

概要

世界のバッテリー市場は最近の電子技術の進歩により大きく成長しています。バッテリーのアプリケーションの中でも携帯電話、ポケットベル、補聴器、電気自動車、衛星などの用途が伸びています。バッテリーは民生用、産業用の様々なアプリケーションで使われ、その試験要求は一般に、化学組成、サイズ、用途で決まります。二次（充電）電池は充放電サイクルを繰り返して試験されるのが一般的です。二次電池の放電特性は、そのバッテリーの容量や寿命を知るのに重要です。製造試験では、バッテリー品質の検証やショートチェックに充放電サイクルがよく行われます。

代表的なバッテリーの充放電試験には、プログラマブル電源、電子負荷、電圧計、電流計が使われます。このアプリケーションノートは、2400型デジタルソース・メータがいかによりこれらの個別機器を置き換えるかを述べています。2400シリーズは電圧、電流を出力および測定でき、充放電サイクルを1台で行え、ラックススペースを節約し、プログラミング時間を最短にします。このアプリケーションノートは充放電の方法、パルス電流を使った放電（GSM携帯電話用バッテリーの例を含む）、複数のバッテリー試験について述べています。

試験について

二次電池は、その用途に応じた様々な方法で充放電されます。このアプリケーションノートは、定電圧源か定電流源を使って、それをどのように行うかを論じます。

定電圧法では、プログラマブル電源がバッテリーへ、その定格電圧を印加し、電流は安全な充電レートにリミットされます。バッテリーがフル充電されるにつれ、充電電流はゼロ近傍まで減衰します。安全上およびバッテリーのダメージを防ぐために、バッテリーを過充電しないよう注意を払わなくてはなりません。またバッテリーを規定電圧以下まで放電させると、充電に長い時間を要したり、ダメージを与えることがあります。

バッテリーは定電流で充電することもできます。この方法では、プログラマブル電流源が一定の電流を供給します。その電流源は、設定された電圧レベルになるまで、バッテリーを充放電します。

充放電する電流レートはバッテリーの容量（C）によって決まります。バッテリーの容量は、電流の流れはじめ（ $t=0$ ）からカットオフ電圧に到達するまでの電流の時間積分で定義され、次のように表されます。

$$\text{容量 (C)} = \int_{\Delta t} i dt$$

容量（C）はA（アンペア）・h（時間）で規定され、負荷電流に対して表記されるはずですが。たとえば、500mA・hのバッテリーを50mAで放電させることは0.1CもしくはC/10レートで放電させることになります。

500mA・h@50mAのバッテリーは負荷に10mAの電流を50時間供給できます。容量に影響を与えるファクタとしては、バッテリーのサイズ、化学組成、温度、放電レート、開回路時間等があります。充電レートは用途により変わります。

試験手順について

ソースメータを定電圧源として充放電する

図1はソースメータに接続されたバッテリーを表しています。2400が定電圧モードのとき、電圧または電流を測定できます。出力電圧値はバッテリーの充電あるいは放電の最終値にセットされます。電流コンプライアンスは、バッテリーの種類に応じた充放電の電流値にセットされます。ソースメータはバッテリーを希望する電圧レベルまで充電または放電させますが、その電圧レベルになると、いずれ電流コンプライアンス状態になります。その状態のときは、実際にはソースメータは低電流源として働きます。そのとき、前面パネルの**CMPL**インジケータが点灯します。

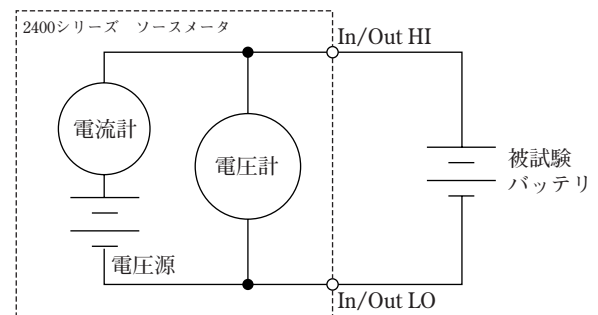


図1. 2400シリーズ ソースメータを定電圧源として使い、バッテリーを充放電する

ソースメータを定電流源として充放電する

図2に示す通り、ソースメータが定電流モードの時、電流または電圧が測定できます。まず適切な出力電流値を選びます。充電するときは正の電流値、放電するときは負の電流値を使います。コンプライアンス電圧の設定値は充電時と放電時で異なります。

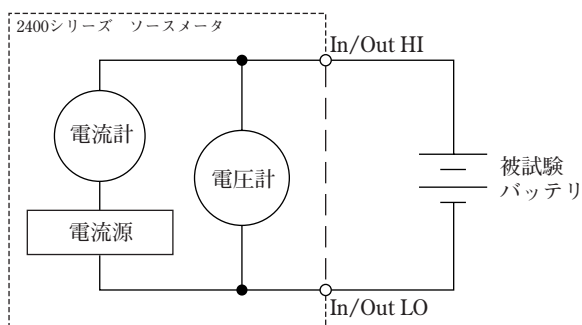


図2.2400シリーズ ソースメータを定電流源として使い、バッテリーを充放電する

充電時は、コンプライアンス電圧は希望するバッテリーの電圧レベルにセットされます。出力がオンされると、ソースメータはバッテリーがあらかじめ設定された電圧レベルになるまで定電流を出力します。その後、ソースメータはコンプライアンス状態、すなわち定電圧状態になります。それは、前面パネルの**CMPL**インジケータが点灯し、知らせます。

放電時は、コンプライアンス電圧を常にバッテリーのノミナル電圧値より大きくセットします。決してバッテリー電圧よりも低くセットしてはいけません。というのも、ソースメータがコンプライアンス状態になり、その結果、望まない大きな電流が流れてしまうからです。コンピュータでソースメータを制御しているときは、バッテリー電圧をソースメータでモニタし希望値と比較できます。

警告：バッテリーを充放電させるのにソースメータから電流出力させるとき、バッテリー電圧が電圧コンプライアンス設定値を越えていないかを確認してください。さもないと被試験バッテリーから過大な電流が流れてしまいます。また電流源の出力オフ状態がHIGH-IMPEDANCE(Hi-Z)に設定されているかも確認ください。これにより電流源の出力がオフのとき、出力リレーがオープンにされます。電流源の出力オフ状態がNORMAL(デフォルト状態)のとき、出力をオフにすると、電圧コンプライアンスはゼロに設定されます。この0Vコンプライアンスにより過大な電流がバッテリーから流出してしまいます。

出力オフ時にHIGH-IMPEDANCE(Hi-Z)状態にするには前面パネルから**CONFIG**ボタンを押し、続けて**OUTPUT**ボタンを押します。矢印キーを使い**OFF・STATE**を選び、**ENTER**を押します。**HIGH・IMPEDANCE**を選び、**ENTER**を押します。**EXIT**を押し、メインディスプレイに戻ります。 **GPIB**で行うには、**:OUTP:SMOD HIMP** コマンドを送ります。

7 放電サイクルの自動化

充放電サイクルは、しばしば数時間を要するため、コンピュータとプログラマブル計測器を使い、試験の自動化が望まれます。ソースメータはGPIBインターフェイスを内蔵し、同じくGPIBインターフェイスを備えたコンピュータを使い、バッテリーの充放電サイクルを自動化できます。次の2400BAT.BASは、12Vバッテリーを10Vまで、100mA電流負荷を使い放電させるプログラムです。6秒ごとに電圧が測定され、対応するタイムスタンプと共にデータファイル(BATTERY.DAT)に保存されます。このデータファイルから放電サイクル中の電圧対時間をプロットできます。電圧が測定される度に、それが10Vのスレッシュホールド電圧と比較され、それに到達したら出力はオフにされます。

図3は12Vバッテリーが放電される間に測定されたデータのプロット例です。電圧ばかりでなく時間もソースメータによって測定され、コンピュータのタイマを使うようにプログラムする必要がありません。データはMicrosoft Excel™を使ってプロットしました。

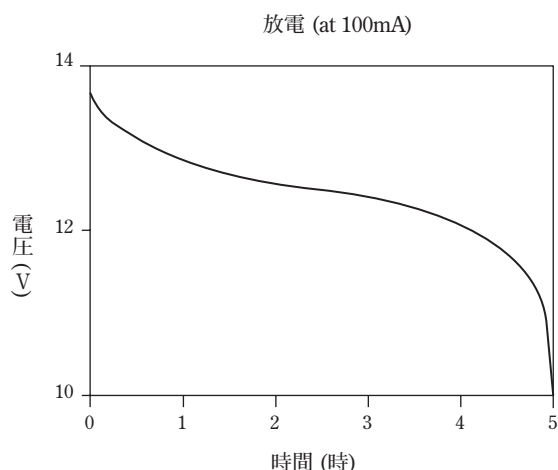


図3.バッテリーの放電曲線

// パルス電流を使った充放電

重要な注意：プログラムを実行する前に、ソースメータの電源をオンにし、出力モードをHIGH-IMPEDANCE(Hi-Z)に設定しなければなりません。これにより、バッテリーが電源オフ状態のソースメータにつながれ、電流が流出するのを防ぎます。一旦ソースメータの出力モードがHIGH-IMPEDANCE(Hi-Z)に設定されたら、その設定を2400のメモリロケーション0にストアします。それには、**MENU**を押し、**SAVESETUP**を選び、**ENTER**を押します。続けて**GLOBAL(ENTER)**、**SAVE(ENTER)**、**O(ENTER)**を押します。GPIBで行うには、***SAV O**コマンドを送ります。電源投入と同時にこの設定になるようプログラムします。前面パネルからまず**MENU**を押し、次の通り選択します。**GLOBAL(ENTER)**、**POWERON(ENTER)**、**USER-SETUP-NUMBER(ENTER)**、**O (ENTER)**。GPIBで行うには、**:SYST:POS O**コマンドを送ります。プログラムが実行される時、プログラムが***RCL O**を送り、設定用メモリ0番の内容がリコールされ、バッテリーからの電流流出は起こらなく安心です。

いくつかの二次電池は携帯電話やポケットベルなど負荷電流がデジタル信号のアプリケーションで使われます。この種のアプリケーション用のバッテリーの充放電試験は、パルス電流を使って行うことが望まれます。

パルス波形は、2400シリーズで、その出力値を変化させるか、またはソースリストコマンドを使い作成できます。パルス幅が約1秒以下の時は、ソースリスト機能を使います。それ以上の時は、単に固定レベル出力コマンドを使い、その出力値を変えたほうがいいでしょう。その2つのパルス出力アプリケーション例を示します。

例1：ソースメータを使い、パルス電流を出力して、電圧をモニタする

図4はパルス波形の例を示しています。-1Aで6.7msのピーク値と-0.2Aで13.3msのアイドル電流で、周期は20msです。

```
'program name is 2400bat.bas
'this program uses Microsoft Quicbasic 4.5 and the KPC0488.2AT IEEE Interface Card

'$INCLUDE: 'ieeeqb.bi'
CALL initialize(21, 0)
r$ =SPACE$(100)
OPEN "C:\qb45\battery.dat" FOR OUTPUT AS #1

'*RCL 0 was already programmed to come up in HI-Z standby
'and it was setup to be the power-on default
CALL send(24, " RCL 0", status%)

REM*****set source
CALL send(24, ":SOURCE:FUNC:MODE CURR", status%)
CALL send(24, ":SOURCE:CURR:RANG 100E-3", status%)
CALL send(24, ":SOURCE:CURR -100E-3", status%)
CALL send(24, ":SOURCE:CURR:MODE FIXED", status%)

REM*****set function
CALL send(24, ":SENS:FUNC:OFF:ALL", status%)
CALL send(24, ":SENS:FUNC 'VOLT'", status%)      'measure voltage
CALL send(24, ":VOLT:RANG 20", status%)         '20V range
CALL send(24, ":VOLT:PROT 15", status%)         'compliance 15V

REM*****set trigger
CALL send(24, ":TRIG:SOURCE IMM", status%)
CALL send(24, ":TRIG:DELAY 6", status%)         '1 reading every 6 s

REM*****set global parameters
CALL send(24, ":FORM:ELEM VOLT,TIME", status%)  'read back colts, time
CALL send(24, "OUTP ON;SYST:TIME:RES", status%) 'output ON, reset clock

DO
  CALL send(24, "READ?", status%)               'take readings and check if reached
  CALL enter(r$, length%, 24, status%)         '10V level
  PRINT r$
  PRINT #1, r$
  r = VAL(r$)
LOOP UNTIL r <= 10

CALL send(24, ":OUTP OFF", status%)
```

```

'program name is 2400puls.bas
'This program creates a current squarewave...-1A for 7ms and -.2A for 13ms.
'the waveform repeats until the measured voltage becomes less than 1 volt.
'the timing may vary slightly depending on the computer; the source delay
'time can be adjusted.
'This program uses Microsoft Quickbasic 4.5 and the KPC-488.2AT IEEE Interface Card

'$INCLUDE: 'ieeeqb.bi'
CALL initialize(21, 0)

CALL transmit("UNT UNL MTA LISTEN 24 SDC UNL UNT", gpib.status%)

'*RCL 0 was already programmed to come up in HI-Z standby
'and it was setup to be the power-on default
CALL send(24, " RCL 0", gpib.status%)

REM*****set source:
CALL send(24, ":source:func:mode curr", gpib.status%)
CALL send(24, ":source:curr:rang 1", gpib.status%)
CALL send(24, ":source:list:curr -1,-.2", gpib.status%)
CALL send(24, ":source:curr:mode list", gpib.status%)
CALL send(24, ":source:delay .0055", gpib.status%)

REM*****turn display off:
CALL send(24, ":disp:enable off", gpib.status%)

REM*****set measure:
CALL send(24, ":SENS:FUNC 'VOLT:DC'", gpib.status%)
CALL send(24, ":VOLT:RANG 10;NPLC 0.01:", gpib.status%)

REM*****global parameters
CALL send(24, ":FORM:DATA ASCII;ELEM VOLT", gpib.status%)
CALL send(24, ":syst:azero 0", gpib.status%)
CALL send(24, ":syst:rsen 1", gpib.status%)
data$ = SPACE$(80)

REM*****set trigger:
CALL send(24, ":ARM:COUNT 1", gpib.status%)
CALL send(24, ":TRIG:COUN 2", gpib.status%)
CALL send(24, ":TRIG:DELAY 0", gpib.status%)
CALL send(24, "":OUTP ON", gpib.status%)

DO
  CALL send(24, ":READ?", gpib.status%)
  CALL enter(data$, length%, 24, gpib.status%)
  r$ = LEFT$(data$, 13)

LOOP UNTIL VAL(r$) < 1!
PRINT VAL(r$)

CALL send(24, "":OUTP OFF", gpib.status%)
END

```

この波形を生成するコマンドシーケンスを記述したサンプルプログラム (2400plus.bas) をリストします。プログラム中、電圧が各ソースリストロケーション毎に測定させます。その電圧はあらかじめ指定された電圧値と比較され、それに到達すると、出力はオフにされます。

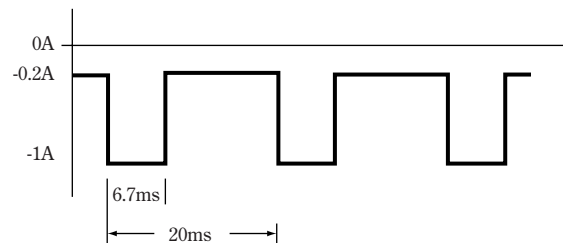


図4.2400型ソースメータが発生するパルス電流波形

ソースメモリリストコマンドは100個までの出力値のリストを定義するのに使われます。そのリストが実行されると、2400は順次ソースリストの値をソースディレイコマンドで指定された時間出力します。ソースリストの1つの制約は、リスト中の各々の値が同じ時間出力されることです。

図4の波形を作るには2つの（3つではなく）ソースリスト値が各々6.7ms出力されます。第一ロケーションは-1A、第二ロケーションは-0.2Aにセットされます。ソースディレイ時間は6.7msではなく5.5msにセットしました。というのは、ソフトウェアのオーバーヘッド時間を加味しないとイケないからです。この波形は必要なだけ繰り返されます。また、さらにオーバーヘッド時間とジッタがリストの最終値に加えられ、最終値は初めの値と比べ長くなります。このケースではオーバーヘッド時間は約6msになります。このGPIOオーバーヘッド時間は、データ転送や次の読み取り要求に使われます。この時間は使用するコンピュータに依存します。正確な波形を作成するには波形をオシロスコープで観測しながら、ソースディレイ値を調整して行います。

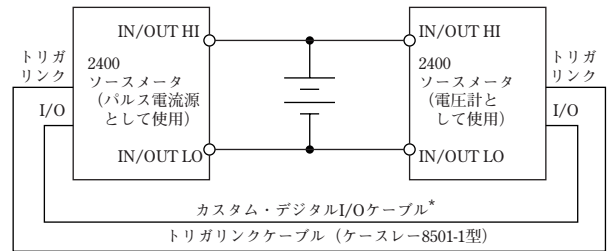
重要な注意：プログラムを実行する前に、“放電サイクルの自動化”の章の最後に述べたように、ソースメータの出力オフモードをHIGH-IMPEDANCE(Hi-Z)に設定してください。

例2：GSM携帯電話用のバッテリー試験

GSMの通話プロファイル試験のようないくつかのバッテリー試験では、数ミリ秒もしくは数百マイクロ秒のパルスを発生する必要があります。ソースメータはそのようなパルスを出力できますが、電圧測定を同時に行うことはできません。そのような場合、もう1台のソースメータを使い電圧を測定します。そのセットアップを図5に示します。2420型高電流ソースメータがパルスを発生し、2400型ソースメータがバッテリー両端の電圧降下を測定します。

2400GSM.basプログラムは-1.42Aを550 μ s、-0.22Aを4.1msの連続パルス電流波形を2420型に発生させます。このプログラムはGSM規格に沿っています。例1と同様、ここでもソースリスト機能を使い、パルス波形を作っています。パルスは、2400で測定したバッテリー両端の電圧が、あらかじめ決められたレベルに到達するまで、繰り返されます。それぞれのソースメータはトリガリンクコネクタを通じてトリガされます。2400は-1.4Aのパルス時間の最中に電圧を測定、比較します。データファイルのサイズを扱いやすくするため、2秒に1回ストアします。2400が、あらかじめ決められた電圧リミットに到達すると、5Vの信号をデジタルI/Oポートを通して、2420のデジタルI/Oポートにあるソースインターロック・ピンへ送り、出力をオフにします。そしてSRQが発生し、プログラムは停止します。

このプログラムが正しく動作するためには、2420型のバージョンがC9以降でなければなりません。このファームウェアはトリガモデル：ARM:COUNTのINFINITEをサポートし、パルスのタイミングを大変高速にしました。



* カスタム・ケーブルは2つのDB-9メスコネクタを使い、次のようにピン配線する。(ケースレーP/N CS-761-9)

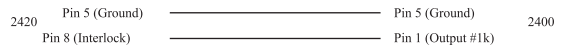


図5.GSM規格携帯電話用バッテリーの試験システム

出力をオフにする方法には限りがあります。このプログラムでは、前述のように、2400から2420の安全インターロック・ピンへ信号を送り行っています。もう一つの方法は、前面パネルの**OUTPUT**を押したり、**ABORT**やデバイス・クリアを送って出力をオフにし、***RST**または***RCL**を送ります。

図6は2420の実際の出力を、クランプオン電流プローブを使い、オシロスコープで観測したものです。

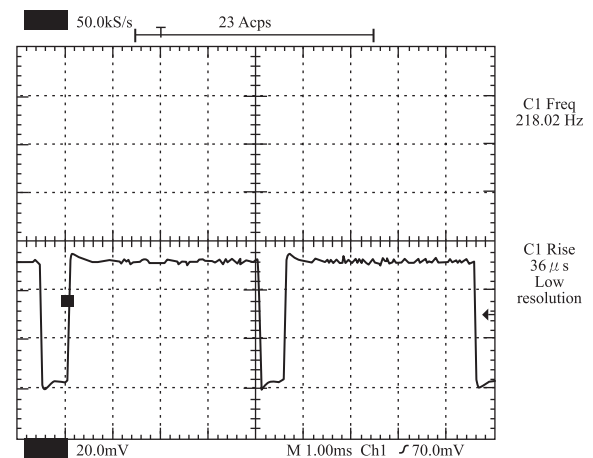


図6.2420型の電流出力波形（クランプオン電流プローブで観測）

```

'$INCLUDE: 'ieeeqb.bi'

'program name is 2400gsm.bas.
'This program creates a current squarewave for GSM profile.
'Press local and output to turn off output prior to reaching threshold.
'This program uses Micro Quickbasic 4.5 and the KPC-488.2AT IEEE Interface Card.

TYPE Rdg2400
    Reading AS SINGLE
    Time AS SINGLE
END TYPE
CONST NUMRDGS = 2400    'max value is 2500
DIM Rdg(2) AS Rdg2400
CLS

INPUT "Enter the 2400's Threshold Voltage: "; tval!

CALL initialize(21, 0)

CALL transmit("UNT UNL MTA LISTEN 24 LISTEN 16 SDC UNL UNT", gpib.status%)

' *RCL 0 was already programmed to come up in HI-Z standby
' and it was setup to be the power-on default
CALL send(24, " RCL 0", GPIB.STATUS%)
CALL send(16, " RCL 0", GPIB.STATUS%)

REM*****set-up 2400 to be a voltmeter:
CALL send(16, "form:elem volt,time;bord swap:data sreal", GPIB.STATUS%)
CALL send(16, ":sour:func:mode curr", GPIB.STATUS%)
CALL send(16, ":sour:curr:rang min", GPIB.STATUS%)
CALL send(16, ":sens:func 'volt'", GPIB.STATUS%)
CALL send(16, ":sens:volt:dc:rang 10", GPIB.STATUS%)
CALL send(16, "sens:volt:dc:nplc .01", GPIB.STATUS%)
CALL send(16, ":arm:count 1;:trig:count " + STR$(NUMRDGS), GPIB.STATUS%)
CALL send(16, "trig:sour tlink;asyn:ilin 2", GPIB.STATUS%)
CALL send(16, ":syst:azer:stat off", GPIB.STATUS%)

REM*****set-up limits for 2400
CALL send(16, ":calc2:lim2:stat 1", GPIB.STATUS%)
CALL send(16, "sour2:t1 0", GPIB.STATUS%)
CALL send(16, ":calc2:lim2:UPP:data 10;sour2 1", GPIB.STATUS%)
CALL send(16, ":calc2:lim2:LOW:data " + STR$(tval!) + ";sour2 1", GPIB.STATUS%)
CALL send(16, ":calc2:clim:clear:auto 1", GPIB.STATUS%)
CALL send(16, ":calc2:clim:pass:sour2 0", GPIB.STATUS%)
CALL send(16, ":calc2:lim2:stat on", GPIB.STATUS%)
CALL send(16, "sour:del 0.00035", GPIB.STATUS%)

REM*****setup 2420 to be a current source:
CALL send(24, ":sour:func:mode curr", GPIB.STATUS%)
CALL send(24, ":source:curr:rang 3", GPIB.STATUS%)
CALL send(24, ":source:list:curr -1.42,-.224,-.224,-.224,-.224,-.224,-.224,-.224", GPIB.STATUS%)
CALL send(24, ":SOURCE:curr:mode list", GPIB.STATUS%)

REM*****set measure of 2420:
CALL send(24, ":SENS:FUNC:OFF:ALL", GPIB.STATUS%)
CALL send(24, ":VOLT:RANG 10;PROT 10", GPIB.STATUS%) '(see note below)
'NOTE: PROT must be greater than the battery voltage

REM*****global paramters
CALL send(24, ":outp:int:stat on", GPIB.STATUS%) ' Enable Interlock

```

```

'Setup Status Model to SRQ on goto IDLE
'routine checks if test is done running
CALL send(24, ":stat:oper:enab 1024", GPIB.STATUS%)
CALL send(24, "SRE 128", GPIB.STATUS%)
' Clear Pending SRQ's
CALL spoll(24, poll%, GPIB.STATUS%)
CALL send(24, ":stat:oper?", GPIB.STATUS%)
A$ = SPACE$(80)
CALL enter(A$, 1%, 24, GPIB.STATUS%)

REM*****set trigger for 2420:
CALL send(24, "ARM:COUNT INF", GPIB.STATUS%)
CALL send(24, "arm:olin 2;outp trig", GPIB.STATUS%)
CALL send(24, ":source:delay .00035", GPIB.STATUS%)
CALL send(24, "TRIG:COUN 8", GPIB.STATUS%)
CALL send(24, "TRIG:DELAY 0", GPIB.STATUS%)

REM*****turn display off:
CALL send(16, ":disp:enable off", GPIB.STATUS%)
CALL send(24, ":disp:enable off", GPIB.STATUS%)

OPEN "Testdata.txt" FOR OUTPUT AS #1
PRINT #1, "Reading (V), Time (s)"
CALL settimeout(500)
R$ -=SPACE$(2)
Init2400% = 1
CALL send(24, ":OUTP ON", GPIB.STATUS%)
CALL send(16, ":SYST:TIME:RESET;:OUTP ON", GPIB.STATUS%)
DO
CALL send(16, ":READ?", GPIB.STATUS%)
IF Init2400% THEN
CALL send(24, ":INIT", GPIB.STATUS%) 'synchronize 2400 & 2420
Init2400% = 0
END IF
CALL transmit("UNT UNL MLA TALK 16", GPIB.STATUS%)
CALL receive(r$, 1%, GPIB.STATUS%)
FOR x% = 1 TO NUMRDGS
CALL rarray(Rdg(1), 8, 1%, GPIB.STATUS%)
IF GPIB.STATUS% = 8 THEN
x% = NUMRDGS
'Display and save the last Reading
PRINT #1, Rdg(1).Reading; ", "; Rdg(1).Time
PRINT Rdg(1).Reading; ", "; Rdg(1).Time
ELSEIF x% MOD 400 = 0 THEN
'Display and save reading every 2 seconds (as indicated by 400)
' NOTE: if 100 then .5s, 200 then 1s, 400 then 2s, 600 then 3s, etc.
'Make sure this number is an interger multiple of CONST NUMRDGS
'defined at the beginning of the program
PRINT #1, Rdg(1).Reading; ", "; Rdg(1).Time
PRINT Rdg(1).Reading; ", "; Rdg(1).Time
END IF
NEXT
CALL rarray(Rdg(1), 1, 1%, GPIB.STATUS%)
CALL transmit("UNT UNL", GPIB.STATUS%)
LOOP UNTIL (srq%) ' Wait for 2400 to finish
CALL transmit("UNT UNL MTA LISTEN 24 LISTEN 16 SDC UNL UNT", GPIB.STATUS%)
CALL send(16, ":OUTP OFF", GPIB.STATUS%)
CLOSE #1
END

```

重要な注意：プログラムを実行する前に、“放電サイクルの自動化”の章の最後に述べたように、2400、2420の出力オフモードをHIGH-IMPEDANCE(Hi-Z)に設定しなければなりません。

7 複数のバッテリーの試験

一つのバッテリーの充放電は、しばしば数時間かかるので、数個のバッテリーを直列につなぎ、同時に試験することが望まれます。図7は、40個のバッテリーと一台のソースメータの入出力を直列に接続した回路を示しています。

個々のバッテリーの電圧をモニターするために、DMMと各バッテリーの切り替え用スイッチが使われます。このアプリケーションでは、ソースメータを独立した電流源と電圧計として使うことはできません。というのも、ソースメータの入力HiとセンスHiの間の電圧差が5Vに制限されているからです。

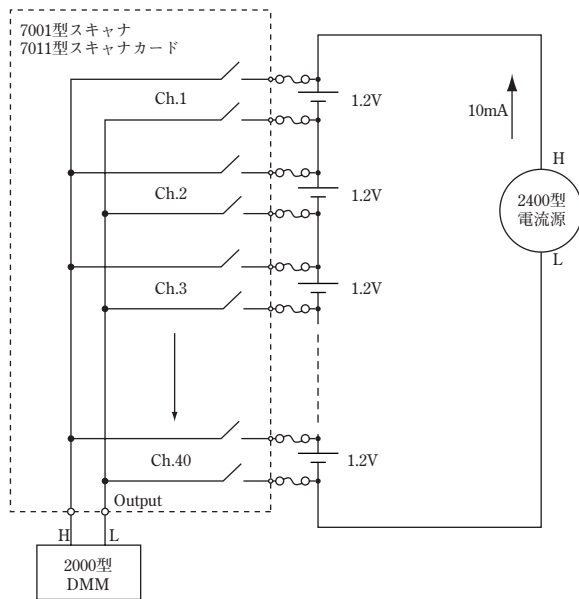


図7.直列接続されたバッテリーの充放電

この例では、ソースメータは40個のバッテリーに±10mAを供給し、すべてのバッテリーは同時に充電と放電が行えます。全バッテリーの電圧ドロップ合計は48Vになり、各バッテリーの電圧を測るのに2000型DMMを使います。2000型を各バッテリーへ切り替えるのに、7001型スキャナと7011型リレーカードを使います。測定された電圧値は、最終電圧値に達しているかを判断するために、あらかじめ決められた値と比較されます。ソースメータ、スイッチ本体、DMMすべてがGPIBインターフェイスを持ち、同じくGPIBを装備したコンピュータによってセットアップは自動で行えます。

リレーへのダメージを避けるために、スイッチ切替えと測定シーケンスを同期させるのが重要です。特に、特定のチャンネルを閉じる前に、他のすべてのリレーがオープンになっていることを確認します。不注意で複数のチャンネルを閉じてしまうと、バッテリーはショートし、リレーもダメージを受けます。この種のダメージからリレーを守るよう、電流リミット抵抗やヒューズを各スイッチに直列に挿入することをお奨めします。

7 他のソリューション

もし、充放電に3A以上の電流が必要なときは、10A (100W) までのソースまたはシンクが行える228A型電流/電圧源が使えます。また228Aは電流や電圧の読み取りも行えます。

もし個別の電子負荷を使うときは、バッテリー両端の電圧測定のみが必要です。その場合は、ケースレーの2000シリーズDMM (2000,2001,2002,2010) は高い入力抵抗を持ち、GPIBインターフェイスも標準装備で、高精度で高速なDC電圧測定が行えます。

Specifications are subject to change without notice.

All Keithley trademarks and trade names are the property of Keithley Instruments, Inc.
All other trademarks and trade names are the property of their respective companies.

KEITHLEY

ケースレーインスツルメンツ株式会社

〒105-0022 東京都港区海岸1-11-1 ニューピア竹芝ノースタワー13F TEL : 03-5733-7555 FAX : 03-5733-7556
Web site : www.keithley.jp · Email : info.jp@keithley.com

Keithley Instruments, Inc

28775 Aurora Road · Cleveland, Ohio 44139 · 440-248-0400 · Fax: 440-248-6168
1-888-KEITHLEY (534-8453) · www.keithley.com

© Copyright 2002 Keithley Instruments, Inc

Printed in the Japan

0502300034