ユーザーズ・ガイド
マルチセル・チャージャ/ディスチャージャ
AgilentモデルE4370A
パワーバス負荷
AgilentモデルE4371A
64チャネル・チャージャ/ディスチャージャ
AgilentモデルE4374A

Agilent Technologies

Agilent Part No. 5964-8140

2000年9月

原　典

本書は"USER'S GUIDE Multi-Cell Charger/Discharger Agilent Model E4370A Powerbus Load Agilent Model E4371A 64-Channel Charger/Discharger Agilent Model E4374A" (Part No. 5964-8138) (Printed in USA: May 2000)を翻訳したものです。

詳細は上記の最新マニュアルを参照して下さい。

ご　注　意

- 本書に記載した内容は、予告なしに変更することがあります。
- 当社は、お客様の誤った操作に起因する損害については、責任を負いかねますのでご了承ください。
- 当社では、本書に関して特殊目的に対する適合性、市場性などについては、一切の保証をいたしかねます。
- また、備品、パフォーマンス等に関連した損傷についても保証いたしかねます。
- 当社提供外のソフトウェアの使用や信頼性についての責任は負いかねます。
- 本書の内容の一部または全部を、無断でコピーしたり、他のプログラム言語に翻訳することは法律で禁止されています。
- 本製品パッケージとして提供した本マニュアル、フレキシブル・ディスクまたはテープ・カートリッジは本製品用だけにお使いください。プログラムをコピーをする場合はバックアップ用だけにしてください。プログラムをそのままの形で、あるいは変更を加えて第三者に販売することは固く禁じられています。

納入後の保証について

- ハードウェア製品に対しては部品及び製造上の不具合について保証します。又、当社製品仕様に適合していることを保証します。
  ソフトウェアに対しては、媒体の不具合（ソフトウェアを当社指定のデバイス上適切にインストールし使用しているにもかかわらず、プログラミング・インストラクションを実行しない原因がソフトウェアを記録している媒体に因る場合）について保証します。又、当社が財産権を有するソフトウェア（特注品を除く）が当社製品仕様に適合していることを保証します。
  保証期間中にこれらの不具合、当社製品仕様への不適合がある旨連絡を受けた場合は、当社の判断で修理又は交換を行います。
- 保証による修理は、当社営業日の午前８時 45 分から午後５時 30 分の時間帯でお受けします。なお、保証期間中でも当社所定の出張修理地域外での出張修理は、技術者派遣費が有償となります。
- 当社の保証は、製品の動作が中断されないことや、エラーが皆無であることを保証するものではありません。保証期間中、当社が不具合を認めた製品を相当期間内に修理又は交換できない場合お客様は当該製品を返却して購入金額の返金を請求できます。
- 保証期間は、製品毎に定められています。保証は、当社が据付調整を行う製品については、据付調整完了日より開始します。但し、お客様の都合で据付調整を納入後 31 日以降に行う場合は 31 日目より保証が開始します。又、当社が据付調整を行わない製品については、納入日より保証が開始します。
- 当社の保証は、以下に起因する不具合に対しては適用されません。
  (1) 不適当又は不完全な保守、校正によるとき
  (2) 当社以外のソフトウェア、インターフェース、サプライ品によるとき
  (3) 当社が認めていない改造によるとき
  (4) 当社製品仕様に定めていない方法での使用、作動によるとき
  (5) お客様による輸送中の過失、事故、滅失、損傷等によるとき
  (6) お客様の据付場所の不備や不適正な保全によるとき
  (7) 当社が認めていない保守又は修理によるとき
  (8) 火災、風水害、地震、落雷等の天災によるとき
- 当社はここに定める以外の保証は行いません。又、製品の特定用途での市場商品価値や適合性に関する保証は致しかねます。
- 製品の保守修理用部品供給期間は、製品の廃止後最低５年です。

安全サマリ

以下の一般的な安全に関する注意事項は、本製品のすべての操作段階において遵守しなければなりません。これらの注意事項または本書のその他の箇所にある特定の警告に従わないと、本器の設計、製造、目的用途の安全基準に違反します。ヒューレット・パッカード社は、顧客がこれらの条件に従わなかった場合の責任を負いません。

一般的な注意事項

本製品は、安全クラス1(感電防止用アース端子付き)の測定器です。取扱い説明に明記されていない方法で使用した場合、本製品の保護機能が損なわれる恐れがあります。

本製品に使用されているLEDはすべて、IEC 825-1に準拠したクラス1のLEDです。

環境条件

本器は、屋内の、設置カテゴリII、汚染度2の環境において使用するように設計されています。最大相対湿度95%、最高高度2000メートルにおける操作が可能です。主電源のAC電圧要件および周囲動作温度の範囲については、仕様一覧を参照してください。

電源を投入する前に

すべての安全予防策が取られていることを確認してください。「安全記号」に説明されている測定器の外部マークにご注意ください。

本器をアース接続してください

感電事故を最小限に抑えるため、本器のシャーシとカバーは、必ず、電気グランドに接続してください。本器は、アース付きの電源ケーブルを介して主AC電源に接続し、アース線をコンセントの電気グランド(安全用アース)にしっかりと接続する必要があります。感電防止用(グランド)導線が遮断されたり、感電防止用アース端子が切断されると、感電による人身事故の恐れがあります。Agilentパワーバス負荷のグランド端子を外部DC電源のグランド端子に接続してください。

ATTENTION: Un circuit de terre continu est essentiel en vue du fonctionnement sécuritaire de l'appareil. Ne jamais mettre l'appareil en marche lorsque le conducteur de mise … la terre est d,branch,.

爆発の恐れのある環境で操作しないでください

本器を可燃性ガスや煙のある場所で使用しないでください。

本器のカバーを取り外さないでください

オペレータは本器のカバーを取り外さないでください。部品交換や内部調整は、必ずサービスマンが行うようにしてください。

損傷または欠陥があると思われる測定器は、サービスマンが修理を行うまで使用を中止し、誤って操作されないようにしてください。

安全記号

安全記号			
	直流		注意、感電の危険有り
	アース (グランド) 端子		注意、表面が高温
	感電防止用アース (グランド) 端子 (外部感電防止用導線への接続用)		注意 (添付の文書を参照してください)
	電源オン (主AC電源への接続を示します)。		機器オン (機器の一部がオン状態にあることを示します)。
	電源オフ (主AC電源からの切断を示します)。		機器オフ (機器の一部がオフ状態にあることを示します)。

本書の目的

本書ではAgilentマルチセル・チャージャ/ディスチャージャ・システムの「標準」版について説明します。本書には、設置方法、接続情報、プログラミング情報、サンプル・プログラム、および仕様が記載されています。Agilent MCCDユーザ・インタフェースに関する情報は、オンラインで提供されます。システム・オプションについては、本書に付属のオプション・シートで説明しています。本書の情報はすべて変更されることがあります。印刷の日付の新しい方がアップデート版です。

ご注意

目次

1

概説

Agilent MCCDシステムの機能

Agilentマルチセル・チャージャ/ディスチャージャ (MCCD) システムは、リチウム・イオン電池の製造に関する固有の要件やニーズに対処することを目的としています。Agilent MCCDシステムは、リチウム・イオン電池の正確な充放電および測定を行うことができます。本製品は、Agilent E4370Aマルチセル・チャージャ/ディスチャージャ・メインフレームと、最大4個のAgilent E4374A 64チャネル・チャージャ/ディスチャージャ・カードから構成されます。すべてロードした場合、各メインフレームは256の入出力チャネルを持ちます。メインフレームとモジュールをさまざまな構成で組み合わせることにより、電池製造プロセス用の高性能の充放電装置を低価格で実現できます。

下の図は、Agilent MCCDシステムの簡単なブロック図です。次にシステムの基本機能と高度な機能について簡単に説明します。

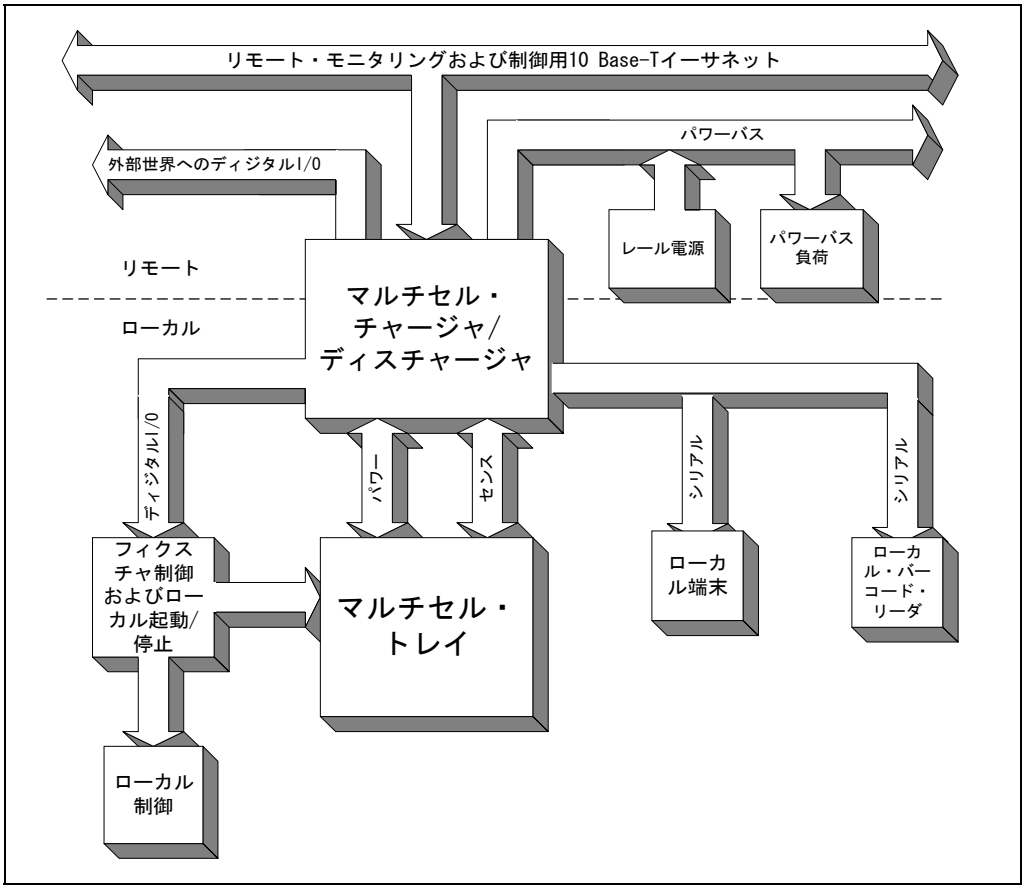


図1-1. Agilent MCCDシステムのブロック図

基本機能

- **チャージャ**－正確に制御された電流および電圧を電池に供給することにより、適切なフォーミングを実行します。各電池は、電池のフォーミング・シーケンスを独立した速度で進みます。したがって、電池がシーケンスの異なるポイントにある場合、充電中の電池と放電中の電池が存在する可能性があります。
- **ディスチャージャ**－正確に制御された電流を電池から引き出すことにより、フォーミングや容量測定を実行します。
- **測定**－充電、放電、休眠中に、電池のパラメータをモニタできます。電圧、電流、時間、内部抵抗、アンペア時、ワット時の測定が可能です。これらの測定値に基づいて、電池のフォーミング・シーケンスを調整することにより、安全性と信頼性を向上させ、適切な電池フォーミングを実現できます。
- **ディジタルI/O制御**－Agilent MCCDに接続されたディジタルI/Oを通じて、信号のモニタやスティミュラスの供給が可能です。これにより、接続が簡略化され、拡張が容易になり、集中制御システムよりも信頼性が向上します。高速動作を利用して、障害をいち早く検出してシステムを停止することが可能です。
- **RS-232制御**－Agilent MCCDのシリアル・ポートに周辺機器を接続することにより、プリンタ、バーコード・リーダ、ローカル端末、ロボットなどのローカル増設機器を、ホスト・コンピュータからパススルー制御できます。
- **機器保護**－広範囲の安全機能によって、フォーミング中の電池とハードウェアの両方を、機器故障、プログラミング・エラー、電池の不良、その他の外部不良から保護します。

その他の機能

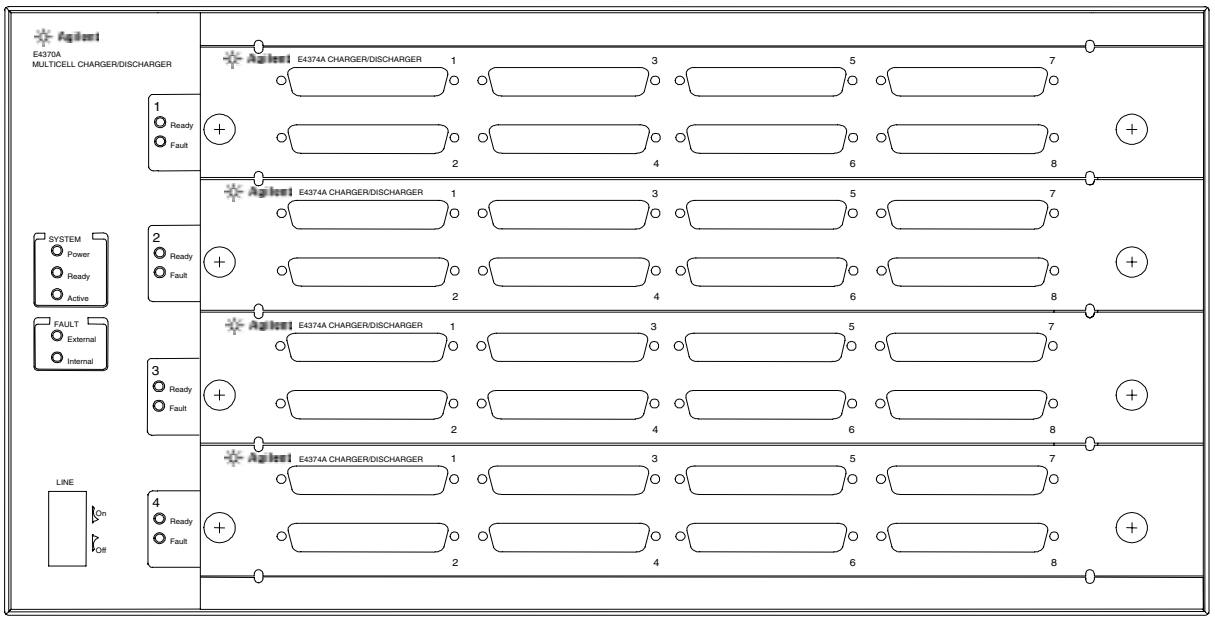
- Webサーバのグラフィカル・ユーザ・インタフェースとAPI (Application Programming Interface) の両方を使用できる、LAN 10 base-T制御
- 包括的なデータ記憶機能と、リモート・データ収集
- 修理の際のダウンタイムを最小限に抑える、取外しが簡単なチャージャ/ディスチャージャ・カード
- ソフトウェアで変更可能な充放電シーケンスにより、システム・ハードウェアを変更せずに製造プロセスを簡単に変更可能
- 電池またはチャネルの隣接ブロックをグループとして定義、構成。これにより、電池グループで異なるシーケンスを同時に実行可能
- 充放電シーケンス中に常時プログラミング回路の校正を行うことにより、温度ドリフトによる誤差を除去
- 放電中の電池のエネルギーを使って充電中の電池にエネルギーを供給することにより、双方向パワー伝送およびエネルギーの再利用を実現

ハードウェア解説

Agilent E4370A/E4374A MCCD

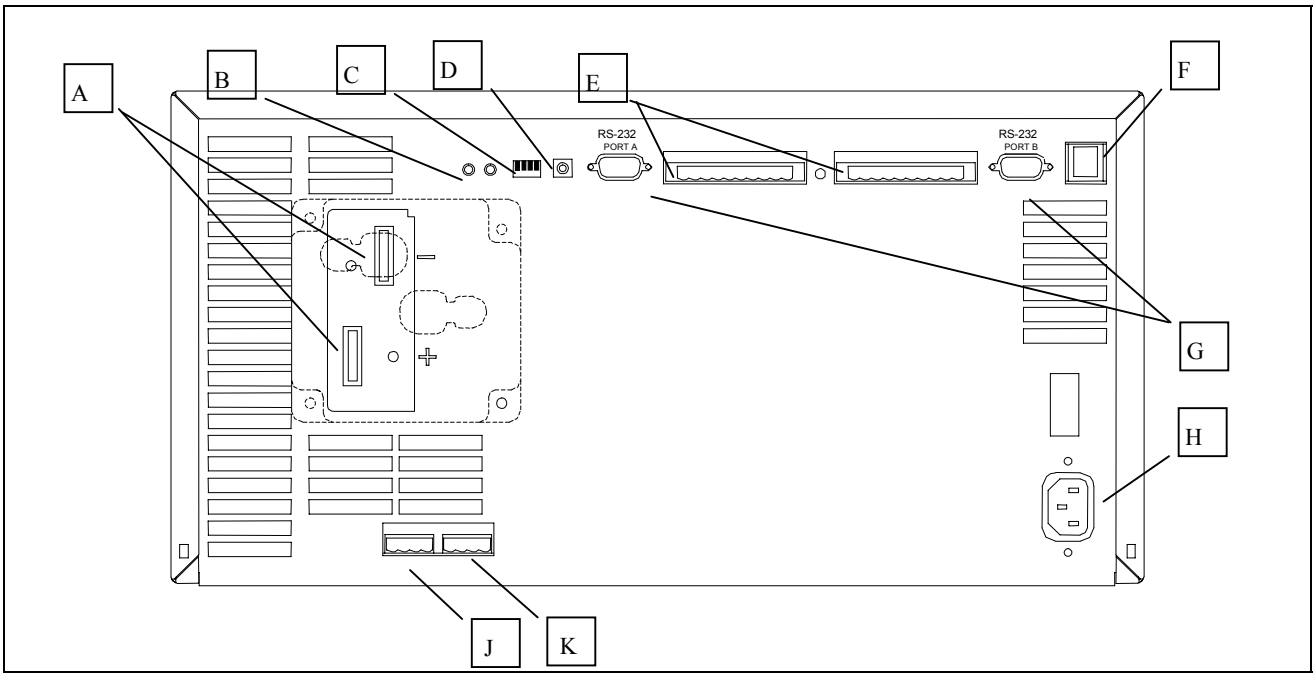
Agilent E4370A MCCDメインフレームは、フル・ラック・ボックス型であり、Agilent E4374A 64チャネル・チャージャ/ディスチャージャ・カードを収納するためのスロットが4つ装備されています。Agilent E4374A 64チャネル・チャージャ/ディスチャージャ・カードには、電池を個別に最高5V/2Aで充放電する回路があります。

注記: 各出力チャネルの最大可能コンプライアンス電圧は5.5Vです。コンプライアンス電圧とは、電池で要求される電圧にフィクスチャ/配線によるすべての電圧降下を足した値です。この高いコンプライアンス電圧により、最大0.5Vの配線損失で、5 Vを直接電池に供給できます。



LINE	Agilent MCCDのAC電源をオン/オフします。電源をオフにすると、パワーバスとの接続に用いられる内部リレーが切断されます。したがって、このスイッチはユニットとパワーバスとを遮断する役割も果たします。
SYSTEM	
Power	点灯している場合、メインフレームの電源がオンであることを示します。
Ready	点灯している場合、ユニットの操作準備が完了していることを示します。 消灯している場合、外部パワーバス電圧が高すぎるか低すぎることを示します。
Active	点灯している場合、LANケーブルでデータ通信が行われていることを示します。 点滅している場合、LAN通信が進行中であることを示します。
FAULT (不良状態のクリア方法については、付録Eを参照してください)	
External	点灯している場合、次のような外部不良を示します。 外部ディジタル不良信号を受信 停電シャットダウン信号を受信 電源投入後の高パワーバス電圧 電源投入後の低パワーバス電圧 過熱
Internal	点灯している場合、次のような内部ハードウェア故障を示します。 セルフテスト不良 校正エラー ハードウェア・エラー
1, 2, 3, 4	
Ready	カードの電源がオンになっており、使用可能であることを示します。
Fault	点灯している場合、次のような内部ハードウェア故障を示します。 セルフテスト不良 校正エラー ハードウェア・エラー

図1-2. Agilent E4370A/E4374A MCCDの前面パネルのコントロールとインジケータ



A	＋およびーパワーバス・コネクタ (–バス・バーはシャーシ・グランドに接続) 校正ステータスLED	G H	RS-232コネクタ (ポートAおよびB) AC電源接続 (87Vac～250Vac、50/60Hzの電源電圧に対するユニバーサルAC入力)
B	設定スイッチ	J	補助出力コネクタ
C	伝送校正スイッチ	K	校正ポート
D	ディジタルI/Oコネクタ		
E	LAN接続		
F			

図1-3. Agilent E4370A/E4374A MCCD裏面パネルの接続

Agilent E4371Aパワーバス負荷

放電サイクルでは、放電中の電池からの余剰パワーを消費するために、Agilent E4371Aパワーバス負荷が必要です。負荷は、定電圧モードのみで動作し、内部抵抗のオン/オフを順次切り替えて、パワーバスの電圧を中間点の26.75V付近に調整します。必要な負荷の数は、システム内のAgilent MCCDメインフレームの数によって決まります。Agilent E4371Aパワーバス負荷1個で、256チャネルのAgilent E4370A MCCDメインフレーム2個の全パワーに対応できます。

Agilent E4371Aパワーバス負荷の裏面パネルには＋とーのパワーバス・コネクタがあります。そのほかグランド接続もあります。安全要件を満たすには、Agilentパワーバス負荷のグランド端子を外部DC電源のグランド端子に接続します。負荷は、その動作電力をパワーバスから得ています。パワーバスのDC電圧が23.8Vより下がった場合、またはパワーバス上に使用可能な電力がない場合、負荷は動作しません。負荷をプログラミングすることはできません。出荷時に正しい動作電圧に設定されており、校正は不要です。

負荷上のオン/オフ・スイッチによって、負荷とパワーバスの接続と切断を行います。内部ファンにはパワーバスから約1.5Aの電流が流れます。

注意:　Agilent E4371Aパワーバス負荷は、最大定格パワーを放電する際、手で触れないほど熱くなります。

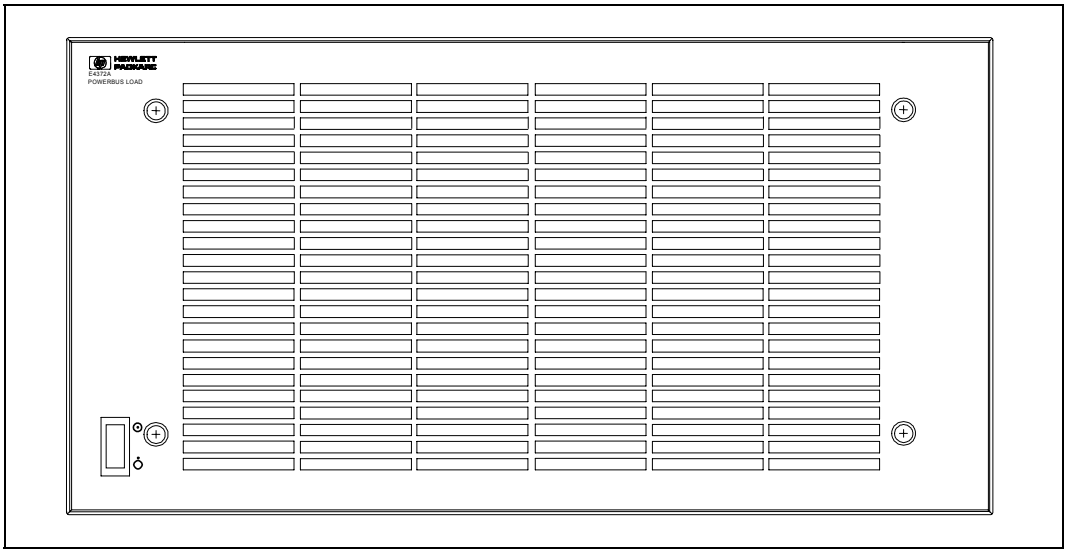


図1-4. Agilent E4371Aパワーバス負荷の前面パネル

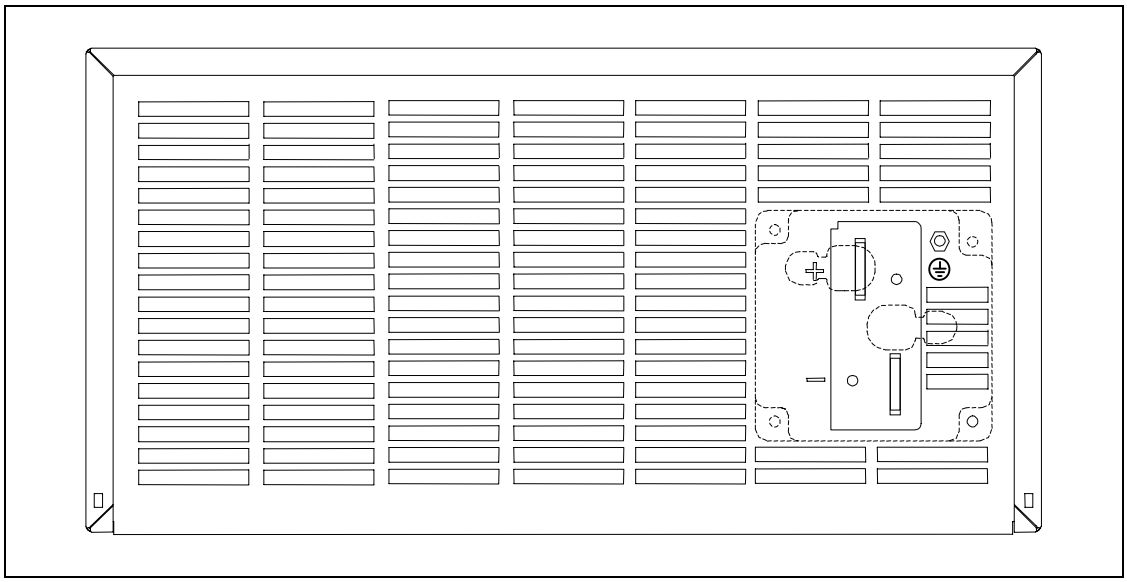


図1-5. Agilent E4371Aパワーバス負荷の裏面パネル

外部電源

充電サイクルでは、電池に電力を供給するための外部DC電源が、Agilent MCCDメインフレームのそれぞれに対して必要です。外部電源は、メインフレーム裏面のパワーバス端子に接続されます。電源の定格電圧は24Vで、必要な電池充電パワーの125%を供給できなければなりません。たとえば、256チャネル・システムで、チャネルあたり5V、2A (すなわち2.56kW) の電池充電パワーを供給するには、約3.2kW (24V、133A) の電力を各Agilent MCCDメインフレームに供給するDC電源が必要です。

充電電流が小さい場合は、それに応じて電源の定格電流も小さくできます。たとえば、前述のシステムの最大出力電流が電池1個あたり1Aの場合は、定格63Aの電源を使用できます。

また、十分な定格電流を持つ1個の電源を、共通パワーバスに接続された複数のメインフレームで共有することができます。ただし、全電流を供給したときでも、Agilent MCCD裏面のパワーバス端子で電圧仕様が満たされる必要があります。

注記: 外部DC電源に過電圧保護回路が装備されている場合、放電サイクル中にシャットダウンしないように30Vより高く設定してください。

複数のAgilent MCCDによる構成

次の図は、8個のメインフレームをフル装備したAgilent E4370A MCCDシステムです。

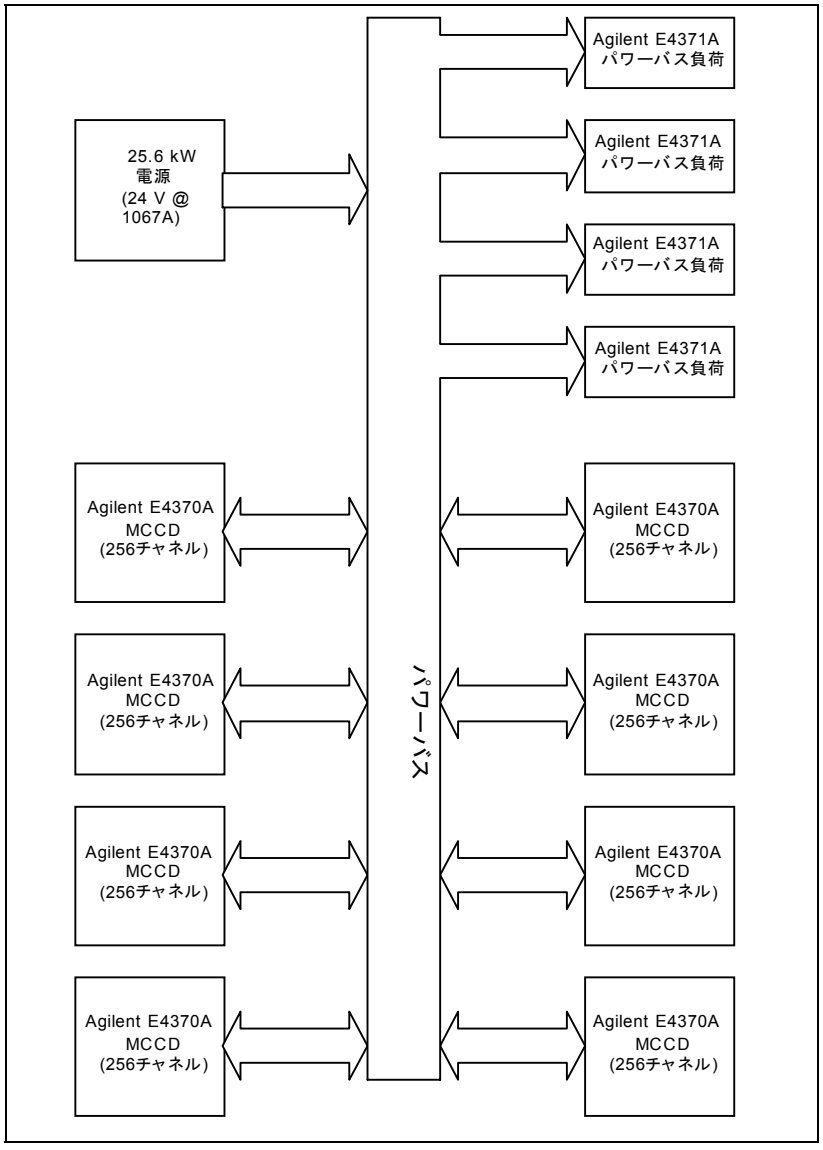


図1-6. 最大システム構成

このようなシステムに必要な最大パワーは25.6kWです。十分な定格電流を持つ1個の電源を、パワーバスに接続された複数のメインフレームで共有することができます。ただし、全電流を供給したときでも、すべてのメインフレーム裏面のパワーバス端子で24VのDC入力条件が満たされる必要があります。図に示す1個の24V、25.6kW DC電源の代わりに、複数の24V DC電源を並列に接続することもできます。

Agilent E4370A MCCDシステムをマルチユニット・システムで使用する場合、エネルギー効率を高めるため、外部電源の補助として放電エネルギーを他の電池の充電に利用できます。このために、共通のパワーバスにつながれた充電中の電池と放電中の電池の間で、双方向のパワー伝送機能を利用します。このエネルギー伝送を利用するためには、放電モードで動作するメインフレームと充電モードで動作するメインフレームが、1つのシステムに同時に存在する必要があります。

この構成に特別な制御システムは必要ありません。24V DC電源のレギュレーション回路、Agilent E4370A MCCD、およびAgilent E4371Aパワーバス負荷は、特別なハードウェア制御線や付加ソフトウェアがなくても正しく動作します。

注記:　大電流を伝送するためには、十分なサイズのパワーバス配線が必要です。表2-5を参照してください。

測定機能

Agilent MCCDメインフレームおよびチャージャ/ディスチャージャ・カードには、全チャネルで電圧および電流測定を実行する高速スキャン・システムが装備されています。測定システムのテクニカル・データについては、付録Aを参照してください。以下の測定が可能です。

電圧測定

Agilent MCCDは、校正済みの内部測定回路を使って各チャネルの電圧を測定します。ローカル・センシング・モードでは、電圧測定はパワー・コネクタで行われます。リモート・センシング・モードでは、電圧はリモート・センス・リード線の終端で測定されます。リモート・センシングがローカル・センシングより優れている点は、リモート・センス・リード線を電池に接続すると、電池の実際の電圧が測定されることです。負荷リード線の電圧降下は測定には影響しません。詳細については、第2章の「リモート・センシング」の項を参照してください。

注記:　Agilent MCCDシステムをローカル・センシング用に設定すると、測定した出力電圧は電池の実際の電圧を示しません。これは、配線抵抗、プローブ抵抗、コネクタ抵抗などによるワイヤ内の電圧降下により、電池で使用可能な電圧が減少するからです。

電流測定

Agilent MCCDは、校正済みの内部測定回路を使って各チャネルの出力電流経路における実際の電流を測定します。

容量測定

アンペア時容量－Agilent MCCDは、連続電流測定に基づいて計算を行い、電池のアンペア時容量を求めます。

充電中、測定を行うたびに測定電流に測定間隔を掛けて、各測定間隔中に電池に流れた実際の増分アンペア時容量を計算します。次に、この増分量を累積アンペア時値に加算して、電池に流れた総アンペア時容量を求めます。アンペア時容量は、充電中、正の値になります。このため、定電圧充電など充電電流が一定でないときでも、正確なアンペア時容量が測定できます。

放電中は、測定を行うたびに測定電流に測定間隔を掛けて、電池から引き出された実際の増分アンペア時容量を計算します。次に、この増分量を累積アンペア時値に加算して、電池から引き出された総アンペア時容量を求めま

す。アンペア時容量は、放電中、負の値になります。このため、放電電流が一定でないときでも、正確なアンペア時容量が測定できます。

ワット時容量－Agilent MCCDは、連続電流および電圧測定に基づいて計算を行い、ワット時電池容量を求めます。

充電中、測定を行うたびに測定電流と測定電圧と測定間隔を掛けて、各測定間隔中に電池に流れた実際の増分ワット時容量を計算します。次に、この増分量を累積ワット時値に加算して、電池に流れた総ワット時容量を求めます。ワット時容量は、充電中、正の値になります。このため、充電電流と電圧が変化しているときでも、正確なワット時容量が測定できます。

放電中は、測定を行うたびに測定電流と測定電圧と測定間隔を掛けて、各測定間隔中に電池から引き出された実際の増分ワット時容量を計算します。次に、この増分量を累積ワット時値に加算して、電池から引き出された総ワット時容量を求めます。ワット時容量は、放電中、負の値になります。このため、放電電流と電圧が変化しているときでも、正確なワット時容量が測定できます。

電池抵抗

連続した電圧、電流、および容量測定に加えて、ACおよびDC電池抵抗の測定も可能です。この測定は、シーケンスが実行中でないときにはコマンドで実行したり、フォーミング・シーケンス内の独自のステップとして実行することができます。

AC電池抵抗を測定する場合、Agilent MCCDは、まず、充放電回路をすべての電池から切り離します。Agilent MCCDメインフレーム内のAC波形発生器が各電池に順次接続されます。AC波形発生器はそれぞれの電池に一瞬だけ小さな励振電流を流し、測定システムが電池の出力電圧と出力電流を測定します。電池のAC抵抗を得るには、狭帯域同調フィルタを用い、電流に対する電圧の振幅と位相角度を計算します。この方法は、LCRメータで用いられる方法と類似しています。測定が各チャネルに対して順次行われるため、テストの間、他のチャネルは休眠状態になります。

DC電池抵抗を測定する場合、Agilent MCCDは、まず、充放電回路をすべての電池から切り離します。Agilent MCCDメインフレーム内のパルス・ジェネレータが各電池に順次接続されます。パルス・ジェネレータはそれぞれの電池に持続時間の短いパルス電流を流し、測定システムが高確度の高速A/Dコンバータを使って電池の電圧と電流をディジタイズします。電池のDC (またはパルス) 抵抗を得るには、専用のアルゴリズムを用い、パルス電流の変化に対する電圧の変化を計算します。測定は各チャネルに対して順次行われるため、テストの間、他のチャネルは休眠状態になります。

プローブ抵抗

プローブ抵抗測定も実行可能です。Agilent MCCDはリモート・センス機能を使ってパワー・プローブとセンス・プローブの両方の抵抗を測定します。プローブ抵抗測定は、シーケンスが実行中でないときはコマンドで実行することができます。

測定されたプローブ抵抗は、配線抵抗、プローブ抵抗、信号経路内のコネクタの抵抗など、信号経路内の全抵抗を表します。センス・プローブ測定の場合、内部スキャナ抵抗 (通常は1000Ω)も含まれます。パワーおよびセンス・プローブ測定では、実際の測定値がΩ単位で返されます。

シーケンスの実行中には、オンコマンド・プローブ抵抗測定に加えて、プローブが常時チェックされます。プローブ・チェック検証の詳細については、第5章の「プローブのチェック」を参照してください。

データ・ロギング

充放電シーケンスの実行中、Agilent MCCDは電圧、電流、および容量測定を定期的に実行します。あらゆる測定をデータ・バッファにログするのではなく、重要な測定だけがデータ・バッファにログされるようにデータ・ロギングを制御することも可能です。これはイベント・ベースのデータ・ロギングと呼ばれ、重要なイベントが発生する

たびに、データ・ログ・レコードがデータ・バッファに書き込まれます。重要な測定だけを保存することによって、バッファ・メモリを最も効率的に使用できます。

以下のイベントを、重要な測定のトリガとして使用できます。

電圧の変化 (ΔV)
トリガが**ΔV**の場合、ユーザ指定の電圧変化を上回ると、データ・ログ・レコードがバッファに書き込まれます。ΔVを100mVに設定した場合、電圧表示値の変化が100mVより大きくなるたびに、レコードがバッファに書き込まれます。

電流の変化 (ΔI)
トリガが**ΔI**の場合、ユーザ指定の電流変化を上回ると、データ・ログ・レコードがバッファに書き込まれます。ΔIを100mAに設定した場合、電流表示値の変化が100mAより大きくなるたびに、レコードがバッファに書き込まれます。

時間の変化 (Δt)
トリガが**Δt**の場合、ユーザ指定の時間間隔を上回ると、データ・ログ・レコードがバッファに書き込まれます。Δtを1秒に設定した場合、レコードは毎秒バッファに書き込まれます。Δtは、事実上、クロック方式のデータ・ログです。

ΔV、ΔI、およびΔtの値の許容範囲は、0から無限です。値を0または0付近に設定した場合、すべての表示値がΔV、ΔIまたはΔtの値0を上回るため、全部バッファにログされます。このため、測定ログがすぐに一杯になってしまいます。値を大きな値か無限に設定した場合には、表示値がΔV、ΔIまたはΔtの値を上回ることはないので、バッファにログされる表示値はありません。

ΔV、ΔI、およびΔtの値を上回ったかどうかを確認するための比較テストが、各測定間隔の終わりに実行されます。したがって、データ・バッファへレコードを書き込む際の最高速度が、Agilent MCCDの測定速度になります。任意の組合わせのイベントを指定し、指定したイベントのうちのいずれかが発生したら、データ・ログ・レコードをデータ・バッファに書き込むようにすることもできます。

データ・バッファ内の各レコードには、情報としてステータス (CV/CCおよびステップ番号を含む)、経過時間、電圧、電流、アンペア時、およびワット時が含まれています。保存できる表示値の総数については、仕様テーブルを参照してください。データ・ログは巡回待ち行列で、データを連続してデータ・バッファに記録していきます。データ・バッファが一杯になると、バッファ内の一番旧いデータが新しいデータによって上書きされます。データの損失を防ぐには、データが上書きされる前にコントローラがバッファからデータを読み取る必要があります。データは、テスト・シーケンス中にいつでもデータ・バッファから読み取ることができます。

注記: AC電源で停電が発生すると、データ・バッファ内の情報が失われます。停電によるデータの損失を防ぐには、cfShutdown関数を使ってデータを不揮発性メモリに保存します。詳細については、第5章の「停電時の操作」を参照してください。AC電源を一時的な遮断から保護するには、Agilent E4370Aのメインフレームを600 VA無停電電源装置 (UPS) に接続してください。

Agilent E4373Aマニュアル・パッケージに付属のソフトウェアに測定ログ・ユーティリティが含まれています。このユーティリティを使えば、データ・ログを読み取り、その情報をPC上のファイルに保存することができます。Agilent MCCD測定ログ・ユーティリティの使用法については、第4章を参照してください。

保護機能

Agilent MCCDには、ハードウェアとフォーミング中の個々の電池を致命的損傷から保護するために、広範囲の保護機能が装備されています。また、保護ステータスを製造システムの他の部分に伝達して、より高度な保護を実現することもできます。

内部保護機能

パワーバスとAgilent E4374Aチャージ/ディスチャージ・カードの間には、内部リレーが存在します。これらのリレーは、パワーバス上の過電圧または不足電圧条件からAgilent MCCDを保護します。リレーは、外部不良条件が検出された場合にもAgilent MCCDを保護します。出力レギュレータには、ハードウェア故障から電池を保護するための機能がいくつかあります。回路がすべてのチャネルに対して直列に接続されており、電池の逆極性、電池故障、レギュレータ故障からシステムを保護します。内蔵の温度センサが最大温度上昇をチェックし、過熱による故障を防ぎます。また、ファンによって、内部温度が許容範囲に保たれます。

さらに、Agilent MCCDには安全性のレベルを高める内蔵ハードウェア・ウォッチドッグ・タイマがあります。ハードウェア・ウォッチドッグ・タイマは、CPU、ソフトウェア、ファームウェアの動作とは関係なく機能します。内部ファームウェアまたはソフトウェア不良のためにAgilent MCCDのCPUが数秒以上機能を停止した場合、ハードウェア・ウォッチドッグ・タイマがAgilent MCCDを電源投入時の状態にリセットします。このステートでは、チャネル出力は電池から切断されます。

注記: 過電圧および過電流保護を実現するために、テスト・シーケンスの一部として過電圧テストと過電流テストを含めることが可能です(第5章を参照)。

外部ディジタルI/O保護機能

Agilent MCCDのディジタルI/Oサブシステムは、設定により保護機能を実現できます。ディジタルI/O信号は独立して動作するので、コンピュータまたはLAN接続に問題が発生しても、Agilent MCCDの保護機能は影響を受けません。第2章で説明するように、16個のディジタルI/O信号それぞれに次の保護機能の1つを個別に設定できます。

外部不良入力 この機能を使えば、外部不良条件によって入力が真に設定された場合に、電池フォーミング・シーケンスを停止することができます。

外部不良出力 この機能を使えば、外部不良条件または内部不良条件が発生したことを外部回路または別のAgilent MCCDに信号で知らせることができます。

外部インターロック この機能を使えば、外部不良条件以外の理由で電池フォーミング・シーケンスを停止することができます。

外部トリガ この機能を使えば、電池フォーミング・シーケンスを起動することができます。

保護機能に加えて、ディジタルI/Oを汎用I/Oとして使用することも可能です。汎用I/Oとして設定した場合、ディジタル・コネクタの入力信号や出力信号は、LANを経由してAPIプログラミング・コマンドで直接制御されます。

AC電源に不具合が発生した場合

AC電源に不具合が発生した場合、Agilent MCCDのCPUはシャットダウンします。充放電動作がすべて停止し、現在のシーケンス、テスト・データ、プログラム設定は失われます。

注記: 600 VA無停電電源装置 (UPS) を使ってAgilent E4370A MCCDメインフレームにAC電力を供給することにより、停電中のデータの損失を防ぐことができます。

停電時には、Agilent MCCD内のバイアス電源式リレーによってパワーバスもAgilent MCCDから切り離されます。したがって、停電のためにAgilent MCCDにAC電力が供給されなくなった場合、パワーバスに電力が供給されており、依然アクティブであったとしても、安全を確保するために、内部リレーが切断され、充放電も停止されます。

また、停電がAgilent MCCDに影響しない場合でも、パワーバスの電圧降下が起きたときには、Agilent MCCDによってパワーバスの不足電圧条件として検出され、リレーが開きます。このため、接続されている電池がさらに充電または放電されることはありません。

リモート・プログラミング・インタフェース

Agilent MCCDのリモート・プログラミング・インタフェースは、LANベースのTCP/IP通信プロトコルに基づきます。LANへの接続には、裏面パネルの標準8ピン10Base-Tコネクタを使います。第3章の指示に従って、LANの設定を行ってください。LAN通信プロトコルは、2種類の方法で実現されています。

アプリケーション・プログラミング・インタフェース (API)

アプリケーション・プログラミング・インタフェースは、Windows 95またはWindows NT 4.0上で、付属のC言語関数呼び出しを使って実行されます。これらの関数呼び出しについては、第5章および第6章に記載されています。この方法では、Agilent MCCDに関するもっとも包括的な制御を実現できます。Agilent MCCDがリモート・コンピュータに接続されて自動製造プロセスの一部として用いられる場合は、APIインタフェースによる制御が最適です。

Web互換のAgilent MCCDユーザ・インタフェース

Agilent MCCDにはグラフィカル・ユーザ・インタフェースを持つWebサーバ機能が内蔵されており、Netscape Navigator 3.03またはMicrosoft Internet Explorer 3.02などの標準的なWebブラウザを使ってアクセスします。このAgilent MCCDユーザ・インタフェースからは、個々の電池ステートのモニタ、テスト実行中の電池の電圧と電流の測定、およびテスト・ステータスの完全なモニタと制御が可能です。テスト・システムの評価、プロセスのプロトタイプ作成、プログラムのデバッグには、Agilent MCCDユーザ・インタフェースによる制御が最適です。

電池フォーミング・プロセスの例

Agilent E4370A MCCDは、図1-7に示すような電池フォーミング・プロセスに統合できるように設計されています。図からも分かるように、すでに説明したAgilent E4370A MCCDの保護および外部信号機能の多くは、ディジタルI/O接続によって実現されます。Agilent MCCD裏面のシリアル・ポートを使うと、ローカル周辺機器をホスト・コンピュータから直接制御できます。Agilent MCCDへのリモート・プログラミング・インタフェースを使えば、これらの機能をすべて、電池フォーミング・プロセスにシームレスに統合できます。

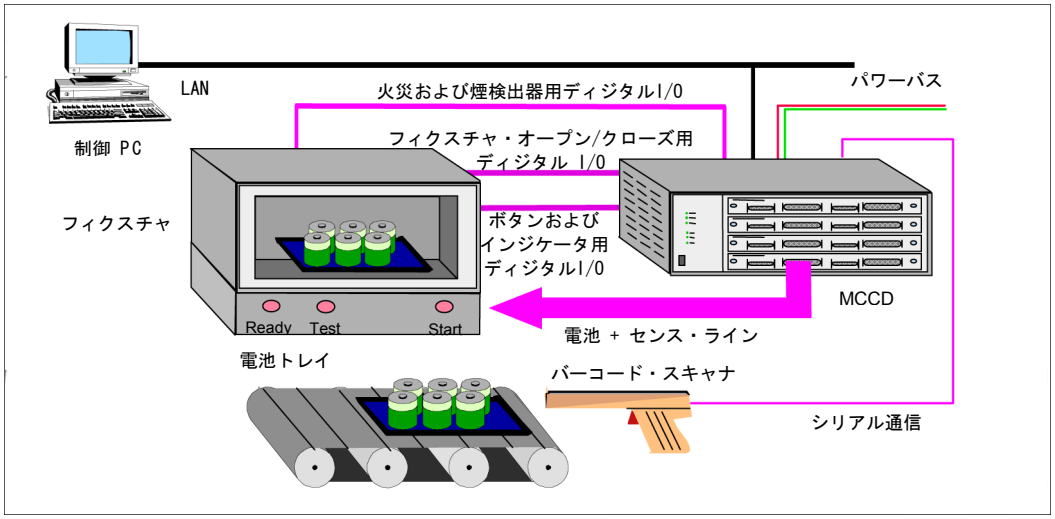


図1-7. 代表的な電池フォーミング・ステーション

次の電池フォーミング例は、Agilent E4370A MCCDを使って半自動プロセスを実現したものです。このシステムで人間が実行するのは、バーコード・スキャナによるデータ入力、テスト・フィクスチャのロードとアンロード、および電池フォーミング・プロセスの開始だけです。電池フォーミング・プロセスを実現するための関数呼び出しの詳細については、第5章および第6章を参照してください。

- 制御PCが、LAN経由でディジタルI/Oに信号を送り、テスト・フィクスチャのReadyライトをオンにします。これは、システムが次の電池トレイを受け入れ可能なことをオペレータに通知するためです。さらに、制御PCは、Agilent MCCDのRS-232バッファにあるシリアル・データのポーリングを開始します。
- オペレータは、コンベア・ベルト上の電池トレイのバーコードをスキャンします。次に、トレイをテスト・フィクスチャにロードし、フィクスチャをクローズします。
- RS-232バッファにデータがあるのを検出すると、制御PCはバーコード・データを読み取ります。PCはデータに基づいて適切なフォーミング・シーケンスをAgilent MCCDにダウンロードします。また、オンにするチャネル出力、プローブ・チェック設定、トリガ源などのセットアップ情報もダウンロードします。
- 制御PCは、Startボタンに接続されたディジタルI/Oラインをポーリングします。
- オペレータがStartボタンを押すと、制御PCがそれを検出し、ディジタルI/Oラインをポーリングしてフィクスチャがクローズされていることを確認します。PCは信号を送信して、Readyライトをオフ、Testライトをオンにします。これは電池フォーミング・シーケンスが開始されたことをオペレータに通知するためです。
- 制御PCは、Agilent MCCDにトリガを送信して、フォーミング・シーケンスを開始します。さらに、機器ステータスのポーリングを開始して、テスト・シーケンスの終了をモニタします。
- 電池フォーミング・シーケンスが実行されます。テスト・シーケンスは、自動的に電池にスティミュラスを供給し、電池のパラメータをモニタして合否を判定し、テスト結果を保存します。テスト・シーケンス中は、Agilent MCCDは火災および煙検出器に接続された専用ディジタルI/Oラインをモニタします。このため、問題が発生しても迅速に対応できます。
- Agilent MCCDの機器ステータスがシーケンスの終了を示すと、制御PCはAgilent MCCDにコマンドを送信してすべての電池の内部抵抗を測定させ、すべての測定データをアップロードします。
- 最後に、制御PCは信号を送信して、Testライトをオフ、Readyライトをオンにします。これは、トレイをフィクスチャから取り出して、次のバッチを開始してもよいことをオペレータに通知するためです。

第7章に、C言語で書かれたプログラミング例を紹介しています。これらの例では、ユーザが独自のアプリケーション・プログラムを開発できるように、Agilent MCCDの各種機能の実現方法を示します。プログラム2が、ここで説明した例に一致します。

2

設置

検査

装置を受け取ったら、輸送中に目に見える損傷を受けていないかどうか検査してください。損傷があった場合、ただちに運送業者と最寄りのAgilentセールス・サポート・オフィスに連絡してください。Agilentセールス・サポート・オフィスの一覧は、本書の巻末にあります。保証内容については、本書の最初に記載されています。

Agilent MCCDの検査が済むまでは、返送に備えて輸送用カートンと梱包材料を保管しておいてください。Agilent MCCDを修理のために返送する際には、モデル番号、シリアル番号、所有者を記載した札を付けてください。また、問題の内容を簡単に記したものを同封してください。

部品およびアクセサリ

表2-1は、Agilent E4370A/E4374A MCCDの付属品リストです。これらの一部はユニットに取り付けられています。

表2-1. 付属品

品名	部品番号	説明
電源コード (1)	最寄りのAgilentセールス・サポート・オフィスにお問い合わせください	各地域向けの電源コード

表2-2に、Agilent E4370A/E4374A MCCDに付属していない別売のアクセサリ・リストを示します。ユーザーズ・ガイド以外のすべてのアクセサリは、Agilent MCCDと、コンピュータ、テスト・フィクスチャ、またはAgilent MCCDによって制御される外部デバイスを接続する際に必要となります。

これらのアイテムについては、該当するキットをオーダするか、メーカに直接オーダしてください。コネクタ部品のメーカの所在地については、表2-3をご覧ください。

表2-2. アクセサリ

品名	メーカ部品番号	説明
ディジタル・コネクタ (2)	Phoenix MSTB-2.5/10-STF	ユニット裏面のディジタル・コネクタに接続される10ピン端子プラグ
校正コネクタ (1) 補助バイアス・コネクタ (1)	Phoenix MSTB-2.5/4-ST	ユニット裏面の校正および補助コネクタに接続される4ピン端子プラグ
マニュアル・パッケージ	Agilent E4373A	ユーザ・マニュアル、ソフトウェア・ドライバ、ユーティリティ・プログラムを含む
シリアル・ケーブル	Agilent 34398A	ポートAまたはB用のRS-232ヌル・モデム・ケーブル (回路図については図2-4を参照)
37ピン	AMP 205210-2	Agilent E4374A前面パネル・チャネル・コネクタ用の差

D-subコネクタ		込みコネクタ。Agilent E4374Aカード1枚当たり、8個のコネクタが必要 (Agilent E4374Aカード上のコネクタ・ピンの最大定格電流は5 Aです。)
37ピン・コネクタ用 コネクタ・フード	AMP 749916-2	Agilent E4374Aカード1枚当たり、8個のコネクタ・フードが必要
37ピン・コネクタ用 クリンプ式接点	AMP 66506-9	37ピン・コネクタ用クリンプ接点 コネクタ1個当たり16接点が必要 (ピンに使用可能なワイヤの線径は20-24 AWGだけです)
クリンプ式接点用 クリンプ・ツール	AMP 58448-2	ハンド・クリンプ・ツール
37ピン・コネクタ用 はんだ式接点	AMP 66570-2	37ピン・コネクタ用クリンプ接点 コネクタ1個当たり16接点が必要 ツールは不要 (ピンに使用可能なワイヤの線径は18 AWGだけです)
前面パネル・フィラー・ パネル	Agilent部品番号 5002-1505	Agilent MCCDメインフレームの各空きスロットに1枚のブランク・フィラー・パネルが必要
ラックマウント・ フランジ・キット	Agilent部品番号 5062-3979	フランジ2個、ファスナ、マウント用ネジを含む
ラックマウント・フランジ・キット(ハンドル付き)	Agilent部品番号 5062-3985	ハンドル2個、フランジ2個、ファスナ、マウント用ネジを含む

表2-3. メーカ所在地

会社名	住所	連絡先
Phoenix Contact	P.O. Box 4100 Harrisburg, PA 17111-0100	電話:717-944-1300 ファックス:717-944-1625 http://www.phoenixcontact.com/index.html
AMP	Harrisburg, PA 17111	http://www.amp.com/
アジレント・テクノロジー株式会社	マニュアル裏面のリストを参照してください。	http://www.agilent.com/

配置

Agilent E4370A MCCDメインフレーム

付録Cの外形図に、Agilent MCCDの外形寸法を示します。メインフレームはラックを使わずに自立型で設置することもできますが、側面および裏面に通気のための十分な空間を確保する必要があります。メンフレームは、標準の600mm幅のシステム・キャビネットにマウントできます。マウントすると十分な通気用の空間が確保されます。メインフレームをラックにマウントする際には、サポート・レールも必要です。レールは、通常、キャビネットと一緒にオーダします。

Agilent MCCDに装備されている冷却ファンは、ユニット左側から空気を吸い込み、裏面と側面から排出します。両側面には最低9cmの空間が必要です。メインフレームの裏面には最低23cmの空間が必要です。**裏面または側面の排気孔をふさがないでください。**

注記: Agilent MCCDメインフレームを正しく冷却するには、メインフレームの前面パネルに空きスロットがあってはいけません。Agilent E4374Aチャージャ/ディスチャージャ・カードをインストールしていないか、スロットから取り外した場合、空いているスロットにはブランク・フィラー・パネルをインストールする必要があります。表2-2を参照してください。

Agilent E4371Aパワーバス負荷

注意: Agilentパワーバス負荷を冷却するために十分な通気を確保するには、最低でも負荷の前後に0.6mの空間を確保する必要があります。ラックにマウントする際には、ラックのドアを取り外したままにしておいてください。

Agilent E4371Aパワーバス負荷は、最大定格パワーを放電する際、手で触れないほど熱くなります。

付録Cの外形図に、Agilentパワーバス負荷の外形寸法を示します。負荷はラックを使わずに自立型で設置することもできますが、前面および裏面に通気のための十分な空間を確保する必要があります。負荷に装備されている冷却ファンは、前面から空気を吸い込み、裏面から排出します。最大通気量は1分当たり10立方メートルです。

Agilent E4371Aパワーバス負荷は、裏面のドアを取り外せば、標準の600mm幅のシステム・キャビネットにマウントできます。マウントすると十分な通気用の空間が確保されます。ラックマウント・キットについては、表2-2を参照してください。ユニットをラックにマウントする際には、サポート・レールが必要です。安全要件を満たすには、Agilentパワーバス負荷のグランド端子を外部DC電源のグランド端子に接続してください。

チャネル接続

1台のAgilent E4370A MCCDメインフレームは、4台のAgilent E4374Aチャージャ/ディスチャージャ・カードをインストールしたときに、最大256個の電池の充放電を制御できます。各Agilent E4374Aチャージャ/ディスチャージャには、64のチャネルが存在します。本書のプログラミングの項では、チャネルを出力とも呼んでいます。フル装備の状態では、256の充放電チャネルが以下のように構成されます。

表2-4. チャネル構成

カード番号	コネクタ番号							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1～8	9～16	17～24	25～32	33～40	41～48	49～56	57～64
2	65～72	73～80	81～88	89～96	97～104	105～112	113～120	121～128
3	129～136	137～144	145～152	153～160	161～168	169～176	177～184	185～192
4	193～200	201～208	209～216	217～224	225～232	233～240	241～248	249～256

Agilent E4374Aカードのパワー接続には、8個の37ピンD-subコネクタを使います。コネクタには、シールドとストレーン・リリーフを付けることができます。コネクタでは対応するセンス接続も可能です。差込みコネクタのオーダ情報については、表2-2を参照してください。表で示すように、差込みコネクタには、使用するコネクタの種類に応じてAWG 24からAWG 18までの線径のワイヤを使用できます。ワイヤ接続を行うには、差込みコネクタにワイヤを取り付ける必要があります。取り付けが終了したら、Agilent E4374Aカードの前面に差込みコネクタを装着してください。前面パネル・コネクタのピン配置については、付録Dを参照してください。

使用されていない特定のチャネルがあれば、それらのチャネルを非アクティブに設定できます。非アクティブ・チャネルは開放回路です。チャネル出力の構成方法には2通りあり、ユニットの電源投入時にはそれぞれ違う働きをします。

- cfSetOutputConfig()関数 (第6章を参照) を使ってチャネル出力を構成した場合、設定は不揮発性メモリに保存されません。ユニットの電源を投入するたびに、設定をプログラムし直す必要があります。
- Agilent MCCDユーザ・インタフェース (第4章を参照) のシーケンス・セットアップ・ページを使ってチャネル出力を構成した場合、設定は不揮発性メモリ内に保存されます。電源を入れ直しても、ユニットの設定はそのままです。

注記: メインフレームに空きカード・スロットがあると、通常、これらのカード・スロットに対して予約されているチャネルは、非アクティブなチャネルとして処理されます。

電圧降下と配線抵抗

注記: **各チャネルのパワー・コネクタで、最大5.5V、2Aが得られます。**定格出力では、Agilent E4374Aチャージャ/ディスチャージャは、配線抵抗、プローブ抵抗、コネクタ抵抗などによる負荷リードの電圧降下を0.5Vまで許容します。電圧降下が大きいほど、電池で得られる電圧は低下します。太いワイヤ線を使い、フィクスチャ接点の抵抗を下げることにより、配線による電圧損失を最小限に抑え、電池の充電に使用できる電圧を最大限に高めることができます。

パワー・コネクタから電池までのリード線の長さは、システムで許容される電圧降下の大きさによって決まります。電圧降下は、ワイヤ、コネクタ、プローブ抵抗 (表2-5を参照) によって決まります。詳細については、「リモート・センス接続」の項を参照してください。

性能を最適化し、出力の不安定性と出力ノイズを減少させるには、次の指針に従ってください。

- センス・ワイヤ線およびパワー・ワイヤ線をよったり、シールドするのは、良い技術練習となります。
- パワー・ワイヤ線をよりあわせ、できるだけ短くします。
- センス・ワイヤ線をよりあわせます。センス・ワイヤ線をパワー・ワイヤ線とよりあわせないでください。
- 可能であればセンス・ワイヤ線をシールドします。シールドはケースに接続してください。
- ケーブルの全長をできるだけ短くします。
- 抵抗の小さいフィクスチャ接点を使います。

リモート・センス接続

センス接続は、フィクスチャでのリモート・センス機能を実現します。各カードのセンス接続は、パワー接続を収容するコネクタと同じコネクタを介して行います。

リモート・センス機能を使うと、出力電圧を電池で測定できるため、配線の損失が補正されます。Agilent E4374Aカードのコンプライアンス電圧 (Agilent MCCDがプログラム可能な定格以上に供給できる電圧) は最大5.5Vで、チャネル出力と電池接続間の配線におけるIR電圧降下をすべて補正することができます。この高いコンプライアンス電圧により、最大0.5Vの配線損失で、5Vを直接電池に供給することが可能です。リチウム・イオン電池の充電電圧が4.0～4.1Vの範囲内にある場合、コンプライアンス電圧は、最大1.4～1.5Vの配線による損失を補正できます。

下表に、さまざまな線径の抵抗値を示します。この値を使って、いろいろな長さや直径のワイヤ線の電圧降下を計算できます。太くて短いワイヤほど、電圧降下は小さくなります。下表には、最大電流2Aで電圧降下を1.4Vに制限する最大ワイヤ長も示されています (1.4Vは、パワー・コネクタで得られる5.5Vのコンプライアンス電圧と、典型的なリチウム・イオン電池を充電するのに必要な4.1Vとの差です)。

表2-5. 銅より線の抵抗

AWG番号	mm^2	抵抗 (20℃で)		2Aで電圧降下を1.4Vに制限する最大ワイヤ長 (m) (＋と－のリード線の全長)
		Ω/m	Ω/ft	
18	0.825	0.022	0.0066	30
20	0.519	0.034	0.0105	20
22	0.324	0.055	0.0169	12
24	0.205	0.087	0.0267	8

たとえば、パワー接続にAWG番号24のワイヤ線を使用しており、充電電圧が2Aで4.1Vであると仮定します。この直径のワイヤ線を使い、最大電流を2Aと仮定した場合、パワー・コネクタと電池の間の最長距離は約4mに制限されます。これは、＋と－のパワー・リード線の合計長が8mのとき、配線の最大電圧降下が1.4V (2A×0.7Ω) になるためです。電池に必要な充電電圧が4.1Vで、コンプライアンス電圧が5.5Vの場合は、これがAgilent MCCDの最大許容電圧降下になります。

注記: 本例では、フィクスチャ接点抵抗などのリード・パス抵抗や、フィクスチャ・リレーを一切考慮していません。その他の抵抗が存在する場合、リード線の長さをさらに短くする必要があります。

パワーバス接続

注意: Agilent MCCDメインフレームとAgilentパワーバス負荷の両方にパワーバスを接続するときは、極性を守ってください。逆の極性で接続すると、Agilent MCCDメインフレームと Agilentパワーバス負荷の両方に損傷を与える結果となります。Agilent MCCDメインフレームの負の (－) バス・バーは、シャーシ・グランドに接続します。

パワーバスとの接続には、Agilent E4370A MCCDメインフレームおよびAgilent E4371Aパワーバス負荷の裏面にある＋と－のバス・バーを使います。これらのバス・バーを用いれば、複数のメインフレーム、外部電源、およびその他の負荷を相互接続できます。これらのバス・バーには、直径7mmのボルトが入る取付け穴が装備されています。

注記: 端子突出部を各パワーバス・ケーブルに適切に固定してください。裸線をバス・バーに直接接続しないでください。細いワイヤを数多くよりあわせたより線を用いた方が、少数の太いワイヤをよりあわせたより線を用いるよりも作業が簡単です。

パワー接続を行う際には、個別終端ワイヤ、バス・バー、またはその両方を組み合わせて用いることができます。正しく動作させるためには、すべてのパワーバス構成で、磁気放射を抑えるためにループ領域を最小限にし、CRTから遠ざける必要があります。以下の指針を参考にワイヤとバス・バーのどちらを使用するか判断してください。

個別終端ワイヤ:

- 個々の機器に接続する場合に適しています。全電流搬送要件は120A以下です。
- 整列、絶縁、またはルーティング上の問題がほとんどありません。
- 小型の電池充電システムに最適です。

バス・バー:

- 大電流搬送要件に適しています。
- カスタム設計やカスタム購入が可能です。標準的な大電流機能部品を使用できます。
- 接続に、ナットやボルト、またはタッピン穴を使用します。
- 表面を注意深く整え、接続ポイントを清掃する必要があります。

警告: **危険です。大電流パワーバス接続が互いに接触した場合には、相当なアークが発生し、部品が焼けたり、発火したり、溶接される恐れがあります。パワーバスに電気が流れている場合、パワーバスへの接続は行わないでください。**

パワーバスの配線方法

次ページの表に、パワーバス接続に適した数種類の標準線経の抵抗と電流容量を示します。パワーバス配線の抵抗はパワーバス配線で電圧降下を引き起こすため、この情報は重要です。電圧降下が大きいと、Agilent E4370A MCCDメインフレームが充電モードで正しく機能しなくなったり、Agilent E4371Aパワーバス負荷が放電モードで正しく機能しなくなる恐れがあります。

表2-6. 銅より線の電流容量および抵抗

AWG番号	領域 (mm^2)	電流容量	抵抗 (Ω/m)	抵抗 (Ω/ft)	注記
10	5.26	40	0.00327	0.00099	1. ワイヤの電流は30℃の周囲温度に基づき、導線の定格は60℃におけるものです。 2. 抵抗は、20℃のワイヤ温度における公称値容量です。
8	8.36	60	0.00206	0.00062	
6	13.3	80	0.00129	0.00039	
4	21.1	105	0.00081	0.00025	
2	33.6	140	0.00051	0.000156	
1/0	53.5	195	0.00032	0.000098	
2/0	67.4	225	0.00025	0.000078	
3/0	85.0	260	0.00020	0.000062	
4/0	107	300	0.00016	0.000049	

図2-1は、1台のAgilent E4371Aパワーバス負荷と2台の外部DC電源に接続された2台のAgilent E4370A MCCDメインフレームから構成される、2種類の代表的なパワーバス構成を示したものです。図からもわかるように、電流要件は機器とパワーバスの接続方式によって大きく異なります。

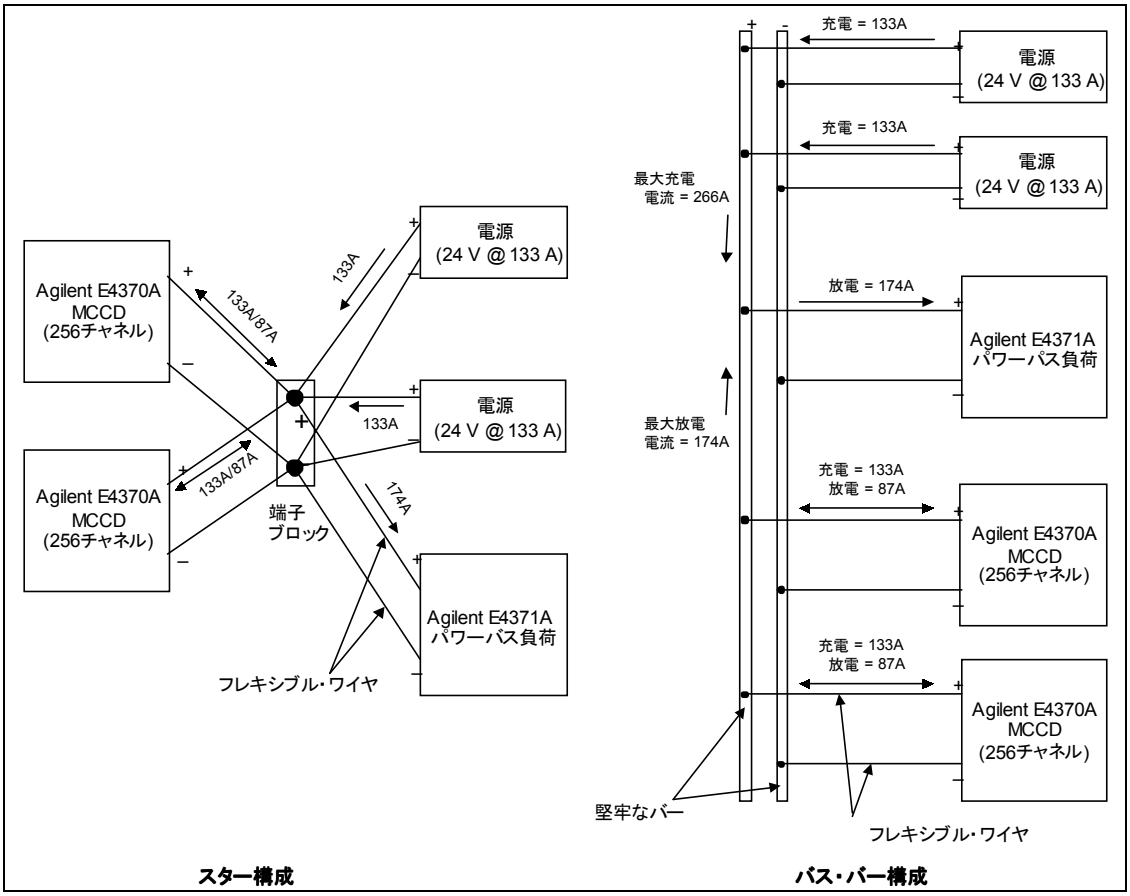


図2-1. 代表的なパワーバス構成

左側のスター構成は、パワーバスの各セクションに、接続されている機器の定格以上の電流が流れないように設計されています。この構成では、各リード線の電圧降下がリード線を流れる電流の量と直接関係しているため、より長いリード線を使用できます。ただし、この場合、各Agilent MCCDメインフレームと負荷および電源を個別のリード線で結ぶ必要があります。このため、必要な配線の量も増加します。

右側のバス・バー構成は、機器間の配線の量を最小限に抑えるように設計されていますが、太いワイヤ線またはバス・バーが必要です。これは、電源からのリード線と負荷までのリード線で、2台のAgilent E4370A MCCDメインフレームの全充放電電流を伝送する必要があるからです。電流量が多ければ多いほど配線の電圧降下が大きくなるので、長いリード線を使用できないことが分かります。

充電モードに関する指針:

パワーバス配線は、パワーバスに接続されているすべてのAgilent E4370A MCCD機器の全充電電流要件に適合していなければなりません。以下に示す例では、ワーストケースの電流要件を求めます。1台のフル装備されたAgilent E4370A MCCDの入力電流要件の計算方法は以下のとおりです。

1個の電池の消費電力にAgilent MCCD内の電池の個数を掛け、その結果を機器効率で割って、メインフレームに必要な全入力パワーを求めます。充電モードにおける機器効率は80%と仮定します。これは、メインフレームが必要とする全パワーの計算に関するワーストケース値です。

$$\frac{\text{電池の個数} \times \text{1個の電池当たりのパワー}}{0.8} = \text{最大パワー入力}$$

1. Agilent MCCDの入力パワー要件をAgilent MCCDの入力端子の最小必要電圧で割ります。その結果が、Agilent MCCDで必要な最大充電電流となります(図2-1に示すように、2台のAgilent MCCDメインフレームを同時に充電する場合、この電流を2倍します)。

$$\frac{\text{最大パワー入力}}{\text{電源電圧}} = \text{最大パワーバス電流}$$

2. 表2-6の抵抗値を用い、最大電流によるパワーバス・リード線の電圧降下を求めます。
3. この電圧降下をAgilent MCCDの入力端子で必要な最小電圧に加算して、DC電源の出力電圧設定を決めます。
4. 充電モード中のAgilent MCCDの入力端子電圧は、25.2 Vと22.8Vの間でなければなりません。＋と－の両方のパワーバス・リード線で発生した電圧降下によって、メインフレームのパワー端子電圧が22.8Vより低下した場合、電圧不足のためにAgilent E4370A MCCDはシャットダウンします。電圧降下を小さくするには、より太いワイヤを使用します。

放電モードに関する指針:

パワーバス配線は、パワーバスに接続されているすべてのAgilent E4370A MCCD機器の全放電電流要件に適合していなければなりません。以下に示す例でも、ワーストケースの電流要件を求めます。1台のフル装備されたAgilent E4370A MCCDの出力電流の計算方法は以下のとおりです。

1. 1個の電池が生成する電力にAgilent MCCD内の電池の個数を掛け、その結果を機器効率で割って、メインフレームが生成する全出力パワーを求めます。放電モードにおける機器効率は100%と仮定します。これは、メインフレームが生成する全パワーの計算に関するワーストケース値です。

$$\frac{\text{電池の個数}\times\text{1個の電池当たりのパワー}}{1.0} = \text{最大パワー出力}$$

2. Agilent MCCDが生成する電力をAgilentパワーバス負荷の入力電圧で割ります。26.5Vの入力電圧では、その結果が、Agilentパワーバス負荷が吸収する最大放電電流となります(図2-5に示すように、2台のAgilent MCCDメインフレームを同時に放電する場合、この電流を2倍します)。

$$\frac{\text{最大パワー出力}}{26.5} = \text{最大パワーバス電流}$$

3. 表2-6の抵抗値を用い、最大電流によるパワーバス・リード線の電圧降下を求めます。
4. **＋と－の両方のパワーバス・リード線の電圧降下の合計が1. 5Vを超えてはなりません。**放電モードにおける電圧降下が1.5Vを超えた場合、メインフレーム端子で過電圧状態となり、Agilent MCCDはシャットダウンします。電圧降下を小さくするには、より太いワイヤを使用します。

ディジタル接続

各Agilent E4370A MCCDメインフレームには、16ビット・ディジタルI/Oポートが1つあります。ディジタルI/Oの設定には、Agilent MCCD設定画面 (第3章を参照) またはAgilent MCCDユーザ・インタフェース (第4章を参照) を使います。すべてのピンを同じ設定にする必要はありません。あるピンを絶縁出力に、別のピンをシングルエンド入出力に設定することもできます。機能を混在させることも可能であり、あるピンを汎用I/Oに、別のピンを特定目的に割り当てることもできます。ビットの極性を正論理または負論理に設定することも可能です。ディジタルI/O設定の種類については、以下のリストを参照してください。

汎用I/O

汎用I/Oでは、ディジタルI/Oがパススルー機能としてプログラムされます。これは、ディジタル・コネクタ上の入出力信号をAPIプログラミング・コマンドから直接制御することを可能にします。これらの信号は電池フォーミング・シーケンスに何の影響も与えません。

ディジタル出力 出力として設定されたラインはそれぞれ、内蔵のオープン・コレクタ・トランジスタによってドライブされます。出力ラインは、TTL互換入力、またはソレノイド、インジケータ・ライト、リレーなどのハイパワー負荷をドライブできます。これらは24V/300mA互換オープン・コレクタ出力です。

ディジタル入力 入力として設定されたラインはそれぞれ、外部ソースによってドライブできます。ラインはすべて、TTL互換入力であり、接点およびスイッチ・クロージャ型入力を容易にするため、5Vへのプルアップを内蔵しています。

ディジタル入出力 入出力として設定されたラインはそれぞれ、入力と出力の両方として使用できます。ライン・ハイをプログラミングした場合、外部デバイスによってラインをドライブできます。ライン・ローをプログラミングした場合、ラインがローにドライブされます。ラインを読み取ると、ラインの実際のステートが返されます。

絶縁出力 出力を光学絶縁モードに設定すると、0.4Vで1.6mAのシンク電流が可能なオープン・コレクタ出力になります。24Vまで使用できます。ピン0から始まる隣接ピン・ペアが、光アイソレータの＋出力と－出力にあたります。0-1、2-3、4-5などの隣接ピン・ペアで、最高8つの絶縁出力が可能です。これらは専用ペアなので、ピン1と2を絶縁出力として結合することはできません。

特殊機能

外部不良入力 この信号が真の場合、電池フォーミング・シーケンスは外部不良条件により停止します。さらに、外部不良出力信号が真になります。この信号は、火災探知器などのセンサに接続できます。また、別のメインフレームの故障に応答できるように、別のAgilent MCCDの外部不良出力に接続することも可能です。

外部不良出力 この信号が真になるのは、外部不良が発生したときです。この信号は、火災報知器などの外部機器に接続できます。また、他のメインフレームの外部不良入力に接続して、1つのメインフレームが故障したときに他のメインフレームをシャットダウンさせることも可能です。この信号は、cfProtectClearコマンドによってクリアします。

外部インターロック この信号が真の場合、電池フォーミング・シーケンスは停止しますが、不良条件による停止ではないので、外部不良出力信号は真にはなりません。このレベル依存型の信号は、外部の停止または休止スイッチに接続できるので、オペレータまたはメカニカル・デバイスが電池フォーミング・シーケンスを停止できます。この信号がオフになると、シーケンスは続行されます。

外部トリガ この外部トリガ入力は、電池フォーミング・シーケンスを起動するために用いられます。

停電 この信号が真の場合、システムの設定に応じ、入力信号によってAgilent MCCD がシャットダウンを実行します。このとき、Agilent MCCDはリスタートに備えてステートを保存します。シャットダウン・ステートが保存されているときには、それを示す停電出力信号も得られます。

配線に関する指針

16本のディジタルI/O信号の接続には、2個の10ピンPhoenix/Weidmuller型コネクタを使用します。これらのコネクタは、ワイヤ接続用のねじ込み端子付きで、取外しが可能です。ディジタルI/Oライン0～7は、コネクタ1にあります。ディジタルI/Oライン8～15は、コネクタ2にあります。各コネクタには、2個のグランド接続またはリターン接続用の共通端子があります。

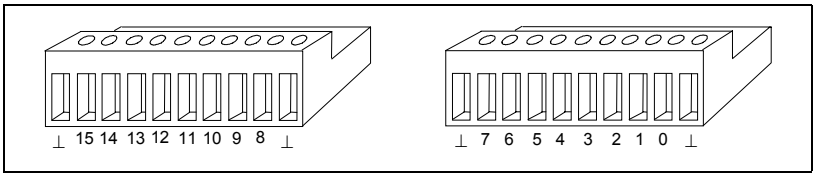


図2-2. ディジタルI/O接続

次ページの図は、ディジタルI/Oコネクタの内部回路を示したものです。ディジタル入力 (A) として使用する場合、ディジタル入力ピンに接続される外部回路には、TTL、AS、CMOS、HC、またはディジタル入力と共通端子を接続する簡単なスイッチがあります。

ディジタル出力 (B) として使用する場合、出力ピンに接続される外部回路には、TTL、AS、CMOS、HC、または警告ライト (ライトが外部バイアス電源を持つ場合) があります。

絶縁ペア(C) として使用する場合、各ピン・ペアを外部回路に接続することができますが、一緒に使用できるのは隣接ペアだけです。各ペアの偶数ピンだけ、プログラミングによってロジック・レベルを負論理または正論理に設定できます。負論理をプログラムした場合、ピンが短絡すると出力が真になります。正論理をプログラムした場合、ピンがオープンすると出力が真になります。

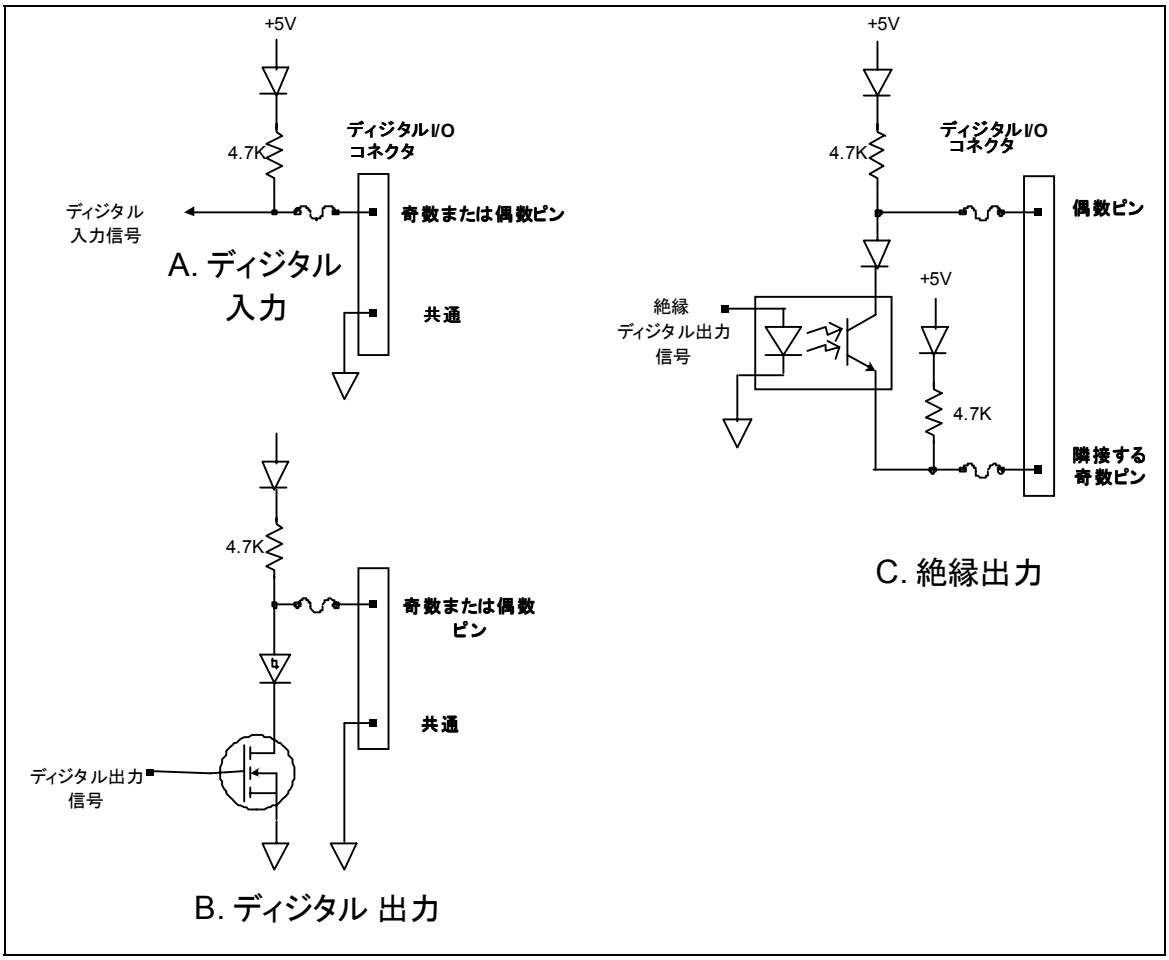


図2-3. 等価ディジタルI/O回路

次の図は、代表的なDIO ハードウェア接続を示したものです。

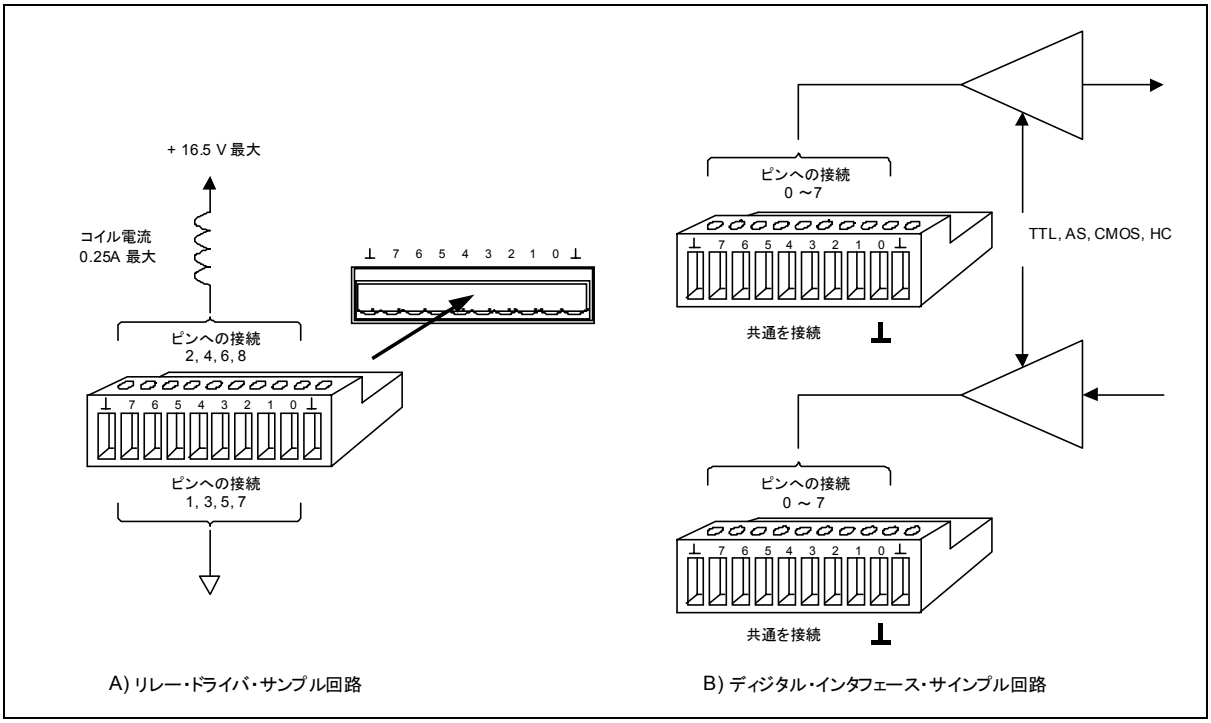


図2-4. 代表的なハードウェア接続

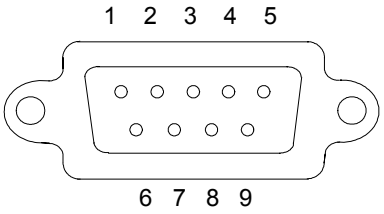
RS-232接続

Agilent MCCDには2個のRS-232ポートがあり、ローカル周辺機器の接続に利用できます。通常の動作では、2個のポートは両方とも汎用の通信に使用でき、LAN経由での設定が可能です。初期設定と構成の際には、RS-232ポートは次のように使用されます。

初期設定 RS-232ポートBが端末に接続され、Agilent MCCD設定画面の表示に用いられます。これによって、初期設定が実行されます。

校正 RS-232ポートBが端末に接続され、Agilent MCCD設定画面の表示に用いられます。この画面から校正プロセスを実行します。RS-232ポートAは、校正用電圧計の接続に用いられます。

どちらのポートも、コンピュータと同じボーレートに設定します。両ポートは、XON/XOFFサポートと、ハードウェア制御用のRTS/CTSにより、完全なハードウェア・フロー制御およびソフトウェア・フロー制御をサポートします。



DB-9オス型コネクタ

ピン	入力/出力	説明
1		接続なし
2	入力	Receive Data (RxD)
3	出力	Transmit Data (TxD)
4		未使用
5	共通	信号グランド
6		未使用
7	出力	Request to Send (RTS)
8	入力	Clear to Send (CTS)
9		接続なし

図2-5. RS-232 AおよびBコネクタ

下図は、Agilent MCCD RS-232ポートと、PCやバーコード・スキャナなどのローカル周辺機器との間のケーブル接続を示したものです。ケーブル・キットについては、表2-2を参照してください。

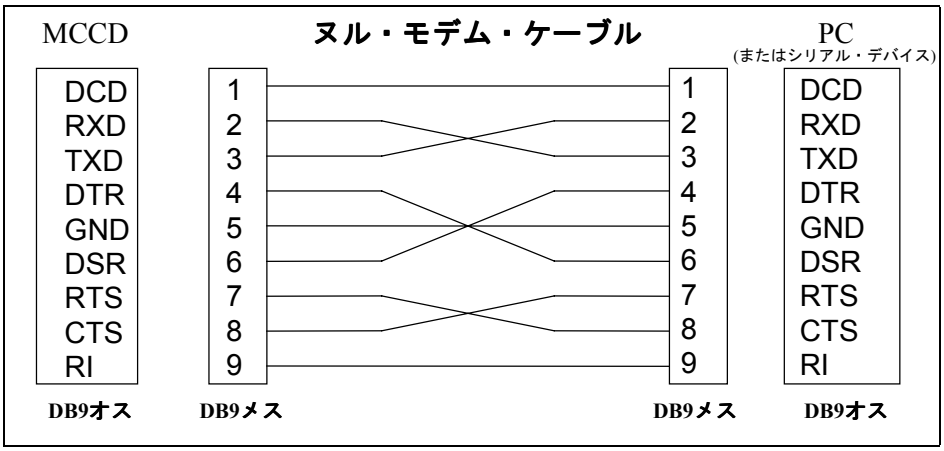


図2-6. ヌル・モデム・ケーブル接続

補助出力接続

汎用の絶縁補助出力が用意されており、テスト・フィクスチャ上の各種アクチュエータおよび回路類へのパワー供給に使用できます。また、外部プルアップ・ソースを必要とするディジタルI/O接続の、プルアップ・ソースとして用いることも可能です。補助出力には、裏面パネルの4ピンPhoenix/Weidmuller型コネクタを使用します。これらのコネクタは、ワイヤ接続用のねじ込み端子付きで、取り外しが可能です。

Agilent MCCD設定画面 (第3章を参照) で、この出力を5V～24Vの範囲に0.1V刻みで設定できます。最大出力パワーは10Wです。補助出力は、シャーシ・コモン (グランド) に対して最高42Vまで絶縁されています。

APIライブラリおよび測定ログ・ユーティリティのインストール

Agilent MCCD用のソフトウェアは、APIライブラリと測定ユーティリティから構成されています。このソフトウェアは、Agilent E4373Aマニュアル・パッケージに付属しています。付属のC言語関数呼び出しを使ってAgilent MCCDを自動製造プロセスの一環として制御するためには、このプログラムをインストールする必要があります。

ディスクに収録されたセットアップ・プログラムを使って、クライアントAPIファイルと測定ユーティリティをPCにインストールします。セットアップ・プログラムによって、新しいディレクトリと新しいWindowsプログラム・グループが作成されます。ソフトウェアをインストールには、以下の手順を実行します。

1. ディスクNO.1をコンピュータのA:ドライブに挿入し、A:SETUP.EXTを実行します。
2. 画面上の指示に従って、ソフトウェアをインストールします。デフォルトのインストール選択では、コンピュータのC:\hpmccdディレクトリに全ファイルがインストールされます。Agilent MCCDクライアントAPIと測定ログ・ユーティリティ・プログラム・フォルダも作成されます。これらのデフォルト選択はどちらも変更可能です。

アップデートについては、\hpmccdディレクトリにあるREADME.TXTファイルをご覧ください。\hpmccdディレクトリには、以下のファイルも含まれています。

mccdcfg.exe	Windows 95またはWindows NTクライアントPCから、Agilent MCCDファームウェアをアップデートできます。
mccdlog.exe	Agilent MCCDのデータ・ログ・メモリからWindows 95またはWindows NTクライアントPCにデータを転送できます。
mccd.dll **mccd.lib** **mccd.h**	Cプログラミング・クライアント・アプリケーションを開発するのに必要なファイルです。

C言語で書かれた2つのサンプル・プログラムもPCのC:\hpmccd\c\samplesにインストールされます。

Agilent MCCD測定ログ・ユーティリティの使用法については、第4章を参照してください。

Visual C++構成

Agilent MCCD APIライブラリを使ってアプリケーションを構築するには、Visual C++開発環境を次のように構成する必要があります。

1. mccd.dll、mccd.lib、およびmccd.hファイルを作業用プロジェクト・ディレクトリにコピーします(これは、前述のようにライブラリをインストールした時点でも実行可能です)。
2. Visual C++ (4.x) で、mccd.libライブラリを追加する必要があります。そのためには、BuildメニューでSettingsを選択してから、Object Modulesを選択して、mccd.libファイルを追加します。
3. アプリケーションでは、このライブラリの呼び出しを含むすべてのファイルの先頭に、mccd.hヘッダ・ファイルを定義する必要があります。ソース・ファイルでこれを行うには、#include "mccd.h" と入力します。

3

設定

LANの設定

LANへの接続には、裏面パネルの標準8ピン10Base-Tコネクタを使います。本項の指示に従って、LANの設定を行ってください。LANの設定を行うには、次の3つのステップを実行します。

1. Agilent E4370A MCCDと通信するために、PCのHyperTerminalプログラムを設定します。HyperTerminalプログラムは、Windows 95およびWindows NTに付属しています。他のASCII端末やターミナル・エミュレーション・プログラムも、HyperTerminalプログラムの場合と同じように設定すれば機能します。
2. 表2-2および図2-5に示すように、ヌル・モデム・シリアル・ケーブルを用い、PCをAgilent E4370A MCCDの裏面のRS-232ポートBコネクタに接続します。
3. Agilent MCCD設定画面に必要事項を入力します。

注記: この手順は、Agilent MCCDの校正、ディジタルI/Oの設定(LAN経由でイネーブルにしていない場合)、および言語オプションの設定にも使用します。

1. HyperTerminalプログラムの設定

以下のようにHyperTerminalプログラムを起動して、設定を行います。

Windows NTの場合:	**Start**ボタンを押し、次の項目を選択します: **Programs> Accessories> Hyperterminal> HyperTerminal**
Windows 95の場合:	**Start**ボタンを押し、次の項目を選択します: **Programs> Accessories> Hyperterminal** 次に、HyperTerminalプログラム・グループのHypertrm.exeアイコンをダブルクリックします。HyperTerminalを初めて実行する場合は、場所に関する情報を入力します。モデムは使用しないので、電話番号を入力する必要はありません。
Connection Descriptionsボックス:	名前を入力し、必要であればアイコンを選択します。 次に、OKをクリックします。
Connect Toボックス (Windows NT) **または Phone Numberボックス** (Windows 95):	**Connect using**フィールドでCOM1またはCOM2を選択することにより、コンピュータ裏面のCOMコネクタを指定します。このフィールドで選択したCOMポートに、Agilent E4370A MCCD裏面のRS-232ポートBを接続します。次に、OKをクリックします。

COM Properties ボックス:	以下のポート設定を選択します。 Bits per second 9600 Data bits 8 Parity None Stop Bits 1 Flow control None 済んだらOKをクリックします。
Fileメニュー	Propertiesコマンドを選択します。
Propertiesボックス:	Settingsタブを選択します。Emulationの下の**Auto detect**が選択されていることを確認します。**ASCII Setup**ボタンをクリックし、以下の項目を選択します。 Send line ends with line feeds チェックなし Echo Typed Characters locally チェックあり Line delay 0 Character delay 0 Append line feeds to incoming cone ends チェックなし Force incoming data to 7-bit ASCII チェックなし Wrap lines that exceed terminal width チェックなし OKをクリックしてASCII Setupを終了します。
	OKをクリックしてPropertiesを終了します。

2. Agilent E4370A MCCDをPCのCOMポートに接続

Agilent MCCD裏面のLINEコネクタにAC電源コードを接続します。これは、87Vac～250Vacの範囲の電源電圧、50/60Hzをサポートする汎用AC入力です。

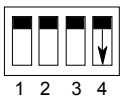
Agilent MCCDの電源をオンにします。

Agilent E4370A MCCDをPCのCOMポートに接続します。Agilent E4370A MCCD裏面のポートBと、HyperTerminalプログラムで指定したコンピュータのCOMポートを、RS-232ケーブルで接続します。

Agilent E4370A裏面のポートBスイッチ (No.4) を上から下 (NormalからConfigure) に切り替えます。

Normal

Configure



注記: スイッチ1～3は常に上にしておいてください。

注記: スイッチ4が下になっている時は、Agilent MCCD設定プログラムはアクティブ状態にあります。Agilent MCCD設定画面がPC上に表示されない場合は、Enterキーを押してください。

---

3. Agilent MCCD設定画面への入力

次ページのようなAgilent MCCD設定画面がHyperTerminalウィンドウに表示されます。

```
MCCD Configuration Screens


1)   Network Configuration

2)   Identification Configurations

3)   Digital I/O Configuration

4)   Perform Calibration

5)   Miscellaneous Configuration









Type a number and press Enter
```

この時点では、**Network Configuration** (ネットワーク設定) と**Identification Configuration** (識別設定) の2つの画面だけを使用します。

ネットワーク設定

注記: この画面で入力する設定の内容は、ネットワーク管理者が決定します。

初期画面で1を選択して、ネットワークを設定します。1、2、3、4のどれかを選択し、IPアドレス、サブネット・マスク、デフォルト・ゲートウェイ、またはパスワードを入力します。

```
Network Configuration

IP Address                  000.000.000.000
Subnet Mask                 000.000.000.000
Default Gateway             000.000.000.000
Network Password

1)   To change IP Address

2)   To change Subnet Mask

3)   To change Default Gateway

4)   To change Password









Type a number and press Enter or ctrl-G to return to initial screen
```

IP Address (IPアドレス) は、Agilent E4370A MCCDのIPアドレスを設定します。ドット区切り10進表記法を使用します。

Subnet Mask(サブネット・マスク)は、Agilent E4370A MCCDが接続されているサブネットワークを表します。サブネット・マスクを0に設定すると、このネットワークではサブネットワークを使用しないことを表します。

Default Gateway(デフォルト・ゲートウェイ)は、Agilent E4370A MCCD用のゲートウェイのIPアドレスです。これにより、ユニットを接続したコンピュータとは別のコンピュータとLAN経由で通信できるようになります。

Agilent MCCDのパスワードは、出荷時には設定されていません。Agilent E4370A MCCDにネットワーク・パスワードを割り当てれば、権限のないユーザがネットワーク経由でユニットを制御するのを防止できます。このパスワードは、API関数を使ってAgilent MCCDのプログラミングを行うときに使用する必要があります。

識別設定

初期画面で2を選択して、Agilent MCCDの識別設定に入ります。この画面の設定は、識別のためだけに用いられます。これはLAN上に複数のAgilent MCCDが存在する場合に特に有用です。Identification Configuration画面に以下のように入力します。

1、2、3のどれかを選択して、ユニット名、場所、その他のユニット識別情報を入力します。

```
Identification Configuration

Unit Name              MyName
Unit Location          Third on left
Other ID               Identification000.000


1)   To change Unit Name

2)   To change Unit Location

3)   To change Other ID









Type a number and press Enter or ctrl-G to return to initial screen
```

Unit Name(ユニット名)は、Agilent MCCDに割り当てるネットワーク名です。ユニット名の最初は英字で、最後は英字または数字です。途中の文字は、英字、数字、ピリオド、またはハイフンです。

Unit Location(ユニット位置)は、Agilent MCCDの物理的位置を示します。印字可能なASCII文字だけが使用できます。

Other ID(その他の識別情報)は、このAgilent MCCDの識別に必要なその他の情報です。印字可能なASCII文字だけが使用できます。例としては、資産番号、部門名や生産ラインなどがあります。

その他の設定

初期画面で5を選択し、Agilent MCCDユーザ・インタフェースで使用する言語を設定します。英語または日本語を選択できます。

この画面では、Agilent MCCDメインフレーム裏面の補助バイアス出力もプログラムできます。バイアス電圧は、5V～24Vの範囲内に0.1V刻みでプログラム可能です。

```
Miscellaneous Configuration

Web Page Language is presently ENGLISH

1)   To Change Web Page language to English

2)   To Change Web page language to Japanese


AUX Bias Supply voltage is 10.0000

3)   Set Aux Bias Supply voltage









Type a number and press Enter or ctrl-G to return to initial screen
```

ディジタルI/Oの設定

ここでは、Hyperterminalを使ったディジタルI/Oの設定について説明します。LAN経由でAgilent MCCDユーザ・インタフェースまたはAPI関数呼び出しを使ってディジタルI/Oを設定することも可能です。ディジタルI/Oラインの機能の詳細については、第6章の"cfSetDigitalConfig"を参照してください。

PCでHyperTerminalプログラムを使ってディジタルI/Oを設定するには、本章の初めで説明したように、Agilent E4370A裏面のポートBスイッチ(No.4)を上から下(NormalからConfigure)に切り替え、HyperTerminalプログラムを実行します。Agilent MCCD設定画面が表示されたら、3を選択してAgilent MCCDメインフレームのディジタルI/Oポートを設定します。

1または2を選択して、LAN経由のディジタルI/O設定をイネーブルまたはディスエーブルにします。ディジタルI/O設定をイネーブルにすると、Agilent MCCD設定画面を起動しなくても、Webを利用したAgilent MCCDユーザ・インタフェースまたはAPIによって、ディジタルI/Oの設定がいつでも可能になります。

注記: ディジタルI/O機能を電池フォーミング・プロシージャの安全機能のモニタまたは実現に使用している場合、LAN経由のディジタルI/Oアクセスをディスエーブルにしておくと、ディジタルI/O機能が誤って再プログラミングされるのを防止できます。

```
Digital I/O Configuration

Digital I/O Configuration over the LAN is    ENABLED

1)   To ENABLE Digital I/O Configuration over the LAN

2)   To DISABLE Digital I/O Configuration over the LAN

3)   To configure Digital I/O









Type a number and press Enter or ctrl-G to return to initial screen
```

ディジタルI/Oの設定を続行するには、3を押します。ディジタルI/Oコネクタのピン番号が画面上に表示されます。ピンの物理的な位置については、図2-2を参照してください。コネクタ両端にある2本のピンは、グランド付き出力として設定されているすべてのピンに共通の接続です。

```
Digital I/O Configuration

Pin,     Function
0        General Purpose I/O,       Grounded, High True
1        General Purpose I/O,       Grounded, High True
2        General Purpose I/O,       Grounded, High True
3        General Purpose I/O,       Grounded, High True
4        General Purpose I/O,       Grounded, High True
5        General Purpose I/O,       Grounded, High True
6        General Purpose I/O,       Grounded, High True
7        General Purpose I/O,       Grounded, High True
8        General Purpose I/O,       Grounded, High True
9        General Purpose I/O,       Grounded, High True
10       General Purpose I/O,       Grounded, High True
11       General Purpose I/O,       Grounded, High True
12       General Purpose I/O,       Grounded, High True
13       General Purpose I/O,       Grounded, High True
14       General Purpose I/O,       Grounded, High True
15       General Purpose I/O,       Grounded, High True


Type a pin number and press Enter or ctrl-G to return to initial screen
```

ピンを設定するには、ピン番号を選択してから、Enterを押します。選択したピンごとに、次ページの選択肢が画面上に表示されます。この画面で行った選択は、前の画面に反映されます。

```
Pin  0   Digital I/O Configuration

1)   Change to External Fault Input      Grounded
2)   Change to External Fault Output     Grounded
3)   Change to External Interlock        Grounded
4)   Change to General Purpose I/O       Grounded
5)   Change to General Purpose Input     Grounded
6)   Change to General Purpose Output    Grounded
7)   Change to External Trigger          Grounded
8)   Change to External Fault Output     Isolated
9)   Change to General Purpose Output    Isolated
10)  Change to Power Fail Input          Grounded
11)  Change to Power Fail Output         Grounded
12)  Change to Power Fail Output         Isolated
13)  Change to Output and not Fault In   Grounded
14)  Change to Output and not Fault In   Isolated


Type a number and press Enter or ctrl-G to return to initial screen
```

すべてのピンを同じ設定にする必要はありません。あるピンを絶縁出力に、別のピンをシングルエンド入出力に設定することもできます。機能を混在させることも可能であり、あるピンを汎用ディジタルI/Oに、別のピンを特定目的に割り当てることもできます。ディジタルI/O信号の目的や用途については、第2章の「ディジタル接続」で詳しく説明しています。1から7までの選択肢は、どのピンにも設定可能です。これらの選択肢については、共通グランド・ピンがリターンとして用いられます。

10と11の選択肢は停電信号です。1つは入力で、停電が発生したことを知らせます。もう1つは出力で、シャットダウン・ステートが保存されていることを示します。ピン13は特殊な目的の信号であり、第5章のcfSetDigitalConfig に説明があります。

8、9、12および14の選択肢は絶縁出力であり、2本1組のピンを割り当てる必要があります。ピンの組は、隣接している必要があり (0-1、2-3、4-5など)、最大8個の組が使用可能です。たとえば、0-1の組を絶縁出力として使う場合、ピン0を絶縁出力に設定し、ピン1は設定しません。ピン1は絶縁出力のマイナス・コネクタになります。ピン1に対する読み書きは無効です。絶縁出力を入力として使うことはできません。

Enterを押してから、ピンを正論理に設定するか、負論理に設定するかを選択します。

```
Pin  0   Digital I/O Configuration

1)   Change to High True

2)   Change to Low true









Type a number and press Enter or ctrl-G to return to initial screen
```

混合設定の例

以下に示すのは、混合ディジタルI/O設定の例です。この例は次のように設定されています。

- ピン0、2、4、6は外部不良絶縁出力、正論理(選択肢11)に設定されています。
- ピン1、3、5、7は絶縁出力の2本目のピンです。
- ピン8～10は、汎用I/O、正論理(選択肢7)、共通端子基準に設定されています。
- ピン11および12は、外部不良入力、負論理(選択肢2)、共通端子基準に設定されています。
- ピン14および15は、外部不良出力、正論理(選択肢3)、共通端子基準に設定されています。

```
Digital I/O Configuration

Pin,     Function                                Polarity
0        External Fault Output,        Isolated, High True
1        second pin of isolated pair
2        External Fault Output,        Isolated, Low True
3        second pin of isolated pair
4        General Purpose Output,       Isolated, High True
5        second pin of isolated pair
6        General Purpose Output,       Isolated, Low True
7        second pin of isolated pair
8        General Purpose I/O,          Grounded, High True
9        General Purpose I/O,          Grounded, High True
10       General Purpose I/O,          Grounded, High True
11       General Purpose I/O,          Grounded, High True
12       External Fault Input          Grounded, Low True
13       External Fault Input          Grounded, Low True
14       External Fault Output         Grounded, High True
15       External Fault Output         Grounded, High True



Type a pin number and press Enter or ctrl-G to return to initial screen
```

校正の実行

便宜上、Agilent MCCD設定画面を使ったAgilent MCCDの校正について説明します。LAN経由で、Agilent MCCDユーザ・インタフェースまたはAPI関数呼び出しを使って校正を実行することも可能です。Agilent MCCD設定画面を使った校正方法の詳細については、付録Bを参照してください。

4

Agilent MCCDユーザ・インタフェース

概要

Agilent MCCDユーザ・インタフェースにより、Agilent E4370A/E4374A MCCDシステムを対話的にモニタしたり制御することができます。LAN上のどこに配置されたPCからでも、標準のWebブラウザを使ってこのインタフェースにアクセスできます。このインタフェースを使用するために、Webブラウザ以外に特別なソフトウェアをPCにインストールする必要はありません。

PCの要件

PCには、以下のWebブラウザのいずれかをインストールしなければなりません。

- Netescape Navigator 3.03以降
- Microsoft Internet Explorer 3.02以降

ブラウザの設定

Agilent MCCDユーザ・インタフェースを使用する場合、以下のブラウザ設定を用いることをお勧めします。

- 256色を表示するように設定されたグラフィック・カード
- 800×600ピクセル以上のビデオ解像度
- Netscape Navigatorを使用しているときは、View/Network Preferences/Languageを表示し、Javascriptをイネーブルにします。
- Microsoft Internet Explorerを使用しているときは、View/Options/Securityを表示し、Run ActiveXスクリプトをイネーブルにします。

注記: 標準のWebブラウザ・ボタン (Back、Forward、Home) を使ってAgilent MCCDユーザ・インタフェース内を移動すると、予期しない結果になる場合があります。

セキュリティ

Agilent MCCDの内蔵サーバには、Webブラウザでサポートされる基本的なパスワード認証スキームが実装されています。出荷時には、Agilent MCCDユーザ・インタフェースにパスワードは設定されていません。Agilent MCCD設定画面を使ってパスワードを設定すれば、Agilent MCCDへのアクセスを制限することができます。第3章で説明したように、パスワードの設定はインストール中に行います。Agilent MCCD設定画面で設定したパスワードが、Agilent MCCDユーザ・インタフェースによっても使用されます。API関数呼び出しを使ってAgilent MCCDをプログラミングする場合にも、このパスワードを使用する必要があります。

ローカライズ

ユーザ・インタフェース・ページは、英語と日本語で提供されます。Agilent MCCDのインストール中に、デフォルトの言語を指定することができます(Agilent MCCDユーザ・インタフェースの起動後は、Systemページから言語を変更することも可能です)。

アクセス

ユーザ・インタフェースにアクセスするには、LANに接続されたPCからWebブラウザを起動し、以下のURLを指定します。

```
http://<address>/
```

ここで、<address>はモニタしている特定のAgilent MCCDのIPアドレスまたは名前です。

インタフェースの使用法

Agilent MCCDユーザ・インタフェースには、基本的なシステムのモニタおよび制御機能があります。このインタフェースを用いれば、個々の電池ステートのモニタ、テスト実行中の電池の電圧と電流の測定、完全なテスト・シーケンスのモニタおよび制御が可能です。

Agilent MCCDユーザ・インタフェースによって、コンピュータ・テスト・プログラムとは関係なく、電池フォーミング・ステーションを制御できます。Agilent MCCDユーザ・インタフェースを使ってプログラムした電池フォーミング・シーケンスは、プログラムにAPI関数を使用して構築したシーケンスと同じになります。既存のシーケンスをPCからダウンロードし、Agilent MCCDユーザ・インタフェースを使って表示したり修正することも可能です。

Agilent MCCDの各種機能に慣れるために、Agilent MCCDユーザ・インタフェースを学習用ツールとして用いることもできます。

最後に、Agilent MCCDユーザ・インタフェースのDiagnosticsページでは、個々のチャネル出力を直接、すぐにプログラムできます。これはデバッグ専用です。Agilentでは、直接出力制御機能を使って電池フォーミング・シーケンスを実行することはお勧めしません。

注意: **直接出力制御は電池の充電には使用しないでください。直接出力制御を用いた場合、過充電を防ぐことはできません。このモードは診断やデバッグにだけ使用してください。**

Agilent MCCDユーザ・インタフェースの使用法については、インタフェースからアクセス可能なオンライン・ヘルプに詳しい説明があります。Helpボタンをクリックしてください。

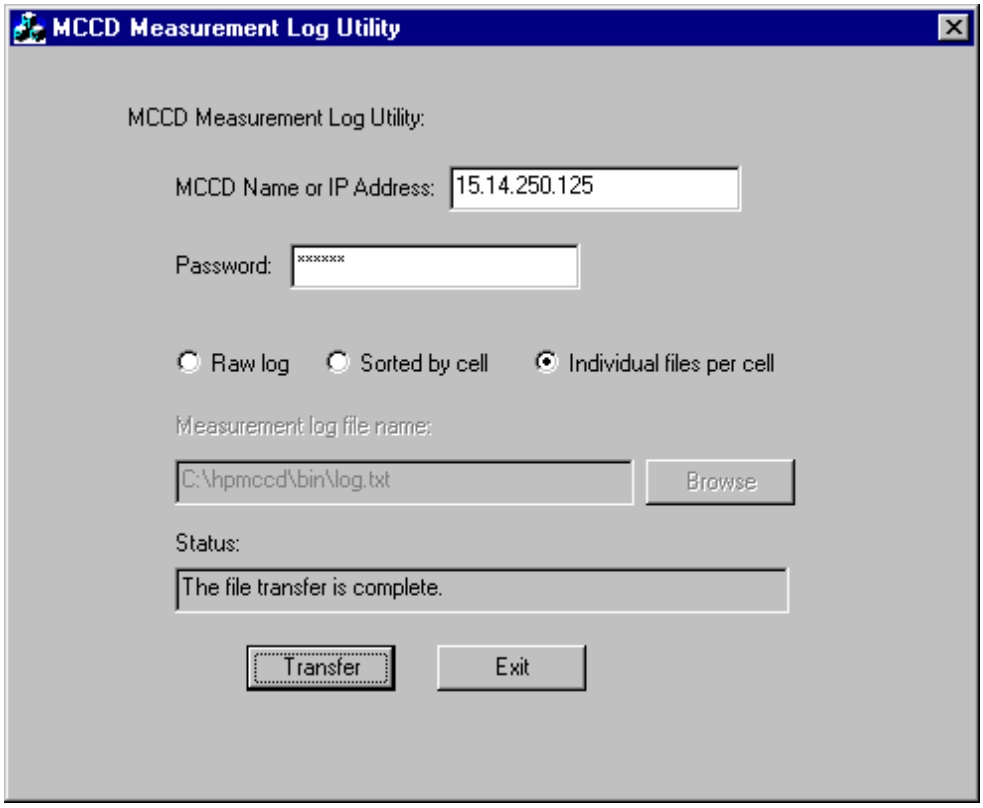
Agilent MCCD測定ログ・ユーティリティの使用法

Agilent MCCDユーザ・インタフェースを使って電池フォーミング・シーケンスを作成、実行する場合、データ・ログ・メモリのデータをPCに転送し、電池フォーミング・シーケンスの完了時に解析や保存を行うことができます。Agilent MCCD測定ユーティリティを使ってデータをデータ・メモリ・ログからクライアントPCのファイルに転送します。

注記: データ・ログ・メモリは、イニシエート関数を実行したとき、Agilent MCCDユーザ・インタフェースを終了したとき、またはユニットの電源をオフにしたときにクリアされます。データを保存したい場合は、データをPCに転送してください。

Agilent MCCD測定ログ・ユーティリティを実行するには、
Start> Programs> Agilent MCCD Client API and Measurement Log Utility> Measurement Log
をクリックします。

次のようなウィンドウがコンピュータ画面に表示されます。



測定ユーティリティを使用するには、以下の手順を実行します。

1. 最初のフィールドに、アクセスするユニットのMCCD名またはIPアドレスを入力します。
2. ユニットがパスワードでプロテクトされている場合、パスワード (Password) フィールドにパスワードを入力します。
3. 次のデータ・ログ・フォーマットのいずれかを選択します。

   Raw log ログされた順番でログ・データをすべて転送します。

   Sorted by cell 電池順に並べ替えられたログ・データをすべて転送します。データは、最初の電池から最後の電池まで順番に編成されます。

   Individual files per cell ログ・データをすべて転送し、電池ごとに個別のデータ・ファイルを作成します。

4. Raw logまたはSorted by cellを選択した場合、データを保存するファイル名を入力する必要があります。Browseボタンを選択して、ファイルを置くディレクトリを選択します。デフォルトのディレクトリはC:＼hpmccd＼binです。
5. Individual files per cellを選択した場合、ユーティリティが最大256個のデータ・ファイルを自動的に作成します(各アクティブ電池に1つ)。ファイル名はc001.txt～c256.txtです。ファイルはすべて、C:＼hpmccd＼bin＼dataディレクトリに入れられます。
6. Transferをクリックして、データ転送を開始します。ステータス(Status)フィールドに、転送に関するステータス情報が表示されます。
7. Exitをクリックして、ユーティリティを終了します。

測定ログ・ユーティリティによって作成されたデータ・ファイルには、次の情報が含まれます。

セル番号 1～256

ステップ番号 1～n。シーケンスの総ステップ数

時間 フォーミング・シーケンスがトリガされてから以降の時間 (秒)

ステータス 電池のステータスを示す値

値	ステータス
1	定電圧モード
2	定電流充電モード
4	定電流放電モード

入力タイプ 次のいずれか1つ:"Charge"(充電)、"Discharge"(放電)、"Rest"(休眠)、"ACR"、"DCR"

電圧の表示値 電池の電圧 (V) (充電、放電、休眠ステップの場合のみ)

電流の表示値 電池の電流 (A) (充電、放電、休眠ステップの場合のみ)

アンペア時 ステップ番号の初めからの累積アンペア時間 (充電、放電、休眠ステップの場合のみ)

ワット時 ステップ番号の初めからの累積ワット時間 (充電、放電、休眠ステップの場合のみ)

抵抗 ACまたはDC抵抗測定 (Ω) (ACRおよびDCRステップの場合のみ)

5

プログラミング概要

電池フォーミングの概要

Agilent E4370A MCCDの電池フォーミング・プロセスは、プロセスが完了するまで電池のグループに対して実行される、一連のステップや動作から構成されています。この電池フォーミング・プロセスを、本書ではシーケンスと呼びます。シーケンスは本質的に3つのステップから構成されています。電池の**充電**、**休眠**、および**放電**です。これらのステップは、プロセスに応じて、シーケンス中に任意の順序で何回でも実行できます。1つのステップから次のステップへの遷移は、ステップ内のテストによって制御されます。このテストには、測定基準を指定します。テストにおける測定の実行時間や、測定基準が満たされた場合に実行する動作を指定することができます。２つの追加ステップ、AC抵抗とDC抵抗が使用可能です。これらのステップは、電池のAC抵抗またはDC抵抗の測定に使用します。測定は、電池の充放電中には実行できません。

ステップには、電池に供給される電圧および電流スティミュラスと、スティミュラスが供給される時間の長さを定義します。ステップ内の**テスト**は、電池を測定し、測定制限を定義します。次に、測定結果を制限と比較し、比較結果に基づいて実行する動作を指定します。所定のステップで実行可能なテストの一覧については、第6章のcfSetSeqTest関数を参照してください。AC抵抗テストとDC抵抗テストは、それぞれ、AC抵抗ステップとDC抵抗ステップでのみ実行できます。

電池のテストは、ステップに指定された時間の前または後、あるいは特定の時間に1度だけ行われます。ステップの最初でスティミュラスを供給する前に電池をテストし、充電や放電を行っても安全か確かめることもできます。

テスト結果に基づき、電池は残りのテストを飛ばしてシーケンスの**次**のステップに進むか、**不良**のフラグがセットされてシーケンスから除かれます。図5-1を参照してください。電池がシーケンスから除かれると、その電池への出力はオフになり、その電池に対する以降のテストは実行されません。

注記: 個々の電池に対するシーケンスの制約はありません。すなわち電池は、他のすべての電池と独立して、電池フォーミング・シーケンス内の任意のポイントに存在できます。電池は、現在いる特定のステップに対して指定された定格で、充電、放電、または休眠することができます。AC抵抗測定とDC抵抗測定も、電池がACRまたはDCRステップに入ったときに、個々の電池で実行されます。電池への出力も、個別にオンまたはオフにできます。

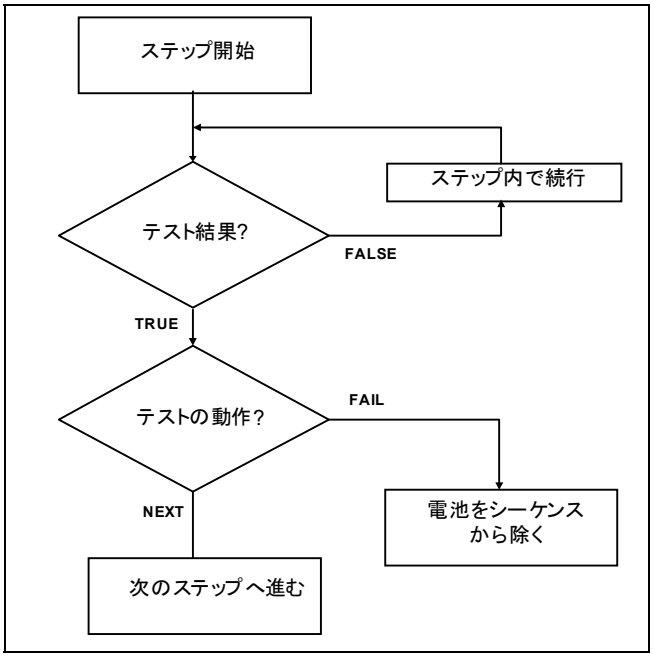


図5-1. テスト結果フローチャート

電池フォーミングの例

下表に、4つのステップから成るシーケンスを示します。図5-2は、シーケンスの実行中に3個の電池がどのように動作するかを示したものです。シーケンス内の各ステップは、すべての電池に対して同時に実行されます。シーケンスのステップと動作は次のとおりです。本例のCプログラミング・コードについては、第7章を参照してください。

関数	ステップ	ステップの動作/テスト・タイプ	電圧	電流	時間	テスト結果
Set Seq Step	1	で充電	4.2V	0.295A	20分間	
Set Seq Test	1	電圧≧	3.8V		5分前	FAIL (電池はシーケンスから除去される)
Set Seq Test	1	電流≦		0.02A	5分後	NEXT (電池はステップ2、休眠に進む)
Set Seq Step	2	休眠			10分間	
Set Seq Step	3	で放電	3.0V	0.295A	15分間	
Set Seq Test	3	電圧≦	3.0V		5分前	FAIL (電池はシーケンスから除去される)
Set Seq Test	3	電圧≦	3.0V		5分後	NEXT (電池はステップ4、休眠に進む)
Set Seq Test	3	電圧≧	3.0V		15分で	FAIL (電池はシーケンスから除去される)
Set Seq Step	4	休眠			5分間	

ステップ1

ステップ1では、電圧が4.2Vに達するまで、すべての電池が0.295Aの定電流で充電されます。電池は4.2V制限で充電を続けますが、充電電流は0.295Aの制限設定から減少し始めます。電池の充電は、電流が0.02Aに下がるまで続けられます。0.02Aになると、電池は次のステップである休眠状態に進みます。図5-2では、これは電池1と電池2に見られます。電池3の場合、電流テストが真にならなかったので、最大充電時間である20分間、充電ステップにとどまります。

電池が5分以内に3.8Vの電圧設定に達した場合、テスト結果はFAILになります。これは、電池の充電が速すぎることを表します。

ステップ2

ステップ2では、すべての電池が少なくとも10分間休眠状態に置かれ、出力にスティミュラスは供給されません。このように、休眠ステップを使用して、テスト基準を満たした後、現在のスティミュラス設定を適用したくない場合、あるいはその他の電池が現在のステップを終了するまで次のステップに進みたくない場合に、電池を休眠状態に移すことができます。

ステップ3

ステップ3では、電圧が3Vに降下するまで、すべての電池が0.295Aの定電流で放電されます。この電圧は放電電圧の終わり (EODV) と呼ばれます。5分経過した後に電圧が3Vまで降下した場合、電池は休眠状態になります。図5-2では、これは電池1と電池2に見られます。放電ステップの最大制限時間は15分間ですが、電池1および2の場合、ステップはこれより速く完了しています。

5分経過する前に電圧が3Vまで降下した場合、電池のテスト結果はFAILになります。図5-2では、これは電池3に見られ、電池の放電が速すぎることを表します。15分後に電圧が3V以下に下がらない場合にも、電池のテスト結果はFAILになります。これは、テスト・フィクスチャまたは配線に問題があるために、電池の放電が遅すぎることを表します。

ステップ4

ステップ4は、5分間の休眠ステップです。この例では、前の放電ステップとシーケンスの次のステップとの間のバッファとして含まれています。各電池は独立した速度で進むので、この方法で休眠ステップを使用する必要はありません。

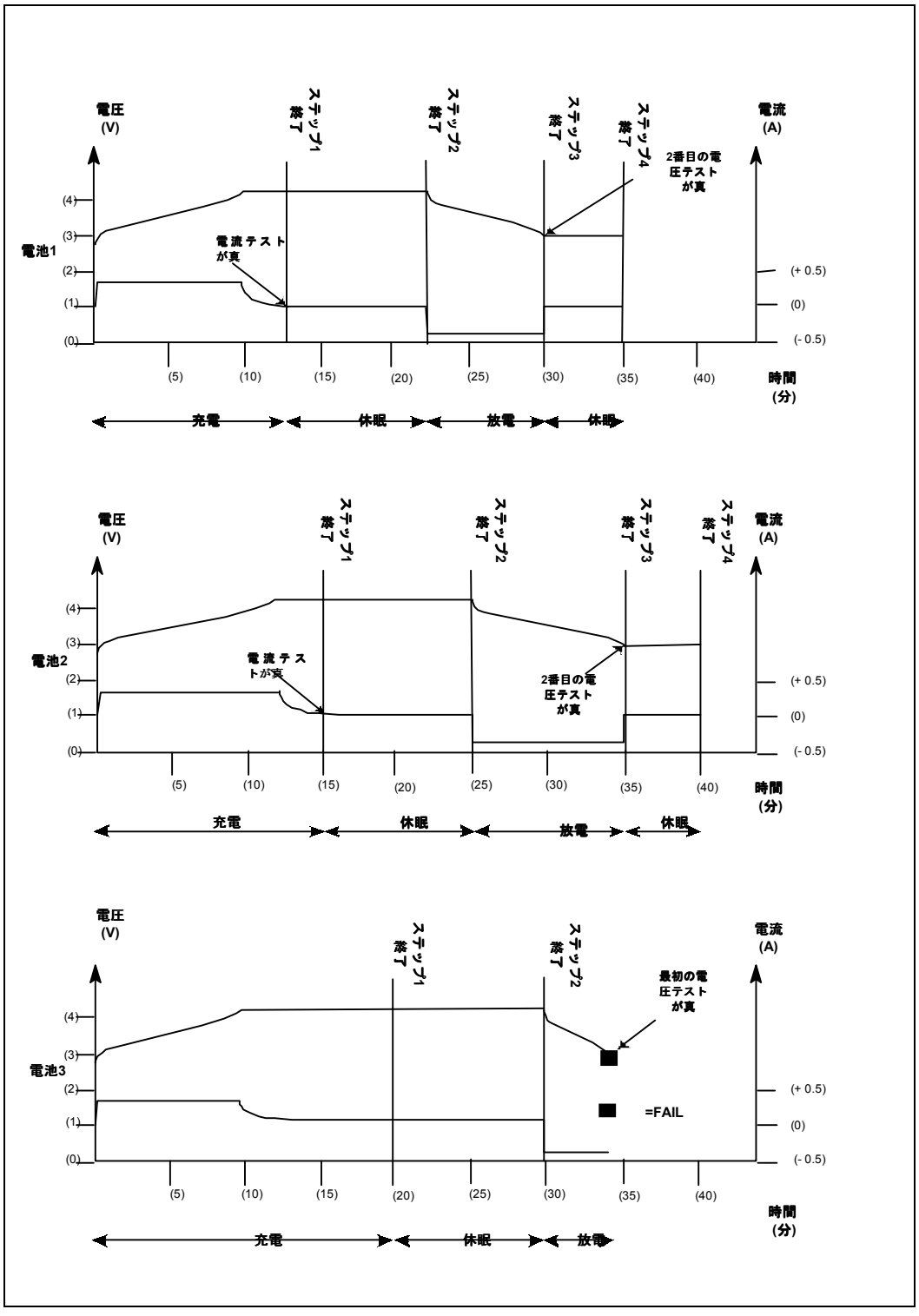


図5-2. 簡単な電池フォーミングの例

関数呼び出しの概要

Agilent E4370A MCCDの電池フォーミング・プロセスを制御するドライバ関数呼び出しは、次のカテゴリに大きく分けられます。

電池グループ化関数－独立したシーケンス制御を行うための電池グループの構成
ステップ/テスト関数－電池フォーミング・シーケンスの各ステップのセットアップと制御
シーケンス制御関数－機器のラン・ステートの制御
保護関数－機器の保護ステートのセットアップ
データ記憶関数－シーケンス実行中に発生した測定ログの制御
直接制御関数－シーケンスを実行しない場合の電池のプログラミング
サーバ関数－通信パラメータの設定と読み取り
ディジタルI/O関数－外部ディジタル制御信号の構成と制御
シリアル・ポート関数－2個のシリアルI/Oポートの構成と制御

第6章の関数定義の項に、すべての電池フォーミング (cf) 関数がアルファベット順に記載されています。

電池のグループ化

Agilent E4370A MCCDには電池またはチャネルの隣接ブロックをグループ化する機能があります。定義した電池の各グループは、他の定義電池のグループとは独立に制御することができます。これは、Agilent E4370A MCCDメインフレームに接続された電池のグループに、異なる電池フォーミング・シーケンスが割り当て可能であることを意味します。割り当てられた全部のシーケンスを同時に実行することができます。

各グループは、開始電池番号とグループ内の電池の総数によって定義されます。グループには、1個の電池からメインフレーム内のすべての電池までを含めることができます。256チャネルのメインフレームそれぞれに対して、最大8グループを定義できます。グループが定義されていない場合、メインフレームに送信されたコマンドは、メインフレーム内のすべてのアクティブ・チャネルに適用されます。1個の電池グループを定義している場合、メインフレーム内の残りの電池を制御するために、残りの電池もすべて、グループに割り当てる必要があります。

グループを作成するには、以下のコマンドを使用します。

```
int cfSetGroup(CF_HANDLE server, char *name, int start, int size);
```

引数nameは、グループの識別に用いられます。グループを作成したら、その後のAPI関数による制御のために、ハンドルを取得する必要があります。この関数は、既存グループの変更にも使用できます。既存グループ名を用いると、グループの定義が、新しいデータで上書きされます。

グループ・ハンドルを取得するには、以下の関数を使用します。

```
int cfOpenGroup(CF_HANDLE server, char *name, CF_HANDLE
*group_handle);
```

既存グループを削除するには、以下の関数を使用します。

```
cfDeleteGroup
```

全定義グループの問い合わせには、以下の関数を使用します。

```
int cfGetGroups(CF_HANDLE server, char
names[CF_MAX_GROUPS][CF_MAX_GROUP_NAME_LEN], int start[CF_MAX_GROUPS],
int size[CF_MAX_GROUPS]);
```

注記: グループは揮発性であり、AC電源を切ると消滅します。また、cfResetはすべての揮発性設定を電源投入時の状態にリセットするので、グループがすべて削除されます。

グループ化関数

cfOpenGroupによって返されるグループ・ハンドルは、以下のリストに示す関数に使用できます。これらの関数は、特定グループの制御や問い合わせを行います。関数がこのリストにない場合、関数をcfOpenGroupから取得されたグループ・ハンドルで使用することはできません。

```
cfAbort                  cfGetSenseProbeTest      cfSetSenseProbeTest
fGetCurrent              cfGetSeqTime             cfSetSeqStep
cfGetMeasLogInterval     cfGetTrigSource          cfSetSeqTest
cfGetOutputProbeTest     cfGetVoltage             cfSetSeqTestAnd
cfGetOutputState         cfInitiate               cfSetTrigSource
cfGetSeqStep             cfReadMeasLog            cfSetVoltage
cfGetSeqTest             cfSetCurrent             cfStateRecall
cfGetSeqTestAnd          cfSetMeasLogInterval     cfStateSave
cfResetSeq               cfSetOutputProbeTest     cfTrigger
cfGetRunState            cfSetOutputState
```

cfSetGroupによって1つまたは複数のグループを定義した場合、上記リストの関数は、cfOpenGroupから取得したグループ・ハンドルでのみ使用することができます。これらの関数をcfOpenから取得したハンドルを使って呼び出すと、エラーが返されます。グループが定義されていない場合は、cfOpenから返されたハンドルを使ってすべての電池が制御できます。

ステップ/テスト関数

充放電シーケンスは、計測器またはグループが自動的に従うユーザ定義のステップのシーケンスです。各ステップでは、指定された時間、充電、放電、またはスティミュラスなしのいずれかが適用されます。その他のパラメータは、電圧および電流の設定値、シーケンス内のステップの番号、ステップの時間の長さを決定します。ステップは、AC抵抗とDC抵抗の測定にも用いられます。

ステップ・パラメータの設定および問い合わせには、以下の関数を使用します。

```
cfSetSeqStep();
cfGetSeqStep();
```

シーケンスの実行中に、現在実行中のステップ、およびそのステップに電池がどれだけの期間置かれていたかを問い合わせるには、以下の関数を使用します。

```
cfGetStepNumber();
```

ステップ1を0.295A、電圧制限4.2Vで30分間充電するように、またステップ2を0.5A、電圧制限2.0Vで15分間放電するようにプログラムするには、以下の関数を使用します。

```
cfSetSeqStep(server, 1, CF_CHARGE, 4.2, 0.295, 30.0 *
SECONDS_PER_MINUTE, 0.0);
cfSetSeqStep(server, 2, CF_DISCHARGE, 2.0, 0.5, 15.0 *
SECONDS_PER_MINUTE, 0.0);
```

各ステップごとにテストを定義して、電池が正しく機能していることを確認したり、電池の性能基準に従って次のステップへの遷移を制御することができます。各テストでは、ステップ番号、測定の種類 (電圧、電流、またはAC抵抗が制限より大きいか小さいか)、測定制限、時間テストの種類と制限、およびテストが真である場合に実行すべき動作が参照されます。

テストを定義するには、以下の関数を使用します。

```
cfSetSeqTest();
```

定義済みのテストをリードバックするには、以下の関数を使用します。

```
cfGetSeqTest();
```

30分以内に電圧が4Vを超えなければ電池を不良とみなすテストや、5分以内に電圧が4Vに達した場合にテスト結果がFAILになるテストをプログラムするには、以下の関数を使用します。

```
cfSetSeqTest(server, 1, CF_VOLT_LE, 4, CF_TEST_AT, 30 *
SECONDS_PER_MINUTE, CF_FAIL);
```

```
cfSetSeqTest(server, 1, CF_VOLT_GE, 4, CF_TEST_BEFORE, 5 *
SECONDS_PER_MINUTE, CF_FAIL);
```

時間テストの種類と時間制限によって、いつ測定が行われるかが決まります。CF_TEST_BEFOREの場合、測定はステップの最初から時間制限まで連続して実行されます。CF_TEST_AFTERの場合、測定は時間制限からステップが完了するまで連続して実行されます。CF_TEST_ATの場合、測定が時間制限で一度だけ実行されます。

テストが真であれば、NEXTによって出力が次のステップに進み、残りのテストはすべて無視されます。FAILの場合は、特定の出力が開放回路にされ、シーケンスから削除され、不良としてタグが付けられます。測定テストが偽である場合は、何も起こりません。

シーケンスの制御

次の図に、計測器またはグループのさまざまなラン・ステートを示します。電源投入時にCF_NOT_READYステートになり、セルフテストと初期化が終了し、パワーバスのDC電源がオンになるまでこのステートが続きます。これには数秒かかります。CF_NOT_READYステートの間は、出力をプログラムできません。校正中も、計測器はこのステートになります。

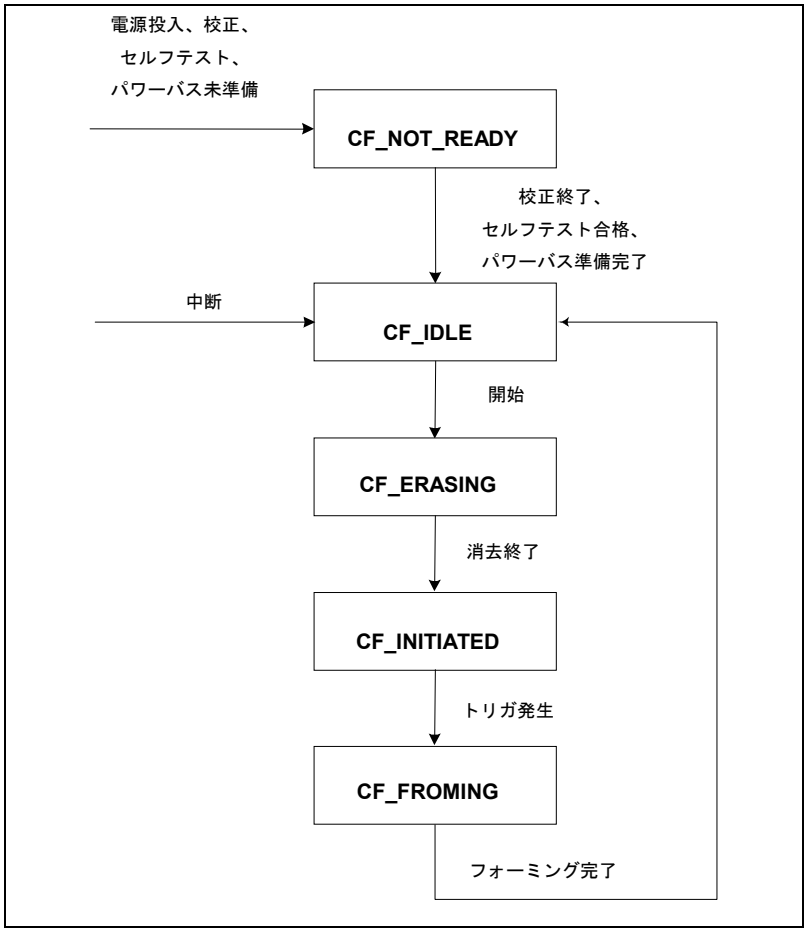


図5-3. 計測器のラン・ステート

セルフテストが完了し、パワーバスにDC電圧が供給されると、計測器はCF_IDLEステートに移行します。このステートで、計測器は電池フォーミング・シーケンスが開始されるのを待ちます。電池フォーミング・シーケンスが完了すると、計測器はCF_IDLEステートに戻ります。また、Abort関数によってもCF_IDLEステートになります。電

池フォーミング・シーケンスは、Agilent MCCDがCF_IDLEステートにあるときだけ、定義したり不揮発性メモリからリコールすることができます。

シーケンスの実行を開始するには、イニシエート関数を呼び出す必要があります。これによって、計測器はシーケンスをチェックし、実行可能な場合は測定ログに使用されているメモリの消去を開始し、CF_ERASINGステートに移行します。メモリの消去には5秒～50秒かかります。この後、CF_INITIATEDステートに移行します。この段階から、計測器はトリガを受信すると、シーケンスの実行を開始します。トリガ源として、LANまたはディジタルI/Oポートが使用できます。

シーケンスが完了すると、計測器はCF_IDLEステートに戻り、結果を計測器から読み出すことができます。イニシエート関数が測定ログをクリアするため、シーケンスを再度イニシエートする前に測定ログを読む必要があります。

計測器のステートの問い合わせには、以下の関数を使用します。

```
cfGetRunState();
```

シーケンスをチェックし、CF_INITIATEDステートに移行するには、以下の関数を使用します。

```
cfInitiate();
```

シーケンスを開始するには、以下の関数を使用します。

```
cfTrigger () ;
```

または、設定済みディジタルI/Oラインを使ってトリガを生成します。

トリガ源の設定および問い合わせには、以下の関数を使用します。

```
cfSetTrigSource();
cfGetTrigSource();
```

トリガ発生後の時間を問い合わせるには、以下の関数を使用します。

```
cfGetSeqTime();
```

シーケンスをアボートし、CF_IDLEステートに戻るには、以下の関数を使用します。

```
cfAbort();
```

出力の構成

特定の出力を使用しない場合は、それらの出力を非アクティブにプログラムすることができます。非アクティブに設定された出力はオフ状態のままになり、シーケンス設定関数やステータス関数には含まれません。これらの出力は、セルフテスト中にテストされず、機器の校正中にも校正されません。非アクティブ出力の測定を行うと、特殊値CF_NOT_A_NUMBERが返されます。

出力構成の設定および問い合わせには、以下の関数を使用します。

```
cfSetOutputConfig();
cfGetOutputConfig();
```

注記: cfSetOutputConfig()を使ってプログラムされたアクティブ/非アクティブ出力設定は、不揮発性メモリには保存**されません**。ユニットの電源を投入するたびに、出力設定をプログラムし直さなければなりません。一方、Agilent MCCDユーザ・インタフェースを使ってプログラムされた出力設定は、不揮発性メモリに保存**されます**。ユニットの電源投入時には、保存されている設定が用いられます。

計測器の保護

下図に、計測器のさまざまな保護ステートを示します。

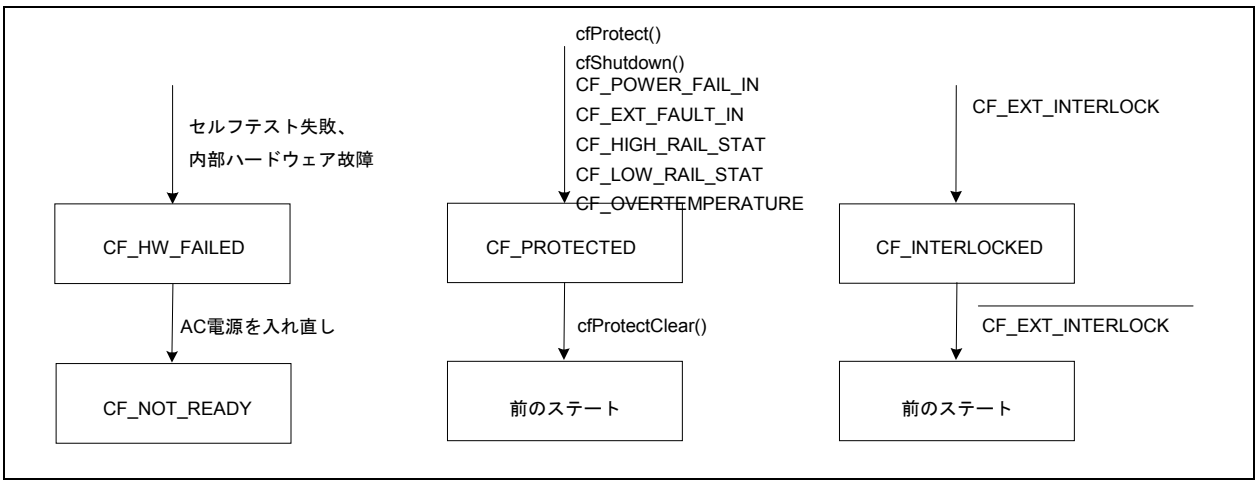


図5-4. 計測器の保護ステート

セルフテストが完了し、パワーバスにDC電圧が供給されると、計測器は通常、CF_IDLEステートに移行します。しかし、電源投入時のセルフテストに失敗した場合には、CF_HW_FAILEDステートに移行し、AC電源を入れ直すかcfSelftest()に合格するまでそのステートを維持します。CF_HW_FAILEDステートでは、出力をプログラムすることはできませんが、LAN通信は正常に機能します。

計測器を保護ステートにすることも可能です。保護ステートでは、すべての出力が開放回路になります。これは、プログラム・コマンドcfProtect()を使うか、CF_EXT_FAULT_INとして構成されたディジタル入力で偽-真エッジを表明することによって実現されます。計測器は、内部の温度が上がり過ぎた場合や、パワーバスの電圧が高くなり過ぎたり、低くなり過ぎた場合にも、保護ステートになります。計測器は、CF_PROTECTEDステートになるときに元のステートを記憶するので、cfProtectClear()コマンドが送信されると前のステートに戻ります。不良がある場合は、cfProtectClear関数が呼び出されても、計測器はCF_PROTECTEDステートに置かれたままになります。CF_PROTECTEDにされたときに計測器がCF_FORMINGステートにある場合、フォーミング・プロセスおよびシステム・タイマは、割込みが発生したところからフォーミングが再開されるように一時停止されます。

CF_PROTECTEDに似た保護ステートとしてはCF_INTERLOCKEDがあります。外部CF_EXT_INTERLOCK入力が真である場合には必ず、計測器はCF_INTERLOCKEDステートになります。また、CF_EXT_INTERLOCKが偽である場合、計測器は前のステートに戻ります。

機器ステートの問い合わせには、以下の関数を使用します。

```
cfGetRunState();
```

計測器を強制的にCF_PROTECTEDステートにするには、以下の関数を使用します。

```
cfProtect();
```

CF_PROTECTEDステートを出るには、以下の関数を使用します。

```
cfProtectClear();
```

注記: 計測器がCF_PROTECTEDステートにある間にcfAbort()コマンドを送信した場合、前のステートに関係なく、cfProtectClear()は計測器をCF_IDLEステートにします。

停電時の操作

Agilent E4370A MCCDは、2つの停電シャットダウン・モードの1つで操作できます。モードは、cfSetShutdownMode()コマンドによって設定します。モードをCF_AUTOに設定すると、CF_POWER_FAIL_INディジタル入力の真信号によって、Agilent MCCDがシャットダウンを実行します。その際、Agilent MCCDは不揮発性メモリにステートを保存します。モードをCF_MANUALに設定すると、自動シャットダウンは実行されません(cfShutdownなどのAPIコマンドを使って、手動で実行する必要があります)。CF_POWER_FAIL_INディジタル入力信号は、cfShutdownMode = AUTOの場合にだけ得られます。

Agilent MCCDファームウェアがディジタルCF_POWER_FAIL_IN信号のステートを認識するには約20msかかります。cfSetShutdownDelay()コマンドによって設定されるプログラマブル遅延によって、CF_POWER_FAIL_IN信号がどれだけの間真であれば自動シャットダウンが発生するかが決まります。CF_POWER_FAIL_IN信号が真である時間がcfShutdownDelayで設定した遅延時間よりも長ければ、Agilent MCCDはシャットダウンを実行します。cfShutdownDelay時間が経過する前にCF_POWER_FAIL信号が偽になると、シャットダウンは発生しません。

この遅延を使って、Agilent MCCDが20msを超える停電で毎回シャットダウンしないようにします。たとえば、Agilent MCCDを5分間継続して動作させられるUPSにAgilent MCCDを接続した場合、cfShutdownDelayを4分間に設定します。この場合、4分間停電しないとAgilent MCCDはシャットダウンしません。4分は、UPSホールドアップ時間の5分よりも短いので、電力が失われる恐れがありません。4分の遅延時間が経過する前に電力が戻った場合(CF_POWER_FAIL_IN = falseによって示されます)、シャットダウンは発生せず、AC電力が短時間失われてもシステムに問題はありません。

停電シャットダウンのプログラムと制御には、以下の関数を使用します。

```
cfSetShutdownMode();
cfSetShutdownDelay();
cfSetServerTimeout();
cfSetAutoConnect();
cfShutdown();
```

以下の関数にも停電シャットダウンに使用される機能があります。

```
cfSetDigitalConfig();
cfGetInstStatus();
cfSaveOutputConfig
```

保存したシャットダウン・ステートをリコールして計測器をリスタートするには、以下の関数を使用します。

```
cfRestart();
```

機器のステートの記憶

計測器は、複数の機器ステートを保存することができます。定義済みシーケンス・ステップやテストを含めて、計測器の全ステートがユーザ定義の名前で不揮発性メモリに保存されます。

機器のステートの制御には、以下の関数を使用します。

```
cfStateSave();
cfStateRecall();
cfStateList();
cfStateDelete();
```

計測器を電源投入時のステートにリセットするには、以下の関数を使用します。

```
cfReset();
```

電源投入時およびcfReset実行時の機器設定は、次のとおりです。

```
Output State = OFF
Output Voltage = 0 volts
Output Current = 0 amperes
Sequence Step = <all steps>
        Type = <undefined>
        Voltage = 0 volts
```

```
            Current = 0 amperes
            Time    = 0 seconds
    Measurement Interval = All steps
                ΔV = Infinity
                ΔI  = Infinity
                Δt  = Infinity
                ΔV = Infinity
    Trigger Source = LAN
    Digital Port = 0
    Probe Test Resistance = Infinity
    Sense Probe Test = Off
```

電源の投入およびcfReset()によってテスト設定はすべてクリアされます。

ステータス

計測器には、さまざまな機器条件をレポートするステータス・レジスタが装備されています。ステータス・レジスタを読み取るには、cfGetInstStatus()を使用します。レポートされる条件は、それぞれレジスタ内の1ビットで表されます。これらの条件を以下に示します。

CF_EXT_FAULT_IN_STAT	CF_EXT_FAULT_INとして構成された入力が表明されました。
CF_EXT_INTERLOCK_STAT	CF_EXT_INTERLOCKとして構成された入力は真です。
CF_SERIALB_SWITCH_STAT	シリアル・ポートBがハードウェア・スイッチによって構成ポートとして設定されています。
CF_LOW_RAIL_STAT	レール電圧が低くなり過ぎたか、現在低過ぎます。
CF_HIGH_RAIL_STAT	レール電圧が高くなり過ぎたか、現在高過ぎます。
CF_OVERTEMPERATURE_STAT	内部温度が高くなり過ぎたか、現在高過ぎます。
CF_CALIBRATING_STAT	機器は校正中です。
CF_POWER_ON_STAT	機器の電源投入時の初期化が完了していません。
CF_POWER_FAIL_STAT	CF_POWER_FAIL_INとして構成された入力は真です。
CF_RAIL_NOT_READY_STAT	レールはまだオンになっていません。
CF_RESTART_STAT	シャットダウン・ステートが保存されています。
CF_SELFTEST_STAT	セルフテストが進行中です。
CF_SELFTEST_ERROR_STAT	セルフテスト・エラーが発生しました。
CF_SHUTDOWN_STAT	停電シャットダウンが発生しました。
CF_CAL_ERROR_STAT	校正エラーが発生しました。

シーケンス中の電池の現在のステート (合格、不合格、進行中) を返すには、以下の関数を使用します。

```
cfGetCellStatus();
```

測定ログ

Agilent E4370A MCCDは、開始、終了で測定データをログします。各シーケンス・ステップの間、測定データをログするようプログラムすることも可能です。各出力の電圧と電流が継続的にモニタされ、ユーザ指定のしきい値によって出力が変化したか、指定時間が経過したときには、その出力に関するログが入力されます。ログ入力を引き起こす電圧または電流の基準はプログラム可能であり、シーケンス内の各ステップごとに異なる値を設定することもできます。データ・ロギングの時間周期パラメータは、1秒から596時間までの範囲の時間間隔に設定できます。特殊値CF_INFINITYをプログラムすると、特定のパラメータに対するロギングが効率的にオフになります。

測定ログに保存されるデータは、入力を提供する出力の番号、入力されたときに実行していたステップの番号、シーケンスを開始してからの時間、ステータス・ビットの合計、電圧および電流測定値、ステップ開始以降の累積ワット時およびアンペア時などです。AC抵抗ステップとDC抵抗ステップに関しては、抵抗測定値だけがログに入れられます。

測定ログには、タグ付きシーケンス・ステップ・タイプからのデータも入れられます。タグ付き測定には、AC抵抗、DC抵抗、開回路電圧、累積アンペア時、累積ワット時があります。測定ログからタグ付き入力だけを選択的に読み取るための特殊フィルタが用意されています。タグ付き測定の詳細については、第6章のcfSet SequenceStep()およびcfReadMeasLog()を参照してください。

測定ログは、349,504入力を保持できる巡回待ち行列です。測定ログの総使用可能メモリの各部分が、グループ内の電池の数に基づいてそれぞれのグループに割り当てられます。測定ログのメモリは64個のブロックに組織化されており、各ブロックの容量は5461入力です。各グループは、4個以上の測定ブロックを使用します。電池が16個を超えるグループの場合、最初の16電池の後は電池4個ごとに、別のメモリ・ブロックが使用されます。数が4個未満の場合でも、追加メモリ・ブロックが使用されます。たとえば、16個の電池のグループは4個のブロックを、17、18、19、または20個の電池のグループは5個のメモリ・ブロックを使用します。21個の電池のグループは、6個のメモリ・ブロックを使用します。グループの詳細については、本章のはじめにある「電池のグループ化」を参照してください。

注記: シーケンスをイニシエートすると、測定ログの内容はクリアされます。

測定ログの電圧および電流の記録間隔基準の設定および問い合わせには、以下の関数を使用します。

```
cfSetMeasLogInterval();
cfGetMeasLogInterval();
```

測定ログを読み取るには、以下の関数を使用します。

```
cfReadMeasLog();
```

読み取りポインタをログの開始または書き込みポインタの直前にリセットするには、以下の関数を使用します。

```
cfReset();
```

測定ログをクリアするには、以下の関数を使用します。

```
cfInitiate();
```

タイムスタンプ関数

測定ログは、電池フォーミング・シーケンスの開始からの時間を秒単位で記録するだけです。フォーミング・シーケンスが実際に開始した時刻を求めるには、cfGetSeqTime()関数をコントローラのクロックと一緒に使用します。

cfGetSeqTime()関数は、シーケンスが開始されたか、トリガされてからの経過時間を秒単位で返します。プログラムで、シーケンスが動作中であると判断したらこの関数を呼び出し、以下のアルゴリズムを使ってシーケンスの開始時刻を計算できます。

StartTime = CurrentTime – cfGetSeqTime()

出力測定

計測器は、コマンドによって出力端子での測定を行うことができます。以下の測定を、単一の出力、または全出力に対して行うことができます。

電圧測定の場合は、以下の関数を使用します。

```
cfMeasVoltage();
```

電流測定の場合は、以下の関数を使用します。

```
cfMeasCurrent();
```

以下の抵抗測定が完了するまでには数秒かかります。場合によっては、実行時間の増加を考慮してcfSetTimeout()関数を一時的に調整する必要があります。何らかの理由で測定が完了しなかった場合、特殊値CF_NOT_A_NUMBER (9.91E37) が返されます。

AC抵抗測定の場合は、以下の関数を使用します。

```
cfMeasACResistance();
```

DC抵抗測定の場合は、以下の関数を使用します。

```
CfMeasDCResistance () ;
```

出力プローブ抵抗測定の場合は、以下の関数を使用します。

```
csMeasOutputProbeResistance () ;
```

センス・プローブ抵抗測定の場合は、以下の関数を使用します。

```
csMeasSenseProbeResistance () ;
```

出力電圧の調整と測定は、リモート・センス関数が用いられている場合以外は、パワー出力端子で行われます。Agilent MCCDは、電圧の調整と測定をパワー端子かセンス端子のいずれかで行うように構成できます。

電圧測定場所の設定および問い合わせには、以下の関数を使用します。

```
cfSetSense();
cfGetSense();
```

直接出力制御

注意: **直接出力制御は充電中の電池には用いないでください。直接出力制御を用いた場合、過充電やプローブ・チェックの使用を防ぐことができません。このモードは、診断やデバッグのためだけに使用してください。**

ステップとテストから成るシーケンスを定義せずに、Agilent MCCDの出力を直接制御して、診断を行うことができます。直接出力制御コマンドは、Agilent MCCDがCF_IDLEステートにある間だけ使用できます。電圧、電流、および出力ステート設定は、すべての出力に対して同時に設定されます。Agilent MCCDシステムがCF_IDLEステートを出るたびに、これらの設定は電源投入時の値にリセットされます。

電圧制御には、以下の関数を使用します。

```
cfSetVoltage();
cfGetVoltage();
```

電流制御には、以下の関数を使用します。

```
cfSetCurrent();
cfGetCurrent();
```

出力のオン/オフ・ステートには、以下の関数を使用します。

```
cfSetOutputState();
```

```
cfGetOutputState();
```

一般サーバ関数

計測器に関連する一般的な関数がいくつかあります。計測器の制御は、LAN接続を行った後で可能になります。接続を開くにはパスワードが必要です。パスワードは、Agilent MCCD設定画面で設定します (第3章を参照)。

計測器のLAN接続を開くか閉じるには、以下の関数を使用します。

```
cfOpen () ;
cfClose () ;
```

以下の関数を使って、計測器の応答の最大待ち時間を設定します。

```
cfSetTimeout();
```

以下の関数を使って、計測器の識別文字列を読み取ります。

```
cfInstIdentify();
```

以下の関数を使って、Agilent MCCD画面から入力された識別文字列を読み取ります。

```
cfUserIdentify();
```

以下の関数を使って、計測器の他の関数がエラーを返したとき、その関数から呼び出される関数（省略可能）を定義します。

```
cfSetErrorFunction();
```

セルフテスト

Agilent E4370A MCCDには、電源投入時に実行されるセルフテスト機能が内蔵されています。この限定されたセルフテストによって、メモリ機能、シリアル通信機能、A/Dコンバータ機能、および各出力レギュレータの電圧プログラミングが正しく機能するか検証されます。セルフテストでは、パワーバス上の外部DC電源の存在も確認されます。テストでは電池に電力が供給されないので、実行時に、Agilent MCCDに電池が接続されているか否かは関係しません。

次のコマンドを実行することによって、より完全なセルフテストを実行できます。

```
cfSelftest();
```

cfSelftest()は、ほとんどの電源投入時テストを実行するほか、すべての出力レギュレータの定電流充放電機能や、電流測定機能のテストも行います。

注意: cfSelftest()を用いると、電圧が出力に加えられます。cfSelftest()の実行時には、接続されている電池がないことを確認してください。

セルフテストの完了までには数秒かかるので、cfSelftest()関数はセルフテストが完了するのを待ちません。セルフテストの開始直後に元の状態に戻ります。セルフテストの実行中は、cfGetInstStatus()によって返されるステータス・ワードのCF_SELFTEST_STATビットは真になります。セルフテストが完了すると、CF_SELFTEST_STATビットが偽になり、テストの失敗がステータス・ビットCF_SELFTEST_ERROR_STATによって示されます。したがって、セルフテストの実行中に機器ステートをポーリングすることによって、セルフテストの状況を確認することができます。

CF_SELFTEST_ERROR_STATビットによってセルフテスト・エラーがあることが示された場合、エラーの詳細をテスト・ログから入手することができます。テスト・ログを読み取るには、以下の関数を使用します。

```
cfReadTestLog();
```

別のセルフテストまたは校正コマンドが指定されるまで、そのセルフテスト・エラー情報はテスト・ログに保持されます。

校正

Agilent MCCDの校正は、Agilent MCCDがCF_IDLEステートにある場合にだけ実行可能です (図5-3を参照)。校正の詳細については、付録Bを参照してください。校正手順は2つのステップから成ります。まず外部DMMを使って内部基準が校正され、次に内部基準を使って各チャネルに校正が転送されます。新しい、または修理したチャージャ/ディスチャージャ・カードをメインフレームにインストールしたときは、常に2番目のステップを実行する必要があります。

内部メインフレーム基準を校正するには、電圧計をシリアル・ポートAに接続する必要があります (付録Bを参照)。メインフレーム基準の校正を開始するには、以下の関数を使用します。

```
cfCalStandard();
```

基準を各チャネルに転送するには、電池の出力およびセンス端子に対する外部接続をすべて開く必要があります。伝送校正を開始するには、以下の関数を使用します。

```
cfCalTransfer();
```

単一のコマンドで、cfCalStandard()とcfCalTransfer()の組合わせを実行することも可能です。

```
cfCal();
```

注意: cfCalTransfer()またはcfCal()を実行する際には、接続されている電池がないことを確認してください。

256チャネルのAgilent MCCDの校正には15分かかるので、校正関数は校正が完了するのを待ちません。校正の開始直後に元の状態に戻ります。校正の実行中は、cfGetInstStatus()によって返されるステータス・ワードのCF_CALIBRATING_STATビットは真になります。校正が完了すると、CF_CALIBRATING_STATビットが偽になり、校正エラーがステータス・ビットCF_CAL_ERROR_STATによって示されます。したがって、校正の実行中に機器ステートをポーリングすることによって、校正の状況を確認することができます。

CF_CAL_ERROR_STATビットによって校正エラーがあることが示された場合、エラーの詳細をテスト・ログから入手できます。テスト・ログを読み取るには、以下の関数を使用します。

```
cfReadTestLog();
```

別の校正またはセルフテスト・コマンドが指定されるまで、その校正エラー情報はテスト・ログに保持されます。

シリアル・ポート

計測器にはシリアル・ポートが2つあり、LANからのパススルー・ポートとして使用できます。これらのポートは、アプリケーション・プログラムの制御下で、ローカル周辺機器として使用することができます。パススルー・モードでは、ポートに対する読み書きに使用される関数が計測器に直接的な影響を及ぼすことはありません。シリアル・ポートの構成を設定する関数もあります。

ポートにアクセスするには、以下の関数を使用します。

```
cfReadSerial();
cfWriteSerial();
```

シリアル・ポート構成の設定および問い合わせには、以下の関数を使用します。

```
cfSetSerialConfig();
cfGetSerialConfig();
```

シリアル・ポートのステータスを返すには、以下の関数を使用します。

```
cfGetSerialStatus();
```

シリアル・ポートBは、構成ポートとしても使用されます。これは、計測器のハードウェア・スイッチを使って選択します (第3章を参照)。また、校正の実行中には、シリアル・ポートAは外部電圧計の専用通信ポートとして再構成されます。

ディジタル・ポート

計測器には、16ビット・ディジタルI/Oポートがあります。このポートを汎用I/Oとして使用することも、特定の目的に合わせてビットを構成することもできます。各ピンをシャーシ基準入力または出力として使用したり、ピン・ペアを1つの絶縁出力として定義することも可能です。詳細については、第6章の"cfSetDigitalConfig()"を参照してください。

ディジタルI/O構成は、API関数以外にも、Agilent MCCD設定画面またはAgilent MCCDユーザ・インタフェースを使用することによって設定できます。詳細については、第2章および第4章を参照してください。

構成の設定および問い合わせには、以下の関数を使用します。

```
cfSetDigitalConfig();
cfGetDigitalConfig();
```

ラインを直接読み書きするには、以下の関数を使用します。

```
cfSetDigitalPort();
cfGetDigitalPort();
```

プローブのチェック

プローブ・チェックは、導通チェック、パワー・プローブの抵抗チェック、センス・プローブの抵抗チェックの3つの個別の機能から構成されています。プローブ・チェック機能を使用する場合には必ず、チャネル出力に電池を接続しなければなりません。

導通チェックは、パワー・プローブのテストやセンス・プローブのテストの前に実行すべき低電流スティミュラス・テストです。主な役目は調整不良プローブまたは異常プローブを識別することにあります。プローブがこの初期テストに合格した場合、テスト・シーケンスを安全に開始し、電池に電力を供給することができます。プローブ導通テストを実行するには、以下の関数を使用します。

```
cfMeasProbeContinuity();
```

プローブ抵抗チェックは、製品のリモート・センス機能と共にプラスとマイナスの出力端子を使用して、パワー・プローブとセンス・プローブの両方の抵抗が許容制限の範囲内にあることを確認するための内部手順です。この機能をイネーブルにすると、プローブ抵抗のチェックがシーケンスの実行中に継続的に行われます。

パワー・プローブの抵抗をチェックするためには、ユーザ定義の抵抗制限を指定する必要があります。パワー・プローブ抵抗には、プローブ、フィクスチャ配線、および接続抵抗が含まれます。いずれかのチャネルで既定義の抵抗制限値を上回った場合、そのチャネルは不良と見なされ、フォーミング・シーケンスから除去されます。抵抗制限を指定し、パワー・プローブ・チェックをイネーブルにするには、以下の関数を使用します。

```
cfSetOutputProbeTest();
```

センス・プローブ抵抗のチェックでは、内部測定が行われ、リモート電圧センス回路のリードバック確度の範囲内で最大許容抵抗値と比較されます。最大許容抵抗には、プローブ、フィクスチャ配線、接続抵抗、および内部スキャナ抵抗が含まれます。最大許容抵抗は通常、1000Ωです。いずれかのチャネルで最大許容抵抗値を上回った場合、そのチャネルは不良と見なされ、フォーミング・シーケンスから除去されます。センス・プローブ・チェックをイネーブルにするには、以下の関数を使用します。

```
cfSetSenseProbeTest();
```

実際のパワー・プローブ抵抗とセンス・プローブ抵抗を測定するには、以下のコマンドを使用します。

```
cfMeasOutputProbeResistance();
cfMeasSenseProbeResistance();
```

6

言語ディクショナリ

APIの使用に関する指針

このアプリケーション・プログラミング・インタフェース (API) を使えば、LAN経由で1台以上のAgilent MCCDユニットの操作を制御するためのアプリケーション・プログラムをPC上で作成することができます。APIは、一連のドライバ関数を提供するダイナミック・リンク・ライブラリ (DLL) によって構成されます。これらの関数は、アプリケーション・プログラムがAgilent MCCD機能にアクセスするために呼び出します。本項には、Agilent MCCDによって使用されるすべてのAPI電池フォーミング (cf) 関数の構文とパラメータを記載します。

オペレーティング・システムおよび言語のサポート

クライアントAPIライブラリは、Windows 95およびWindows NTをサポートします。使用するには、これらのオペレーティング・システムに含まれるTCP/IPサービスをインストールし、構成する必要があります。このAPIは、32ビット・アプリケーションだけをサポートします。テスト・プログラムはMicrosoft Visual C/C++で記述しなければなりません。

ブロッキング関数

このAPIの関数はすべてブロッキング関数なので、関数の操作が完了するまで戻りません。多くの関数はネットワークを介して通信を行うため、完了までにかなり時間を要することがあります。ネットワーク要求の進行中に、呼び出し側アプリケーションが実行すべきタスクを持っている場合、ネットワーク接続をサービスするスレッドを作成します。

バッファ管理

アプリケーションは、関数を呼び出すときに、関数にデータを渡すために必要なバッファを割り当てなければなりません。関数が戻った場合には、呼び出し側がバッファの割り当てを解除しなければなりません。

順次関数呼び出し

すべてのAPI関数はマルチスレッド操作をサポートしています。2つのスレッドそれぞれが、異なるAgilent MCCDに対する関数呼び出しを行った場合、関数呼び出しは同時に処理されます。ただし、2つのスレッドが同じAgilent MCCDに対して関数呼び出しを行った場合は、呼び出しは順番に処理されます。2番目のスレッドによって呼び出された関数は、最初のスレッドによる呼び出しが完了するまでブロックされます。

エラー・レポート

いずれの関数も、成功した場合にはCF_OKを返し、エラーが発生した場合には0以外のコードを返します。必要に応じてエラー処理関数をAPIに登録し、中央でエラーを処理することができます。この関数は、いずれかのAPI関数でエラーが発生した場合に必ず呼び出されます。

サーバ接続数

Agilent MCCDサーバでは、最大3つのクライアントの接続が可能です。3つのクライアントすべてが同じ機能を備えているので、いずれのクライアントも本章に記載されているAPIプログラミング関数を使ってAgilent MCCDのモニタと制御を行うことができます。WebベースのAgilent MCCDユーザ・インタフェースでは、3つのクライアント接続とは別に、個別の接続を使用します。

パスワードによるプロテクト

アプリケーション・プログラムは、サーバへの接続を開くためにパスワードを用意しなければなりません。Agilent MCCDは、出荷時にはパスワードによってプロテクトされていません。Agilent MCCDサーバのパスワードは、インストール手順の実行中に、Agilent MCCD設定画面を使って設定することができます。

APIの概要

cfAbort	フォーミング・シーケンスをアボートします
cfCal	完全校正 (メインフレームおよびカード) を開始します
cfCalStandard	標準校正 (メインフレーム) を開始します
cfCalTransfer	伝送校正 (カード) を開始します
cfClose	サーバ接続を閉じます
cfDeleteGroup	Agilent MCCDからグループを削除します
cfGetCellStatus	個々の電池のステータスを返します
cfGetCellStatusString	不良電池に関する詳細情報を返します
cfGetCurrent	cfSetCurrentによってプログラムされた電流設定を返します
cfGetGroups	全定義グループに関する情報を返します
cfGetDigitalConfig	個々のディジタルI/Oポートの設定を返します
cfGetDigitalPort	ディジタル・ポートからデータ・ワードを読み取ります
cfGetInstIdentify	機器の説明を返します
cfGetInstStatus	機器ステータスを返します
cfGetMeasLogInterval	データのログ時期を決定する基準を返します
cfGetOutputConfig	出力の構成を返します
cfGetOutputProbeTest	出力プローブ抵抗制限を返します
cfGetOutputState	Agilent MCCDの出力ステートを返します
cfGetRunState	現在の機器のラン・ステートを返します
cfGetSense	リモート・センスまたはローカル・センスの設定を返します
cfGetSenseProbeTest	センス・プローブ・テストの設定を返します
cfGetSeqStep	シーケンスのステップ番号のパラメータを返します
cfGetSeqTest	シーケンス・テストの1つのパラメータを返します
cfGetSeqTestAnd	AND 関数で結合された複数のテストのパラメータを返します
cfGetSeqTime	シーケンスのトリガ後の経過時間を返します
cfGetSerialConfig	シリアル・ポートの通信パラメータを返します
cfGetSerialStatus	シリアル・ポートのステータスを返します
cfGetStepNumber	電池の現在のシーケンス・ステップおよびステップ開始後の時間を

	返します
cfGetShutdownDelay	cfSetShutdownDelayによって設定された遅延値を返します
cfGetShutdownMode	シャットダウン・モードの設定を返します
cfGetTrigSource	選択されているトリガ源を返します
cfGetUserIdentify	Name (名前)、Location(場所)、およびDescription (説明) 情報を返します
cfGetVoltage	cfSetVoltageによってプログラムされた電圧設定を返します
cfInitiate	フォーミング・シーケンスをイニシエートします
cfMeasACResistance	1個の電池または全電池のAC抵抗を測定します
cfMeasCapacityAS	現在のステップにおける1個の電池の累積アンペア時容量を測定します
cfMeasCapacityWS	現在のステップにおける1個の電池の累積ワット時容量を測定します
cfMeasCurrent	1個の電池または全電池の電流を測定します
cfMeasDCResistance	1個の電池または全電池のDC抵抗を測定します
cfMeasOutputProbeResistance	1個の電池または全電池の出力プローブ抵抗を測定します
cfMeasProbeContinuity	1個の電池または全電池のセンスおよび出力プローブ接続をチェックします
cfMeasSenseProbeResistance	1個の電池または全電池のセンス・プローブに対する抵抗を測定します
cfMeasVoltage	1個の電池または全電池の電圧を測定します
cfOpen	Agilent MCCDへの接続を開きます
cfOpenGroup	サーバ・ハンドルを定義グループと結合します
cfProtect	Agilent MCCDを強制的に保護ステートにします
cfProtectClear	Agilent MCCDの保護ステートをクリアします
cfReadMeasLog	フォーミング・シーケンス中に捕捉された測定値を返します
cfReadTestLog	テスト・ログからのエラー・メッセージを返します
cfReadSerial	シリアル・ポートのうちの1つからデータを読み取ります
cfReset	プログラム可能なファンクションを電源投入時のステートに設定します
cfResetSeq	定義済みのシーケンスをアボートし、クリアします
cfRestart	前に保存したリスタート・ステートをリコールします
cfSaveOutputConfig	出力構成を不揮発性メモリに保存します
cfSelftest	機器のセルフテストを開始します
cfSetAutoConnect	mccd.dllファイルの自動リコネクト機能をオンまたはオフにします
cfSetCurrent	出力電流を設定します
cfSetDigitalConfig	個々のディジタルI/Oポートの動作を設定します
cfSetDigitalPort	ディジタルI/Oポートにデータ・ワードを書き込みます
cfSetErrorFunction	エラー関数を定義します

cfSetGroup	電池のグループを定義します
cfSetMeasLogInterval	測定ログ入力の生成に関する基準を定義します
cfSetOutputConfig	出力電池をアクティブまたは非アクティブとして構成します
cfSetOutputProbeTest	出力プローブの抵抗制限を設定します
cfSetOutputState	機器の出力ステートをオフ、充電、または放電に設定します。
cfSetSense	電圧センスをリモートまたはローカルに設定します
cfSetSenseProbeTest	センス・プローブ抵抗の自動テストを設定します
cfSetSeqStep	出力シーケンス・ステップを定義します
cfSetSeqTest	シーケンス・ステップ中に実行するテストを定義します
cfSetSeqTestAnd	論理積関数で結合された複数のテストを定義します
cfSetSerialConfig	シリアル・ポートの通信パラメータを設定します
cfSetServerTimeout	接続の非アクティブのタイムアウト時間を設定します
cfSetShutdownDelay	シャットダウン遅延周期を設定します
cfSetShutdownMode	シャットダウン・モードを自動または手動に設定します
cfSetTimeout	クライアントがサーバからの応答を待つ時間を設定します
cfSetTrigSource	トリガ源をLANまたは外部に設定します
cfSetVoltage	出力電圧を設定します
cfShutdown	Agilent MCCDをシャットダウン前に安全なステートに移行させます
cfStateDelete	以前に作成した機器ステートを削除します
cfStateList	機器ステート名のリストを返します
cfStateRecall	以前に作成した機器ステートをロードします
cfStateSave	現在の機器設定を不揮発性メモリに保存します
cfTrigger	LAN経由でトリガを送信します
cfWriteSerial	シリアル・ポートにデータ・ワードを書き込みます

API関数の定義

cfAbort

構文

```
int cfAbort(CF_HANDLE server);
```

解説

フォーミング・シーケンスをアボートします。これはラン・ステートを CF_IDLE に設定します。アイドル・ステートでは、各電池の出力条件は関数 cfSetVoltage、cfSetCurrent、および cfSetOutputState で定義します。引数 server

には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

cfCal

注意: cfCalを実行する際には、接続されている電池がないことを確認してください。

構文

```
int cfCal(CF_HANDLE server);
```

解説

完全校正シーケンスを開始します。この関数は、Agilent MCCD の内部基準を校正し、各チャネルに基準の校正を伝送します。つまり、cfCal は、cfCalStandard および cfCalTransfer 関数と同じ働きをします。

校正を行うには、サポートされている電圧計のうちの 1 つをシリアル・ポート A に接続する必要があります。チャネル出力に負荷や電池を接続することはできず、電圧計の入力は校正出力に接続しなければなりません。

完全校正には最大 15 分かかるので、校正関数は校正が完了するのを待ちません。校正関数は、校正開始直後に戻ります。校正の進捗度および結果をモニタするには、cfGetInstStatus を使用します。校正進行中は、CF_CALIBRATING_STAT ビットが真になります。校正中に何らかのエラーが発生した場合には、CF_CAL_ERROR_STAT ビットが真になります。エラーの詳細は、cfReadTestLog を使って入手できます。

cfCalStandard

構文

```
int cfCalStandard(CF_HANDLE server);
```

解説

機器の内部基準の校正を開始します。この場合、サポートされている電圧計のうちの 1 つをシリアル・ポート A に接続する必要があります。

校正関数は校正が完了するのを待ちません。校正関数は、校正開始直後に戻ります。校正の進捗度および結果をモニタするには、cfGetInstStatus を使用します。校正進行中は、CF_CALIBRATING_STAT ビットが真になります。校正中に何らかのエラーが発生した場合には、CF_CAL_ERROR_STAT ビットが真になります。エラーの詳細は、cfReadTestLog を使って入手できます。

cfCalTransfer

注意: cfCalTransferを実行する際には、電池が接続されていないことを確認してください。

構文

```
int cfCalTransfer(CF_HANDLE server);
```

解説

伝送校正シーケンスを開始します。この関数は計測器の内部基準を使って、計測器およびチャネルの出力回路を校正します。このコマンドを使用する場合は、出力に負荷や電池を接続しないでください。

伝送校正には最大 15 分かかるので、校正関数は校正が完了するのを待ちません。校正関数は、校正開始直後に戻ります。校正の進捗度および結果をモニタするには、cfGetInstStatus を使用します。校正進行中は、CF_CALIBRATING_STAT ビットが真になります。校正中に何らかのエラーが発生した場合には、CF_CAL_ERROR_STAT ビットが真になります。エラーの詳細は、cfReadTestLog を使って入手できます。

cfClose

構文

```
int cfClose(CF_HANDLE server);
```

解説

サーバ接続を閉じます。Agilent MCCD サーバは、限られた数のクライアント接続だけを開くことができます。接続を閉じると、それを他のクライアントが使用できます。接続を閉じてもサーバの出力関数には影響せず、計測器のその他の関数は cfOpen や cfClose コマンドに影響されずに動作を続けます。

cfDeleteGroup

構文

```
int cfDeleteGroup(CF_HANDLE server);
```

解説

Agilent MCCD からグループを削除します。引数 server は、cfOpenGroup()に対する呼び出しによってあらかじめ取得しておいたハンドルです。

cfGetCellStatus

構文

```
int cfGetCellStatus(CF_HANDLE server, int cell, CF_CELL_STATUS *status);
```

解説

ステータスが指し示す変数の値を返します。ステータスは、フォーミング・プロセスにある電池の現在の状態を示しています。以下の戻り値があります。

CF_UNTESTED	AC電源を入れて以来、電池はシーケンスを開始していません。
CF_PASSED	電池は最後のフォーミング・シーケンスを終了し、すべてのテストに合格しました。
CF_SEQUENCE_FAILED	電池は最後のフォーミング・シーケンス中のテストで不合格になりました。
CF_OUT_PROBE_FAILED	電池はフォーミング中の出力プローブ抵抗テストで不合格になりました。
CF_SENSE_PROBE_FAILED	電池はフォーミング中のセンス・プローブ抵抗テストで不合格になりました。
CF_INACTIVE	電池はcfSetOutputConfigによって非アクティブに設定されました。
CF_SEQUENCING	電池はフォーミングの最中です。
CF_ABORTED	最後のフォーミング・シーケンスがアボートされました。

引数 cell には、個々の電池番号 1～256 を指定するか、すべての電池のステータス条件を読み取るために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定した場合、引数 status が戻り値を受け取る 256 個の列挙型配列を指している必要があります。

cfGetCellStatusString

構文

```
int cfGetCellStatusString(CF_HANDLE server, int cell, char *status);
```

解説

ステータスが CF_SEQUENCE_FAILED である電池の詳細を含む ASCII 文字列を返します。引数 cell には、個々の電池番号 1～256 を指定しなければなりません。定数 CF_MAX_CELL_STATUS_LEN は、戻りステータス文字列の最大長を定義します。

cfGetCurrent

構文

```
int cfGetCurrent(CF_HANDLE server, float *current);
```

解説

cfSetCurrent が設定したアイドル状態の電流設定値を返します。アイドル状態の電流とは、フォーミング・シーケンスがアイドル状態にあり、出力ステートがイネーブルのときに設定される電池電流の制限値です。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

cfGetDigitalConfig

構文

```
int cfGetDigitalConfig(CF_HANDLE server, int bitnum, CF_EXT_SIGNAL *signal,
CF_POLARITY *polarity,CF_PEFERENCE *reference);
```

解説

ディジタル I/O ポートの 16 ピンをマッピングするファンクションおよびロジック・センスを返します。ディジタル I/O ポート構成の詳細については、関数 cfSetDigitalConfig を参照してください。以下の信号が定義されています。

CF_EXT_FAULT_IN	External Fault Input（外部不良入力）
CF_EXT_FAULT_OUT	External Fault Output（外部不良出力）
CF_EXT_INTERLOCK	External Interlock（外部インターロック）
CF_EXT_TRIGGER	External Trigger（外部トリガ）
CF_DIG_IN	General purpose input（汎用入力）
CF_DIG_OUT	General purpose output（汎用出力）
CF_DIG_IN_OUT	General purpose input/output（汎用入出力）
CF_POWER_FAIL_IN	Power fail input（停電入力）
CF_POWER_FAIL_OUT	Power fail output（停電出力）
CF_DIG_OUT_AND_NOT_FAULT_IN	Digital output and not fault input（ディジタル出力および非不良入力）

関連項目

cfSetDigitalPort, cfGetDigitalConfig

cfGetDigitalPort

構文

```
int cfGetDigitalPort(CF_HANDLE server, int *data);
```

解説

ディジタル I/O ポートからデータ・ワードを読み取ります。ディジタル I/O ポートの詳細については、関数

cfSetDigitalConfig を参照してください。

関連項目

```
cfSetDigitalPort, cfGetDigitalConfig
```

cfGetGroups

構文

```
int cfGetGroups(CF_HANDLE server, char
names[CF_MAX_GROUPS][CF_MAX_GROUP_NAME_LEN], int start[CF_MAX_GROUPS],
int size[CF_MAX_GROUPS]);
```

解説

全定義グループに関する情報を返します。引数 names、start、および size は、サイズ CF_MAX_GROUPS の配列で、定義されたグループ名、その開始電池番号およびサイズを保持します。定義が CF_MAX_GROUPS より少ない場合、最後の有効グループ入力後の配列 size の入力には、値 0 が格納されます。

使用例

```
void query_groups(CF_SERVER server)
{
    char names[CF_MAX_GROUPS][CF_MAX_GROUP_NAME_LEN];
    int starts[CF_MAX_GROUPS];
    int sizes[CF_MAX_GROUPS];

    cfGetGroups(server, names, starts, sizes);
}
```

cfGetInstIdentify

構文

```
int cfGetInstIdentify(CF_HANDLE server, char *idstring);
```

解説

このコマンドは、計測器の説明を返します。この文字列は、モデル番号で始まり、製品名、ファームウェアのバージョン番号、計測器のオプションの記述または略語と続きます。idstring の最大長は CF_MAX_ID_LEN 文字で、ヌルで終わる C 文字列として返されます。

cfGetInstStatus

構文

```
int cfGetInstStatus(CF_HANDLE server, int *status);
```

解説

計測器のステータスを返します。ステータス・ワード内の個々のビットは、さまざまなステータス条件を示すように定義されます。以下の定数を使って、さまざまなステータス条件をテストすることができます。

CF_EXT_FAULT_IN_STAT	最後にcfProtectClearが呼び出されてから後に、CF_EXT_FAULT_IN関数に構成された入力が表明されました
CF_EXT_INTERLOCK_STAT	CF_EXT_INTERLOCK関数に構成された入力は真です
CF_SERIAL1_SWITCH_STAT	シリアル・ポート1がハードウェア・スイッチによって構成端末ポートとして設定された場合に真になります
CF_LOW_RAIL_STAT	最後にcfProtectClearが呼び出されてから後に、レール電圧が低くなり過ぎました

CF_HIGH_RAIL_STAT	最後にcfProtectClearが呼び出されてから後に、レール電圧が高くなり過ぎました
CF_OVERTEMPERATURE_STAT	最後にcfProtectClearが呼び出されてから後に、内部温度が高くなり過ぎました
CF_CALIBRATING_STAT	計測器は校正中です
CF_POWER_ON_STAT	計測器の電源投入時の初期化が完了していません
CF_POWER_FAIL_STAT	CF_POWER_FAIL_IN信号が真のときには常に真です
CF_RAIL_NOT_READY_STAT	パワーバスにDC電圧がありません
CF_RESTART_STAT	cfRestart()でリスタートできる保存されたシャットダウン・ステートがある場合は常に真です
CF_SELFTEST_STAT	セルフテストの進行中は真になります
CF_SELFTEST_ERROR_STAT	セルフテストの進行中は真になります
CF_SHUTDOWN_STAT	自動停電シャットダウンが発生したか、最後のcfProtectClear()呼び出しの後にcfShutdown()が呼び出されると真になります
CF_CAL_ERROR_STAT	セルフテストの進行中は真になります

cfGetMeasLogInterval

構文

```
int cfGetMeasLogInterval(CF_HANDLE server, int step_number, float
*volt_interval, float *curr_interval, float *time_interval);
```

解説

データのログ時期を決定するのに使用される電圧、電流および時間変化基準を返します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。引数 step_number には、個々のステップを指定するか、またはすべてのステップの変化基準を得るために定数 CF_ALL_STEPS を指定します。CF_ALL_STEPS を指定する場合は、引数 volt_interval、curr_interval および time_interval が、戻り値を受け取る CF_MAX_STEPS 個の実数型配列を指している必要があります。

cfGetOutputConfig

構文

```
int cfGetOutputConfig(CF_HANDLE server, int cell, CF_OUTPUT_CONFIG *config);
```

解説

出力の構成を返します。出力構成は CF_SET_ACTIVE または CF_SET_INACTIVE です。引数 cell には、個々の電池番号 1～256 を指定するか、またはすべての電池の出力構成を得るために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定する場合は、引数 config が、戻り値を受け取る 256 個の CF_OUTPUT_CONFIG の配列を指している必要があります。

CF_SET_INACTIVE に設定された電池は、出力およびフォーミング・コマンドの大部分を無視します。これらの電池の測定では、常に特別な値 CF_NOT_A_NUMBER が返されます。

cfGetOutputProbeTest

構文

```
int cfGetOutputProbeTest(CF_HANDLE server, float *resistance);
```

解説

フォーミング・シーケンスの出力プローブ・テスト時に使用される抵抗制限を返します。プローブ抵抗が抵抗制限を上回ると、その電池には不良のマークが付けられます。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

cfGetOutputState

構文

```
int cfGetOutputState(CF_HANDLE server, CF_OUTPUT_STATE *state);
```

解説

ラン・ステートが CF_IDLE である場合の Agilent MCCD の出力ステートを返します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。戻り値は以下のとおりです。

CF_OUTPUT_OFF	Agilent MCCDの出力ステートはオフです。
CF_OUTPUT_CHARGE	Agilent MCCDの出力は充電ステートにあります。
CF_OUTPUT_DISCHARGE	Agilent MCCDの出力は放電ステートにあります。

OFF ステートでは、チャネル出力は開放回路となり、電流は供給されません。充電および放電ステートでは、チャネル出力は cfSetVoltage および cfSetCurrent によって制御されます。

関連項目

```
cfSetOutputState, cfSetVoltage, cfSetCurrent
```

cfGetRunState

構文

```
int cfGetRunState(CF_HANDLE server, CF_RUN_STATE *state);
```

解説

計測器の現在のラン・ステート（CF_IDLE、CF_ERASING、CF_INITIATED、CF_FORMING、CF_PROTECTED、CF_NOT_READY、CF_INTERLOCKED、または CF_HW_FAILED）を返します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

cfGetSense

構文

```
int cfGetSense(CF_HANDLE server, CF_SENSE *sense);
```

解説

リモート・センスまたはローカル・センスの設定を返します。戻り値は CF_SENSE_REMOTE または CF_SENSE_LOCAL です。リモート・センスとローカル・センスの選択は、cfSetSense によって制御されます。

関連項目

```
cfSetSense
```

cfGetSenseProbeTest

構文

```
int cfGetSenseProbeTest(CF_HANDLE server, CF_BOOLEAN *on_off);
```

解説

センス・プローブ・テストの設定を返します。設定は ON と OFF のどちらかです。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。このテストがイネーブルになっているときは、計測器はセンス・プローブの抵抗を定期的に測定し、正確な電圧測定を行うことができるだけの低い値であるかチェックします。プローブ抵抗があまりにも高く、しかもテストがイネーブル状態にある場合、その電池のフォーミング・シーケンスは終了されます。

関連項目

```
cfSetSenseProbeTest
```

cfGetSeqStep

構文

```
int cfGetSeqStep(CF_HANDLE server, int step_number, CF_SEQ_OUT
*out_type, float *voltage, float *current, float *time, float *reserved);
```

解説

与えられたシーケンスの step_number のパラメータを返します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。step_number が定義されていないステップである場合、*out_type に返される値は CF_STEP_UNDEFINED になります。

cfGetSeqTest

構文

```
int cfGetSeqTest(CF_HANDLE server, CF_READP *read_pos, int *step_number,
CF_SEQ_TEST *meas_test_type, float *limit, CF_TIME_TEST
*time_test_type,  float *time, CF_SEQ_ACTION *action);
```

解説

1 つのシーケンス・テストのパラメータを返します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

同一ステップのシーケンス・テストは、time パラメータの昇順で計測器内に保存されます。連続して cfGetSeqTest を呼び出すと、この順序でパラメータが返されます。引数 read_pos によって指し示された値で、どのテストを読み出すかを制御します。 read_pos は、ステップ内の最初のテストを読み出すための特別な値 CF_READ_FIRST を指します。最後のテストが読み出された後は、cfGetSeqTest を呼び出すと、read_pos によって指し示された値に特別な値 CF_READ_EOF が返されます。

cfGetSeqTestAnd

構文

```
int cfGetSeqTestAnd(CF_HANDLE server, CF_READP *read_pos, int *step_number,
CF_SEQ_TEST  *meas_test_type,  float  *limit,  CF_TIME_TEST  *time_test_type,
float *time, CF_SEQ_ACTION *action, int *count);
```

解説

関数 cfSetSeqTest または cfSetSeqTestAnd によって定義されたシーケンス・テストのパラメータを返します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。動作は、関数 cfSetSeqTest と同様です。測定テストの数と制限は、*count に返されます。引数 meas_test_type と limit は、サイズ CF_MAX_AND_TESTS の配列を指している必要があります。

cfGetSeqTime

構文

```
int cfGetSeqTime(CF_HANDLE server, float *time);
```

解説

シーケンスがトリガされてからの経過時間を秒単位で返します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

cfGetSerialConfig

構文

```
int cfGetConfig(CF_HANDLE server, int portnum, int *baudrate,
CF_SERIAL_PARITY *parity, int *wordsize, CF_SERIAL_FLOW *flow_ctrl);
```

解説

1 つのシリアル・ポートの通信パラメータを返します。

Port	CF_PORTAまたはCF_PORTB
Baudrate	1200、2400、4800、9600、または19200
Parity	CF_PARITY_EVEN, CF_PARITY_ODD, または CF_PARITY_NONE
Wordsize	7 または 8
Flow_ctrl	CF_FLOW_RTS_CTS, CF_FLOW_XON_XOFF, またはCF_FLOW_NONE

cfGetSerialStatus

構文

```
int cfGetSerialStatus(CF_HANDLE server, int portnum, int *status);
```

解説

1 つのシリアル・ポートのステータスを返します。ビットの定義は以下のとおりです。

CF_SERIAL_MAV	メッセージ使用可能
CF_SERIAL_PE	パリティ・エラー
CF_SERIAL_FE	フレーミング・エラー
CF_SERIAL_OE	UARTオーバラン・エラー
CF_SERIAL_INBUF_OE	入力バッファ・オーバラン・エラー
CF_SERIAL_OUTBUF_OE	出力バッファ・オーバラン・エラー

シリアル・ステータスを読み出すと、エラー・ビットがクリアされます。ポートの FIFO に読み出す文字がなくなると、MAV ビットがクリアされます。

cfGetShutdownDelay

構文

```
int cfGetShutdownDelay(CF_HANDLE server, float *delay);
```

解説

cfSetShutdownDelay()によって設定された遅延値を返します。

cfGetShutdownMode

構文

```
int cfGetShutdownMode(CF_HANDLE server, int *mode);
```

解説

シャットダウン・モード CF_AUTO または CF_MANUAL を返します。

cfGetStepNumber

構文

```
int cfGetStepNumber(CF_HANDLE server, int cell, int *step_number, float
*time);
```

解説

電池の現在のフォーミング・シーケンスのステップ番号、および電池が現在のステップに置かれている時間 (秒) を返します。引数 cell には、個々の電池番号 1～256 を指定するか、すべての電池のデータを要求するために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定した場合は、引数 step_number が 256 個の整数型配列を、また引数 time が戻り値を受け取る 256 個の実数型配列を指している必要があります。step_number に返される値は、正整数のステップ番号か、CF_NOT_FORMING のいずれかです。

cfGetTrigSource

構文

```
int cfGetTrigSource(CF_HANDLE server, CF_TRIG_SOURCE *source);
```

解説

選択されているトリガ源を返します。返されるトリガ源は CF_LAN または CF_EXTERNAL です。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

cfGetUserIdentify

構文

```
int cfGetUserIdentify(CF_HANDLE server, char *idstring);
```

解説

Agilent MCCD 設定画面を使って設定された Name(名前)、Location(場所)、および Description(説明)テキスト・フィールドを返します。これらのフィールドは、改行で区切られ、ASCII ヌル文字で終わります。idstring の最大長は CF_MAX_USER_ID_LEN 文字です。

cfGetVoltage

構文

```
int cfGetVoltage(CF_HANDLE server, float *voltage);
```

解説

cfSetVoltage が設定したアイドル状態の電圧の設定値を返します。アイドル状態の電圧とは、フォーミング・シーケンスがアイドル状態にあり、出力ステートがイネーブルのときに設定される電池電圧の値です。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

cfInitiate

構文

```
int cfInitiate(CF_HANDLE server);
```

解説

電池フォーミング・シーケンスをイニシエートします。電池フォーミング・シーケンスは、イニシエート後、トリガがかかるまで起動しません。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。トリガ・ステートがアイドル状態にない場合、またはシーケンスが無効の場合、cfInitiate はエラーを返します。

関連項目

```
cfTrigger, cfSetTriggerSource, cfAbort
```

cfMeasACResistance

構文

```
int cfMeasACResistance(CF_HANDLE server, int cell, float *reading);
```

解説

注記: このコマンドの完了までには数秒かかるので、実行時間の増加を考慮して、cfSetTimeout関数を一時的に調整する必要があります。

特定の電池またはすべての電池の AC 抵抗の測定値を返します。引数 cell には、個々の電池番号 1～256 を指定するか、またはすべての電池の読み取りを要求するために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定する場合は、引数 reading が、戻り値を受け取る 256 個の実数型配列を指している必要があります。

電池の出力が OFF ステートにあるか、電圧センスが Local に設定されているか、または測定に十分な電流が流れていないために AC 抵抗の測定ができない場合は、特別な値 CF_NOT_A_NUMBER（9.91E37）が返されます。

cfMeasCapacityAS

構文

```
int cfMeasCapacityAS (CF_HANDLE server, int cell, float *reading);
```

解説

現在のステップにおける電池の累積容量をアンペア秒で返します。容量は、各ステップのはじめでゼロにリセットされます。電池がフォーミング・ステートにない場合、特別な値 CF_NOT_A_NUMBER が返されます。電池の引数には、個々の電池番号 1～256 を指定するか、またはすべての電池の読み取りを要求するために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定する場合は、戻り値を受け取る 256 個の実数型配列を指している必要があります。

cfMeasCapacityWS

構文

```
int cfMeasCapacityWS (CF_HANDLE server, int cell, float *reading);
```

解説

現在のステップにおける電池の累積容量をワット秒で返します。容量は、各ステップのはじめでゼロにリセットされます。電池がフォーミング・ステートにない場合、特別な値 CF_NOT_A_NUMBER が返されます。電池の引数には、個々の電池番号 1～256 を指定するか、またはすべての電池の読み取りを要求するために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定する場合は、戻り値を受け取る 256 個の実数型配列を指している必要があります。

cfMeasCurrent

構文

```
int cfMeasCurrent(CF_HANDLE server, int cell, float *reading);
```

解説

特定の電池またはすべての電池の測定電流を返します。引数 cell には、個々の電池番号 1～256 を指定するか、またはすべての電池の読み取りを要求するために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定する場合は、引数 reading が、戻り値を受け取る 256 個の実数型配列を指している必要があります。

cfMeasDCResistance

構文

```
int cfMeasDCResistance(CF_HANDLE server, int cell, float *reading);
```

解説

注記: このコマンドの完了までには数秒かかるので、実行時間の増加を考慮して、cfSetTimeout関数を一時的に調整する必要があります。

特定の電池またはすべての電池の DC 抵抗の測定値を返します。引数 cell には、個々の電池番号 1～256 を指定するか、またはすべての電池の読み取りを要求するために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定する場合は、引数 reading が、戻り値を受け取る 256 個の実数型配列を指している必要があります。

電池の出力が OFF ステートにあるか、電圧センスが Local に設定されているか、または測定に十分な電流が流れていないために DC 抵抗の測定ができない場合は、特別な値 CF_NOT_A_NUMBER（9.91E37）が返されます。

cfMeasOutputProbeResistance

構文

```
int cfMeasOutputProbeResistance(CF_HANDLE server, int cell, float
*resistance);
```

解説

注記: このコマンドの完了までには数秒かかるので、実行時間の増加を考慮して、cfSetTimeout関数を一時的に調整する必要があります。

特定の電池またはすべての電池の出力プローブの接触抵抗を測定して返します。データはオーム単位で返されます。引数 cell には、個々の電池番号 1～256 を指定するか、またはすべての電池の読み取りを要求するために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定する場合は、引数 resistance が、戻り値を受け取る CF_MAX_CELLS 個の実数型配列を指している必要があります。

プローブ抵抗を効率よく測定するために、プローブと電池の接点に十分な電流を供給してください。cfSetVoltage、cfSetCurrent、および cfSetOutputState コマンドを使って、この測定に合った条件を設定することができます。電池の出力が OFF ステートにあるか、電圧センスが Local に設定されているか、または測定に十分な電流が流れていないために、プローブ抵抗の測定が行えない場合は、特別な値 CF_NOT_A_NUMBER(9.91E37)が返されます。

cfMeasProbeContinuity

構文

```
int cfMeasProbeContinuity(CF_HANDLE server, int cell, CF_CONTINUITY *result);
```

解説

注記: このコマンドの完了までには数秒かかるので、実行時間の増加を考慮して、cfSetTimeout関数を一時的に調整する必要があります。

このコマンドは、特定の電池またはすべての電池のセンスおよび出力プローブ接続をチェックします。引数 cell には、個々の電池番号 1～256 を指定するか、すべての電池の読み取りを要求するために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定した場合は、引数 result が戻り値を受け取る CF_MAX_CELLS 個の配列を指している必要があります。結果の戻り値は以下の定数のいずれかです。

CF_PROBES_OK
CF_SENSE_PROBE_OPEN
CF_OUTPUT_PROBE_OPEN
CF_PROBES_OPEN
CF_CANNOT_TEST (ユニットがローカル・センシングに設定されているか、電池が非アクティブの場合)

Agilent MCCD はリモート電圧センス用に構成し、プローブは、プローブの導通テストを実行するためにバッテリ電池に接続しなければなりません。ローカル電圧センスがプログラムされている場合、テストは一切実行されません。

cfMeasSenseProbeResistance

構文

```
int cfMeasSenseProbeResistance(CF_HANDLE server, int cell, float
*resistance);
```

解説

注記: このコマンドの完了までには数秒かかるので、実行時間の増加を考慮して、cfSetTimeout関数を一時的に調整する必要があります。

特定の電池またはすべての電池のセンス・プローブにさかのぼる抵抗を測定して返します。データはオーム単位で返されます。引数 cell には、個々の電池番号 1～256 を指定するか、またはすべての電池の読み取りを要求するために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定する場合は、引数 resistance が、戻り値を受け取る CF_MAX_CELLS 個の実数型配列を指している必要があります。

プローブ抵抗を効率よく測定するために、電池を出力に接続してください。計測器は、センス接続の抵抗と電池の出力抵抗を区別することができません。

cfMeasVoltage

構文

```
int cfMeasVoltage(CF_HANDLE server, int cell, float *reading);
```

解説

特定の電池またはすべての電池の測定電圧をボルトで返します。電圧は、各電池の選択されたセンス端子で測定されます。引数 cell には、個々の電池番号 1～256 を指定するか、またはすべての電池の読み取りを要求するために定数 CF_ALL_CELLS を指定します。CF_ALL_CELLS を指定する場合は、引数 reading が、戻り値を受け取る CF_MAX_CELLS 個の実数型配列を指している必要があります。

cfOpen

構文

```
int cfOpen(char *server_name, CF_HANDLE *server, char *password);
```

解説

電池フォーミング（cf）関数を使用する前に、目的の電池フォーミング・サーバとの接続を確立しなければなりません。この関数は接続を確立し、他のすべての cf 関数で使用するハンドルを返します。cfOpen へのアクセスは、英数字のパスワードでプロテクトされています。パスワードが照合されてから、接続が許されます。シリアル・ポート B に接続するシリアル・ターミナルを使って、パスワードを変更することができます。パスワードの最大長は 32 文字です。引数 server_name は、IP アドレスまたはサーバ名です。

使用例

```
#include <stdio.h>
#include <mccd.h>

main()
{
        CF_HANDLE server;
        if (cfOpen("15.14.248.100", &server, "mypassword"))
                printf("Cannot connect to MCCD server\n");
}
```

cfOpenGroup

構文

```
int cfOpenGroup(CF_HANDLE server, char *name, CF_HANDLE *group_handle);
```

解説

連続制御のために CF_HANDLE を定義グループと結合します。返されたグループ・ハンドルを使って、API コマンドを指定したグループに向けることができます。

API グループをグループ・ハンドルを使って終了したときは、cfClose を呼び出し、グループ・ハンドル値を渡してハンドルを閉じます。使用可能なグループ・ハンドルの数には制限があり、API プログラムが不要になったグループ・ハンドルを解放しない場合は、プログラムが全部のハンドルを使いきってしまう可能性があります。

サーバ・ハンドル (cfOpen に対する呼び出しによって取得されたハンドル) を閉じると、そのサーバ・ハンドルに結合された全部のグループ・ハンドルが自動的に閉じます。この場合、グループ・ハンドルを明示的に閉じる必要はありません。

使用例

```
#define MY_GROUP "1.5Ahour"
/*
 * Define group named "1.5Ahour" containing 64 cells
 * starting at cell 129.
 * Define a sequence step for the group, then free the group handle.
 */
void group_example(CF_HANDLE server)
{
      CF_HANDLE group_handle;

      cfSetGroup(server, MY_GROUP, 129, 64);
      cfOpenGroup(server, MY_GROUP, &group_handle);
      cfSetSeqStep(group_handle, 1, CF_CHARGE, 5.0, 1.0, 100.0, 0.0);
      cfClose(&group_handle);
}
```

cfProtect

構文

```
int cfProtect(CF_HANDLE server);
```

解説

計測器を強制的に CF_PROTECTED ステートにします。電池出力はディスエーブルになり、電池フォーミングに関連するアクティビティはすべて中断されます。

cfProtectClear

構文

```
int cfProtectClear(CF_HANDLE server);
```

解説

保護条件が発生すると、Agilent MCCD は常に CF_PROTECTED ステートになります。このステートでは、出力はディスエーブルになり、シーケンスは一時停止します。cfProtectClear は、保護条件が存在しなくなれば、Agilent MCCD サーバを以前のステート（保護イベントの前のステート）に戻します。保護条件がまだ真の場合、cfProtectClear は Agilent MCCD サーバのステートを CF_PROTECTED のままにします。

また、cfProtectClear は、偽条件を持つ cfGentlnstStatus によって返されるワード内のすべてのビットをクリアします。真(1)である条件ビットはすべて、イベント・ワード内で真が保持されます。

cfReadMeasLog

構文

```
int cfReadMeasLog(CF_HANDLE server, CF_READP *read_pos, int cell, int step,
int bufsize, char *buffer, int *retcount);
```

解説

測定ログの内容を返します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

測定ログには、フォーミング･シーケンス中に捕捉された測定が含まれます。シーケンス・ステップ・タイプ CF_CHARGE、CF_DISCHARGE、CF_REST では、ログ入力はステップの最初と最後で実行されます。追加のログ入力は、電圧、電流、または時間が関数 cfSetMeasLogInterval で指定した基準に合致するたびに実行されます。これ以外のシーケンス・ステップ・タイプでは、特定の測定を含む 1 つの入力だけが実行されます。

引数 read_pos によって指し示された値が、ログのどの部分を読み取るかを制御します。この値が特別の値 CF_READ_FIRST である場合、ログの先頭から始まるデータが返されます。後続の cfReadMeasLog の呼び出しでは、read_pos によって指し示された場所を使用して読み取り位置がトラッキングされます。read_pos によって指し示される値が特別の値 CF_READ_LAST である場合、指定された電池に対する最後の入力が返されます。この特別の read_pos 値に、引数 cell として CF_ALL_CELLS を組み合わせると、bufsize が十分に大きければ、ログにある最後の 256 個の入力が返されます。bufsize の大きさが十分でない場合は、bufsize 分の最後の入力が返されます。

引数 cell および step は、特定の電池または特定のステップ番号に対して返されるログ入力を制限するフィルタの役割を果たします。引数 cell には、個々の電池番号 1～256 を指定するか、すべての電池のログ入力を読み取るために定数 CF_ALL_CELLS を指定します。引数 step には、個々のステップ番号を指定するか、またはすべてのステップのログ入力を読み取るために定数 CF_ALL_STEPS を指定します。その他の引数 step は、タグ付き測定を返します。引数 step を以下にまとめて示します。

<ステップ番号>	そのステップ番号に対する入力を返します
CF_ALL_STEPS	全ステップの入力を返します
CF_STEP_TRANSITIONS	各ステップの要約入力を返します
CF_TAGGED_ACR	タグ付きAC抵抗入力を返します
CF_TAGGED_DCR	タグ付きDC抵抗入力を返します
CF_TAGGED_OCV	タグ付き開回路電圧入力を返します
CF_TAGGED_CUM_AH	タグ付き累積アンペア時入力を返します
CF_TAGGED_CUM_WH	タグ付き累積ワット時入力を返します

引数 step が特別な値 CF_STEP_TRANSITIONS である場合、関数は、各ステップから非常に省略された数の入力を返します。したがって、シーケンスのクイック・サマリが取得できます。ステップ・タイプ CF_CHARGE、CF_DISCHARGE、CF_REST では、関数はステップの最初の入力とステップの最後の入力を返します。これ以外のシーケンス・ステップ・タイプでは、関数は 1 個の入力を返します。

buffer に読み込んだ文字数は、retcount に返されます。cfReadMeasLog は不完全な測定ログ入力を返さないので、読

み込んだ文字数は、通常、バッファ・サイズよりもわずかに小さくなります。retcount の値が 0 の場合は、測定ログの終わりに達しています。フォーミング・シーケンスが長く、頻繁に入力を実行するようロギング間隔が設定されている場合は、測定ログ全体の読み取りに非常に時間がかかります。読み取り速度を最大にするには、最適なバッファ・サイズを使用する必要があります。このため、ヘッダ・ファイル mccd.h にはマクロ CF_MEAS_LOG_BUFSIZE が用意されています。マクロの使用方法については、第 7 章の例 1 を参照してください。

測定ログ入力は、ASCII タブ ('＼t') 文字によって区切られた ASCII フォーマットの値の列です。各ログ入力は、改行文字 ('＼n) で終わります。測定ログ入力のフォーマットは、対応するシーケンス・ステップのステップ・タイプによって異なります。どのログ入力でも最初の 4 つの値のフォーマットは同じですが、後続の値の意味はステップ・タイプによって異なります。

CF_CHARGE、CF_DISCHARGE、または CF_REST タイプのシーケンス・ステップのフォーマットは以下のとおりです。

cell-number step-number time status entry-type volt-reading
curr-reading amp-hours watt-hours <newline>.

これ以外のシーケンス・ステップ・タイプのフォーマットは以下のとおりです。

```
cell-number step-number time status entry-type value <newline>
```

セル番号 1～256

ステップ番号 1～n。cfSetSeqStepによって定義された番号

時間 フォーミング・シーケンスがトリガされてから以降の時間 (秒)

ステータス 電池のステータスを示す値

定数	値	説明
CF_CV	1	定電圧モード
CF_CC_POS	2	定電流充電モード
CF_CC_NEG	4	定電流放電モード

入力タイプ 次の文字列のいずれか1つ:Charge(充電)、Discharge(放電)、Rest(休眠)、ACR、DCR、TaggedACR(タグ付きAC抵抗)、TaggedDCR(タグ付きDC抵抗)、Tagged OCV(タグ付き開回路電圧)、TaggedCumAH(タグ付き累積アンペア時)、TaggedCumWH(タグ付き累積ワット時)、ResetCumAH(リセット累積アンペア時)、ResetCumWH(リセット累積ワット時)

電圧の表示値 電池電圧の測定値 (V)

電流の表示値 電池電流の測定値 (A)

アンペア時 そのステップ番号の開始時から測った累積アンペア時

ワット時 そのステップ番号の開始時から測った累積ワット時

値 ステップ・タイプ (Tagged ACR、TaggedDCR、TaggedOCV、TaggedCumWH、TaggedCumAHのいずれか)に関連する測定値または値

cfInitiate 関数によって新しいフォーミング・シーケンスが開始されるまで、測定ログは計測器のメモリ内に保持されます。

cfReadSerial

構文

```
int cfReadSerial(CF_HANDLE server, CF_SERIAL_PORT port, int bufsize,
char *buffer, int *retcount);
```

解説

シリアル・ポートの 1 つからデータを読み出します。引数 port は CF_PORTA または CF_PORTB です。最大 bufsize 文字が buffer に返されます。buffer 内の文字数は retcount に返されます。読み出し可能な文字が bufsize より少ない

場合、この関数はその文字だけを返し、buffer が一杯になるまで待ちません。計測器はシリアル・ポートから読み出した文字を、コントローラが読むまで FIFO バッファに保存します。cfSerialStatus 関数を使って、シリアル・エラー条件をテストすることができます。

関連項目

```
cfGetSerialStatus
```

cfReadTestLog

構文

```
int cfReadTestLog(CF_HANDLE server, CF_READP *read_pos, int bufsize,
char *buffer, int *retcount);
```

解説

最大 bufsize 文字まで、テスト・ログの内容を返します。テスト・ログには、校正またはセルフテスト中に発生したエラーの情報が入力されています。buffer に読み込んだ文字数は、retcount に返されます。引数 read_pos によって指し示された値は、ログのどの部分を読み出すかを制御します。この値が特別な値 CF_READ_FIRST である場合、ログの先頭からデータが返されます。後続の cfReadSelftestLog の呼び出しでは、read_pos によって指し示された場所を使用して読み取り位置がトラッキングされます。引数 retcount が 0 の場合は、ログの終わりに達しています。テスト・ログのフォーマットは以下のとおりです。

```
error-number, error message <newline>
```

電源が切られるか、別の cfSelftest、cfCal、cfCalStandard、または cfCalTransfer コマンドが指定されるまで、テスト・ログは計測器で読取り可能です。

cfReset

構文

```
int cfReset(CF_HANDLE server);
```

解説

プログラム可能なファンクションのほとんどを、電源投入時のステートに設定します。このコマンドは、定義済みのシーケンス・ステップやシーケンス・テストをクリアし、進行中のシーケンスをアボートします。すべての電池出力は、OFF ステートに設定されます。

cfReset は、cfGetSerialConfig、cfReadSerial、cfGetSerialStatus、またはディジタル構成の設定には影響を与えません。また、ログやデータ・キューはクリアしません。

cfResetSeq

構文

```
int cfResetSeq(CF_HANDLE server);
```

解説

定義済みのシーケンス・ステップやシーケンス・テストをクリアし、進行中のシーケンスをアボートします。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

cfRestart

構文

```
int cfRestart(CF_HANDLE server);
```

解説

このコマンドによって、Agilent MCCD は前に保存したリスタート・ステートを呼び出します。リスタートを実行するには、Agilent MCCD が CF_IDLE ステートでなければなりません。CfGetInstStatus から返されたステータスの CF_RESTART ビットをテストすることにより、リスタート・ステートの存在を問い合わせることができます。

cfSaveOutputConfig

構文

```
int cfSaveOutputConfig(CF_HANDLE server);
```

解説

このコマンドによって各チャネルの出力構成設定が不揮発性メモリに保存されます。これらの設定は cfSetOutputConfig によって設定されます。

cfSelftest

注意: セルフテストは出力への電圧の供給を引き起こします。cfSelftestを実行する際には、接続されている電池がないことを確認してください。

構文

```
int cfSelftest(CF_HANDLE server, int *reserved);
```

解説

このコマンドは、本器のセルフテストを開始します。このセルフテストは、電源投入時に自動的に実行されるものよりも詳細です。cfSelftest 関数を呼び出す前に、電池やその他の負荷から電池供給出力を切り離してください。

この関数は、reserved 引数が指す位置に整数値を返します。戻り値は使用できませんが、値を格納する整数変数へのポインタは指定する必要があります。

セルフテストの実行にはかなり時間がかかることがあるので、cfSelftest 関数はセルフテストを起動したあと終了を待たずにただちに戻ります。セルフテスト中は、cfGetInstStatus で返されるステータス・ワードの CF_SELFTEST_STAT ビットが真になります。セルフテストが終了すると、CF_SELFTEST_STAT ビットは偽になります。セルフテスト中にエラーが発生すると、CF_SELFTEST_ERROR_STAT ビットが真になります。エラーの詳細を知るには、cfReadTestLog を使います。テスト・ログに記録されたエラー情報は、次のセルフテストまたは校正コマンドが実行されるまで保持されます。

関連項目

```
cfReadTestLog, cfGetInstStatus
```

cfSetAutoConnect

構文

```
int cfSetAutoConnect(CF_HANDLE server, CF_BOOLEAN on_off);
```

解説

このコマンドは、クライアント・コンピュータ上の mccd.dll ファイルの自動リコネクト機能をオンまたはオフにします。Agilent MCCD メインフレーム・サーバは、cfSetServerTimeout によって設定された時間より長い時間、アクティビティがないと、接続を閉じます。自動リコネクト機能を CF_ON に設定すると、API 関数呼び出しが実行されるたびに、クライアントの mccd.dll が、接続が失われた Agilent MCCD メインフレーム・サーバに自動的に再接続しようとします。自動リコネクト機能を CF_OFF に設定すると、サーバ接続が失われている場合、API 関数はエラーを返します。この場合、クライアント・プログラムは、cfSetServerTimeout によって設定された時間より短い間隔で、サーバとの通信を確認する必要があります。そうでないと、サーバはアクティビティがないために接続を閉じてしまいます。

mccd.dll を最初にロードしたときには、自動接続機能は CF_ON に設定されます。

cfSetCurrent

注意: 直接出力制御は電池の充電には使用しないでください。直接出力制御を用いた場合、過充電を防ぐことはできません。このモードは診断やデバッグにだけ使用してください。

構文

```
int cfSetCurrent(CF_HANDLE server, float current);
```

解説

診断またはデバッグを行うために、出力電流を IDLE ステートに設定します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

ステップとテストから成るシーケンスを定義せずに、Agilent MCCD の出力を直接制御して、診断やデバッグを行うことができます。直接出力制御コマンドは、CF_IDLE ステートにある間だけ使用できます。電圧、電流、および出力ステート設定は、すべての出力に対して同時に設定されます。ユニットが CF_IDLE ステートを出るたびに、設定は電源投入時の値にリセットされます。

関連項目

```
cfGetCurrent, cfSetVoltage, cfSetOutputState
```

cfSetDigitalConfig

構文

```
int cfSetDigitalConfig(CF_HANDLE server, int bitnum, CF_EXT_SIGNAL
signal, CF_POLARITY polarity, CF_REFERENCE reference);
```

解説

注記: Agilent MCCD設定画面 (第3章を参照) は、cfSetDigitalConfigの可用性を制御します。このメニュー項目がプログラマブル・アクセスに用心するよう設定されている場合、cfSetDigitalConfigはエラーCF_ACCESS_DENIEDを返します。

ディジタル I/O ポートの 16 ピンの動作を設定します。ディジタル I/O ビットは、入力または出力として使用可能な独立したシャーシ基準ビット、あるいは絶縁出力ペアとして構成できます。絶縁出力ペアとして構成した場合には、偶数番号のビットとその直後の奇数番号のビットがそれぞれペアになります。たとえば、ビット 0 とビット 1 がペアに、ビット 2 とビット 3 がペアになり、最大で 8 つのペアができます。

引数 bitnum は、プログラムされるビットを指定し、0～15 の数値を取ります。0 は、cfSetDigitalPort および cfGetDigitalPort によって設定され、読み出されるディジタル・ワードの最下位ビットを表します。引数 reference は、ビットの構成を CF_GROUNDED または CF_ISOLATED に設定します。cfSetDigitalConfig によって設定された値は、不揮発性メモリに保存されるので、電源を切ったり、cfReset を使っても影響を受けません。

*CF_GROUNDED*演算

CF_GROUNDED 演算では、引数 bitnum によって指定されたビットが、入力、出力、または入出力として使用可能なシングルエンドのシャーシ基準ビットとして構成されます。各ビットは、CF_GROUNDED で個別に構成する必要があります。

ビットの極性は、引数 polarity を用いて設定します。polarity が CF_HIGH_TRUE であり、ビットが出力として構成されている場合、cfSetDigitalPort によって設定されたビットに送られた 1 が、コネクタのハイ・レベルとして出力されます。同様に、ビットが入力として構成されている場合、cfGetDigitalPort によって、コネクタのハイ・レベル

が 1 として返されます。polarity が CF_LOW_TRUE に設定されている場合には、出力と入力の両方に逆の極性が使用されます。CF_GROUNDED 演算の引数 signal の選択肢を以下にまとめて示します。

CF_EXT_FAULT_IN	外部不良入力
CF_EXT_FAULT_OUT	外部不良出力。外部不良入力と同じ論理値を持ちます。
CF_EXTINTERLOCK	外部インターロック入力
CF_EXT_TRIGGER	外部トリガ入力
CF_DIG_IN	汎用入力
CF_DIG_OUT	汎用出力
CF_DIG_INOUT	汎用入出力
CF_POWER_FAIL_IN	外部停電入力
CF_POWER_FAIL_OUT	外部停電出力。停電シャットダウン・ステートが保存されると真になります。電源投入時、cfInitiate()、またはcfRestart()で偽になります。
CF_DIG_OUT_AND_NOT_FAULT_IN	CF_DIG_OUT関数をCF_EXT_FAULT_INからのイネーブリング・ロジックと結合します。外部不良入力が真のとき、OUTPUT_AND_NOT_FAULT_IN ピンは偽に保持されます。

出力信号がプログラムされている場合、ピンはオープン・コレクタ・トランジスタによってドライブされます。cfSetDigitalPort を使ってポートにワードを書き込むと、書き込まれたワードとビットの極性に応じて、トランジスタのオン/オフが切り替えられます。cfGetDigitalPort を使ってポートを読み取ると、最後にビットに書き込まれた値が返されます。

入力信号がプログラムされている場合、cfGetDigitalPort を使って入力ステートを読み取ることができます。ポートに書き込みを行っても、ビットには何の影響もありません。

CF_DIG_INOUT がプログラムされている場合、ビットを入力と出力の両方に使用できます。出力をハイにするワードを書き込むと、出力トランジスタがオフになるので、外部デバイスがポート・ハイまたはポート・ローをドライブすることができます。出力をローにするワードを書き込むと、トランジスタがオンになり、ポート・ビットをローにドライブします。ポートを読み取った場合、設定値ではなく、ポートの実際のステートが返されます。

注記: 以前にCF_ISOLATEDとして構成した偶数ビットをCF_GROUNDEDに設定した場合、特に設定しない限り、隣接するペア・ビット(奇数ビット)の属性はCF_GROUNDED、CF_DIG_INOUT、CF_LOW_TRUEにデフォルト設定されます。

---

*CF_ISOLATED*演算

引数 reference を CF_ISOLATED に設定した場合、偶数ビットとその直後の奇数ビットがそれぞれ組になります。各ビット・ペアに対応する出力コネクタ上のピンが、光アイソレータのプラス出力とマイナス出力になります。cfSetDigitalConfig を呼び出すときには、引数 bit には偶数ビットと奇数ビットのどちらも指定できます。しかし、cfSetDigitalPort を使って**出力の論理レベルを設定する場合、ペアのうちの奇数ビットしか使用できません。**cfSetDigitalPort は、CF_ISOLATED として構成されているペアの奇数ビットは無視します。

reference が CF_ISOLATED に設定されているビット・ペアはすべて、引数 signal を CF_DIG_OUT に設定することにより、汎用出力として使用できます。これらのペアは、引数 signal を CF_EXT_FAULT_OUT に設定すれば、絶縁された不良出力として使用することも可能です。CF_ISOLATED 演算の signal の選択肢を以下にまとめて示します。

CF_EXT_FAULT_OUT	外部不良出力。外部不良入力と同じ論理値を持ちます。
CF_DIG_OUT	汎用出力
CF_POWER_FAIL_OUT	外部停電出力
CF_DIG_OUT_AND_NOT_FAULT_IN	CF_DIG_OUT関数をCF_EXT_FAULT_INからのイネーブリング・ロジックと結合します。

CF_ISOLATED ペアの極性は、引数 polarity を使って設定します。polarity が CF_HIGH_TRUE の場合、cfSetDigitalPort によってペアの偶数ビットに送られた 1 がコネクタのハイ・レベルとして出力されます。polarity が CF_LOW_TRUE の場合には、ペアの偶数ビットに送られた 1 がコネクタのロー・レベルとして出力されます。

CF_ISOLATED に設定されているペアをディジタル入力として使用することはできません。cfGetDigitalPort によって返される CF_ISOLATED ペアに関するデータは、偶数ビットの設定値と奇数ビットの 0 から構成されます。

関連項目

```
cfSetDigitalPort, cfGetDigitalConfig
```

cfSetDigitalPort

構文

```
int cfSetDigitalPort(CF_HANDLE server, int data);
```

解説

ディジタル I/O ポートにデータを書き込みます。データは 16 ビット・バイナリ・ワードとして送信する必要があります。たとえば、値 0 を送信すると、すべてのビットがローに設定されます。値 65,535 を送信した場合、すべてのビットがハイに設定されます。以下の値を使って、個々のディジタル I/O ビットを設定してください。

ビット番号	値	ビット番号	値	ビット番号	値	ビット番号	値
0	1	4	16	8	256	12	4096
1	2	5	32	9	512	13	8192
2	4	6	64	10	1024	14	16,384
3	8	7	128	11	2048	15	32,768

複数のビットをプログラムするには、上記の値を組み合わせて使用します。たとえば、ビット 3、7、10、11 を設定するには、3208 (8+128+1024+2048) を送信します。

関連項目

```
cfGetDigitalPort, cfGetDigitalConfig
```

cfSetErrorFunction

構文

```
int cfSetErrorFunction(void(*ErrorFn)(CF_HANDLE server, char *name, int
errorcode));
```

解説

エラー関数を定義します。これは、計測器の他の関数が、0 以外の戻り値を検出したときに呼び出す関数です。引数 ErrorFn は、3 つの引数を持ち void を返す関数へのポインタです。アプリケーション・プログラムはこの機能を使って、Agilent MCCD ライブラリ関数からエラー値が返されたことを知ることができます。

ヌル・ポインタを引数 ErrorFn として関数に渡すことができます。この場合は、エラー・コールバック機能はオフになります。

サーバを扱う引数を持たないライブラリ関数がエラーを返した場合、CF_NULL_SERVER のサーバ・パラメータ値

によってエラー関数が呼び出されます。

使用例

```
void report_mccd_error(CF_HANDLE server, char *name, int error)
{
                printf("MCCD server returned error %d from API function %s\n",
                                        error, name);
}
main()
{
                /* Initialization code not shown here. */
                cfSetErrorFunction(report_mccd_error);
}
```

cfSetGroup

構文

```
int cfSetGroup(CF_HANDLE server, char *name, int start, int size);
```

解説

開始電池番号とグループ内の電池の総数を指定することによって電池のグループを定義します。name は、グループの識別に用いられるヌル終端文字列です。グループを作成したら、API 関数による後続の制御のために、cfOpenGroup を使ってハンドルを取得することができます。グループは揮発性であり、AC 電源を切ると消滅します。name 内の文字数は、CF_MAX_GROUP_NAME_LEN 未満でなければなりません。

グループ名が CF_MAX_GROUP_NAME_LEN – 1 文字より長いか、名前がヌル文字列の場合、エラーが返されます。引数 size は 0 より大きくなければなりません。そうでないと、エラーが返されます。

cfSetMeasLogInterval

構文

```
int cfSetMeasLogInterval(CF_HANDLE server, int step_number, float
 volt_interval, float curr_interval, float time_interval);
```

解説

新しい測定ログ入力が書き込まれるように、電圧、電流、および時間の変化基準を設定します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。引数 step_number には個々のステップ番号か定数 CF_ALL_STEPS を指定し、全ステップの基準を同じ値に設定します。フォーミング・シーケンスの実行中 (機器ステートが CF_FORMING の場合) は必ず、以下のいずれかが真であれば、測定ログ入力が実行されます。

- 電池の測定電圧が最後の入力からvolt_intervalボルトだけ変化した。
- 電池の測定電流が最後の入力からcurr_intervalアンペアだけ変化した。
- 最後の入力からtime_interval秒が経過した。

これらの値のいずれかを特別な値 CF_INFINITY に設定した場合、その特定パラメータに応じてロギングが効率的にオフになります。

cfSetOutputConfig

構文

```
int cfSetOutputConfig(CF_HANDLE server, int first_cell, int last_cell,
```

```
CF_OUTPUT_CONFIG config);
```

解説

出力の構成を CF_SET_ACTIVE または CF_SET_INACTIVE に設定します。first_cell から last_cell までの範囲内にある電池はすべて、要求された構成に設定されます。

CF_SET_INACTIVE に設定されている電池は、すべての出力およびフォーミング・コマンドを無視します。これらの電池の測定では、常に特別な値 CF_NOT_A_NUMBER が返されます。

注記: このコマンドは、アイドル・ステートの出力に対してのみ有効です。指定した範囲内にアイドル・ステートではない電池が1つでもあれば、エラーが返され、コマンドは無視されます。

---

cfSetOutputProbeTest

注記: 出力プローブ・テストを実行するには、Agilent MCCDをリモート電圧センシングに構成する必要があります。ローカル電圧センシングに構成されている場合、出力プローブ・テストは実行されません。

構文

```
int cfSetOutputProbeTest(CF_HANDLE server, float resistance);
```

解説

このコマンドは、フォーミング・シーケンスの出力プローブ・テスト時に使用される抵抗制限を設定します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。出力プローブの抵抗は、パワー・プローブとセンス・プローブの電圧の差が 50mV を超える限り、フォーミング中に定期的に測定されます。プローブ抵抗が設定された値を上回ると、その電池には不良のマークが付けられます。CF_INFINITY の抵抗値を送信することによって、出力プローブ抵抗の自動チェックをディスエーブルにすることができます。

cfSetOutputState

注意: 直接出力制御は電池の充電には使用しないでください。直接出力制御を用いた場合、過充電を防ぐことはできません。このモードは診断やデバッグにだけ使用してください。

構文

```
int cfSetOutputState(CF_HANDLE server, CF_OUTPUT_STATE state);
```

解説

診断またはデバッグのための出力ステートを設定します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

CF_OUTPUT_OFF のステート値は、電池の出力ステートを OFF に設定します。CF_OUTPUT_CHARGE は、充電できるように出力を設定します。CF_OUTPUT_DISCHARGE は、放電 (シンク電流) できるように出力を設定します。OFF ステートでは、チャネル出力は開放回路になるので、電流は供給されません。充電ステートと放電ステートでは、チャネル出力は cfSetVoltage と cfSetCurrent によって制御されます。

ステップとテストから成るシーケンスを定義せずに、Agilent MCCD の出力を直接制御して、診断やデバッグを行うことができます。直接出力制御コマンドは、CF_IDLE ステートにある間だけ使用できます。電圧、電流、および出力ステート設定は、すべての出力に対して同時に設定されます。ユニットが CF_IDLE ステートを出るたびに、設

定は電源投入時の値にリセットされます。

関連項目

```
cfGetOutputState, cfSetVoltage, cfSetCurrent
```

cfSetSense

構文

```
int cfSetSense(CF_HANDLE server, CF_SENSE sense);
```

解説

電圧センスをリモート・センスまたはローカル・センスに設定します。引数 sense は、CF_SENSE_REMOTE または CF_SENSE_LOCAL を取ります。センス設定は、不揮発性メモリに保存されるので、AC 電源をオフにしても保持されます。

関連項目

```
cfGetSense
```

cfSetSenseProbeTest

注記: 出力プローブ・テストを実行するには、Agilent MCCDをリモート電圧センシングに構成する必要があります。ローカル電圧センシングに構成されている場合、出力プローブ・テストは実行されません。

構文

```
int cfSetSenseProbeTest(CF_HANDLE server, CF_BOOLEAN on_off);
```

解説

リモート・センス・プローブ抵抗の自動テストをイネーブルまたはディスエーブルにします。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

このテストがイネーブルになっているときは、計測器はシーケンス中にセンス・プローブの抵抗を定期的に測定し、正確な電圧測定のできる低い値であるかチェックします。プローブ抵抗があまりにも高く、しかもテストがイネーブル状態にある場合、その電池のフォーミング・シーケンスは終了されます。

計測器は、センス・プローブの抵抗と電池の出力抵抗を区別することができません。出力電圧および電流が抵抗測定に十分でなければ、センス・プローブ抵抗のチェックは行われません。

cfSetSeqStep

構文

```
int cfSetSeqStep(CF_HANDLE server, int step_number, CF_SEQ_OUT out_type, float
voltage, float current, float time, float reserved);
```

解説

出力シーケンス・ステップを、出力レギュレーション・タイプ、電圧制限、電流制限、時間、および今後の拡張用に確保されている追加引数によって定義します(未使用のパラメータを 0 にプログラム設定します)。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。出力レギュレーション・タイプとしては以下のものがあります。

CF_CHARGE	定電圧/定電流充電
CF_DISCHARGE	定電圧/定電流放電
CF_REST	休眠 (ハイ・インピーダンス・ステートで出力)
CF_ACR	AC抵抗測定
CF_DCR	DC抵抗測定
CF_TAG_ACR	測定ログでTaggedACR入力タイプとして識別されるAC抵抗測定の実行
CF_TAG_DCR	測定ログでTaggedDCR入力タイプとして識別されるDC抵抗測定の実行
CF_TAG_OCV	測定ログでTaggedOCV入力タイプとして識別される開回路電圧測定の実行
CF_TAG_CUM_AH	測定ログへの累積アンペア時のTaggedCumAH入力タイプとしての書き込み
CF_TAG_CUM_WH	測定ログへの累積ワット時のTaggedCumAH入力タイプとしての書き込み
CF_RESET_CUM_AH	累積アンペア時測定のゼロへのリセット
CF_RESET_CUM_WH	累積ワット時測定のゼロへのリセット

2 つのクラスのステップ・タイプがあります。充電、放電、休眠の各ステップ・タイプは、電池への特定のスティミュラスで時間周期を定義します。これらのステップ・タイプの場合、ステップの最大持続時間は、秒単位の引数 time によって指定されます。時間周期が経過する前に次のステートへ移行した遷移は、関数 cfSetSeqTest によって定義されたアクションによる影響を受けます。

その他のステップ・タイプは、測定の制御と測定ログへの入力の生成に使用されます。これらのステップ・タイプでは、引数 time は無視され、ステップの持続時間は、単にステップに付随するアクションの実行に必要な時間となります。たとえば、ACR 測定と DCR 測定をシーケンスの特定のステップで実行することができます。これらの測定には、タグ付き測定の場合とタグなし測定の場合があります。タグ付き測定は、タグ付き測定の入力だけを返すクウェリ・フィルタを使って、測定ログから検索できます。タグなし測定もログから読み取り可能ですが、これらの測定を選択的に問い合わせることはできません。測定値は、ログ全体を読み取ったときに、その他のすべての入力と一緒に返されます。

Agilent MCCD は、電池の容量をアンペア時とワット時の両方で測定します。これらの容量は、各シーケンス・ステップの最初にゼロにリセットされます。容量は、各測定ログ入力に標準測定の一部としてレポートされます。Agilent MCCD には、複数のシーケンス・ステップにわたって容量を累積する能力もあります。これらの蓄積容量をリセットし、シーケンスの特定ステップで蓄積容量を測定ログに送信するための特別のステップ・タイプが用意されています。容量はタグ付き測定であり、クウェリ・フィルタを使って検索されます。

シーケンス・ステップはほとんど編集できません。定義済みの step_number を送信すると、前の設定値が新しい step_number の設定値で置き換えられます。2 つの定義済みのステップの間に新たなステップを挿入したり、1 つのステップを削除したりする方法はありません。cfResetSeq は、シーケンスのステップとテストをすべて削除するため、シーケンス・ステップおよびテストの新しいセットを定義する前に送信してください。フォーミング・シーケンスの実行中に cfSetSeqStep を使用すると、エラーが返されます。

関連項目

```
cfReset, cfSetSeqTest, cfGetSeqStep
```

cfSetSeqTest

構文

```
int cfSetSeqTest(CF_HANDLE server, int step_number, CF_SEQ_TEST meas_test_type,
float limit, CF_TIME_TEST time_test_type, float time, CF_SEQ_ACTION action);
```

解説

シーケンス・ステップ中に実行するテストを定義します。これらのテストでは、測定値が得られた場合、電池は次のステップに進みます。不良制限を超えた場合に電池は不良品となり、その後のスティミュラスから除去されます。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

step_number は、cfSetSeqStep の対応する番号を参照します。

meas_test_type は以下のいずれかを取ります。

CF_VOLT_GE	電池電圧はプログラムされた**limit**以上
CF_VOLT_LE	電池電圧はプログラムされた**limit**以下
CF_CURR_GE	電池電流はプログラムされた**limit**以上
CF_CURR_LE	電池電流はプログラムされた**limit**以下
CF_ACR_GE	電池のAC抵抗はプログラムされた**limit**以上
CF_ACR_LE	電池のAC抵抗はプログラムされた**limit**以下
CF_DCR_GE	電池のDC抵抗はプログラムされた**limit**以上
CF_DCR_LE	電池のDC抵抗はプログラムされた**limit**以下
CF_POWER_GE	ワットで表した電池電力 (電池電圧×電池電流) の絶対値はプログラムされた**limit**以上
CF_POWER_LE	ワットで表した電池電力 (電池電圧×電池電流) の絶対値はプログラムされた**limit**以下
CF_AMPH_GE	アンペア時で表した電池容量の絶対値はプログラムされた**limit**以上
CF_AMPH_LE	アンペア時で表した電池容量の絶対値はプログラムされた**limit**以下
CF_WATTH_GE	ワット時で表した電池容量の絶対値はプログラムされた**limit**以上
CF_WATTH_LE	ワット時で表した電池容量の絶対値はプログラムされた**limit**以下
CF_POS_DVDT_GE	標準測定間隔中の電圧変化は正で、プログラムされた**limit**以上
CF_POS_DVDT_LE	標準測定間隔中の電圧変化は正で、プログラムされた**limit**以下
CF_NEG_DVDT_GE	標準測定間隔中の電圧変化は負。変化の振幅はプログラムされた**limit**以上
CF_NEG_DVDT_LE	標準測定間隔中の電圧変化は負。変化の振幅はプログラムされた**limit**以下
CF_POS_DIDT_GE	標準測定間隔中の電流振幅の変化は正で、プログラムされた**limit**以上
CF_POS_DIDT_LE	標準測定間隔中の電流振幅の変化は正で、プログラムされた**limit**以下
CF_NEG_DIDT_GE	標準測定間隔中の電流振幅の変化は負。変化の振幅はプログラムされた**limit**以上
CF_NEG_DIDT_LE	標準測定間隔中の電流振幅の変化は負。変化の振幅はプログラムされた**limit**以下
CF_DVMAX_GE	電圧とステップ中に観察された最大電圧との差の振幅はプログラムされた**limit**以上

CF_DVMAX_LE	電圧とステップ中に観察された最大電圧との差の振幅はプログラムされた**limit**以下
CF_DVMIN_GE	電圧とステップ中に観察された最小電圧との差の振幅はプログラムされた**limit**以上
CF_DVMIN_LE	電圧とステップ中に観察された最小電圧との差の振幅はプログラムされた**limit**以下
CF_DIMAX_GE	電流の絶対値とステップ中に観察された電流の最大絶対値との差の振幅はプログラムされた**limit**以上
CF_DIMAX_LE	電流の絶対値とステップ中に観察された電流の最大絶対値との差の振幅はプログラムされた**limit**以下
CF_DIMIN_GE	電流の絶対値とステップ中に観察された電流の最小絶対値との差の振幅はプログラムされた**limit**以上
CF_DIMIN_LE	電流の絶対値とステップ中に観察された電流の最小絶対値との差の振幅はプログラムされた**limit**以下

time_test_type は以下のいずれかを取ります。

CF_TEST_BEFORE	引数**time**より前に測定テストが真になった場合に、動作が行われます。
CF_TEST_AT	引数**time**の時点で測定テストが真である場合に、動作が行われます。
CF_TEST_AFTER	引数**time**より後に測定テストが真になった場合に、動作が行われます。
CF_TEST_BEFORE_STIMULUS	このステップにスティミュラスが供給される**前に**測定テストが真になった場合に、動作が行われます。出力は、このテスト中、リセット状態にあります。このtime_test_typeの場合、引数**time**は無視されます。

time は、現在のステップ番号の開始時から秒単位で測ります。

action は、meas_test_type と time_test_type の両方が満たされた場合に行われる動作で、以下の選択肢があります。

CF_NEXT	現在のステップにある残りのテストを省略し、できるだけ早く次のステップに進みます。
CF_FAIL	電池を不良品としてマークし、それを切り離します。その電池に対し、以降のテストやステップは適用されません。

注記: CF_ACR_LEとCF_ACR_GEは、CF_ACRステップだけに使用可能です。
CF_DCR_LEとCF_DCR_GEは、CF_DCRステップだけに使用可能です。
その他のmeas_test_typeは、CF_ACRやCF_DCRステップには使用できません。

単一のテストを削除することはできません。cfResetSeqは、すべてのシーケンス・ステップおよびテストを削除するので、一連の新しいシーケンス・ステップおよびテストを定義する前に送信する必要があります。フォーミング・シーケンスの進行中にcfSetSeqTestを使用した場合には、エラーが返されます。

cfSetSeqTestAnd

構文

```
int cfSetSeqTestAnd(CF_HANDLE server, int step_number, CF_SEQ_TEST
*meas_test_type,  float  *limit,  CF_TIME_TEST  time_test_type,  float  time,
CF_SEQ_ACTION action, int count);
```

解説

このコマンドは cfSetSeqTest と同様ですが、シーケンス中に論理積で結合された複数のテストが実行可能です。すべての測定テストが真になるまで動作は実行されません。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。引数 count は、配列引数 meas_tests_type および limit に格納されるテストの数を定義します。

meas_tests_type[]配列と limit[]配列に渡すことができるテストの最大数は、マクロ CF_MAX_AND_TESTS によって定義されます。

使用例

```
/* This example defines a test that will advance to the next step when
 * the voltage is less than 3.1 V and the current is less than 0.5A
 */
CF_SEQ_TEST meas_test[CF_MAX_AND_TESTS];
float limit[CF_MAX_AND_TESTS];
meas_test[0] = CF_VOLT_LE;
limit[0] = 3.1;
meas_test[1] = CF_CURR_LE;
limit[1] = 0.5;
cfSetSeqTestAnd(server, 1, meas_test, limit CF_TEST_AFTER, 0.0,
     CF_NEXT, 2);
```

cfSetSerialConfig

構文

```
int cfSetSerialConfig(CF_HANDLE server, CF_SERIAL_PORT port, int baudrate,
CF_SERIAL_PARITY parity, int wordsize, CF_SERIAL_FLOW flow_ctrl);
```

解説

1 つのシリアル・ポートの通信パラメータを設定します。

Port	CF_PORTAまたはCF_PORTB
Baudrate	1200, 2400, 4800, 9600, または 19200
Parity	CF_PARITY_EVEN, CF_PARITY_ODD, または CF_PARITY_NONE
Wordsize	7 または8
flow_ctrl	CF_FLOW_RTS_CTS, CF_FLOW_DSR_DTR, CF_FLOW_XON_XOFF, または CF_FLOW_NONE

cfSetServerTimeout

構文

```
int cfSetServerTimeout(CF_HANDLE server, float timeout);
```

解説

このコマンドは、接続の非アクティビティのタイムアウト周期を設定します。Agilent MCCD サーバは、タイムアウト値より長い時間アクティビティがないと、接続を閉じます。電源投入時と cfReset 後、タイムアウト値は 60 秒に設定されます。

cfSetShutdownDelay

構文

```
int cfSetShutdownDelay(CF_HANDLE server, float delay);
```

解説

シャットダウン・モードが CF_AUTO に設定されているときに、CF_POWER_FAIL_IN 入力における真信号の表明と Agilent MCCD シャットダウンの開始との間の遅延を設定します。電源投入時には、このタイムアウトは 60 秒に設定されます。シャットダウン・モードを CF_MANUAL に設定していると、遅延は影響しません。

cfSetShutdownMode

構文

```
int cfSetShutdownMode(CF_HANDLE server, int mode);
```

解説

シャットダウン・モードを CF_AUTO または CF_MANUAL に設定します。CF_AUTO に設定すると、有効なリスタート・ステートがまだ存在しない場合、CF_POWER_FAIL_IN 入力の真信号によって、シャットダウン遅延が経過すると Agilent MCCD はシャットダウンを実行します。CF_MANUAL に設定すると、Agilent MCCD は、真 CF_POWER_FAIL_IN 入力に応答して自動シャットダウンを実行しません (ただし、機器ステータスの CF_POWER_FAIL_STAT ビットに入力ステートをレポートします)。

Agilent MCCD は、前に保存したステートがまだ存在する場合、リスタート・ステートを自動的に保存しません。リスタート・ステートは、cfInitiate と cfRestart で削除されます。リスタート・ステートが存在するか問い合わせるには、cfGetInstStatus から返されるステータスの CF_RESTART ビットをテストします。

cfSetShutdownMode の電源投入時の設定は CF_MANUAL です。

cfSetTimeout

構文

```
int cfSetTimeout(float timeout);
```

解説

クライアントが Agilent MCCD サーバからの応答を待つ時間の最大秒数を設定します。はっきり設定されていない場合のデフォルトのタイムアウト値は 30 秒です。

cfSetTrigSource

構文

```
int cfSetTrigSource(CF_HANDLE server, CF_TRIG_SOURCE source);
```

解説

シーケンスのトリガ源を、CF_LAN または CF_EXTERNAL に設定します。CF_LAN に設定すると、cfTrigger 関数を使ってフォーミング・シーケンスにトリガをかけることができます。CF_EXTERNAL に設定すると、External Trigger にマップされているディジタル入力の真信号がトリガになります。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

cfSetVoltage

注意: **直接出力制御は電池の充電には使用しないでください。直接出力制御を用いた場合、過充電を防ぐことはできません。このモードは診断やデバッグにだけ使用してください。**

構文

```
int cfSetVoltage(CF_HANDLE server, float voltage);
```

解説

診断またはデバッグのための出力電圧を設定します。引数 server には、cfOpenGroup によって取得されたグループ

に対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

ステップとテストから成るシーケンスを定義せずに、Agilent MCCD の出力を直接制御して、診断やデバッグを行うことができます。直接出力制御コマンドは、CF_IDLE ステートにある間だけ使用できます。電圧、電流、および出力ステート設定は、すべての出力に対して同時に設定されます。ユニットが CF_IDLE ステートを出るたびに、設定は電源投入時の値にリセットされます。

cfShutdown

構文

```
int cfShutdown(CF_HANDLE server);
```

解説

このコマンドによって Agilent MCCD は以下を実行します。

- CF_PROTECTED ステートに進みます (cfGetInstStatus によって返されるステータスで CF_SHUTDOWN_STAT を真にします)。
- 後で cfRestart と使用するため、現在のステートを不揮発性メモリに保存します。
- CF_POWER_FAIL_OUT ディジタル出力信号を表明します。

保存されたステートの情報には、すべてのシーケンス・ステップとテストが含まれます。シーケンスが動作中の場合 (CF_FORMING ステート)、情報には、現在のステップ番号、ステップ内の時間、容量、合格/不合格ステータスなど、フォーミング・シーケンスにおける各電池の現在のステートが含まれます。測定ログも保存されます。Agilent MCCD が CF_IDLE の状態にある場合は、前のシーケンスの合格/不合格結果とその測定ログが保存されます。

cfStateDelete

構文

```
int cfStateDelete(CF_HANDLE server, char *state);
```

解説

以前に cfStateSave で作成した機器ステートを削除します。

cfStateList

構文

```
int cfStateList(CF_HANDLE server, char *buffer);
```

解説

サーバに保存されている機器ステート名を、カンマで区切られ、ヌルで終わるリストで返します。buffer は、電池フォーミング・サーバに保存できる最大ステート数の名前リストを保持できる大きさでなければなりません。定数 CF_MAX_STATE_LIST_LEN を使って、buffer のスペースを割り当てることができます。

使用例

```
char list_buffer[CF_MAX_STATE_LIST_LEN];

/* Read and print the list of instrument state names. */
cfStateList(server, list_buffer);
printf("%s\n", list_buffer);
```

cfStateRecall

構文

```
int cfStateRecall(CF_HANDLE server, char *state);
```

解説

以前に cfStateSave で作成した機器ステートをロードします。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。計測器のプログラム可能なファンクションは、すべて state に保存された値に設定されます。また、このコマンドは進行中のフォーミング・シーケンスをアボートし、フォーミング・ステートを CF_IDLE にします。

cfStateSave

構文

```
int cfStateSave(CF_HANDLE server, char *state);
```

解説

現在の機器設定を、state パラメータによって指定された名前で不揮発性メモリに保存します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。与えられた名前のステートがすでに存在している場合には、古いステートが上書きされます。ステート名の長さは、CF_MAX_STATE_NAME_LEN 分の文字数に限られています。

cfTrigger

構文

```
int cfTrigger(CF_HANDLE server);
```

解説

トリガ源 CF_LAN からトリガを送信します。引数 server には、cfOpenGroup によって取得されたグループに対するハンドル、またはグループが定義されていない場合、計測器内のすべての電池に対するハンドルを指定します。

cfWriteSerial

構文

```
int cfWriteSerial(CF_HANDLE server, CF_SERIAL_PORT port, char *port_data, int
count);
```

解説

カウント・データ・ワードをシリアル・ポートに書き込みます。

関連項目

```
cfSerialStatus, cfReadSerial, cfSerialConfig
```

7

Cプログラム例

例1

次のCプログラムは、第5章の初めで説明した例を、API電池フォーミング (cf) 関数を使用して実現する方法を示したものです。電池フォーミング関数は、Agilent E4373Aマニュアル・パッケージに付属のドライバ・ソフトウェアに含まれています。

```
#include <stdio.h>
#include <mccd.h>

#define SECONDS_PER_MINUTE 60.0f

void setup(CF_HANDLE server);

main()
{
    CF_HANDLE server;
    int err_ret;
    char buf[CF_MEAS_LOG_BUFSIZE];
    int retcount;
    FILE *fp;
    CF_READP read_pos;
    CF_RUN_STATE presentState;

    /* Open the server connection */
    if (cfOpen("15.14.248.100", &server, "mypassword")) {
        printf("Cannot connect to Agilent MCCD\n");
        exit(1);
    }

    /* Reset the server to power-on defaults */
    cfReset(server);

    /* Set the trigger source to LAN */
    cfSetTrigSource(server, CF_LAN);

    /* Program the charge/discharge sequence */
    setup(server);

    /* Initiate sequence and check for sequence consistency */
    if (err_ret = cfInitiate(server)) {
        printf("Initiate error, code %d\n", err_ret);
        exit(err_ret);
    }

    /* Wait for runstate CF_INITIATED. */
```

```
    do {
        Sleep(1000);
        cfGetRunState(server, &presentState);
    } while(presentState != CF_INITIATED);

    /* Start the sequence */
    cfTrigger(server);

    /* Wait for the sequence to end */
    do {
        cfGetRunState(server, &presentState);
        /* sleep or do something else */
    } while(presentState == CF_FORMING);

    /* Read entire measurement log and write it to a disk file */
    fp = fopen("logfile", "w");
    for (read_pos = CF_READ_FIRST; ; ) {
        cfReadMeasLog(server, &read_pos, CF_ALL_CELLS, CF_ALL_STEPS,
                CF_MEAS_LOG_BUFSIZE, buf, &retcount);
        if (retcount)
                fputs(buf, fp);
        else
                break;
    }
    fclose(fp);

    /* Close the server connection */
    cfClose(server);

    return(0);
}

/* Program the charge/discharge sequence */
void setup(CF_HANDLE server)
{
    /* Step 1 and tests. */
    cfSetSeqStep(server, 1, CF_CHARGE, 4.2, 0.295,
                20 * SECONDS_PER_MINUTE, 0.0);
    cfSetSeqTest(server, 1, CF_VOLT_GE, 3.8, CF_TEST_BEFORE,
                5 * SECONDS_PER_MINUTE, CF_FAIL);
    cfSetSeqTest(server, 1, CF_CURR_LE, 0.02, CF_TEST_AFTER,
                5 * SECONDS_PER_MINUTE, CF_NEXT);

    /* Step 2 is a rest step. */
    cfSetSeqStep(server, 2, CF_REST, 0.0, 0.0);
                10 * SECONDS_PER_MINUTE, 0.0);

    /* Step 3 and tests. */
    cfSetSeqStep(server, 3, CF_DISCHARGE, 3, 0.295,
                15 * SECONDS_PER_MINUTE, 0.0);
    cfSetSeqTest(server, 3, CF_VOLT_LE, 3, CF_TEST_BEFORE,
                5 * SECONDS_PER_MINUTE, CF_FAIL);
    cfSetSeqTest(server, 3, CF_VOLT_LE, 3, CF_TEST_AFTER,
                5 * SECONDS_PER_MINUTE, CF_NEXT);
    cfSetSeqTest(server, 3, CF_VOLT_GE, 3, CF_TEST_AT,
                15 * SECONDS_PER_MINUTE, CF_FAIL);
```

```
/* Step 4 is a rest step. */
cfSetSeqStep(server, 4, CF_REST, 0.0, 0.0);
                5 * SECONDS_PER_MINUTE, 0.0) ;
   }
```

例2

次のCプログラムは、第1章の終わりに説明した例を、API電池フォーミング関数を使用して実現する方法を示したものです。本例には簡単な電池フォーミング・シーケンスしか含まれておらず、各関数呼び出し後のエラー・チェックは含まれていません。本例は、主として、ディジタルI/Oやシリアル・ポートなどのAgilent MCCDの各種ハイ・レベル機能を電池フォーミング・シーケンスに組み込む方法を示しています。

シリアル・ポートAは、パススルー・モードで動作するようにプログラム設定されているので、ポートAに接続されているバーコード・リーダからの情報はPCに直接送られます。

第1章の終わりに示した例に合わせるために、ディジタル・ポートは次のようにプログラムされています。

ピン0、1、2は、負論理汎用ディジタル入力としてプログラム設定されています。これらのディジタル入力を使用するには、ディジタル・ポートのスイッチをグランドに接続します。ピン0はフィクスチャ・スイッチに、ピン1はStartボタンに、ピン2は煙検出器のスイッチにそれぞれ接続されます。スイッチが開いている場合、内部+5Vプルアップ抵抗は、これらの入力をハイすなわちFALSEに設定します。スイッチが閉じている場合、ディジタル入力は共通すなわち負論理に接続されます。

ピン8および9は、負論理汎用出力としてプログラム設定されています。ディジタル出力を使用するには、ディジタル・ポートのライトを補助出力などの+24V電源に接続します。ピン8はReadyライトに接続されます。ピン9はTestライトに接続されます。0 (FALSE) にプログラム設定された場合、ディジタル・ポートは開放回路になり、ライトはオフになります。1 (TRUE) にプログラム設定された場合、ディジタル・ポートは共通端子に接続され、ライトはオンになります。

```
/* sample1.c */

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <io.h>
#include "mccd.h"

/* Default server address and password */
#define DEFAULT_SERVER "15.14.250.125"
#define DEFAULT_PASSWORD "mufasa"

/* Digital port bit definitions */
#define DIG_FIXTURE_READY   0x0001
#define DIG_START_BUTTON    0x0002
#define DIG_SMOKE_DETECT    0x0004
#define DIG_READY_LIGHT     0x0100
#define DIG_TEST_LIGHT          0x0200

#define CLIENT_TIMEOUT 60.0f
#define MAX_BARCODE 256
#define LOG_FILE "MCCDLOG.TXT"
#define MEAS_BUF_SIZE 1024

/* Local function prototypes */
void APIError(CF_HANDLE hServer, char *szName, int nError);
char *RunStateToString(CF_RUN_STATE state);

/*****************************************************************
Main function
*****************************************************************/

void main(int argc, char *argv[])
{
    char *szServerAddr = DEFAULT_SERVER;
```

```
char *szPassword = DEFAULT_PASSWORD;
CF_HANDLE hServer;
int nResult;
int nDigitalPort;
char szBarCodeMsg[MAX_BARCODE];
int nBarCodeCount;
int nRunState;
CF_READP readPos;
FILE *hFile;
char szMeasBuffer[MEAS_BUF_SIZE];
int nMeasBufCount;
float fCellResistance[CF_MAX_CELLS];

/* Open a connection to an MCCD. */
nResult = cfOpen(szServerAddr, &hServer, szPassword);
if (nResult != CF_OK)
{
    printf("Could not connect to MCCD: %s\n", szServerAddr);
    APIError(hServer, "cfOpen", nResult);
}
printf("Connected to MCCD: %s\n", szServerAddr);

/*
 * Setup the client API DLL.
 */

/* Install a central error handler function. */
cfSetErrorFunction(APIError);

/* Set the client timeout period. */
cfSetTimeout(CLIENT_TIMEOUT);

/*
 * Setup the MCCD server.
 */

/* Reset the server to power-on defaults. */
cfReset(hServer);

/* Set voltage sense to remote. */
cfSetSense(hServer, CF_SENSE_REMOTE);

/* Set trigger source to LAN. */
cfSetTrigSource(hServer, CF_LAN);

/* Set measurement log intervals for all sequence steps. */
cfSetMeasLogInterval(hServer, CF_ALL_STEPS, 0.1f, 0.1f, CF_INFINITY);

/* Set serial port A configuration to use with bar code reader. */
cfSetSerialConfig(hServer, CF_PORTA, 9600,
                  CF_PARITY_NONE, 8, CF_FLOW_NONE);

/* Enable probe tip checking. */
cfSetSenseProbeTest(hServer, CF_ON);

/* Configure resistance limit for output probe test. */
cfSetOutputProbeTest(hServer, 0.1f);

/* Mark outputs 65 - 256 as unused by this fixture. */
cfSetOutputConfig(hServer, 1, 14, CF_SET_ACTIVE);
cfSetOutputConfig(hServer, 15, 256, CF_SET_INACTIVE);

/*
 * Turn on the fixture ready light to tell the operator that the
 * system is ready for a new tray of cells.
 */
cfGetDigitalPort(hServer, &nDigitalPort);
```

```
nDigitalPort |= DIG_READY_LIGHT;
cfSetDigitalPort(hServer, nDigitalPort);

/*
 * Poll serial port A for data from the bar code reader.
 */
while (1)
{
    cfReadSerial(hServer, CF_PORTA, MAX_BARCODE,
                szBarCodeMsg, &nBarCodeCount);
    /* Check the data for a token that indicates end of data.
     * When token is found, break out of loop. (not shown)
     */
    break;
    /* Sleep for 1 second to suspend this process, but allow
     * other processes to continue to run.
     */
    Sleep(1000);
}

/*
 * Process the barcode message to determine what forming sequence
 * should be downloaded to the fixture. (not shown in this example)
 */

/* Download the forming sequence. */

cfSetSeqStep(hServer, 1, CF_CHARGE, 4.0f, 1.0f, 300.0f, 0.0f);
cfSetSeqStep(hServer, 2, CF_REST, 0.0f, 0.0f, 60.0f, 0.0f);
cfSetSeqStep(hServer, 3, CF_DISCHARGE, 0.5f, 2.0f, 120.0f, 0.0f);

cfSetSeqTest(hServer, 1, CF_VOLT_LE, 0.5f, CF_TEST_AFTER, 120.0f, CF_FAIL);
cfSetSeqTest(hServer, 1, CF_VOLT_GE, 4.0f, CF_TEST_AFTER, 0.0f, CF_NEXT);
cfSetSeqTest(hServer, 3, CF_VOLT_LE, 1.0f, CF_TEST_AFTER, 0.0f, CF_NEXT);

/* Poll the fixture; wait until it is closed. */
while (1)
{
    cfGetDigitalPort(hServer, &nDigitalPort);
    if (nDigitalPort & DIG_FIXTURE_READY)
        break;
    /* Sleep for 1 second. */
    Sleep(1000);
}

/*
 * Poll digital port for START button pressed.
 * Since lines are not latched, must press button > 1 second.
 */
while (1)
{
    cfGetDigitalPort(hServer, &nDigitalPort);
    if (nDigitalPort & DIG_START_BUTTON)
        break;
    /* Sleep for 1 second. */
    Sleep(1000);
}

/*
 * Turn off the READY light and turn on the TEST light
 * to indicate to the operator that cell forming has started.
 */
cfGetDigitalPort(hServer, &nDigitalPort);
nDigitalPort &= ~DIG_READY_LIGHT;
nDigitalPort |= DIG_TEST_LIGHT;
cfSetDigitalPort(hServer, nDigitalPort);
```

```
    /* Initiate the sequence. */
    cfInitiate(hServer);

    /*
     * Wait for sequencer to transition to initiated state.
     * This may take up to 1 minute while data logs are erased.
     */
    while (1)
    {
        cfGetRunState(hServer, &nRunState);
        if (nRunState == CF_INITIATED)
            break;
        Sleep(1000);
    }

    /* Trigger the sequence. */
    cfTrigger(hServer);
    printf("Forming sequence started.\n");


    /* The sequence is now running.
     * Display the sequencer state until it finishes.
     */
    while (1)
    {
        Sleep(5000);
        cfGetRunState(hServer, &nRunState);
        printf("Runstate = %s\r", RunStateToString(nRunState));
        if (nRunState == CF_IDLE)
            break;
    }
    /* The sequence is finished. */

    /* Read the entire measurement log to a file. */
    if((hFile = fopen(LOG_FILE, "w")) != NULL)
    {
        readPos = CF_READ_FIRST;
        while (1)
        {
            cfReadMeasLog(hServer, &readPos, CF_ALL_CELLS, CF_ALL_STEPS,
                MEAS_BUF_SIZE, szMeasBuffer, &nMeasBufCount);
            if (nMeasBufCount == 0)
                break;
            fwrite(szMeasBuffer, sizeof(char), nMeasBufCount, hFile);
        }
        fclose(hFile);
    }

    /* Measure the internal resistance of all cells. */
    cfMeasACResistance(hServer, CF_ALL_CELLS, fCellResistance);

    /*
     * Turn off the TEST light and turn on the READY light
     * to indicate that it the tray can be removed from the fixture.
     */
    cfGetDigitalPort(hServer, &nDigitalPort);
    nDigitalPort &= ~DIG_TEST_LIGHT;
    nDigitalPort |= DIG_READY_LIGHT;
    cfSetDigitalPort(hServer, nDigitalPort);
    printf("Forming sequence complete.\n");

    /* Close the server connection. */
    cfClose(hServer);
    exit(0);
}
```

```
/*********************************************************************
 API error handler.
*********************************************************************/
void APIError(CF_HANDLE hServer, char *szName, int nError)
{
    printf("\nServer = %d Function = %s Error = %d\n",
            hServer, szName, nError);
    cfClose(hServer);
    exit(1);
}

/*********************************************************************
Convert sequence run state to a description string.
*********************************************************************/
char *RunStateToString(CF_RUN_STATE state)
{
    switch (state) {
    case CF_NOT_READY:
        return "CF_NOT_READY  ";
    case CF_IDLE:
        return "CF_IDLE       ";
    case CF_ERASING:
        return "CF_ERASING    ";
    case CF_INITIATED:
        return "CF_INITIATED  ";
    case CF_FORMING:
        return "CF_FORMING    ";
    case CF_INTERLOCKED:
        return "CF_INTERLOCKED";
    case CF_PROTECTED:
        return "CF_PROTECTED  ";
    case CF_HW_FAILED:
        return "CF_HW_FAILED  ";
    }
    return("Unknown state");
}
```

例3

1台のPCから最大16のAgilent MCCDを制御でき、アプリケーションのプログラム構造に応じて、良好なシステム応答を達成できます。

以下のCプログラム例では、マルチスレッド・プログラムを使用しています。プログラムでは、各スレッドは、1つのAgilent MCCDを独立して制御でき、しかもユーザ・インタフェース・スレッドとデータを共有できます。これにより、各スレッドがAgilent MCCDを 1 つだけ制御するという単純な構造で、すべてのAgilent MCCDの中央システム・ビューが得られます。このタイプのプログラムでは、同期オブジェクトを使って共有データへアクセスする際に注意が必要です。結局、複数のAgilent MCCDを制御するには、シングルスレッド・プログラムを記述するより、マルチスレッド・プログラムを使用する方が簡単です。

```
#include <windows.h>
#include <stdio.h>
#include <time.h>
#include "mccd.h"

#define MEAS_BUF_SIZE CF_MEAS_LOG_BUFSIZE
#define LINE_SIZE 80
#define PASSWORD "mufasa"

// Structure for thread information
typedef struct {
    char *szAddr;
    char *szLogFile;
    HANDLE hStart;
} THREAD_INFO;

THREAD_INFO ThreadInfo[] = {
{"15.14.250.130", "mccdlog0.txt", NULL},
{"15.14.250.124", "mccdlog1.txt", NULL},
};

// Local function prototypes
void GetTimeStamp(char *szTimeStamp);
void ErrorHandler(CF_HANDLE server, char *function, int errorcode);
DWORD WINAPI ReadThread(LPVOID lpvThreadParm);

/*****************************************************************************
 Main function
*****************************************************************************/
void main (int argc, char *argv[])
{
    DWORD dwThreadId;
    HANDLE hReadThread0, hReadThread1;

    // Create events to signal threads to start reading.
    ThreadInfo[0].hStart = CreateEvent(NULL, FALSE, FALSE, NULL);
    ThreadInfo[1].hStart = CreateEvent(NULL, FALSE, FALSE, NULL);

    // Create threads to read log from MCCDs.
    hReadThread0 = CreateThread(NULL, 0, ReadThread, (LPVOID)0, 0,
```

```
&dwThreadId);
    hReadThread1 = CreateThread(NULL, 0, ReadThread, (LPVOID)1, 0,
&dwThreadId);

    // Signal threads to start reading logs.
    SetEvent(ThreadInfo[0].hStart);
    SetEvent(ThreadInfo[1].hStart);

    // Wait for threads to terminate.
    WaitForSingleObject(hReadThread0, INFINITE);
    WaitForSingleObject(hReadThread1, INFINITE);
    printf("Press any key to exit\n");
    getchar();
}
/*****************************************************************************
 Thread to read log from MCCD.
*****************************************************************************/
DWORD WINAPI ReadThread(LPVOID lpvThreadParm)
{
    int nThread = (int)lpvThreadParm;
    CF_READP readPos;
    FILE *hFile;
    char szMeasBuffer[MEAS_BUF_SIZE];
    char szTimeStamp[LINE_SIZE];
    int nMeasBufCount;
    CF_HANDLE hServer;

    // Wait for a signal to start reading the data logs.
    WaitForSingleObject(ThreadInfo[nThread].hStart, INFINITE);

    // Open a connection to MCCD.
    printf("Thread %d trying MCCD: %s\n", nThread,
ThreadInfo[nThread].szAddr);
    if (cfOpen(ThreadInfo[nThread].szAddr, &hServer, PASSWORD) != CF_OK) {
        printf("Thread %d could not connect to MCCD: %s\n",
                nThread, ThreadInfo[nThread].szAddr);
        return 1;
    }
    printf("Thread %d connected to MCCD: %s\n", nThread,
ThreadInfo[nThread].szAddr);

    // Read the measurement log to a file.
    if((hFile = fopen(ThreadInfo[nThread].szLogFile, "w")) != NULL) {
        GetTimeStamp(szTimeStamp);
        fwrite(szTimeStamp, sizeof(char), strlen(szTimeStamp), hFile);
        readPos = CF_READ_FIRST;
        while (1) {
            cfReadMeasLog(hServer, &readPos, CF_ALL_CELLS, CF_ALL_STEPS,
                MEAS_BUF_SIZE, szMeasBuffer, &nMeasBufCount);
            if (nMeasBufCount == 0)
                break;
            fwrite(szMeasBuffer, sizeof(char), nMeasBufCount, hFile);
        }
        GetTimeStamp(szTimeStamp);
        fwrite(szTimeStamp, sizeof(char), strlen(szTimeStamp), hFile);
        fclose(hFile);
    }
```

```
    return 0;
}
```

A

仕様

ハードウェア仕様

表A-1に示されている仕様は保証されています。本仕様は、0～40℃の周囲温度範囲にわたって有効です。充電時には、本仕様は0.5Vから最大電圧までの充電電圧、0Aから最大電流までの充電電流に適用されます。放電時には、本仕様は2Vから最大電圧までの放電電圧、0Aから最大電流までの放電電流に適用されます。確度仕様は、AC電源とパワーバス条件 (電源変動) の全範囲、および充電/放電レベル (負荷変動)にわたって有効です。仕様は予告なしに変更されることがあります。

表A-1. Agilent E4370A/E4374A MCCDの仕様

パラメータ	条件	値
最大プログラマブル出力電圧	充電時	5 V
最大コンプライアンス電圧 (電池電圧 + フィクスチャ/配線電圧降下)	充電時	5.5 V
最大プログラマブル出力電流	充電または放電時、1チャネル当たり	2 A
最大出力リーク電流	ディスエーブル、1チャネル当たり 外部電圧-5V～+5V	± 25 µA
最大パワー	充電時、1チャネル当たり 放電時、1チャネル当たり	11 W 9 W
最大入力電圧	放電時	4.5 V
電圧プログラミング確度	リモート・センシングを用い、 センス・コネクタ入力で測定	±1 mV
電圧リードバック確度	リモート・センシングを用い、 センス・コネクタ入力で測定	±1 mV
電流プログラミング確度	表示値の%＋オフセット≤ 1A > 1A	± (0.05% + 1 mA) ± (0.1% + 1 mA)
電流リードバック確度	表示値の%＋オフセット≤ 1A > 1A	± (0.05% + 1 mA) ± (0.1% + 1 mA)
AC抵抗測定確度	表示値の%＋オフセット	±(1% + 1 mΩ)
DC抵抗測定確度	表示値の%＋オフセット	±(1% + 1 mΩ)

表A-2～A-4は、Agilent E4370A/E4374A/E4371A MCCDシステムの補足特性リストです。外部パワーバス・ソースの要件も示されています。特性は保証値ではありませんが、デザインまたは型式試験によって求められた代表性能を記載しています。

表A-2. Agilent E4370A/E4374A MCCDの特性

パラメータ	条件	値
AC抵抗測定	最大測定可能値 最大測定時間/出力[1]	1 Ω 1 s
DC抵抗測定	最大測定可能値 最大測定時間/出力[1]	1 Ω 1 s
アンペア時容量測定確度	表示値の%＋オフセット	±(0.01% + 1 mAh/h)
ワット時容量測定確度	表示値の%＋オフセット	±(0.01% + 5 mWh/h)
測定出力電流とプログラミング出力電流の最大偏差	定電流モード	±2 mA
測定出力電圧とプログラミング出力電圧の最大偏差	定電流モード	±2 mV
電圧出力ノイズ[2]	rms p-p	30 mV 100 mV
電流出力ノイズ[2]	rms p-p	1 mA 10 mA
電流の最大オーバシュート	最大持続時間5ms未満	5 %
電圧の最大オーバシュート		25 mV
最大電流立ち上がり時間		0.1 s
最小プログラミング電流	充電および放電モード	25 mA
測定間隔	データ・ログ シーケンス・テスト プローブ・チェック/チャネル[3]	1 s 1 s 1 s
ステップの時間間隔	最大 最小	596時間 1 s
データ・バッファの最大表示値数		349,504
最大シーケンス長		596時間
シーケンスの最大ステップ数		100
定義グループの最大数	256 チャネルのメインフレームの場合	8
AC入力電源要件	入力電圧レンジ 入力周波数レンジ 最大入力パワー 100/120 Vacにおける最大電流 220/240 Vacにおける最大電流	95 Vac～250 Vac 47Hz～63Hz 300 W 4 A 2 A
最大センス・プローブ抵抗[3]		1 kΩ
補助バイアス出力パワー	5～24V/0.42～2Aにおける最大パワー	10 W
補助バイアス出力電圧	0～0.42Aにおける最大電圧 0～2Aにおける最小電圧	24 V 5 V
補助バイアス出力電流	5V出力における最大電流	2 A

	24V出力における最小電流	0.42 A
補助バイアス出力電圧確度	任意の電圧および電流における設定値の%	7 %
補助バイアス出力ノイズ	任意の電圧および電流におけるピーク・ツー・ピーク値	100 mV
非絶縁ディジタル入出力特性	最大ローレベル出力電圧 最小ハイレベル出力電圧 最大ハイレベル出力電流 最小ハイレベル入力電圧 最大ローレベル入力電圧 最大ハイレベル入力電流	0.4 V @ 20 mA sink 1 V @ 300 mA sink 3.5 V @ 0 mA source 2.6V @ -200µA source 250 µA @ Voh=24 V 2.1 V 0.5 V 0.8 mA @ Vih min.
絶縁ディジタル入出力特性	最大ローレベル出力電圧 最大ハイレベル出力電流	0.6 V 100 µA
最大通気量	立方メートル/分 立方フィート/分	7.1 250
最高排気温度上昇	通気から排気まで	8℃
外形寸法	高さ 幅 奥行	221.5 mm 425.5 mm 540.5 mm
質量	メインフレーム1台とカード4枚の場合	22 kg

注:

[1] 256個の電池を測定するには5分かかります。

[2]パワー・コネクタにおいて、帯域幅20Hz～20MHzで測定しています。

[3]出力プローブ抵抗を正確に測定するには、+/–電源リードと+/–センス・リードの間が50mVでなければなりません。センス・プローブ抵抗を正確に測定するには、100mVの電池電圧が必要です。

表A-3. Agilent E4371Aパワーバス負荷の特性

Agilent E4371Aパワーバス負荷をシステムに統合する場合には、以下の情報を参考にしてください。

パラメータ	条件	値
推奨する最大電力消費		5,400 W
通常入力電圧		26.5～27Vdc
推奨する最大入力電流		200 A
Agilent MCCDとAgilentパワーバス負荷の間の最大配線電圧降下	最大電流で	1.5 V
最高排気温度上昇	通気から排気まで	40℃
最大通気量	立方メートル/分 立方フィート/分	10 350
外形寸法	高さ 幅 奥行	221.5 mm 425.5 mm 540.5 mm
質量		22.7 kg

表A-4. 外部パワーバス・ソースの要件

パラメータ	条件	値
公称出力電圧		24Vdc
出力電圧レンジ		22.8～25.2Vdc
電圧出力ノイズ[1]	rms p-p	30 mV 100 mV
最大出力電流	256チャネルのAgilent MCCDメインフレーム1台を充電した場合	133 A @ 5V, 2A/**チャネル**
最大出力パワー	256チャネルのAgilent MCCDメインフレーム1台を充電した場合	3,200 W @ 5V, 2A/**チャネル**

注:

[1]出力において、帯域幅20Hz～20MHzで測定しています。

B

校正

校正の種類

Agilent E4370A/E4374A MCCDには、次の3種類の校正があります。

- 完全校正。Agilent E4370A MCCDメインフレームとインストールされているすべてのAgilent E4374Aチャージャ/ディスチャージャ・カードを校正します。
- 伝送校正。メインフレームにインストールされているAgilent E4374Aチャージャ/ディスチャージャ・カードだけを校正します。
- メインフレーム基準校正。Agilent E4370A MCCDメインフレームだけを校正します。

完全校正は、メインフレーム基準校正と伝送校正から成ります。メインフレーム基準校正で外部電圧計を使用してAgilent MCCDメインフレームの内部基準電圧を校正し、伝送校正で、メインフレームの校正済み内部基準を使ってAgilent E4374Aチャージャ/ディスチャージャ・カード上の全チャネルを校正します。256チャネルの場合、全プロセスの完了までに約15分かかります。

注意: 完全校正または伝送校正を実行する際には、接続されている電池がないことを確認してください。

校正間隔:

1年に1回は、各Agilent E4370A/E4374A MCCDの完全校正を行ってください。

伝送校正は、メインフレームに新しい、または修理したチャージャ/ディスチャージャをインストールするたびに実行する必要があります。

完全校正

校正を行う場合は、DMMを電源メインフレーム裏面の校正端子に接続します。パワーバスには、24VのDC電源を接続してください。次に、内蔵プログラムを使って内部基準を校正します。このプログラムは、ポートBのRS-232インタフェースに接続されているシステムの電圧計を直接制御し、Agilent 3458Aコマンド・セットをサポートします。さらに校正済みの基準電圧を使用し、伝送校正手順を使って、内蔵の多重化回路を介して全チャージャ/ディスチャージャ・カードを校正します。Agilent MCCDがCF_IDLEステートにある場合には、常に完全校正を実行することができます。機器の一覧については、表B-1をご覧ください。

伝送校正

注記: 伝送校正には外部電圧計は不要です。伝送校正は、完全校正やメインフレーム基準校正とは関係なく実行できます。ただし伝送校正では、24VのDC電源をパワーバスに接続する必要があります。

伝送校正の実行中には、個々のチャネルが内部基準に順次接続されます。利得およびオフセット補正値が計算され、不揮発性メモリに保存されます。伝送校正は、Agilent MCCDがCF_IDLEステートにある場合にだけ実行可能です。裏面パネルの伝送校正スイッチを押してください。この校正は、修理後にカードをメインフレームに戻す場合に有用です。機器の一覧については、表B-1をご覧ください。

メインフレーム基準校正

メインフレーム基準校正 (標準校正とも言う) では、Agilent MCCDメインフレームの内部基準が校正されます。この校正を行う場合、外部電圧計をAgilent MCCDのシリアル・ポートAに接続する必要があります。機器の接続については、図B-1を参照してください。メインフレーム基準校正は、Agilent E4374Aカードをメインフレームにインストールしなくても実行できます。また、24Vdc電源をパワーバスに接続する必要もありません。

表B-1. 必要な校正機器

機器名	完全校正	メインフレーム基準校正	伝送校正
Agilent 3458A DMM	X	X	
24Vパワーバスまたは24 Vdc電源	X	X [1]	X
National Instruments GP-IB/RS-232コンバータ [2]	X	X	
GPIBおよびRS-232ケーブル	X	X	
Agilent 3458A DMMをAgilent MCCDに接続するためのワイヤ(推奨AWG 16)	X	X	
24 Vdc電源をAgilent MCCDに接続するためのワイヤ (推奨AWG 16)	X		X

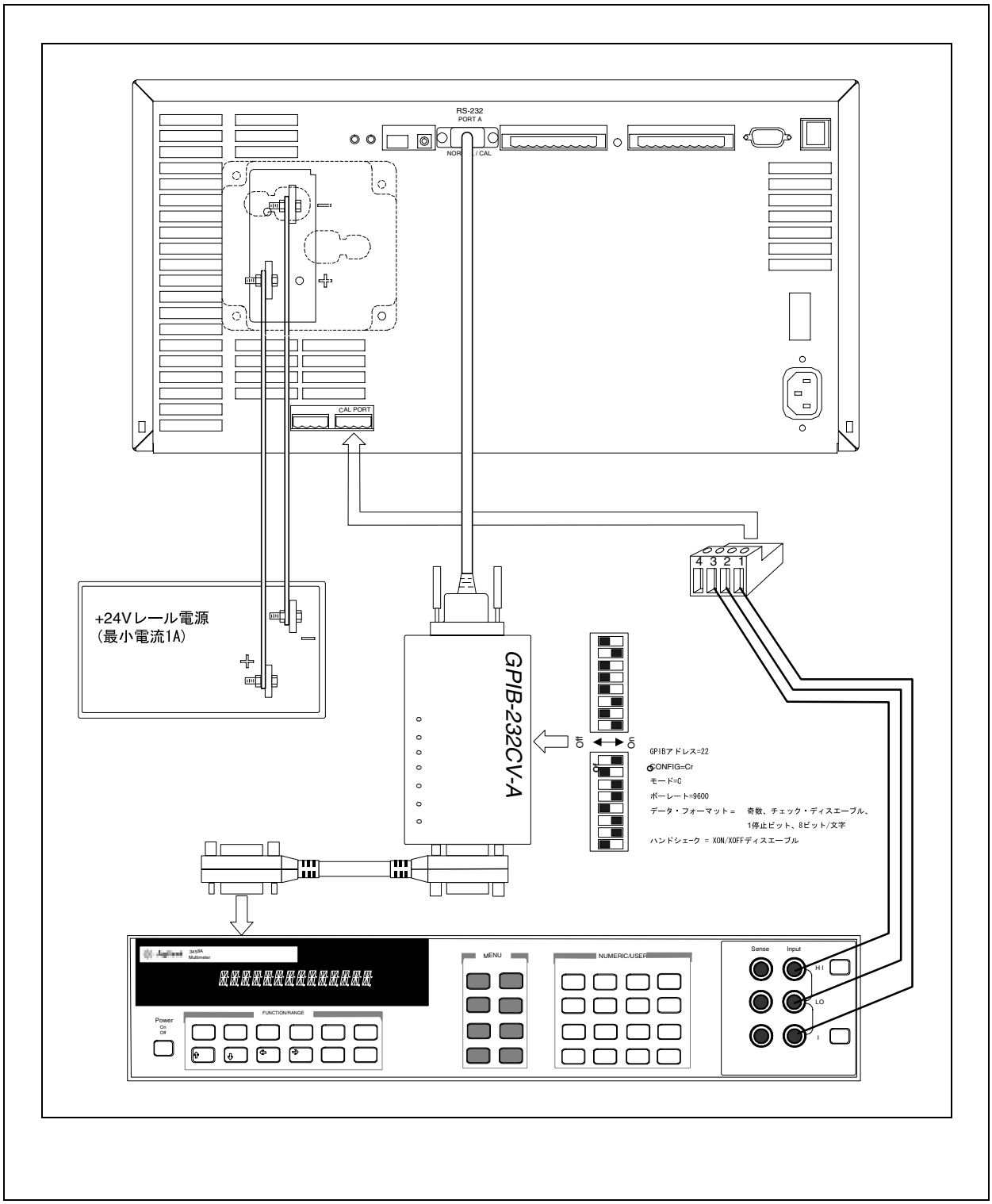
[1] メインフレームにカードがインストールされていない場合、24V DC電源は不要です。

[2] GPIB-232CV-A GP-IB－RS-232コンバータをオーダするには、デバイスの使用国における部品番号を調べる必要があります。適切な部品番号とオーダ情報を得るには、最寄りのNational Instrumentsオフィスに問い合わせるか、ウェブ www.nationalinstruments.comにアクセスしてください。

校正用接続

図B-1は、校正用接続を示したものです。図に従って、National Instruments GP-IB/RS-232コンバータ・ボックス上のスイッチを設定してください。Agilent 3458A GPIBアドレスが22に設定されていることを確かめてください。Agilent MCCDを24Vパワーバスに接続できない場合、代わりに24V、1Aの定格DC電源を使用できます。

注記: 必ず、電圧計の電源を**入れてから**、RS-232コンバータの電源を入れてください。


RS-232
PORT A
CAL PORT
4 3 2 1
+24Vレール電源
（最小電流1A）
GPIB-232CV-A
GPIBアドレス=22
CONFIG=Cr
モード=C
ボーレート=9600
データ・フォーマット＝　奇数、チェック・ディスエーブル、
1停止ビット、8ビット/文字
ハンドシェーク ＝ XON/XOFFディスエーブル
MENU
NUMERIC/USER
Sense
Input
HI
LO
Power
On
Off
FUNCTION/RANGE

図B-1. 校正用接続

校正へのアクセス

校正制御機能へのアクセス方法には、以下の3通りの方法があります。

- Agilent MCCD設定画面
- LAN経由でのAPI呼び出し
- WebベースのAgilent MCCDユーザ・インタフェース

本項では、最初の方法について詳しく説明します。

注記: 伝送校正は、裏面パネルのCalボタンを押すことによっても実行できます。

Agilent MCCD設定画面

Agilent MCCDは、PC上のAgilent MCCD設定画面を使って校正することができます。このプログラムの実行方法については、第2章を参照してください。

Agilent MCCD設定画面を使ってAgilent MCCDを校正するには、Agilent E4370A裏面のポートBスイッチ (No.4) を上から下 (NormalからConfigure) に切り替え、HyperTerminalプログラムを実行します。Agilent MCCD設定画面が表示されたら、4を選択してAgilent E4370A MCCDメインフレームとAgilent E4374Aチャージャ/ディスチャージャを校正します。

完全校正を実行するには1を選択します。伝送校正を実行するには2を選択します。メインフレーム校正を実行するには3を選択します。3を選択した場合には、メインフレーム基準電圧だけが校正されます。

```
During full calibration and mainframe-reference calibration, a DMM
and Powerbus power supply must be connected to the MCCD. For the DMM,
connect to serial Port A with settings: 9600 baud, NO Parity, 8 bits.
Connect the DMM's inputs as follows: Input Hi to Cal Port 3, Input Lo
to Cal Port 2, and Current to Cal port 1.

During transfer calibration the Powerbus power supply must be
connected to the MCCD. Inactive outputs will not be calibrated.
Disconnected sense and load leads before calibrating.

1)   Execute full calibration
     (takes approx 5 seconds per active channel.)

2)   Execute transfer calibration
     (Takes approx 5 seconds per active channel.)

3)   Execute Mainframe-reference calibration
     (takes approx 30 seconds.)

4)   Set DMM model (Agilent3458 currently active)

Type a number and press Enter or ctrl-G to return to initial screen
```

裏面パネルの伝送校正スイッチ

この押しボタン式のスイッチは、裏面パネルのくぼんだ穴にあります。このボタンを押すと、Agilent MCCD内で伝送校正シーケンスがイニシエートされます。これは、メインフレーム内のAgilent E4374Aチャージャ/ディスチャージャ・カードを交換した場合に有用です。伝送校正は、メインフレーム内のすべてのカードを再校正します。伝送校正スイッチの隣にあるインジケータ・ライトが校正ステータスを示します。

LAN経由でのAPI呼び出し

以下のAPI呼び出しによって、校正関数にアクセスすることができます。

cfCal	完全校正を開始します（メインフレームおよびカード）
cfCalStandard	標準校正を開始します（メインフレーム）
cfCalTransfer	伝送校正を開始します（カード）

これらのAPI関数呼び出しの詳細については、第6章を参照してください。

Webベースのグラフィカル・ユーザ・インタフェース

校正情報については、Agilent MCCDユーザ・インタフェースで提供されるオンライン・ヘルプを参照してください。

校正エラー・メッセージ

裏面パネルにある2つのLEDは、校正ステータスを示し、校正エラーをレポートします。Agilent MCCDユーザ・インタフェースとAPI関数を用いれば、より詳細なテキスト・ベースのエラー・レポートを入手できます。

CAL IN PROGESS	点滅しているときは、校正が進行中であることを示します。校正が完了するとオフになります。
CAL FAILED	点灯しているときは、校正が失敗に終わったことを示します。このインジケータとテキスト・ベースのエラー・レポートを使って、カードの問題かメインフレームの問題かを切り分けます。

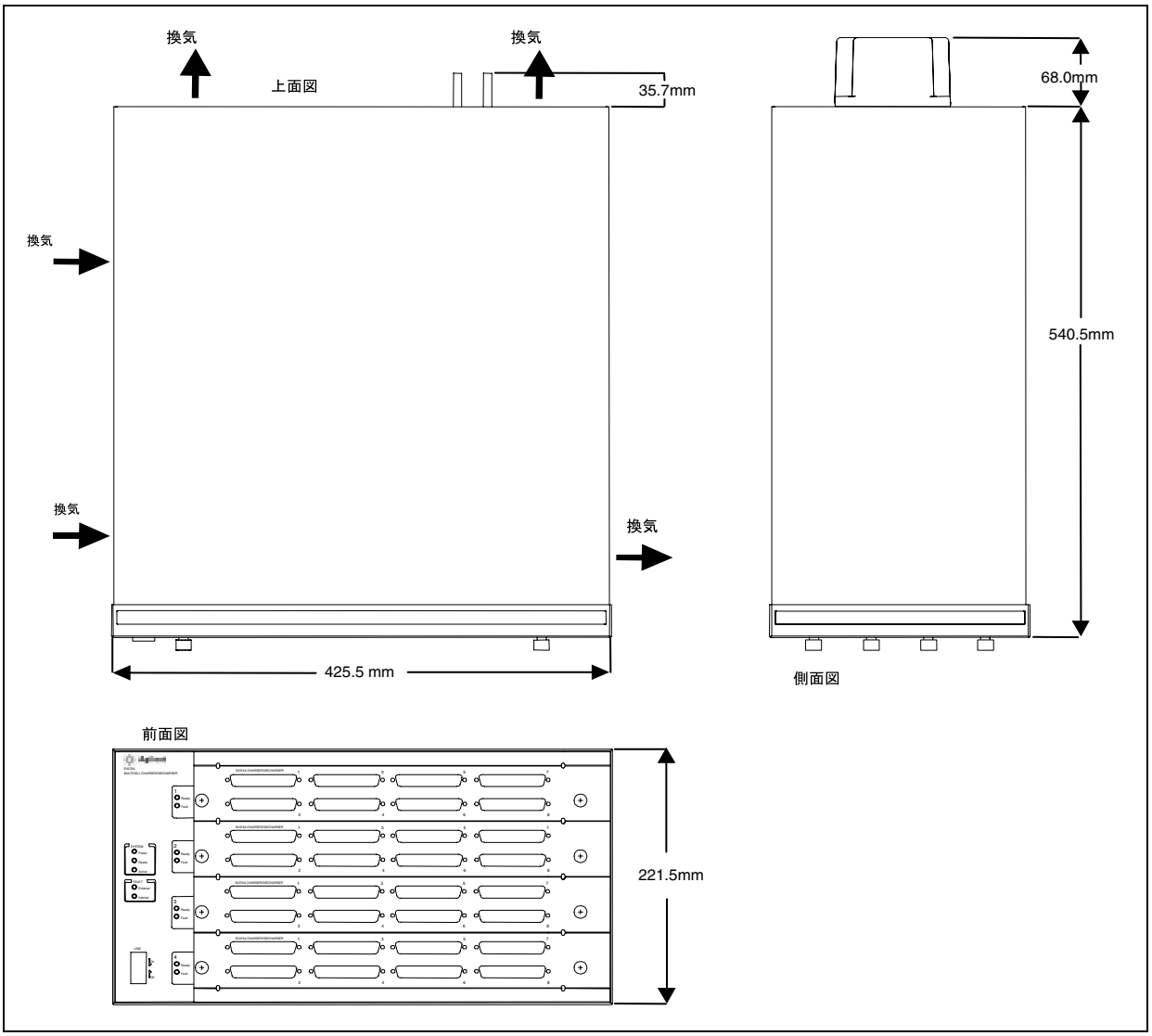
テキスト・ベースの校正エラー・メッセージを読み取るには、cfReadSelftestLog() API関数を使用します。Agilent MCCDユーザ・インタフェースを使用している場合、Systemページにアクセスし、CalibrationとSelftestを選択してから、Calibration Logをクリックすることによって、エラー・メッセージを読み取ることができます。校正エラー・メッセージを書き留め、担当のAgilentサービス・エンジニアにご連絡ください。

C

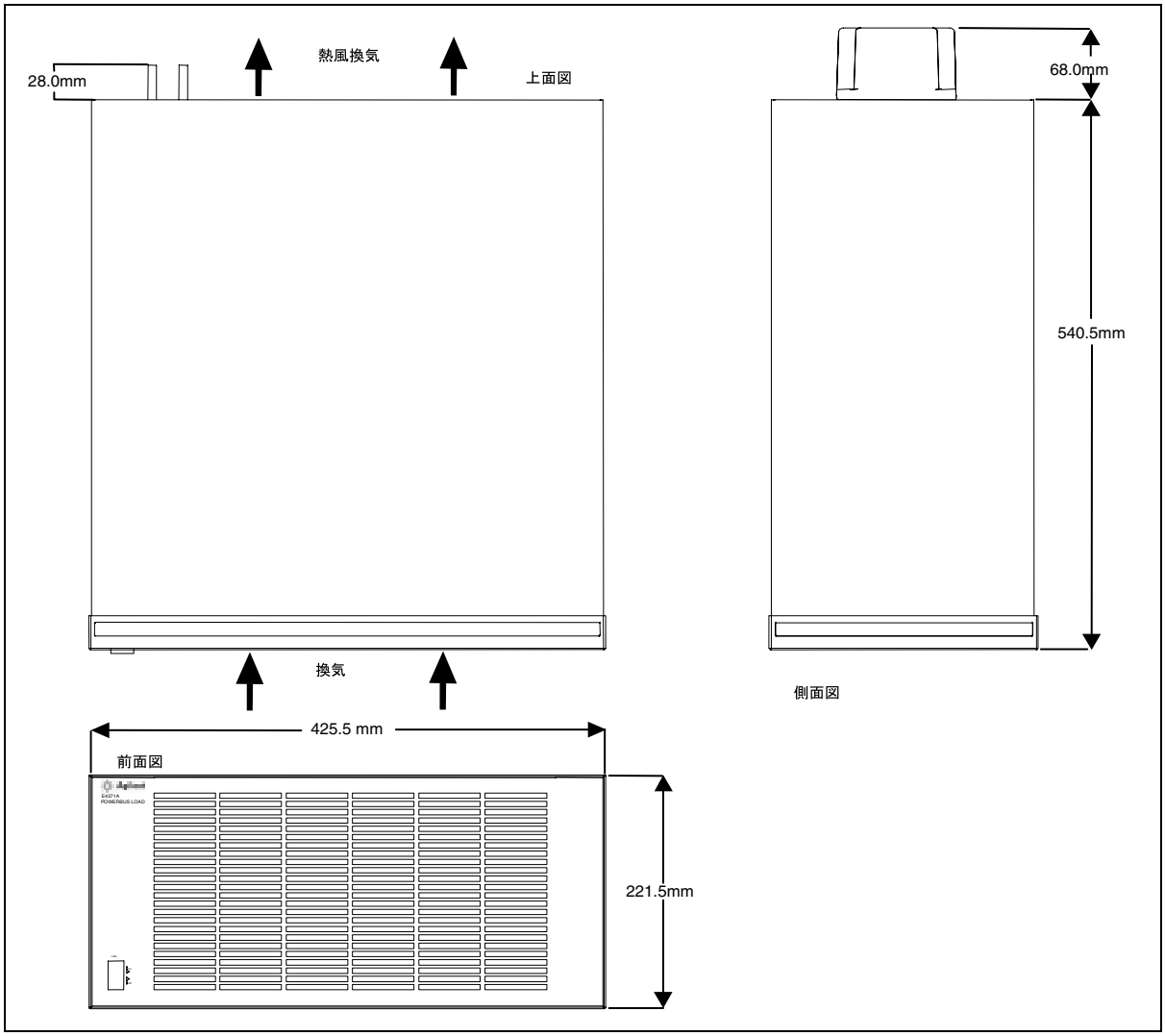
外形寸法図

図C-1は、Agilent E4370A MCCDメインフレームの簡単な外形図を示したものです。一方、図C-2は、Agilent E4371A パワーバス負荷の簡単な外形図を示したものです。

本書のバインダの裏側に掲載されている外形寸法図には、その他の情報が示されています。



図C-1. Agilent MCCDの簡単な外形図



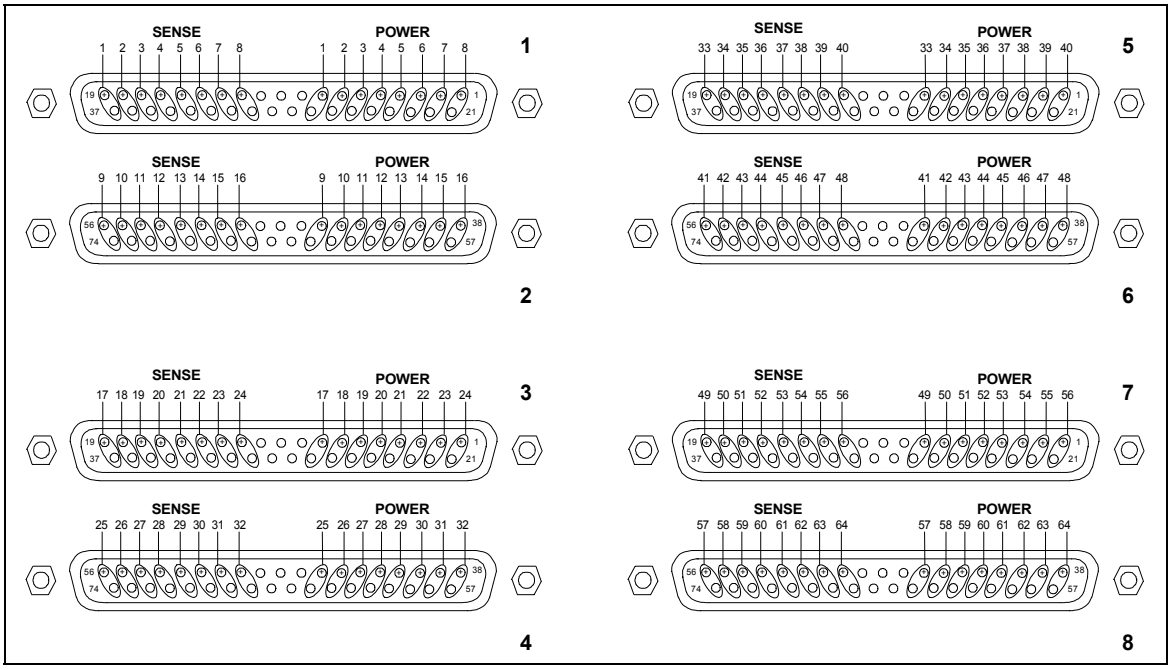
図C-2. Agilentパワーバス負荷の簡単な外形図

D

センスおよびパワー・コネクタのピン配置

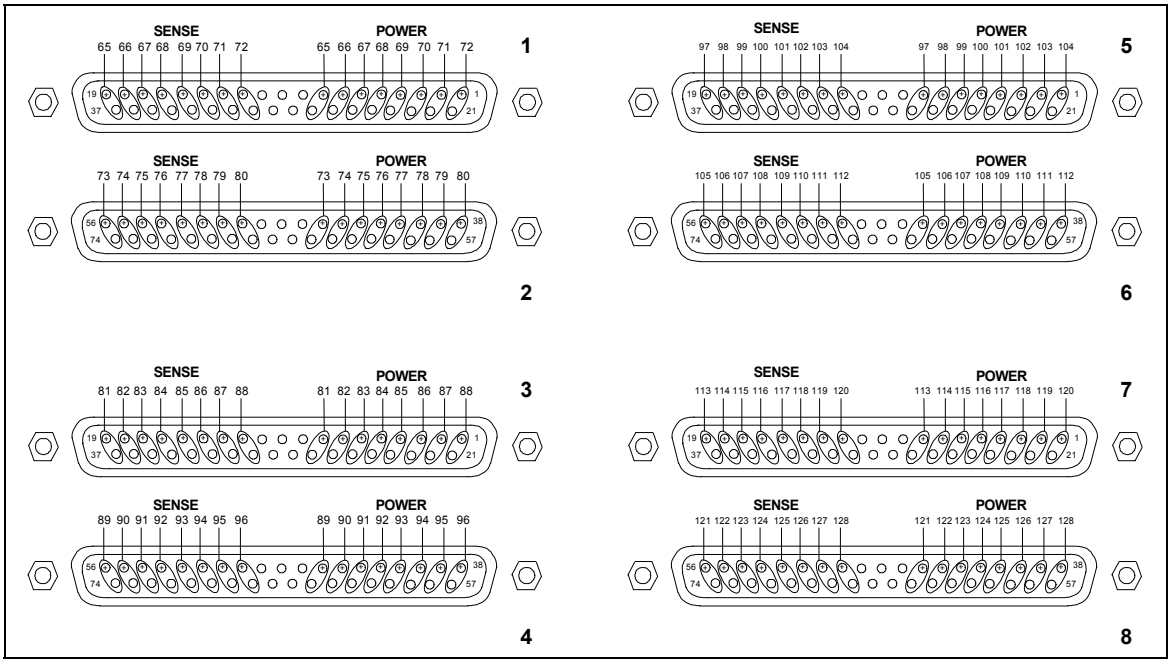
本付録に掲載されている図および表には、Agilent E4370A MCCDメインフレーム (図1-2を参照) のセンスおよびパワー・コネクタのピン配置が示されています。これらの図は、次のように構成されたフル装備の256チャネル・メインフレームをベースとしています。

カード番号	コネクタ番号							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1～8	9～16	17～24	25～32	33～40	41～48	49～56	57～64
2	65～72	73～80	81～88	89～96	97～104	105～112	113～120	121～128
3	129～136	137～144	145～152	153～160	161～168	169～176	177～184	185～192
4	193～200	201～208	209～216	217～224	225～232	233～240	241～248	249～256



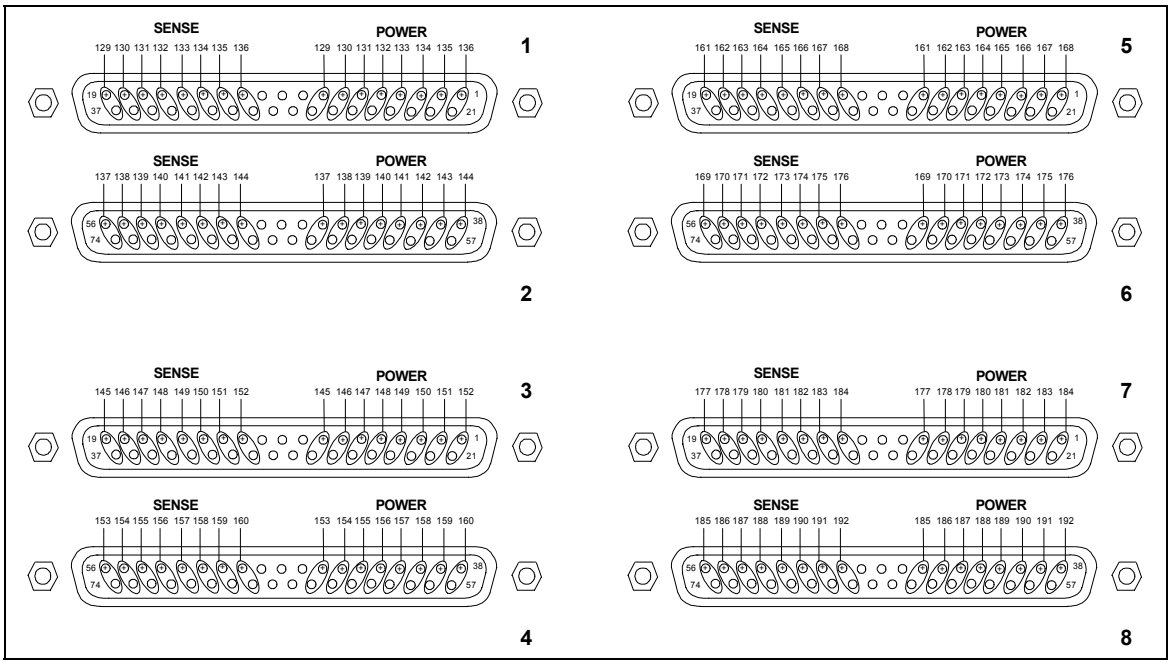
注: ラベルの付いていないピンは、各ペアのマイナス接続です。

図D-1. カード1のセンスおよびパワー・コネクタの電池割当て



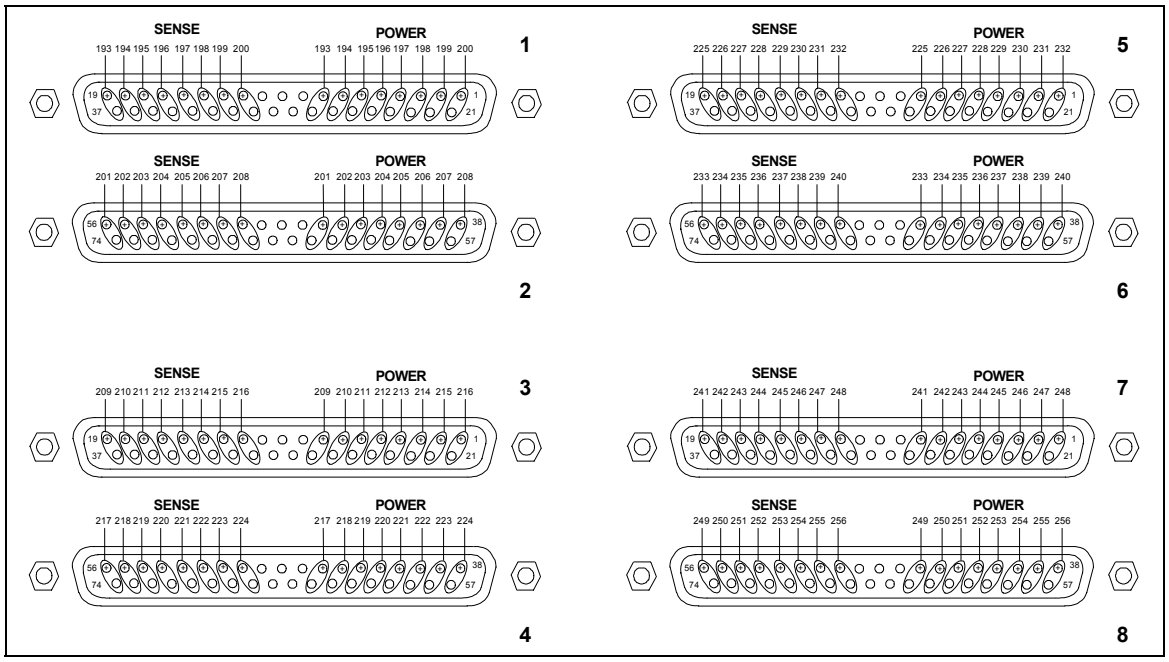
注: ラベルの付いていないピンは、各ペアのマイナス接続です。

図D-2. カード2のセンスおよびパワー・コネクタの電池割当て



注: ラベルの付いていないピンは、各ペアのマイナス接続です。

図D-3. カード3のセンスおよびパワー・コネクタの電池割当て



注: ラベルの付いていないピンは、各ペアのマイナス接続です。

図D-4. カード4のセンスおよびパワー・コネクタの電池割当て

表D-1. カード1のセンスおよびパワー・ピンのピン割当て

センス・ピン	電池番号	パワー・ピン	センス・ピン	電池番号	パワー・ピン
	コネクタ1			**コネクタ5**	
+19, –37	cell 1	+8, –27	+19, –37	cell 33	+8, –27
+18, –36	cell 2	+7, –26	+18, –36	cell 34	+7, –26
+17, –35	cell 3	+6, –25	+17, –35	cell 35	+6, –25
+16, –34	cell 4	+5, –24	+16, –34	cell 36	+5, –24
+15, –33	cell 5	+4, –23	+15, –33	cell 37	+4, –23
+14, –32	cell 6	+3, –22	+14, –32	cell 38	+3, –22
+13, –31	cell 7	+2, –21	+13, –31	cell 39	+2, –21
+12, –30	cell 8	+1, –20	+12, –30	cell 40	+1, –20
	コネクタ2			**コネクタ6**	
+56, –74	cell 9	+45, –64	+56, –74	cell 41	+45, –64
+55, –73	cell 10	+44, –63	+55, –73	cell 42	+44, –63
+54, –72	cell 11	+43, –62	+54, –72	cell 43	+43, –62
+53, –71	cell 12	+42, –61	+53, –71	cell 44	+42, –61
+52, –70	cell 13	+41, –60	+52, –70	cell 45	+41, –60
+51, –69	cell 14	+40, –59	+51, –69	cell 46	+40, –59
+50, –68	cell 15	+39, –58	+50, –68	cell 47	+39, –58
+49, –67	cell 16	+38, –57	+49, –67	cell 48	+38, –57
	コネクタ3			**コネクタ7**	
+19, –37	cell 17	+8, –27	+19, –37	cell 49	+8, –27
+18, –36	cell 18	+7, –26	+18, –36	cell 50	+7, –26
+17, –35	cell 19	+6, –25	+17, –35	cell 51	+6, –25
+16, –34	cell 20	+5, –24	+16, –34	cell 52	+5, –24
+15, –33	cell 21	+4, –23	+15, –33	cell 53	+4, –23
+14, –32	cell 22	+3, –22	+14, –32	cell 54	+3, –22
+13, –31	cell 23	+2, –21	+13, –31	cell 55	+2, –21
+12, –30	cell 24	+1, –20	+12, –30	cell 56	+1, –20
	コネクタ4			**コネクタ8**	
+56, –74	cell 25	+45, –64	+56, –74	cell 57	+45, –64
+55, –73	cell 26	+44, –63	+55, –73	cell 58	+44, –63
+54, –72	cell 27	+43, –62	+54, –72	cell 59	+43, –62

+53, –71	cell 28	+42, –61		+53, –71	cell 60	+42, –61
+52, –70	cell 29	+41, –60		+52, –70	cell 61	+41, –60
+51, –69	cell 30	+40, –59		+51, –69	cell 62	+40, –59
+50, –68	cell 31	+39, –58		+50, –68	cell 63	+39, –58
+49, –67	cell 32	+38, –57		+49, –67	cell 64	+38, –57

注: コネクタ・ピン9～11、28、29、46～48、65、66は未使用です。

表D-2. カード2のセンスおよびパワー・ピンのピン割当て

センス・ピン	電池番号	パワー・ピン	センス・ピン	電池番号	パワー・ピン
	コネクタ1			**コネクタ5**	
+19, –37	cell 65	+8, –27	+19, –37	cell 97	+8, –27
+18, –36	cell 66	+7, –26	+18, –36	cell 98	+7, –26
+17, –35	cell 67	+6, –25	+17, –35	cell 99	+6, –25
+16, –34	cell 68	+5, –24	+16, –34	cell 100	+5, –24
+15, –33	cell 69	+4, –23	+15, –33	cell 101	+4, –23
+14, –32	cell 70	+3, –22	+14, –32	cell 102	+3, –22
+13, –31	cell 71	+2, –21	+13, –31	cell 103	+2, –21
+12, –30	cell 72	+1, –20	+12, –30	cell 104	+1, –20
	コネクタ2			**コネクタ6**	
+56, –74	cell 73	+45, –64	+56, –74	cell 105	+45, –64
+55, –73	cell 74	+44, –63	+55, –73	cell 106	+44, –63
+54, –72	cell 75	+43, –62	+54, –72	cell 107	+43, –62
+53, –71	cell 76	+42, –61	+53, –71	cell 108	+42, –61
+52, –70	cell 77	+41, –60	+52, –70	cell 109	+41, –60
+51, –69	cell 78	+40, –59	+51, –69	cell 110	+40, –59
+50, –68	cell 79	+39, –58	+50, –68	cell 111	+39, –58
+49, –67	cell 80	+38, –57	+49, –67	cell 112	+38, –57
	コネクタ3			**コネクタ7**	
+19, –37	cell 81	+8, –27	+19, –37	cell 113	+8, –27
+18, –36	cell 82	+7, –26	+18, –36	cell 114	+7, –26
+17, –35	cell 83	+6, –25	+17, –35	cell 115	+6, –25
+16, –34	cell 84	+5, –24	+16, –34	cell 116	+5, –24
+15, –33	cell 85	+4, –23	+15, –33	cell 117	+4, –23
+14, –32	cell 86	+3, –22	+14, –32	cell 118	+3, –22
+13, –31	cell 87	+2, –21	+13, –31	cell 119	+2, –21
+12, –30	cell 88	+1, –20	+12, –30	cell 120	+1, –20
	コネクタ4			**コネクタ8**	
+56, –74	cell 89	+45, –64	+56, –74	cell 121	+45, –64
+55, –73	cell 90	+44, –63	+55, –73	cell 122	+44, –63
+54, –72	cell 91	+43, –62	+54, –72	cell 123	+43, –62

+53, –71	cell 92	+42, –61		+53, –71	cell 124	+42, –61
+52, –70	cell 93	+41, –60		+52, –70	cell 125	+41, –60
+51, –69	cell 94	+40, –59		+51, –69	cell 126	+40, –59
+50, –68	cell 95	+39, –58		+50, –68	cell 127	+39, –58
+49, –67	cell 96	+38, –57		+49, –67	cell 128	+38, –57

注: コネクタ・ピン9～11、28、29、46～48、65、66は未使用です。

表D-3. カード3のセンスおよびパワー・ピンのピン割当て

センス・ピン	電池番号	パワー・ピン	センス・ピン	電池番号	パワー・ピン
	コネクタ1			**コネクタ5**	
+19, –37	cell 129	+8, –27	+19, –37	cell 161	+8, –27
+18, –36	cell 130	+7, –26	+18, –36	cell 162	+7, –26
+17, –35	cell 131	+6, –25	+17, –35	cell 163	+6, –25
+16, –34	cell 132	+5, –24	+16, –34	cell 164	+5, –24
+15, –33	cell 133	+4, –23	+15, –33	cell 165	+4, –23
+14, –32	cell 134	+3, –22	+14, –32	cell 166	+3, –22
+13, –31	cell 135	+2, –21	+13, –31	cell 167	+2, –21
+12, –30	cell 136	+1, –20	+12, –30	cell 168	+1, –20
	コネクタ2			**コネクタ6**	
+56, –74	cell 137	+45, –64	+56, –74	cell 169	+45, –64
+55, –73	cell 138	+44, –63	+55, –73	cell 170	+44, –63
+54, –72	cell 139	+43, –62	+54, –72	cell 171	+43, –62
+53, –71	cell 140	+42, –61	+53, –71	cell 172	+42, –61
+52, –70	cell 141	+41, –60	+52, –70	cell 173	+41, –60
+51, –69	cell 142	+40, –59	+51, –69	cell 174	+40, –59
+50, –68	cell 143	+39, –58	+50, –68	cell 175	+39, –58
+49, –67	cell 144	+38, –57	+49, –67	cell 176	+38, –57
	コネクタ3			**コネクタ7**	
+19, –37	cell 145	+8, –27	+19, –37	cell 177	+8, –27
+18, –36	cell 146	+7, –26	+18, –36	cell 178	+7, –26
+17, –35	cell 147	+6, –25	+17, –35	cell 179	+6, –25
+16, –34	cell 148	+5, –24	+16, –34	cell 180	+5, –24
+15, –33	cell 149	+4, –23	+15, –33	cell 181	+4, –23
+14, –32	cell 150	+3, –22	+14, –32	cell 182	+3, –22
+13, –31	cell 151	+2, –21	+13, –31	cell 183	+2, –21
+12, –30	cell 152	+1, –20	+12, –30	cell 184	+1, –20
	コネクタ4			**コネクタ8**	
+56, –74	cell 153	+45, –64	+56, –74	cell 185	+45, –64
+55, –73	cell 154	+44, –63	+55, –73	cell 186	+44, –63
+54, –72	cell 155	+43, –62	+54, –72	cell 187	+43, –62

+53, –71	cell 156	+42, –61		+53, –71	cell 188	+42, –61
+52, –70	cell 157	+41, –60		+52, –70	cell 189	+41, –60
+51, –69	cell 158	+40, –59		+51, –69	cell 190	+40, –59
+50, –68	cell 159	+39, –58		+50, –68	cell 191	+39, –58
+49, –67	cell 160	+38, –57		+49, –67	cell 192	+38, –57

注: コネクタ・ピン9～11、28、29、46～48、65、66は未使用です。

表D-4. カード4のセンスおよびパワー・ピンのピン割当て

センス・ピン	電池番号	パワー・ピン	センス・ピン	電池番号	パワー・ピン
	コネクタ1			**コネクタ5**	
+19, –37	cell 193	+8, –27	+19, –37	cell 225	+8, –27
+18, –36	cell 194	+7, –26	+18, –36	cell 226	+7, –26
+17, –35	cell 195	+6, –25	+17, –35	cell 227	+6, –25
+16, –34	cell 196	+5, –24	+16, –34	cell 228	+5, –24
+15, –33	cell 197	+4, –23	+15, –33	cell 229	+4, –23
+14, –32	cell 198	+3, –22	+14, –32	cell 230	+3, –22
+13, –31	cell 199	+2, –21	+13, –31	cell 231	+2, –21
+12, –30	cell 200	+1, –20	+12, –30	cell 232	+1, –20
	コネクタ2			**コネクタ6**	
+56, –74	cell 201	+45, –64	+56, –74	cell 233	+45, –64
+55, –73	cell 202	+44, –63	+55, –73	cell 234	+44, –63
+54, –72	cell 203	+43, –62	+54, –72	cell 235	+43, –62
+53, –71	cell 204	+42, –61	+53, –71	cell 236	+42, –61
+52, –70	cell 205	+41, –60	+52, –70	cell 237	+41, –60
+51, –69	cell 206	+40, –59	+51, –69	cell 238	+40, –59
+50, –68	cell 207	+39, –58	+50, –68	cell 239	+39, –58
+49, –67	cell 208	+38, –57	+49, –67	cell 240	+38, –57
	コネクタ3			**コネクタ7**	
+19, –37	cell 209	+8, –27	+19, –37	cell 241	+8, –27
+18, –36	cell 210	+7, –26	+18, –36	cell 242	+7, –26
+17, –35	cell 211	+6, –25	+17, –35	cell 243	+6, –25
+16, –34	cell 212	+5, –24	+16, –34	cell 244	+5, –24
+15, –33	cell 213	+4, –23	+15, –33	cell 245	+4, –23
+14, –32	cell 214	+3, –22	+14, –32	cell 246	+3, –22
+13, –31	cell 215	+2, –21	+13, –31	cell 247	+2, –21
+12, –30	cell 216	+1, –20	+12, –30	cell 248	+1, –20
	コネクタ4			**コネクタ8**	
+56, –74	cell 217	+45, –64	+56, –74	cell 249	+45, –64
+55, –73	cell 218	+44, –63	+55, –73	cell 250	+44, –63
+54, –72	cell 219	+43, –62	+54, –72	cell 251	+43, –62

+53, –71	cell 220	+42, –61		+53, –71	cell 252	+42, –61
+52, –70	cell 221	+41, –60		+52, –70	cell 253	+41, –60
+51, –69	cell 222	+40, –59		+51, –69	cell 254	+40, –59
+50, –68	cell 223	+39, –58		+50, –68	cell 255	+39, –58
+49, –67	cell 224	+38, –57		+49, –67	cell 256	+38, –57

注: コネクタ・ピン9～11、28、29、46～48、65、66は未使用です。

E

問題が発生した場合

概要

Agilent E4370A MCCDには、電源投入時に実行されるセルフテスト機能が内蔵されています。さらに、cfSelftest;関数を実行するか、Agilent MCCDユーザ・インタフェースからセルフテストを実行することによって、より完全なセルフテストを行うことも可能です。このセルフテスト機能は、迅速に修理が行えるよう問題を切り分けるための効率的な診断ツールとなります。詳細については、第5章の「セルフテスト」の項を参照してください。

注意: ユーザ・イニシエートのセルフテストを実行するときは、テスト中に危険な電圧が電池に加わらないように、絶対に電池を接続しないでください。

Fault LED (図1-2を参照)

Agilent E4370A/E4374A MCCDシステム前面にある**Fault** LEDのいずれかが点灯している場合、メインフレームか充放電カードに関連する故障があることを示します。カードまたはメインフレームを交換する前に、下の表に示す手順に従ってください。

表E-1. Faultインジケータ

Agilent E4370A Fault	
External	点灯している場合、次のような外部不良を示します。 外部ディジタル不良信号を受信 外部停電シャットダウン信号 電源投入後の高パワーバス電圧 電源投入後の低パワーバス電圧 過熱 ライトをクリアするには、まず故障の原因を除去します。その後、ユニットの電源を入れ直し、APIプログラミング・コマンドProtect Clearを送信するか、Agilent MCCDユーザ・インタフェースで**Protect Clear**を押して、故障レジスタをクリアします。
Internal	点灯している場合、次のような内部ハードウェア故障を示します。 セルフテスト不良 校正エラー ハードウェア・エラー Agilent E4374Aチャージャ/ディスチャージャ・カードのいずれのFault LEDも点灯していないのに**Internal** LEDが点灯している場合は、Agilent E4370A MCCDメインフレームが故障していることを示します。ユニットを返送して、修理を受けてください。 チャージャ/ディスチャージャ・カードのFault LEDが点灯している場合には、メインフレームの電源を切り、つまみ付きネジをゆるめ、故障しているカードをスロットから取り出して交換してください。 故障しているカードを交換してもメインフレームのInternal LEDが点灯したままになっている場合は、メインフレームの故障が考えられます。ユニットを返送して、修理を受けてください。

Agilent E4374A Fault	
1, 2, 3, 4	次のような内部ハードウェア故障を示します。 セルフテスト不良 校正エラー ハードウェア・エラー テキスト・ベースのエラー・メッセージを読み取るには、API関数cfReadSelftestLog()を使用します。Agilent MCCDユーザ・インタフェースを使用している場合には、Systemページにアクセスし、CalibrationおよびSelftestを選択してからSelftest Logをクリックすれば、エラー・メッセージを読み取ることができます。エラー・メッセージを書き留め、担当のAgilentサービス・エンジニアに連絡してください。カードが故障している場合には、カードを返送して修理を受けてください。Agilent E4374Aカードには、ユーザが交換できる部品は搭載されていません。

セルフテスト・エラー・メッセージ

表E-1に掲載されている前面パネルのLEDのほか、Agilent MCCDユーザ・インタフェースとAPI関数を用いることによって、より詳細なテキスト・ベースのエラー・レポートを入手することができます。

テキスト・ベースのセルフテスト・エラー・メッセージを読み取るには、API関数cfReadSelftestLog()を使用します。Agilent MCCDユーザ・インタフェースを使用している場合には、Systemページにアクセスし、CalibrationおよびSelftestを選択してからSelftest Logをクリックすれば、エラー・メッセージを読み取ることができます。セルフテスト・エラー・メッセージを書き留め、担当のAgilentサービス・エンジニアに連絡してください。

索引

― た ―

― な ―

― は ―

― ま ―

― や ―

ー ら ー

ー わ ー