IVI-COM 計測器ドライバ プログラミング・ガイド (LabVIEW 編)

Sep 2005 Revision 2.0

1- 概要

1-1 IVI-C ドライバのサポート

LabVIEW には VXI Plug&Play 計測器ドライバや IVI-C 計測器ドライバをインポートする機能があります。IVI-COM 計測器ドライバを直接利用することも可能ですが、IVI-C 又は VXI Plug&Play 計測器ドライバがあればそれらを利用したほうがプログラミングは簡単になります。当社の IVI-COM 計測器ドライバは、IVI-COM インターフェースだけでなく IVI-C によるプログラミングインターフェースもサポートしています。IVI-C 仕様は VXI plug&play 計測器ドライバ仕様を進化させたもので、LabVIEW で使うのには最も適したドライバタイプです。

Notes:

当社の IVI-COM 計測器ドライバでは、バージョン表記が 2.x.x.x 以上であれば IVI-COM と IVI-C の両方のインターフェースをサポートしています。バージョン表記が 1.x.x.x の場合は IVI-C をサポートしていないので注意してください。

IVI-C 計測器ドライバを使用するには、NI IVI Compliance Package 2.x を別途インストールする必要があります。このパッケージは当社製の IVI 計測器ドライバをインストールしても自動的にはインストールされません。また全てのバージョンの LabVIEW がそれをインストールするわけでもありません。

本ガイドブックでは、IVI-COM Kikusui4800 計測器ドライバ(KIKUSUI PIA4800 シリーズ電源コントローラ)を使用する例を示します。他機種用の IVI-COM 計測器ドライバでも、ほぼ同様の手順で使用できます。

本ガイドブックでは、LabVIEW 7.1 を使用した場合を例に説明しています。

IVI 計測器ドライバを利用する場合、スペシフィック・インターフェースを利用する方法とクラス・インターフェースを利用する方法の 2 種類があります。前者は計測器ドライバの固有インターフェースを利用するもので、使用する計測器の機能を最大限に利用する事ができます。後者は IVI 仕様書で定義されている計測器クラスのインターフェースを利用するもので、インターチェンジャビリティ機能を利用する事ができますが、機種固有の機能を使うことは制限されます。

Notes:

計測器ドライバが所属する計測器クラスについては、ドライバ毎の Readme.txt に記載されています。Readme 文書は、Start ボタン➔Program➔IVI フォルダから開く事ができます。

計測器ドライバが如何なる計測器クラスにも属していない場合、クラス・インターフェースを利用する事はできません。つまりこの場合、インターチェンジャビリティ機能を利用するアプリケーションを作成する事は出来ません。

2- スペシフィック・インターフェースを使用するサンプル

ここでは、スペシフィック・インターフェースを使用したサンプルを示します。スペシフィック・インターフェースを使用すると、計測器ドライバで提供される機能を最大限に利用する事ができますが、インターチェンジャビリティを実現する事はできません。

2-1 計測器ドライバのインポート

IVI-C や VXI Plug&Play 計測器ドライバは LabWindows/CVI 互換の形式(fp、c、h、及び DLL 等)で提供されるため、LabVIEW 統合環境から直接利用することはできません。その為、ドライバのインターフェース情報をインポートして LabVIEW で利用可能な形式(vi 又は llb)に変換する必要があります。

計測器ドライバをインポートするには、LabVIEW の **Tools | Instrumentation | Import CVI Instrument Driver** メニューを選択します。計測器ドライバの fp (CVI ファンクション・パネル) ファイルを指定するように要求されるので、**Program Files/IVI/Drivers/ki4800** ディレクトリに置かれている **ki4800.fp** を選択します。すると、CVI Function Panel Converter ダイアログが表示されるので、変換ファイルの生成先、DLL ファイル名、その他を確認し(通常はデフォルトで問題ありません)、**OK** ボタンをクリックします。

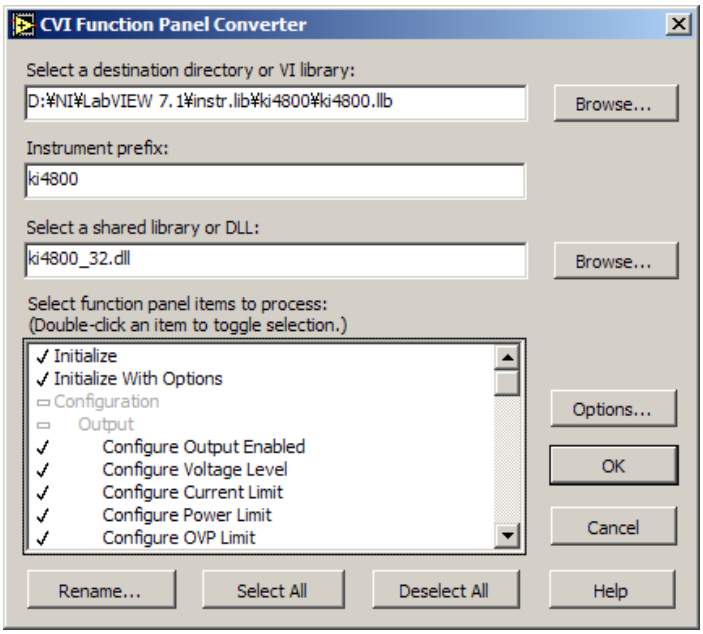


Figure 2-1 CVI Function Panel Converter

変換が終了すると、LabVIEW がインストールされている場所の **instr.lib** サブディレクトリに ki4800.llb(及び幾つかのサポートファイル)が生成されます。これが、LabVIEW から直接利用可能な計測器ドライバ・ラッパーになります。

ki4800 計測器ドライバ・ラッパーは、ブロックダイアグラム上で **Instrument I/O** ファンクション・パレットから参照することが出来ます。

Note:

インポート作業によって生成された llb ファイルはドライバラッパーであり、計測器ドライバの実体ではありません。従って、実行時に IVI 計測器ドライバの実体(DLL)がインストールされている必要があります。

LabVIEW 7.1 で利用可能なアドオン・ソフトウェア「LabVIEW Interface Generator for LabWindows/CVI Instrument Drivers」をインストールした場合、CVI Function Panel Converter の操作が若干異なります。

2-2 コントロールと関数の追加

まず新規アプリケーションを作成します。フロント・パネル・ウインドウを表示し、**error in** クラスタと **error out** クラスタを置いてください。

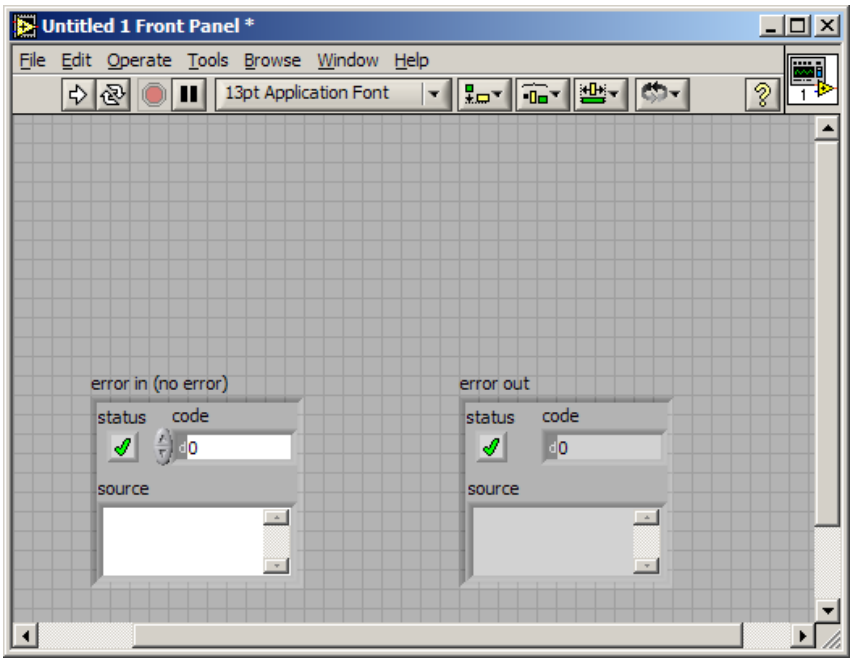


Figure 2-2 Front Panel

次にブロックダイアグラム・ウインドウを表示して、ki4800 ドライバ(ラッパー)のファンクション・パレットを開きます。このファンクション・パレットは、コンテキスト・メニュー → **Instrument I/O** → **Instrument Drivers** → **ki4800** から見つける事ができます。

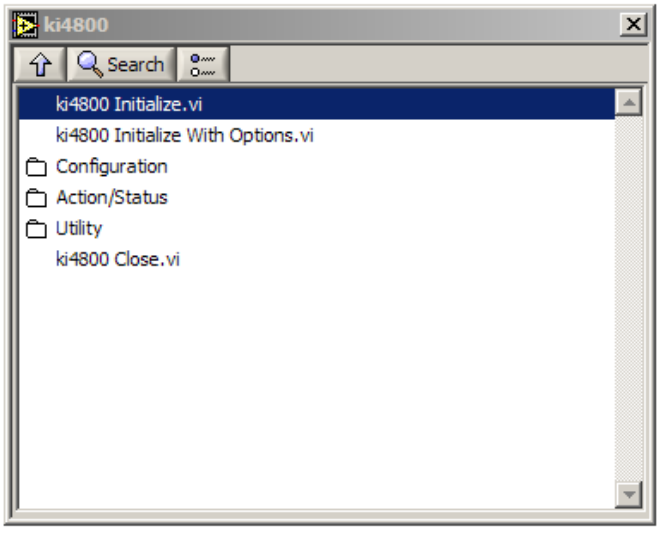


Figure 2-3 ki4800 Function Palette

ブロックダイアグラム上に `Initialize With Options.vi`、`Close.vi` を置きます。更に、**Configuration→Output** パレットの中にある、`Configure Voltage Level.vi`、`Configure Current Limit.vi`、`Configure Output Enabled.vi`、を追加します。

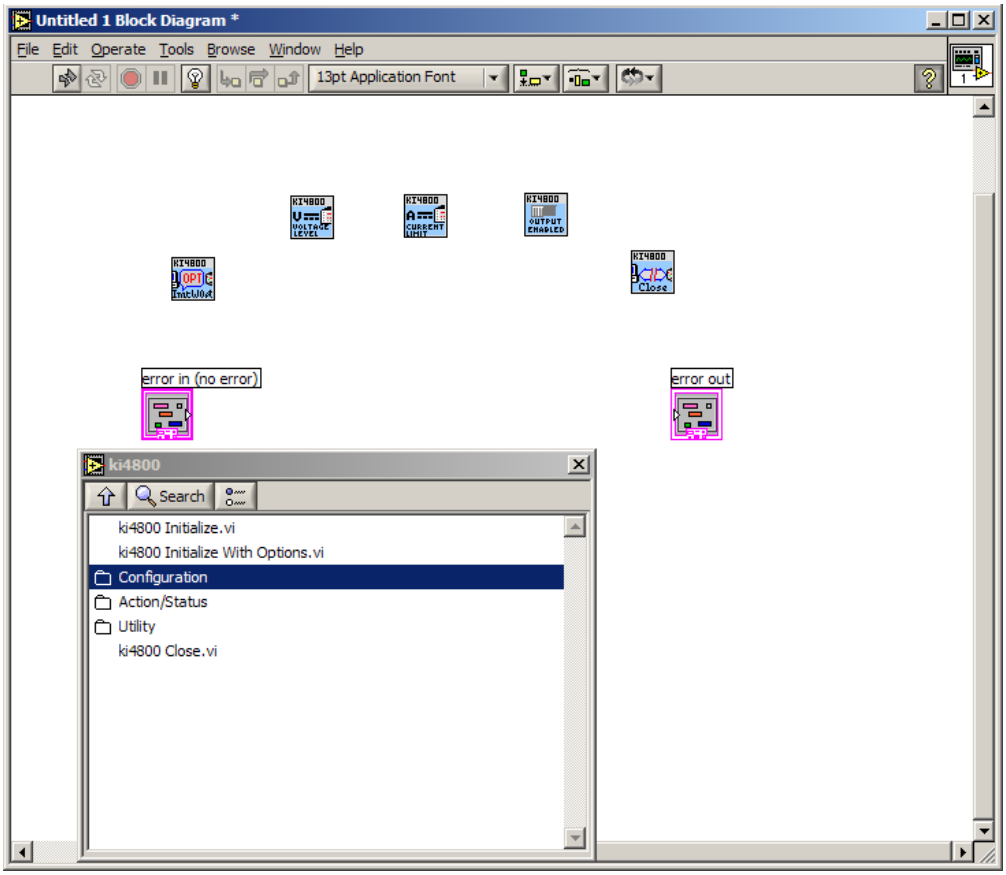


Figure 2-4 Block Diagram

2-3 パラメータの設定

ここでは、PIA4800 シリーズ電源コントローラが GPIB アドレス 3 に設定されているという前提で、`resource name`、`id query`、`reset device` パラメータを `Initialize With Options.vi` に渡します。

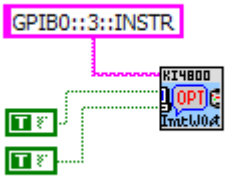


Figure 2-5 Params for Initialize With Options

次に電圧、電流、アウトプットの設定をするパラメータを追加します。ここでは 20V/2A 設定、アウトプット ON 設定を行います。

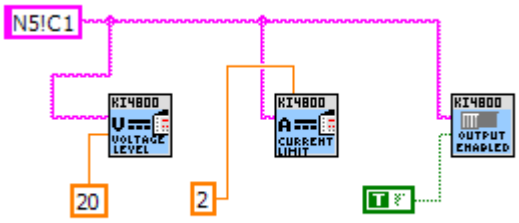


Figure 2-6 Params for Configure Functions

ここで、"N5!C1"という文字列に注意してください。これは制御対象となる DC 電源のチャンネル名です。詳細は後述します。

最後に、**error in** から **error out** クラスタまでを下記のようにワイヤー接続します。エラーin/out の接続だけでなく、計測器セッション(ハンドル)の接続も忘れずに行ってください。

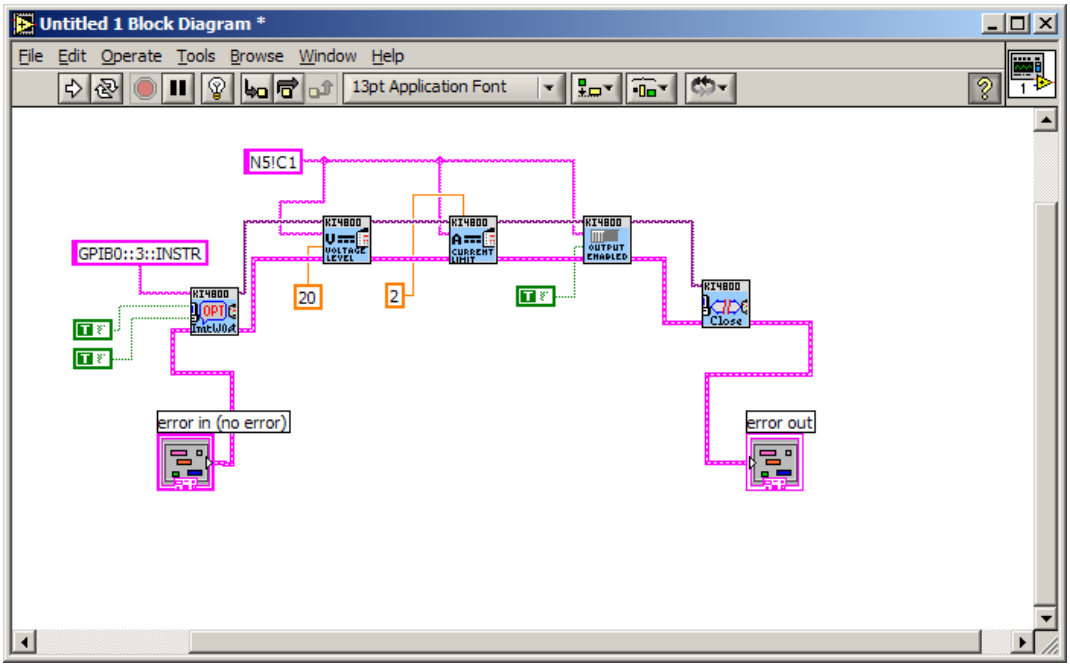


Figure 2-7 Open/Configure/Close

2-4 プログラムの実行

ここまでのコードだけで、とりあえず実行する事は可能です。`Initialize With Options.vi` の `Reset Device` パラメータに TRUE が指定されているので計測器はリセットされます。プログラムを実行すると、即座に計測器との通信が開始されます。実際に計測器が接続されていて `Initialize With Options/Close` の各 vi が成功した場合は、**error out** クラスタ上に表示されるエラー・コードは 0 です。通信に失敗した場合や、VISA ライブラリの設定が正しく行われていない場合などは例外が発生し、**error out** クラスタに表示されます。**error out** クラスタのコンテキスト・メニュー➔**Explain Error** でその詳細を見ることが出来ます。

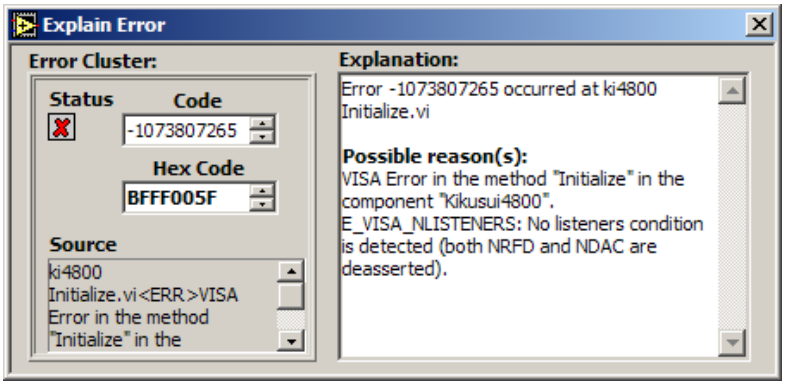


Figure 2-8 実行時エラー

Note:

KIKUSUI PIA4800 シリーズ電源コントローラは、接続されている DC 電源の接続状態の認識(TP-BUS サーチ)を行うのに数分かかります。この処理を Initialize 時に毎回行うわけには行きません。その為、

Kikusui4800(ki4800)ドライバを使用する場合は、Scan Utility を使って制御対象 DC 電源装置の接続状態を事前にコンフィグレーションしておく必要があります。ドライバを始めて使う場合は、必ず Scan Utility を実行してください。実行するには、[Start]ボタン→All Programs→IVI→Kikusui4800→Scan Utility メニューを選択します。

DC 電源装置の台数、ノード・アドレス、通信インターフェース(RS232/GPIB)、及びそれらのポート番号やアドレスを変更した場合は、再度 Scan Utility を実行する必要があります。

Scan Utility によるコンフィグレーションは、IVI-COM Kikusui4800(ki4800)ドライバ固有のものです。他の計測器ドライバでは必要ありません。

3- 解説

3-1 セッションの開始

セッションの開始には `ki4800 Initialize With Options.vi` を使用します。vi(関数)に付く `ki4800` というプレフィックスは計測器ドライバ毎に異なりますが、全ての IVI-C 計測器ドライバにはこのような名前ルールによる関数が用意されています。

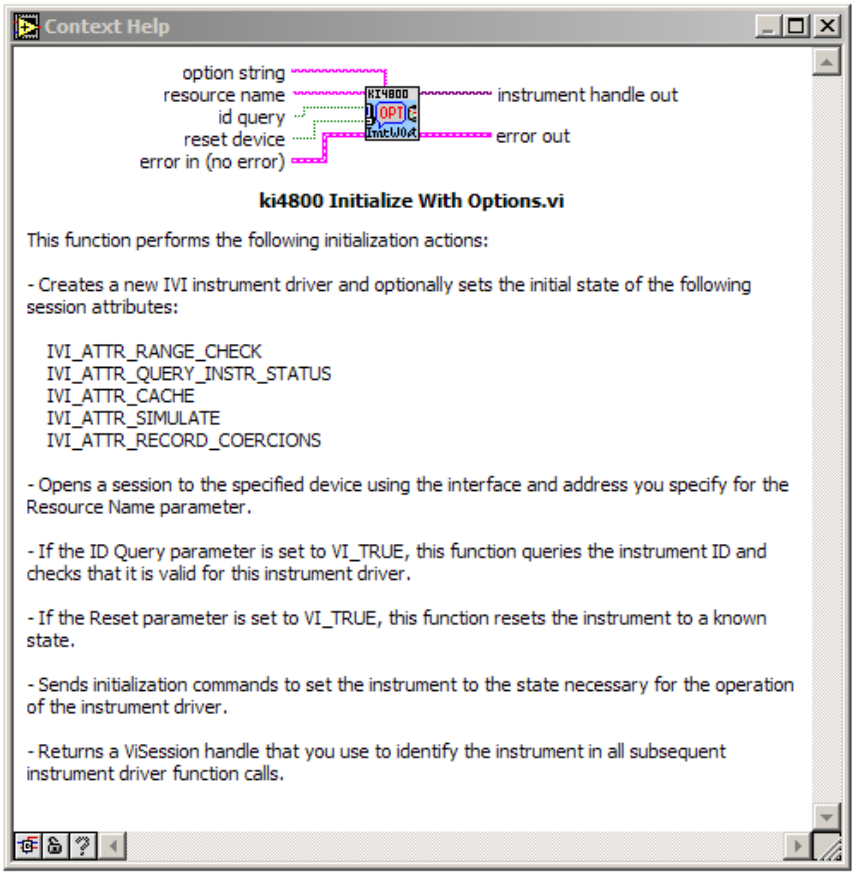


Figure 3-1 Initialize With Options.vi Help

Notes:

IVI-C 及び VXI Plug&Play 計測器ドライバの専門用語として<prefix>という表記が良く使われます。これは各計測器ドライバが固有に持つ識別用の名前で、本書の例では ki4800 がそれに該当します。例えば、<prefix> Initialize.vi という一般的表現は、ki4800 計測器ドライバでは ki4800 Initialize.vi の事を指します。

<prefix> Initialize.vi 及び<prefix> Initialize With Options.vi を除く全ての vi(ドライバ関数)は、左上の入力パラメータが instrument handle in になります。

<prefix> Close.vi を除く全ての vi(ドライバ関数)は、右上の出力パラメータが instrument handle out になります。これは次に来る vi の instrument handle in に接続されるべきものです。

<prefix> Initialize.vi は VXI Plug&Playドライバ仕様との互換性の為に残されています。option string パラメータを指定できない点を除き、<prefix> Initialize With Options.vi と同じ動作をします。

ここで、ki4800 Initialize With Options.vi のパラメータについて説明しましょう。全ての IVI 計測器ドライバは、IVI 仕様書で定義された Initialize With Options.vi を持っています。この vi には、以下のようなパラメータがあります。

Table 3-1 Initialize With Options のパラメータ

パラメータ	タイプ	説明
Resource Name	String	VISA リソース名の文字列。計測器が接続されている I/O インターフェース、アドレスなどによって決定される。例えば、GPIB ボード 0 に接続されたプライマリ･アドレス 3 の計測器であれば、GPIB0::3::INSTR となる。
Id Query	Boolean	VI_TRUE を指定した場合、計測器に対して ID クエリを行う。
Reset Device	Boolean	VI_TRUE を指定した場合、計測器の設定をリセットする。
Option String	String	RangeCheck Cache Simulate QueryInstrStatus RecordCoercions Interchange Check に関する設定を、デフォルト以外に指定できる。更に、計測器ドライバが DriverSetup 機能をサポートする場合、その設定を行うことができる。

Resource Name には VISA アドレス(リソース名)を指定します。ID Query に VI_TRUE を指定した場合は、計測器に対して"*IDN?"クエリなどを発行して機種情報を問い合わせます。Reset Device に VI_TRUE を指定した場合は、"*RST"コマンドなどを発行して計測器の設定をリセットします。

Option String には、2 つの機能があります。1 つは RangeCheck, Cache, Simulate, QueryInstrStatus, RecordCoercions, Interchange Check,などの IVI 定義の動作を設定します。もう 1 つは、計測器ドライバ毎に独自に定義される DriverSetup を指定します。Option String は文字列パラメータなので、これらの設定は下のサンプルのような書式でなければなりません。

```
QueryInstrStatus = TRUE , Cache = TRUE , DriverSetup=12345
```

設定したい機能の名称及び設定値はケース･インセンシティブ(大文字と小文字の区別なし)です。設定値は ViBoolean 型なので、VI_TRUE、VI_FALSE、1、0 の何れかが有効です。複数の項目を設定する場合は、コンマで区切ります。Option String パラメータで特に設定値を指定しない場合、IVI 仕様書で定義されたデフォルト値が適用されます。IVI 仕様書で定義されたデフォルト値は、RangeCheck と Cache だけが VI_TRUE で、その他は全て VI_FALSE です。

計測器ドライバによっては、DriverSetup パラメータが意味を持つ場合もあります。これは、IVI 仕様書では定義されない項目を InitializeWithOptions の呼び出し時に指定するもので、利用目的や書式はドライバ依存です。従って DriverSetup の指定を行う場合、それは optionString の最後の項目として指定される必要があります。DriverSetup の指定内容はドライバ毎に異なるので、ドライバの Readme 文書又はオンライン･ヘルプなどを参照してください。

3-2 チャンネルへのアクセス

IVI 計測器ドライバでは一般に、電源装置や電子負荷装置などの計測器の場合、複数のチャンネルが装備されている事を前提に設計されています。従って、計測器のパネル設定に関する操作を行うドライバ関数は、**channel name** パラメータにチャンネルを指定するケースが多く見られます。

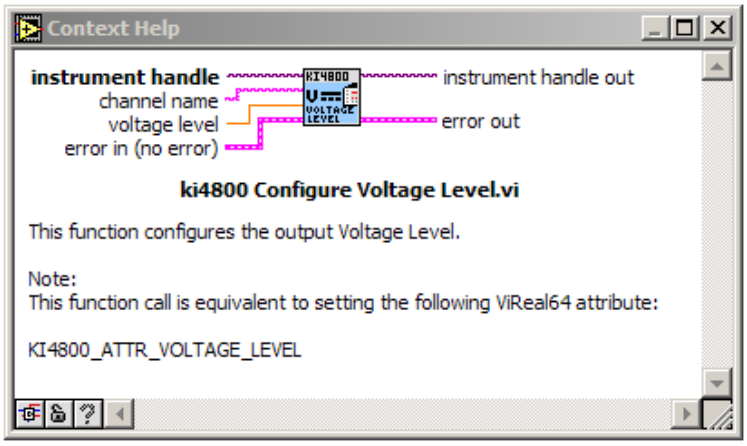


Figure 3-2 Configure Voltage Level.vi Help

本ガイドブックでは直流電源装置を操作する Kikusui4800 (ki4800)ドライバを使用するので、チャンネル名には NODE と CHANNEL を含ませた表現を使用します。上記例の"N5!C1"というチャンネル名はこの計測器ドライバ固有のものであり、ドライバ毎に異なる命名法が使用されます。実際に使用できるチャンネル名の詳細は、各ドライバのオンライン・ヘルプなどを参照してください。

ここでは、PIA4800 シリーズ電源コントローラの NODE5、CHANNEL1 に接続された直流電源を 20V に設定します。

3-3 セッションのクローズ

計測器ドライバによるセッションをクローズするには、`ki4800 Close.vi` 関数を使います。

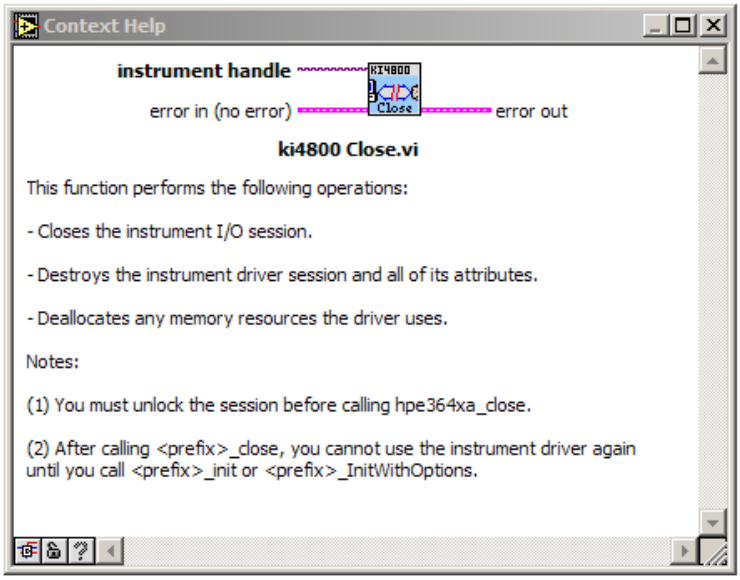


Figure 3-3 Close.vi Help

4- クラス・インターフェースを使用するサンプル

ここでは、クラス・インターフェースを使用したサンプルを示します。クラス・インターフェースを使用すると、アプリケーションを再度コンパイル・リンクすることなく、計測器を交換する事ができます。但しその場合、交換前後の両機種に対して IVI-C 計測器ドライバが提供されており、且つそれらのドライバが同じ計測器クラスに属している必要があります。異なる計測器クラス間でのインターチェンジャビリティは実現できません。

4-1 仮想インストルメント

インターチェンジャビリティ機能を利用するアプリケーションの作成を行う前にやっておかなければならない事は、仮想インストルメントの作成です。インターチェンジャビリティ機能を実現するには、アプリケーション・コード内に特定の IVI-C 計測器ドライバに依存した記述(例えば `ki4800 Initialize.vi` の呼び出し)をしたり、`"GPIB0::3::INSTR"`のような特定 VISA アドレス(リソース名)の記述などをするべきではありません。これらの事柄をアプリケーション内に直接記述すると、インターチェンジャビリティを損ないます。

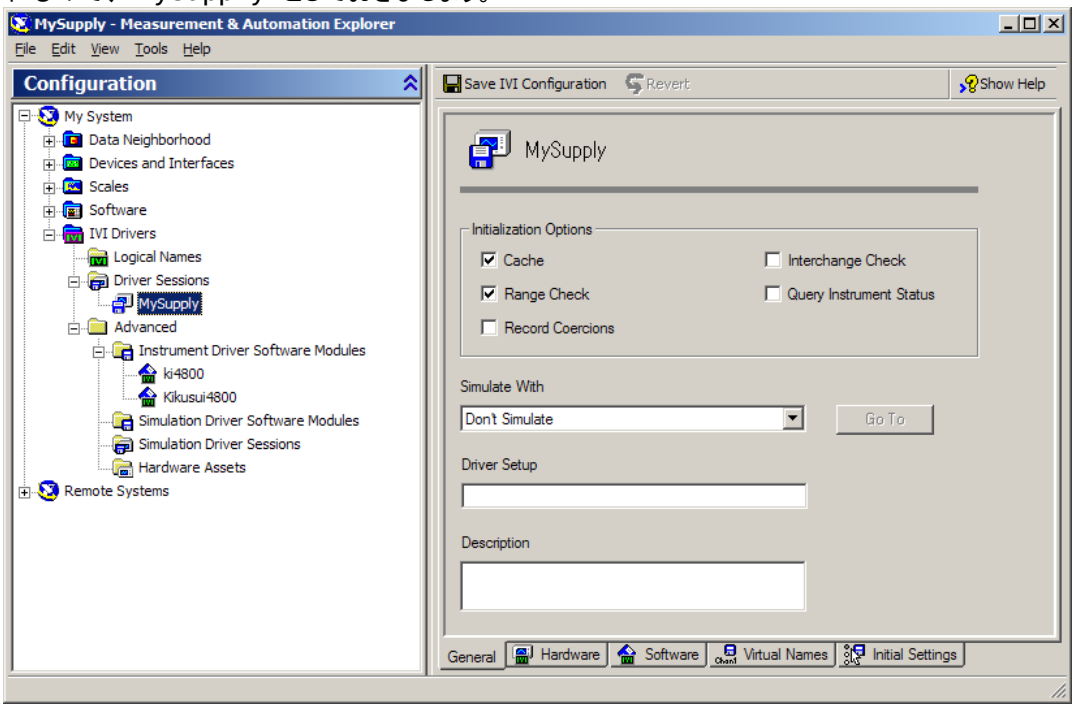
その代わりに、IVI 仕様では、計測器ドライバとアプリケーションの外部に IVI コンフィグレーション・ストアを置く事によってインターチェンジャビリティを実現します。アプリケーションは特定機種用の計測器ドライバを直接使うのではなく、クラスドライバと呼ばれる計測器ドライバを通じて計測器を制御します。その際、IVI コンフィグレーション・ストアの内容に従って計測器ドライバ DLL の選択を行い、間接的にロードされた計測器ドライバを特定機種に依存しないクラスドライバの関数を通じてアクセスします。

IVI コンフィグレーション・ストアは通常、/Program Files/IVI/Data/IviConfigurationStore.XML ファイルで、IVI Configuration Server DLL を通じてアクセスされます。この DLL を利用するのは、主に IVI 計測器ドライバや一部の VISA/IVI コンフィグレーション・ツールであって、アプリケーションからは通常は使いません。LabVIEW を使用する場合は National Instruments 社製のソフトウェア NI-MAX (NI Measurement and Automation Explorer)を使用して IVI ドライバのコンフィグレーションを行います。

Driver Session の作成

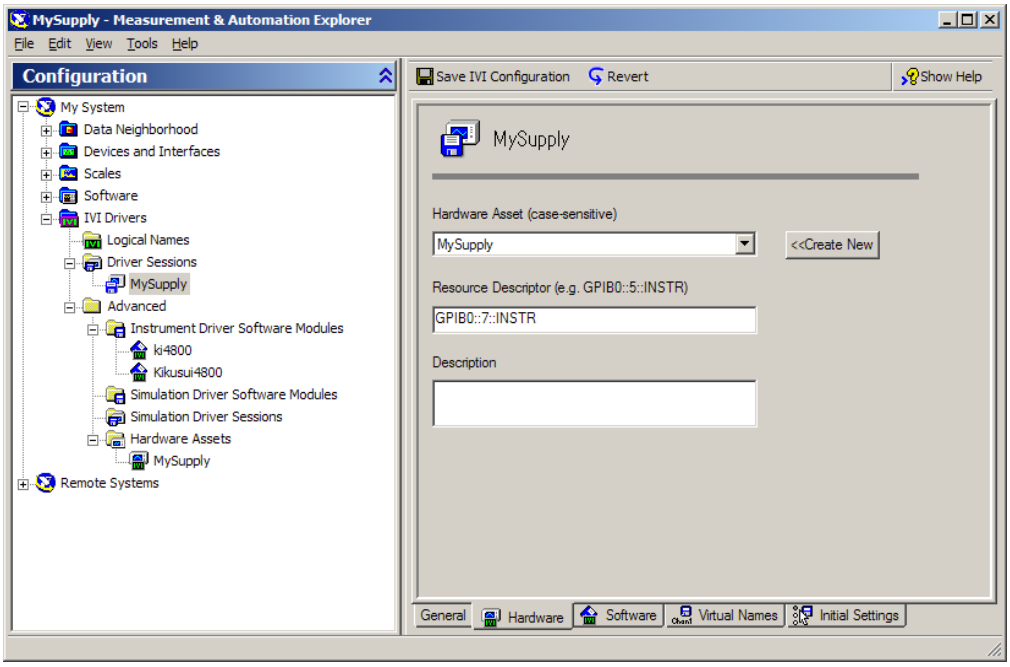
NI-MAX を起動したら、**IVI Drivers** ノードの階層を参照して下さい。**Driver Seesion** の上で右クリックして **Create New** メニューを選択し、Driver Session の新規作成を行います。名前を尋ねら

れるので、"MySupply"としておきましょう。



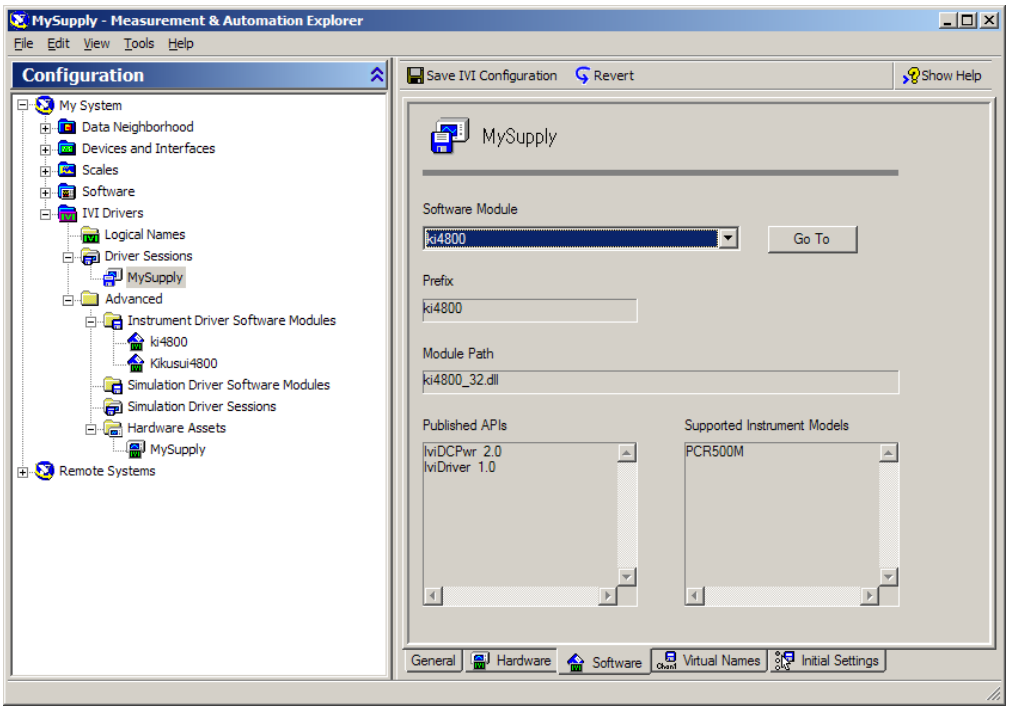
Hardware Asset の作成

引き続き **Hardware** タブを選択すると、ハードウェア・アセットの管理画面になります。ハードウェア・アセットとは、実際の計測器がどのような経路に接続されているかを示すものです。ここで **Create New** ボタンをクリックして Hardware Asset を新規作成します。新しい名前を尋ねられるので、ここでも"MySupply"として **Create** ボタンをクリックします。更に **Resource Descriptor** として、実際の計測器が接続されている VISA アドレスを指定します。



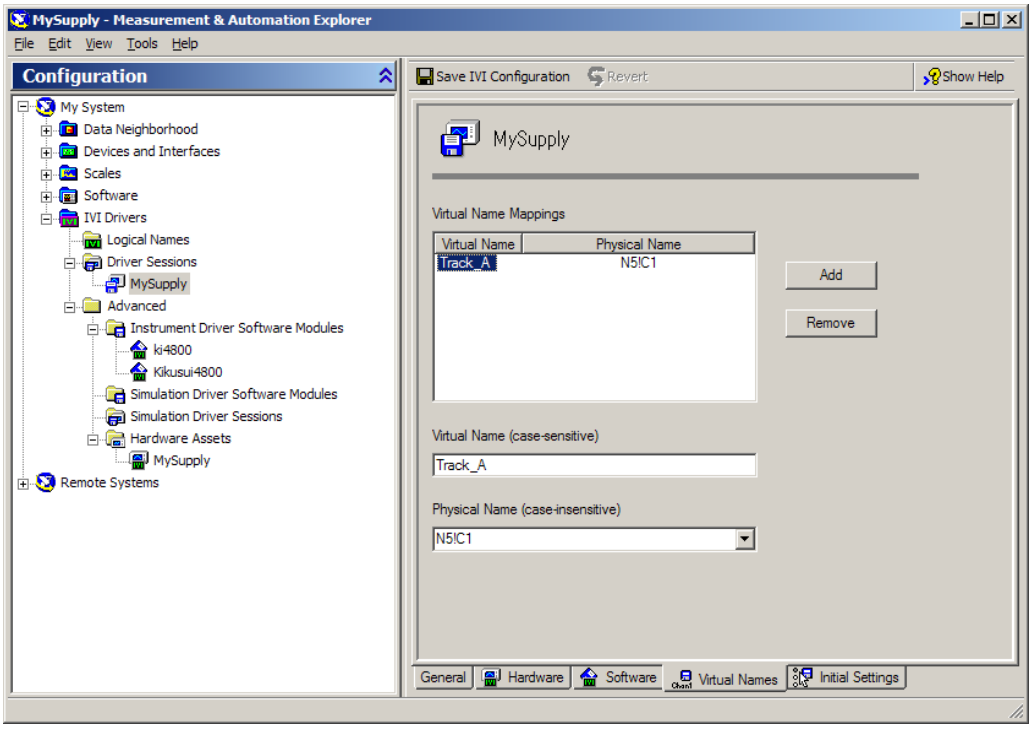
Software Module のリンク設定

引き続き **Software** タブを選択すると、ソフトウェア・モジュールの管理画面になります。ソフトウェア・モジュールとは計測器ドライバモジュール(DLL モジュール)の事を指します。ここで **Software Module** のリストから **ki4800** を選択します。



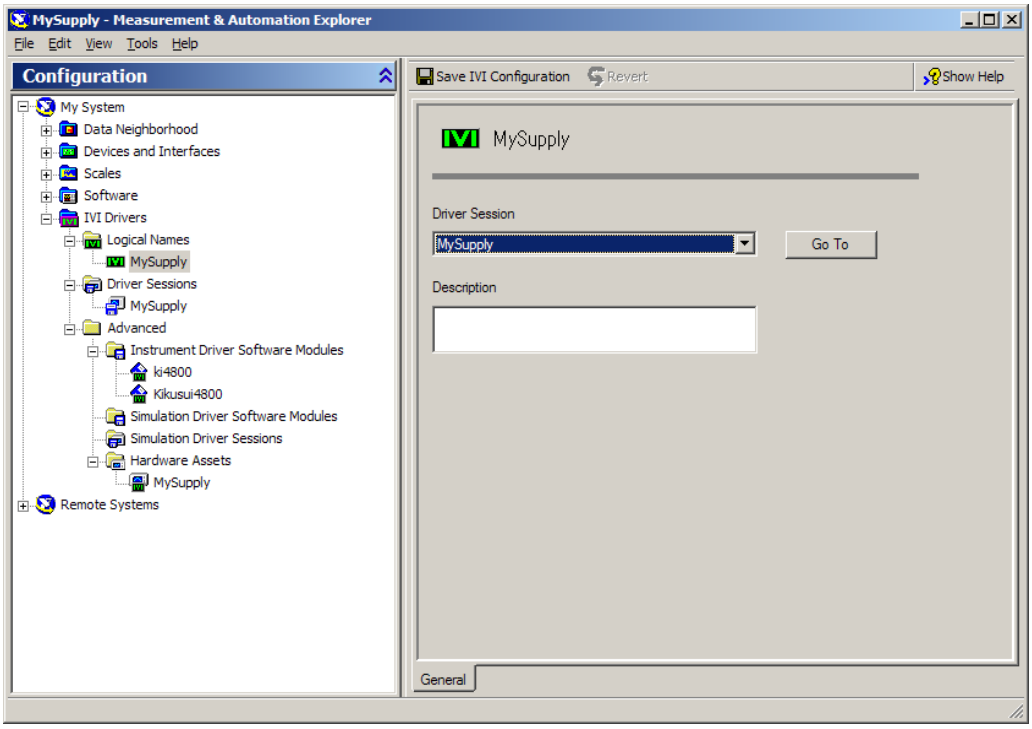
Virtual Name の作成

引き続き **Virtual Names** タブを選択すると、仮想チャンネル名の管理画面になります。通常、電源装置の計測器ドライバのようにチャンネルが関与する場合、有効なチャンネル名は計測器ドライバによって異なります。従ってそれらのチャンネル名も仮想化してやる必要があります。**Add** ボタンをクリックしてバーチャルネームを追加し、**Virtual Name** に"Track_A"と入力します。更に **Physical Name** リストから、実際の直流電源装置が接続されている NODE/CHANNEL に応じた名前を選択します。下の例では NODE 5, CHANNEL 1 に接続された場合で、**N5!C1** を選択しています。



Logical Name の作成とリンク設定

最後にロジカルネームを作成します。ロジカルネームとは、NI-MAX で設定される仮想計測器の名前に相当します。**IVI Drivers** ノードの階層を参照して下さい。**Logical Name** の上で右クリックして **Create New** メニューを選択し、Logical Name の新規作成を行います。名前を尋ねられるので、"MySupply"としておきましょう。更に、**Driver Session** リストから"MySupply"を選択します。



仮想計測器の設定はこれで終了です。NI-MAX 画面上部の **Save IVI Configuration** ボタンをクリックして設定を保存します。

4-2 コントロールと関数の追加

まず新規アプリケーションを作成します。フロント・パネル・ウインドウを表示し、**error in** クラスタと **error out** クラスタを置いてください。

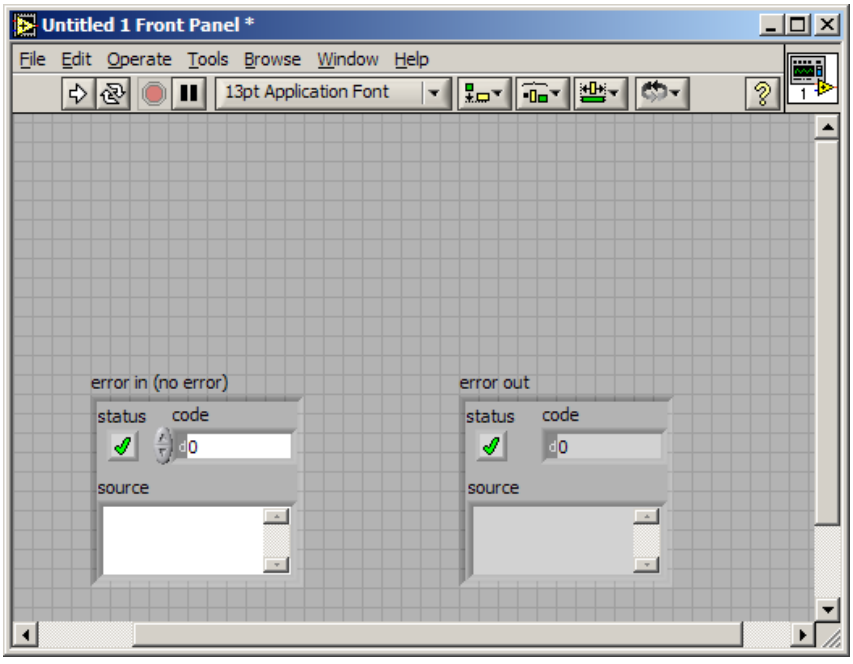


Figure 4-1 Front Panel

次にブロックダイアグラム・ウインドウを表示して、IviDCPwr クラス・ドライバ(ラッパー)のファンクション・パレットを開きます。このファンクション・パレットは、コンテキスト・メニュー → **Instrument I/O** → **IVI** → **IVI DC Power Supply** から見つける事ができます。

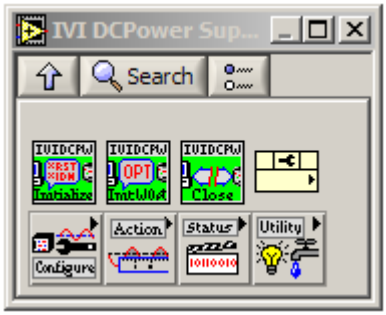


Figure 4-2 IviDCPwr Function Palette

更にブロックダイアグラム上に `Initialize With Options.vi`、`Close.vi` を置きます。更に、**Configuration→Output** パレットの中にある、`Configure Voltage Level.vi`、`Configure Current Limit.vi`、`Configure Output Enabled.vi`、を追加します。

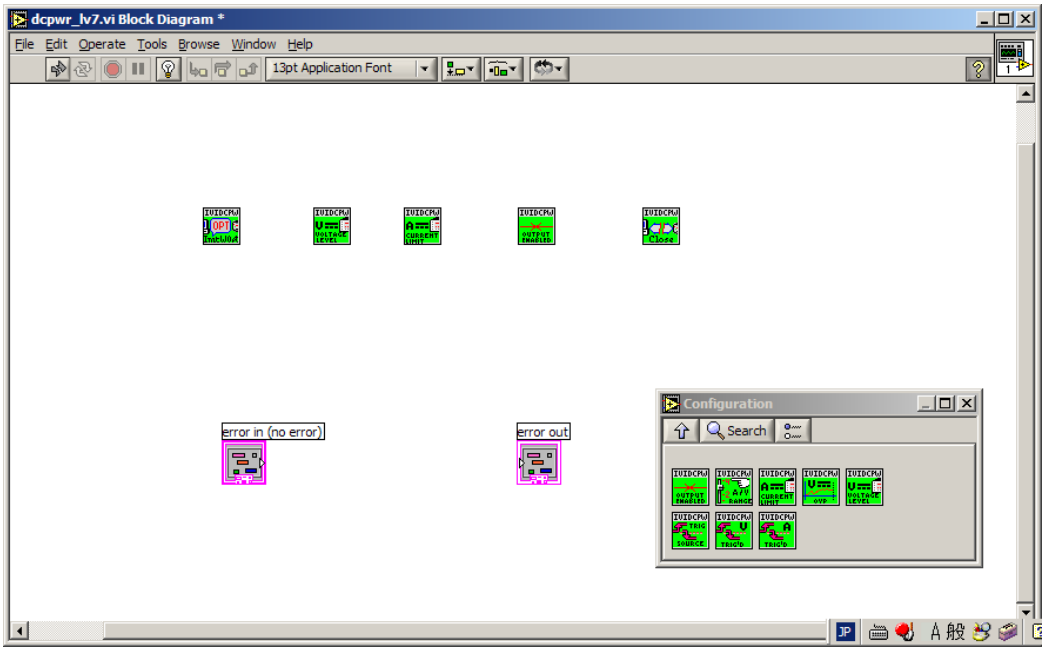


Figure 4-3 Block Diagram

ここでは、PIA4800 シリーズ電源コントローラが GPIB アドレス 3 に設定されているという前提で、`resource name`、`id query`、`reset device` パラメータを Initialize With Options.vi に渡します。

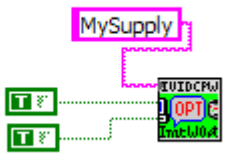


Figure 4-4 Params for Initialize With Options

次に電圧、電流、アウトプットの設定をするパラメータを追加します。ここでは 20V/2A 設定、アウトプット ON 設定を行います。

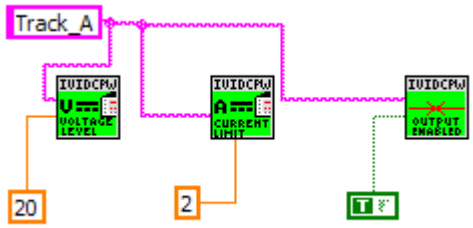


Figure 4-5 Params for Configure Functions

ここで、"Track_A"という文字列に注意してください。これは制御対象となる DC 電源のチャンネル名(バーチャル・ネーム)です。詳細は後述します。

最後に、**error in** から **error out** クラスタまでを下記のようにワイヤー接続します。エラーin/out の接続だけでなく、計測器セッション(ハンドル)の接続も忘れずに行ってください。

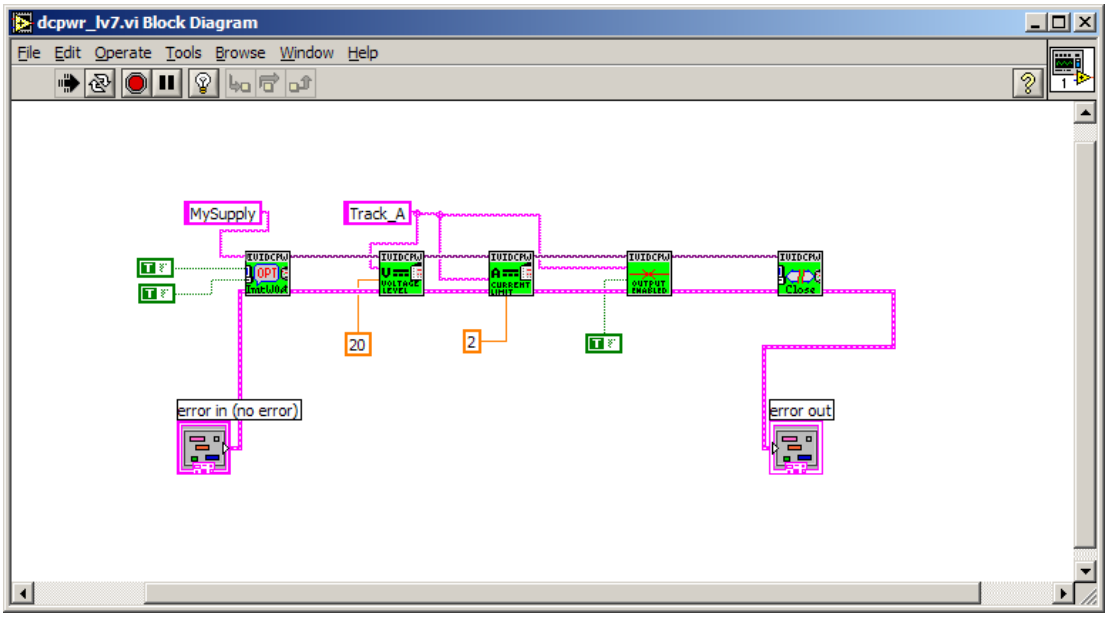


Figure 4-6 Open/Configure/Close

5- 解説

5-1 セッションの開始

セッションの開始には `IviDCPwr Initialize With Options.vi` を使用します。vi(関数)に付く `IviDCPwr` というプレフィックスは IviDCPwr クラスドライバ固有のものです。

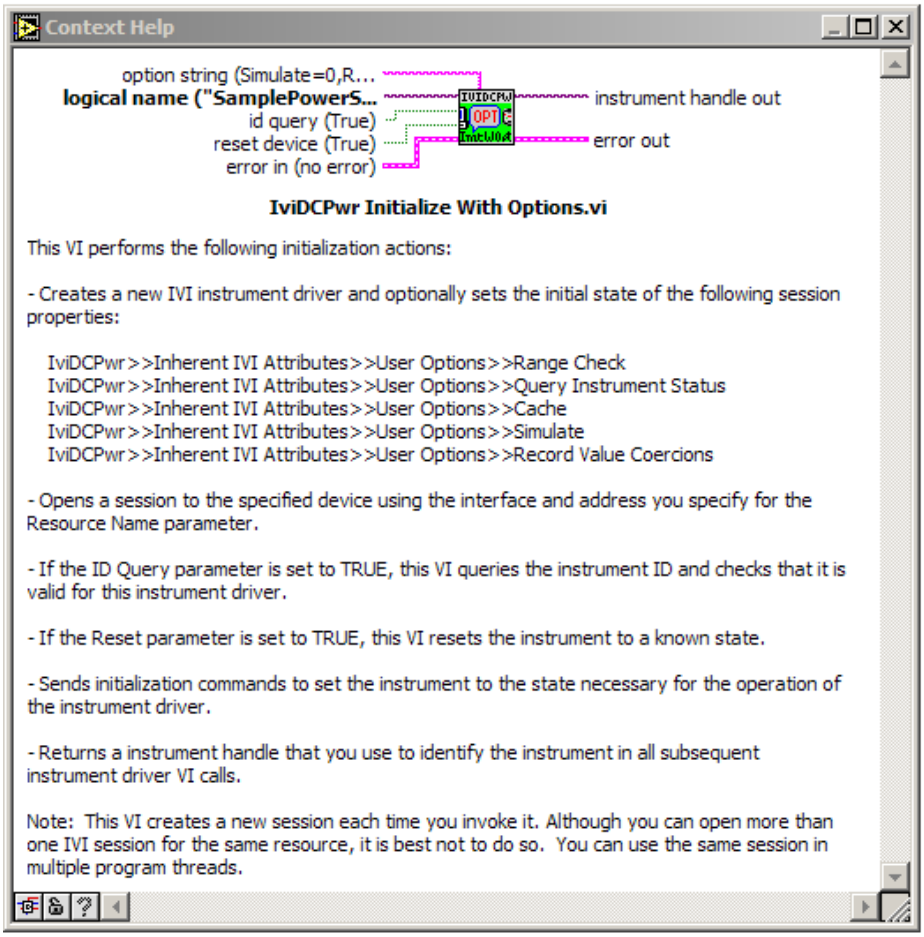


Figure 5-1 Initialize With Options.vi Help

クラスドライバは通常の計測器ドライバとは異なり、`Initialize With Options.vi` に直接 VISA アドレスを渡すことはできません。ここでは NI-MAX のロジカルネームに指定した `"MySupply"`を指定します。クラスドライバはこのロジカルネームを頼りに適切な計測器ドライバ DLL(Software Module)や VISA アドレス(Hardware Asset)を探し当て、最終的に `ki4800 Initialize With Options.vi` を間接的に呼び出します。

Option String パラメータに渡す内容はスペシフィックドライバを直接使用する場合と同じですが、省略された場合のデフォルト値が異なります。スペシフィックドライバを直接使用する場合のデフォルトは IVI 仕様によって定義された値になりますが、クラスドライバを使用する場合のデフォルトは IVI コンフィグレーションストア内の **Driver Session** に指定された値です。

5-2 チャンネルへのアクセス

IVI 計測器ドライバでは一般に、電源装置や電子負荷装置などの計測器の場合、複数のチャンネルが装備されている事を前提に設計されています。従って、計測器のパネル設定に関する操作を行うドライバ関数は、channel name パラメータにチャンネルを指定するケースが多く見られます。

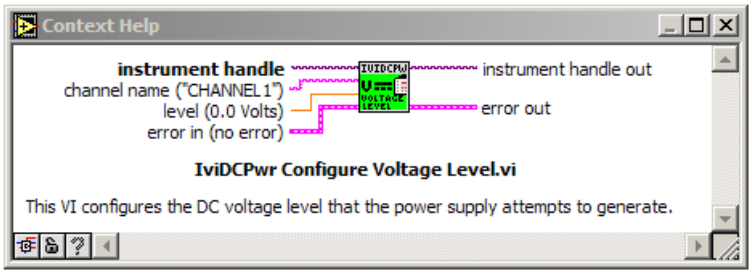


Figure 5-2 Configure Voltage Level.vi Help

チャンネル名を vi に渡す場合、"N5!C1"のような特定の計測器ドライバ(この場合は ki4800 ドライバ)でのみ使用可能な名前を指定する事も可能ですが、この指定方法で計測器の制御を行うと、特定機種の計測器ドライバに依存した名前を使用する事になり、インターチェンジャビリティを損ないます。

先の NI-MAX によるコンフィグレーションでは、バーチャルネームとして"Track_A"という名前を追加し、それが"N5!C1"というフィジカルネームに変換されるように設定しました。従ってここでは、チャンネル名にバーチャルネームを使うことができます。

5-3 セッションのクローズ

計測器ドライバによるセッションをクローズするには、IviDCPwr Close.vi 関数を使います。

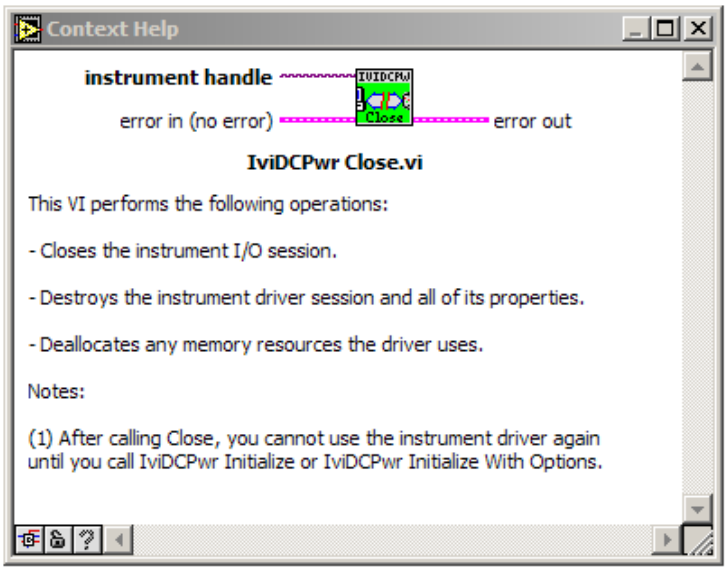


Figure 5-3 Close.vi Help