96 | 3.96 | 3.96 | 0 | 3FFF | 0 | | 0 | 0 |
| 484 | 476 | 15:52 | 3.96 | 3.96 | 3.96 | 3.95 | 4.02 | 4 | 3.96 | 3.96 | 3.96 | 3.96 | 3.95 | 0 | 3FFF | 0 | 15:52:22 | 4 | 0 |
| 485 | 477 | 15:54 | 3.97 | 3.96 | 3.96 | 3.96 | 4.02 | 4.12 | 3.96 | 3.96 | 3.96 | 3.96 | 3.96 | 0 | 3FFF | 0 | | 0 | 0 |
| 486 | 478 | 15:56 | 3.97 | 3.96 | 3.96 | 3.96 | 4.02 | 4.12 | 3.95 | 3.96 | 3.96 | 3.96 | 3.96 | 0 | 3FFF | 0 | | 0 | 0 |
| 487 | 479 | 15:58 | 3.97 | 3.95 | 3.96 | 3.96 | 4.02 | 4.12 | 3.95 | 3.96 | 3.95 | 3.95 | 3.96 | 0 | 3EFF | 0 | | 0 | 0 |
| 488 | 480 | 16:00 | 3.97 | 3.95 | 3.95 | 3.96 | 4.02 | 4.12 | 3.96 | 3.96 | 3.95 | 3.95 | 3.95 | 0 | 3EFF | 0 | | 0 | 0 |
| 489 | 481 | 16:02 | 3.97 | 3.96 | 3.95 | 3.95 | 4.02 | 4.12 | 3.96 | 3.96 | 3.96 | 3.95 | 3.95 | 0 | 3EFF | 0 | | 0 | 0 |
| 490 | 482 | 16:04 | 3.97 | 3.96 | 3.96 | 3.96 | 4.02 | 4.12 | 3.96 | 3.96 | 3.96 | 3.96 | 3.96 | 0 | 3EFF | 0 | | 0 | 1 |
| 491 | 483 | 16:06 | 3.98 | 3.96 | 3.96 | 3.96 | 4.12 | 4.12 | 3.95 | 3.96 | 3.96 | 3.96 | 3.96 | 0.01 | 3EFF | 9 | | 0 | 1 |
| 492 | 484 | 16:08 | 3.98 | 3.96 | 3.96 | 3.96 | 4.12 | 4.12 | 3.95 | 3.96 | 3.95 | 3.96 | 3.96 | 0 | 3EFF | 9 | | 0 | 1 |
| 493 | 485 | 16:10 | 3.98 | 3.96 | 3.96 | 3.96 | 4.12 | 4.12 | 3.96 | 3.96 | 3.95 | 3.95 | 3.96 | 0 | 3EF7 | 9 | | 0 | 1 |
| 494 | 486 | 16:12 | 3.98 | 3.96 | 3.96 | 3.96 | 4.12 | 4.12 | 3.96 | 3.96 | 3.96 | 3.96 | 3.95 | 0 | 3EF7 | 9 | | 0 | 1 |
| 495 | 487 | 16:14 | 3.89 | 3.86 | 3.86 | 3.86 | 4.12 | 4.05 | 3.86 | 3.86 | 3.86 | 3.86 | 3.86 | 0 | 3EF7 | 9 | | 0 | 1 |
| 496 | 488 | 16:16 | 3.89 | 3.86 | 3.86 | 3.86 | 4.12 | 4.05 | 3.86 | 3.86 | 3.86 | 3.86 | 3.86 | 0 | 3EF7 | 9 | | 0 | 1 |
| 497 | 489 | 16:18 | 3.89 | 3.86 | 3.86 | 3.86 | 4.12 | 4.05 | 3.86 | 3.86 | 3.86 | 3.86 | 3.86 | 0 | 3EF7 | 9 | | 0 | 1 |
| 498 | 490 | 16:20 | 3.89 | 3.86 | 3.86 | 3.86 | 4.12 | 4.05 | 3.86 | 3.86 | 3.86 | 3.86 | 3.86 | 0 | 3EF7 | 9 | | 0 | 1 |
| 499 | 491 | 16:22 | 3.89 | 3.86 | 3.86 | 3.86 | 4.12 | 4.05 | 3.86 | 3.86 | 3.86 | 3.86 | 3.86 | 0 | 3EF7 | 9 | | 0 | 1 |
| 500 | 492 | 16:24 | 3.88 | 3.86 | 3.86 | 3.86 | 4.12 | 3.96 | 3.86 | 3.86 | 3.86 | 3.86 | 3.86 | 0 | 3EF7 | 9 | | 0 | 1 |
| 501 | 493 | 16:26 | 3.88 | 3.86 | 3.86 | 3.86 | 4.12 | 3.96 | 3.86 | 3.86 | 3.86 | 3.86 | 3.86 | 0 | 3EF7 | 9 | | 0 | 1 |
| 502 | 494 | 16:28 | 3.88 | 3.86 | 3.86 | 3.86 | 4.12 | 3.96 | 3.86 | 3.86 | 3.86 | 3.86 | 3.86 | 0 | 3EF7 | 9 | 16:28:26 | 9 | 1 |

## 8. むすび

リチウムイオンバッテリーの電圧および温度の経時特性を連続的に測定し，表示し，またデータを保存するための制御プログラムを開発した．机上のプロトタイプシステムを用いてデバッグを行い，設計目標をすべて達成することができた．

本プログラムの開発を始めたのは，USB 接続に関する予備検討が 2012 年 5 月である．またリレーやデータロガーが 8 月中旬以降に納入されてきたので，9 月以降本格的に検討を開始した．

一方測定の対象となる学内の太陽光発電システムは，不幸なことに 2 度に渡って落雷の被害に遇うという憂き目を見，2 月中旬にやっと現状に復したと言う状況にある．

このような事情もあり，十分に時間をかけて入念なデバッグを行うことができた．机上システムを用いてデバッグを始めたのは 10 月下旬以降からであるが，デバッグを行うたびにバグや，検討不足の点が見つかりその都度修正を加えていくことになった．最終的にバグをフィックスできたのは，2013 年 1 月になってからであり，結果的に設計目標に示した項目を全て達成することができた．

現時点での未検討事項を以下に示す．

- 太陽光発電システムからリチウムイオンバッテリーを充放電するシステムの電圧および温度の経時特性の測定
- ミニリレーの ON／OFF 時に，あらかじめステータスを検査してから動作させる方法
- アラーム発出時に保守者に対して通報する方法

[謝辞]

総合情報学部人間情報デザイン学科の菅沼義昇教授には，ドキュメントクラスのポインタをクラスの引数として渡す場合の方法をご教授頂いた．この時期は検討に挫折しかかっていたときでもあり，ご教授を頂いた結果ボトルネックをクリアでき，検討を加速することができた．深く感謝致します．

また電気電子工学科の中田　篤史講師には，データロガーの特性を評価するための標準電圧発生器を借用させて頂いた．記して感謝の意を表します．

[参考文献]

1) 恩田　一，“蓄電型太陽光発電システム用リチウムイオン蓄電池の調査”，静岡理工科大学紀要，Vol.20，pp.37-40，(2012)
2) Turtle 社，”USB インタフェース付き接点出力ユニット取り扱い説明書”，p.12 (2009)
3) グラフテック株式会社，“GL220/820　GBD ファイル仕様書”
4) 村上恭子，” お絵描きソフト作りで学ぶグラフィックプログラミング入門“，pp.158～162(2007)
5) 山本信雄，”プログラミング学習シリーズ VisualC++③はじめての MFC プログラミング”，8.2 章 (2000)
6) 林　晴比古，“新 VisualC++プログラミング入門　シニア編”，ソフトバンクパブリッシング (2003)

[付録] 最終的なプログラムの所在

本資料は約 6 ヶ月間にわたり検討を進めてきて，2013.1.26 にデバグを完了したプロジェクト GLSample107 に基づいて作成した．

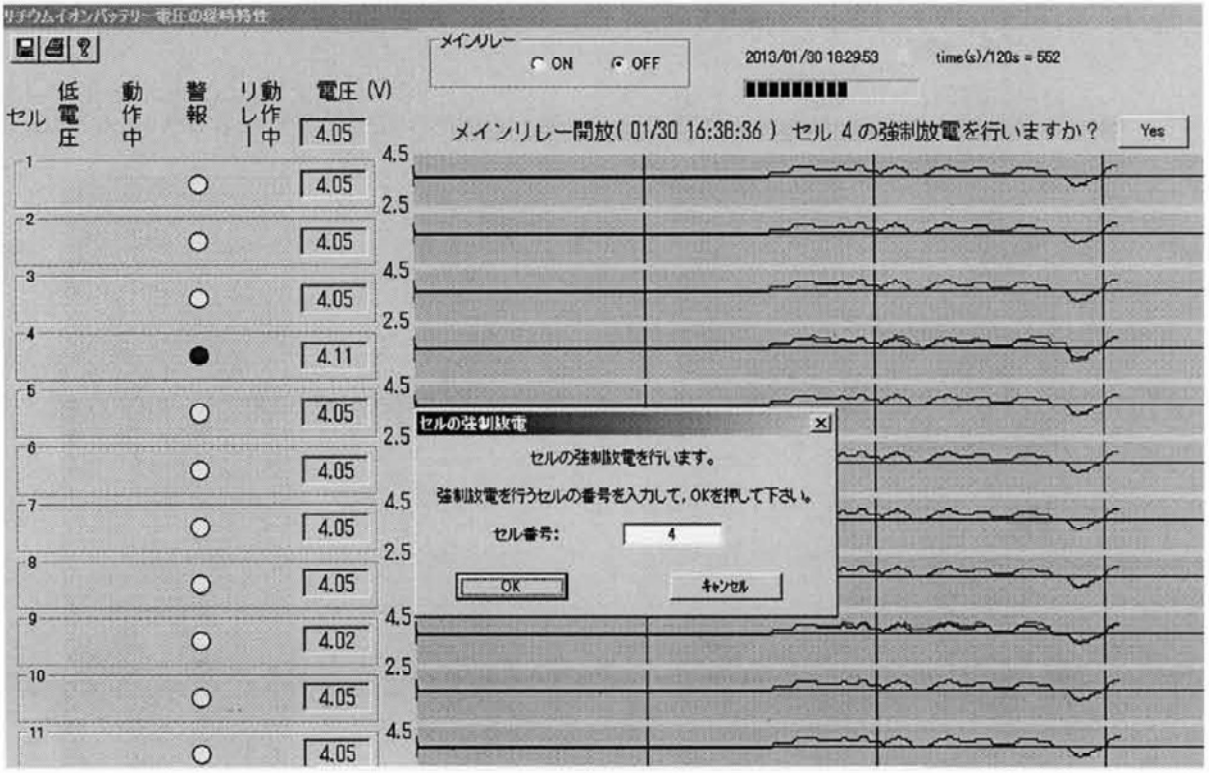


図 7.2　机上システムにおけるセル電圧の経時特性の測定例