NXP Semiconductors
ユーザー・ガイド
ドキュメントID：FMSTERUG
Rev. 4.4、03/2022

FreeMASTER for Embedded Applications

目次

第 1 章
はじめに

1.1 概要

このユーザー・ガイドでは、FreeMASTER アプリケーションについて説明しています。これは、PC 上で実行されるグラフィカル環境を使用して組み込みアプリケーションを制御することを目的に、NXP によって開発されたアプリケーションです。本来はリアルタイム・モーター制御アプリケーションの開発者向けに開発されましたが、多くのユーザーによって、さまざまな分野の開発に非常に役立つと評価されています。

FreeMASTER 3.0 アプリケーションは以前の 2.x および 1.x バージョンとの完全な下位互換性があるほか、最初の「PC Master」1.0 バージョンとも互換性があります。

1.2 サポートされるプラットフォーム

FreeMASTER デスクトップ・アプリケーションは、現時点で Microsoft によってサポートされているすべての Windows® OS ベース・システムにインストールできます。

FreeMASTER 3.0 パッケージには FreeMASTER「Lite」サービスが同梱されています。Lite サービスは他のオペレーティング・システム（Linux OS など）上で動作でき、各種制御ページとターゲット・マイクロコントローラとの間で通信インターフェースとしての役割も果たします。

ターゲット・マイクロコントローラ側では、最新バージョンの FreeMASTER 通信プロトコル V4 をサポートする通信ドライバを、NXP MCUXpresso SDK スイートや他の SDK を通じて、また、ダウンロード可能なスタンドアロンのパッケージとして利用できます。旧バージョンのプロトコルをサポートする旧型ドライバはスタンドアロン・パッケージとして提供されており、HCS08、S12、S12X、S12Z、ColdFire、Power Architecture など、古い Freescale マイクロコントローラ・プラットフォームに対応します。

1.2.1 UART および SCI について

FreeMASTER にはいくつかのプラグイン・モジュールが付属し、代替の通信インターフェースを介してターゲット・ハードウェアにアクセスすることができます。

FreeMASTER で BDM 通信用プラグインを使用すると、ターゲット CPU を介さなくても、HCS08、S12Z、ColdFire、Arm® Cortex®-M の各プラットフォームに対する基本的なメモリ・アクセス操作を実行できます。つまり、組込み側の通信ドライバがなくても、FreeMASTER はその基本的なタスクであるターゲット・メモリの読み取りと書き込みを実行することができます。このプラグインは、BDM と JTAG のデバッグ・プローブ、たとえば P&E Multilink、SEGGER jLink、Arm CMSIS-DAP をサポートしています。

パケットドリブン BDM 通信用プラグインを BDM プラグイン上の追加レイヤとして使用することで、Recorder、TSA、メモリ保護といった高度なプロトコル・コマンドが可能となります。このプラグインは単純なデータの直接アクセスを行うのではなく、JTAG または BDM インターフェースを使用してターゲット・ドライバとの間で通信プロトコルのフレームを交換します。この際にはターゲット上でシリアル・ドライバを実行する必要があるため、そのドライバをパケットドリブン BDM インターフェース向けに適切に設定しなければなりません。

FreeMASTER-over-CAN プラグインを使用すると、FreeMASTER のサービスを CAN トランシーバ経由で使用できます。このプラグインは、msCAN や FlexCAN など各種ペリフェラル・モジュール用に設定されてターゲット上で実行されるシリアル・ドライバと組み合わせて使用します。

FreeMASTER TCP/UDP 通信用プラグインを使用すると、FreeMASTER がネットワーク・プロトコルを介して、イーサネット（または WiFi 接続）と TCP/IP スタックをサポートするターゲットと直接接続できます。このプラグインの最初のプロトタイプが、FreeMASTER 3.1.2 に含まれています。i.MX-RT1xxx および Kinetis K64F プラットフォーム向けのサンプル MCU アプリケーションが、MCUXpresso SDK バージョン 2.10 に同梱されています。このサンプルは、lwIP スタックと有線イーサネット接続を利用したもので、「fmstr_net」という名前が付けられています。将来のバージョンではサポート範囲が拡大され、より多くのプラットフォームと WiFi 通信に対応する予定です。

FreeMASTER ツールを実行している複数のクライアントがネットワーク通信を利用して 1 つのサーバ（マイクロコントローラ・ボード）に接続すると、何らかのアクセス衝突が生じる恐れがあります。MCUXpresso SDK 2.10 でリリースされた FreeMASTER ドライバは、複数のセッションと「機能ロック」をサポートします。これにより、すべてのクライアントが同期化された適切な方法でターゲット・リソースにアクセスできるようになります。

バージョン 3.1.3 では、SEGGER RTT インターフェースを介した通信によって、ネットワーク・プラグインがさらに拡張されています。このインターフェースは JTAG ポートを使用するもので、SEGGER J-Link デバッガ・プローブによってサポートされます。RTT 通信の特長は、アクティブなデバッガ・セッションと並行して FreeMASTER を使用できる点です。

通信用プラグイン・モジュールの詳細については、最新の FreeMASTER アプリケーションに付属する「readme」ドキュメントを参照してください。

1.3 最新バージョンの入手先

FreeMASTER パッケージの最新のインストール・パッケージは、www.nxp.jp/freemaster の「Downloads（ダウンロード）」セクションから入手できます。このページでは、旧バージョンのツールも入手できます。

同じページにターゲット・マイクロコントローラのドライバがあり、単体でダウンロードすることができます。ただし、MCUXpresso SDK（www.nxp.jp/mcuxpresso/sdk）など、NXP から提供されている SDK に付属のドライバの使用をお勧めします。mcuxpresso.nxp.com にアクセスし、FreeMASTER ミドルウェア・コンポーネントとサンプル・アプリケーションを含む SDK パッケージを、SDK ビルダーを使用して生成しましょう。

1.4 FreeMASTER の機能

- グラフィカル環境と分かりやすい操作
- シンプルなシリアル（UART）ネイティブ接続に加え、一部のプラットフォームでは他のオプションも選択可能（BDM、JTAG、CAN、TCP、UDP、SEGGER RTT など）
- 実行中のターゲット・マイクロコントローラ・アプリケーションの C 変数にリアルタイムでアクセス
- オシロスコープウィンドウでリアルタイム・データを可視化
- オンターゲット・レコーダを使用して高速なデータの変化を収集
- 変数の標準型（整数、浮動小数点、ビット・フィールド）と拡張型（小数）を標準サポート
- カスタム定義のテキスト・メッセージを使用して値を解釈可能
- 実数型の変数に対する変換機能を複数搭載
- コンパイラ出力ファイルから変数のアドレスとサイズを自動抽出
- ターゲット・アプリケーションから取得される変数のオブジェクトと型についての TSA（Target-Side Addressing）情報をサポート
- パスワード保護されたデモ・モードのサポート
- HTML ベースの制御ページと説明ページは Internet Explorer または Chromium エンジンによってレンダリング
- ActiveX インターフェースにより、Internet Explorer ページや ActiveX テクノロジをサポートするサード・パーティ・アプリケーションから VBScript または JavaScript で組み込みアプリケーションを制御可能
- JSON-RPC インターフェースにより、Chromium ページ、スタンドアロンの Chrome ブラウザ、node.js スクリプト、Python スクリプトのほか、JSON-RPC プロトコルをサポートする任意のサード・パーティ・アプリケーションからの制御が可能

1.5 FreeMASTER オンライン・コミュニティ

https://community.nxp.com/community/freemaster で、オンラインの FreeMASTER コミュニティ・フォーラムにアクセスできます。質問の投稿、フィードバックの共有、開発チームやサポート・チームへの問い合わせにご利用ください。

第 2 章
質問と回答

このユーザー・ガイドの制作中に、以下の質問が寄せられました。このような質問への回答は、本ユーザー・ガイドの中で説明されているトピックや用語の理解に役立ちますので、こちらのセクションを先に掲載しました。以降のセクションは理解が難しい場合もありますので、こちらに目を通した後で、詳しい解説が記載されたセクションに進むようにしてください。

2.1 FreeMASTER がなぜ必要なのですか?

FreeMASTER は、モーター制御のアルゴリズムやアプリケーションをデバッグしたりデモンストレーションしたりするのためのツールの実現を主な目的として開発されました。その結果、多目的のアルゴリズムやアプリケーションに使用できる用途の広いツールとなっています。以下に実際の使用例を示します。

- リアルタイム・デバッグ:FreeMASTER には変数を監視する機能があるため、文字どおりリアルタイムでアプリケーションをデバッグできます。また、アルゴリズム・レベルでのデバッグが可能であるため、開発フェーズの短縮につながります。
- 診断ツール：FreeMASTER はリモート制御機能を備えているため、お客様のアプリケーションをリモートからネットワーク経由でデバッグするための診断ツールとして使用できます。
- デモ用途：FreeMASTER は、アルゴリズムやアプリケーションの実行デモ、さらに変数の出力デモを行う際に極めて役立つツールです。
- 教育用途:FreeMASTERは教育にも使用できます。アプリケーション制御機能を備えているため、デモンストレーション・モードで試行錯誤しながら、プログラムの実行制御の方法を学ぶことができます。

2.2 FreeMASTER は何をするのですか?

FreeMASTER は、ターゲット・システムのアプリケーションとのシリアル通信によって、アプリケーションの内部変数の読み取りと書き込みを行います。FreeMASTER は以下の可視化機能を備えており、変数の情報を見やすい形式で表示することができます。

- 変数ウォッチ:選択した変数の値が、あらかじめ定義されたテキスト・ベース形式でグリッドに表示されます。値は、グリッド内で直接変更できます。
- オシロスコープ：CRT を備えた一般的なオシロスコープと同じ感覚でアプリケーションの変数を監視 / 可視化できます。この場合、監視の速度は、シリアル通信の速度によって制限されます。
- レコーダ：オシロスコープのサンプル・レートを超える速度で変化するアプリケーションの変数を監視 / 可視化できます。オシロスコープが定期的に FreeMASTER の変数値を読み取ってリアルタイムでプロットするのに対し、レコーダは、ターゲット・ボード上で実行されます。変数の値はサンプリングされてボード上のメモリ・バッファに格納され、サンプリングされたデータがボードから FreeMASTER にダウンロードされます。このメカニズムによってサンプリング周期を大幅に短縮でき、ごく短い変動のサンプリングとプロットが可能となります。

2.3 FreeMASTER がそのように優れたデモンストレーション・ツールであるのはなぜですか?

組込み側アルゴリズムのデモを 1 ブロックで実行できるほか、複数のブロックに分けた方がアルゴリズムの構造をより良く反映するのであれば、そのような方法でデモを実行することもできます。各ブロックの入力パラメータを詳しく見ることで、それらが出力パラメータに及ぼす影響を観察できます。各ブロックには説明タブがあり、マルチメディア対応のスクリプタブルな HTML 形式でアルゴリズムの詳細が説明されます。

2.4 FreeMASTER では、インストラクションに従った場合、何ができるのですか?

FreeMASTER の使い方は、組込み側の実装に付属のデモ・プロジェクトを使用して、プロジェクトに定義されているブロックやパラメータに手を加えながら簡単に覚えることができます。デモ・プロジェクトを通じて、アプリケーションの制御方法を理解することができます。それぞれの項目の細部に踏み込んでそのプロパティを確認したり、パラメータを変更したり、自分のアプリケーションでのその使い方を調べたりすることができます。パラメータの詳細については、「FreeMASTER の使用法」を参照してください。

2.5 FreeMASTER はターゲット開発ボードにどのように接続されるのですか?

FreeMASTER を使用するためには、ターゲット開発ハードウェアにシリアル通信ポートが必要です。接続には、標準の RS-232 シリアル・ケーブルを使用します。シリアル・ケーブルの一方を PC のシリアル・ポート(COM1、COM2 など)に、もう一方をターゲット開発ボードのシリアル・コネクタに接続します。

RS-232 リンクに加え、カスタム通信用プラグイン・モジュールを作成して、FreeMASTER で使用することもできます。CAN Calibration Protocol、56F800E の JTAG Real-time Data Exchange ポート、HCS08/12 デバイスの BDM インターフェースなどに対応する通信用プラグインが提供されています。

また、FreeMASTER バージョン 3.1.2 と MCUXpresso SDK バージョン 2.10 には、ターゲット・ボードと TCP または UDP で直接通信する新しい通信オプションも導入されています。

2.6 さまざまなダイアログ・ボックスの目的を教えてください

「FreeMASTER の使用法」には、さまざまなダイアログ・ボックスの図が記載されています。このようなダイアログ・ボックスは情報の入力に使用されます。たとえば、特定のアルゴリズム・ブロックやアプリケーション変数、その可視化を記述するパラメータをダイアログ・ボックスに入力します。

2.7 プロジェクトとアプリケーションの関係を教えてください

1つのターゲット・ボード・アプリケーションに対して、FreeMASTER プロジェクトを複数使用する場合があります。たとえば、3 つの FreeMASTER プロジェクトがそれぞれ、同じボード・アプリケーションと連携して以下の 3 種類の役割を担うことができます。

デバッグ・プロセス中に使用された情報を提供する

サービスのメンテナンス機能を提供する

アプリケーションについての知識を身に付ける（オペレータのトレーニング目的）

2.8 リモート制御のセットアップ方法と用途について教えてください

リモート制御を行うためには、ネットワークで接続された少なくとも 2 台のコンピュータが必要です。1 台は FreeMASTER Remote Communication Server と呼ばれるスタンドアロンのミニアプリケーションを実行し、もう 1 台は標準の FreeMASTER アプリケーションを実行します。そのうえで、FreeMASTER Server が実行されているコンピュータにターゲット開発ボードを接続します。

リモート制御操作は、リモート・デバッグやリモート診断に活用できます。ターゲット開発ボードをリモート PC に接続した後、お客様のアプリケーション用のサービス・プロジェクトと FreeMASTER をローカル実行することによってアプリケーションをリモートから診断できます。

2.9 ウォッチ・グリッドとは何ですか？

ウォッチ・グリッドは、FreeMASTER アプリケーション・ウィンドウにある部分の 1 つです。選択したアプリケーションの変数とその内容が見やすい形式で表示されます。表示するアプリケーションの変数は、各プロジェクト・ブロックのプロパティ設定で個別に選択します。

2.10 レコーダとは何ですか?

レコーダは、ターゲット開発ボード上のソフトウェア内に作成され、変数の変化をリアルタイムで格納します。ユーザーが定義した一連の変数を、組込み側のタイマ周期割込みサービス・ルーチンによって記録できます。要求された変数のサンプルは、ターゲット・ボード上のレコーダ・バッファ内に格納された後、ボードからダウンロードされて、FreeMASTER の［Recorder（レコーダ）］部分にグラフとして表示されます。レコーダの大きな特長は、極めて高速なアクションをサンプリングする能力です。

2.11 オシロスコープとは何ですか?

FreeMASTER オシロスコープは、一般的なハードウェア・オシロスコープに似ています。グラフィカルに選択された変数がリアルタイムで表示されます。変数の値は、ボード・アプリケーションからシリアル通信ラインを介してリアルタイムで読み取られます。オシロスコープの GUI は、レコーダの GUI と似ていますが、変数のサンプリング速度は通信データ・リンクによって制限されます。

2.12 制御ページとは何ですか?

制御ページは、HTML タグ言語でエンコードされたページです。JavaScript コードを備えています。このページは、ビルトインの Internet Explorer または Chromium ブラウザ・コンポーネントを使用して FreeMASTER ウィンドウ内に表示されます。制御ページは、ローカル・コンピュータのファイルから読み込まれた一般的なウェブページとまったく同じように動作します。

FreeMASTER のユーザーは自由に制御ページを作成し、「親」である FreeMASTER とやり取りしてターゲット MCU の変数にアクセスするスクリプトを記述することができます。たとえば、グラフィカル・オブジェクト(ゲージ、スライダなど)を使用してページ内に変数の値を表示したり、プッシュ・ボタンなど、組み込みアプリケーションを制御するためのコントロールを配置したりすることが考えられます。

2.13 FreeMASTER 3.0 で制御ページは変更されましたか?

FreeMASTER 3.0 は、以前の Internet Explorer レンダリング・エンジンと ActiveX インターフェースを引き続きサポートしており、JavaScript コードから FreeMASTER の機能にアクセスすることができます。これによって以前のプロジェクトとの完全な後方互換性が確保されます。

ただし、バージョン 3.0 では、ビルトイン Chromium ブラウザ・コンポーネントが新たにサポートされています。今後は、同じコンピュータ（リモート・コンピュータでも可）で実行されているスタンドアロンの Chrome ブラウザに制御ページを表示し、最新の JSON-RPC プロトコルを使用して FreeMASTER ツールと連携させることもできます。Chromium は以前の ActiveX テクノロジをサポートしていないため、既存のページやスクリプトは、JSON-RPC 通信を使用するように修正する必要があります。一方、JSON-RPC は、Promise インターフェースを使用した非同期 JavaScript プログラミングへの対応、そして、現在の先進的なウェブ・アプリケーションで使用されている最新のコーディング・テクニックを数多くもたらします。

スタンドアロンの Chrome ブラウザは、JavaScript デバッガ環境として使用でき、新しいスタイルの FreeMASTER 制御ページの開発に最適です。

2.14 FreeMASTER Lite とは何ですか?

FreeMASTER Lite は、FreeMASTER 3.0 パッケージで初めてのバージョンがリリースされた新しいソフトウェア・サービスです。

このサービスは、ターゲット・マイクロコントローラ・ボードに物理的に接続されたコンピュータ上で起動でき、リモート・クライアントがボードにアクセスするための JSON-RPC インターフェースを備えます。このサービスが実装する JSON-RPC インターフェースは、標準の FreeMASTER ツールに備わっている JSON-RPC インターフェース（前出の質問を参照）とほぼ同じです。ただし、フルバージョンの FreeMASTER アプリケーションとは異なり、Lite サービスにはユーザー・インターフェースがありません。ローカルの設定ファイルで設定され、ユーザーのコンピュータで目に見えない状態で実行されます。

制御ページやその他のクライアント（node.js や Python で記述されたスクリプト・アプリケーションなど）は、ローカルまたはリモートのコンピュータからこのサービスに接続して、ターゲット・マイクロコントローラ・アプリケーションにアクセスできます。Lite サービスは標準的なウェブ・サーバとしても機能し、タブレットや携帯電話などのリモート・クライアント向けに制御ページやそのリソースを提供できる点が、標準の FreeMASTER アプリケーションとの主な違いです。

2.15 複数のターゲット・ボードに同時にアクセスして接続することはできますか?

実行中の FreeMASTER インスタンスが接続できるターゲット・ボードは、1 インスタンスにつき 1 つだけです。ただし、複数のインスタンスを実行して、それぞれを異なるボードに接続することはできます。

TCP または UDP ネットワーク通信を使用する場合、ターゲット・ボードは複数のセッションに対応し、複数の FreeMASTER クライアント・インスタンスからの接続を受け入れることができます。ターゲット・メモリの読み取りと書き込みは、接続されたすべてのインスタンスが行えますが、「機能ロック」というメカニズムがあります。レコーダ、パイプ通信など高度な機能の使用を保護し、最初のクライアントにのみアクセスを許可するものです。機能ロックが解除され、インスタンスによって使われなくなると、後続のインスタンスが機能ロックを使用できるようになります。

より一般的なケースとして、ホスト PC 上で実行されている 1 つのアプリケーション内で複数のターゲットのデータを収集しなければならない場合があります。FreeMASTER では、実行中の複数の FreeMASTER から、複数の JSON-RPC 接続または ActiveX 接続を使用してデータを収集し、それらすべてを単一のページに表示する制御ページを作成できます。ただしその場合、それぞれの FreeMASTER インスタンスが異なる JSON-RPC サーバ・ポートを使用する必要があります。

バージョン 3.1.3 は、JSON-RPC ポートの自動割り当てをサポートしており、実行中のインスタンスごとに異なるサーバ・ポートが使用されます。FreeMASTER 内の HTML アプリケーションによって実行されているスクリプトで、接続パラメータを検出できます。「FreeMASTER JSON-RPC インターフェース」および「HTML ページに埋め込まれた JavaScript と JSON-RPC」セクションのサンプル・コードを参照してください。

第 3 章
インストール

3.1 システムの要件

FreeMASTER アプリケーションの要件は、現在販売されている Windows OS ベースのホスト PC であれば、ほとんどの場合、容易に満たすことができます。最新バージョンの Windows OS（7、8、10）でテスト済みです。

オペレーティング・システム： Microsoft Windows 7 以降

必要なソフトウェア： Internet Explorer 10 以降

ハードディスク空間： 400 MB（インストール中は別途 500 MB）

その他のハードウェア要件： マウス、シリアル RS-232 または USB ポート（ターゲット・ボードとの通信用）、ネットワーク・アクセス（リモート操作用）

3.2 ターゲット・アプリケーションへの接続

FreeMASTER をターゲット・ボード・アプリケーションに接続できるようにするには、MCUXpresso SDK（www.nxp.jp/mcuxpresso/sdk から入手可能）または www.nxp.jp/freemaster のホームページから入手したドライバを使用します。

初めて FreeMASTER を使用する場合、いずれかのサンプル・アプリケーションを使用するのが最も簡単です。

FreeMASTER 通信プロトコルをサポートするマイクロコントローラのドライバを使用すると、このプロトコルの新しいバージョン（v4）と以前のバージョンが参照されていることがわかります。v4 プロトコル・バージョンは 2019 年に定義されたもので、FreeMASTER バージョン 2.5 以降でサポートされます。

FreeMASTER デスクトップ・アプリケーション・バージョン 2.5 および今後リリースされるすべてのバージョンで v4 プロトコルがサポートされ、以前のプロトコル・バージョンとの下位互換性も維持される予定です。

3.3 FreeMASTER をインストールする方法

FreeMASTER アプリケーションは、スタンドアロン、単一ファイル、自己解凍型の実行ファイルとして配布されます。FreeMASTER のホームページからインストーラ・ファイルをダウンロードして実行し、画面の手順に従って進んでください。

FreeMASTER デスクトップ・アプリケーションは、Windows OS でのみ利用できます。Linux OS のインストーラも提供されていますが、含まれているのは FreeMASTER Lite サービスのみです。

FreeMASTER Lite をインストールするには、ウェブページ（www.nxp.jp）から無償で入手できるアクティベーション・コードが必要です。インストール・プロセスのコンポーネント選択ウィンドウに、FreeMASTER 製品ライセンス・ページへの直接リンクが表示されます。また、ライセンス・キーも、メインの「ソフトウェアのライセンスとサポート」ページの「FreeMASTER Lite 製品情報」セクションで入手できます。どちらの場合も、NXP アカウントにログインする必要があります。

3.4 FreeMASTER の実行

インストールの完了後、Windows OS の［スタート］メニューにあるアイコンを使用して FreeMASTER を実行します。FreeMASTER がデフォルト・モードで実行され、すべての機能が有効になり、JSON-RPC と ActiveX サーバの両方が起動して、受信接続を待機する状態になります（詳細については、「HTML とスクリプティング」を参照）。FreeMASTER のメインの実行ファイル（pcmaster.exe）をコマンドラインから起動することや、Windows OS ショートカットを手動で作成することもできます。そのため、コマンドライン・オプション（複数可）や、デフォルトで開かれるプロジェクト・ファイル名も指定できます。指定できるコマンドライン・オプションをすべて表示するには、「/help」オプションを使用します。

第 4 章
FreeMASTER の使用法

4.1 アプリケーション・ウィンドウの説明

アプリケーションを起動すると、メイン・ウィンドウが画面に表示されます。プロジェクトがロードされていない場合、ようこそページがウィンドウのメイン部分に表示されます。以下の図は、最初に表示されるメイン・ウィンドウの外観を示しています。

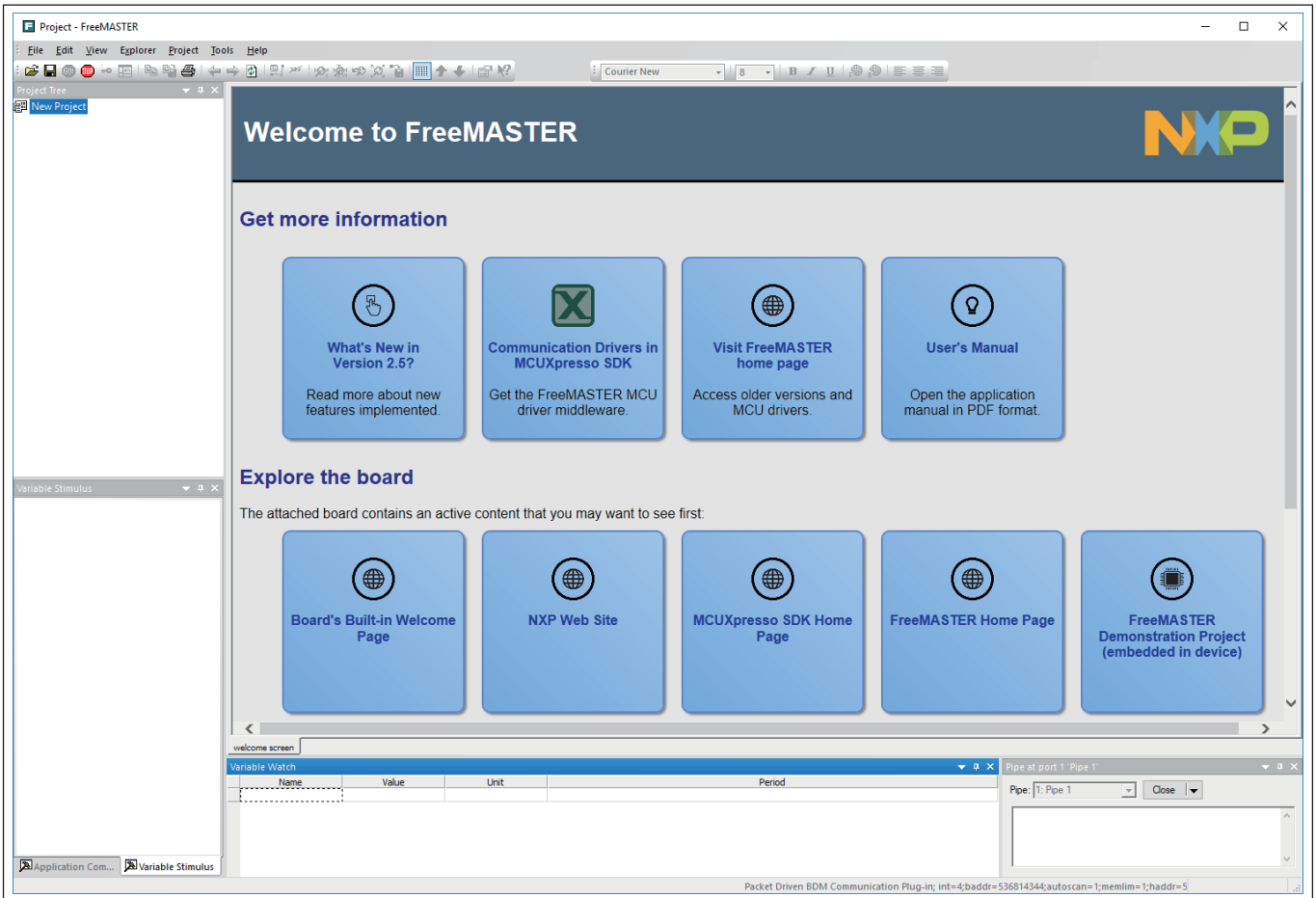

図 1. 最初に表示されるアプリケーション・ウィンドウ

ようこそページには、ドキュメントやアプリケーション・ヘルプへのリンクが表示されます。その他、標準メニュー・コマンド（プロジェクトを開くためのコマンドなど）に対応するリンクもいくつか表示されます。

この章では、MCUXpresso SDK のサンプル・アプリケーションに同梱されている単純なデモ・アプリケーションを例として使用しながら、FreeMASTER の使用法を具体的に紹介します。

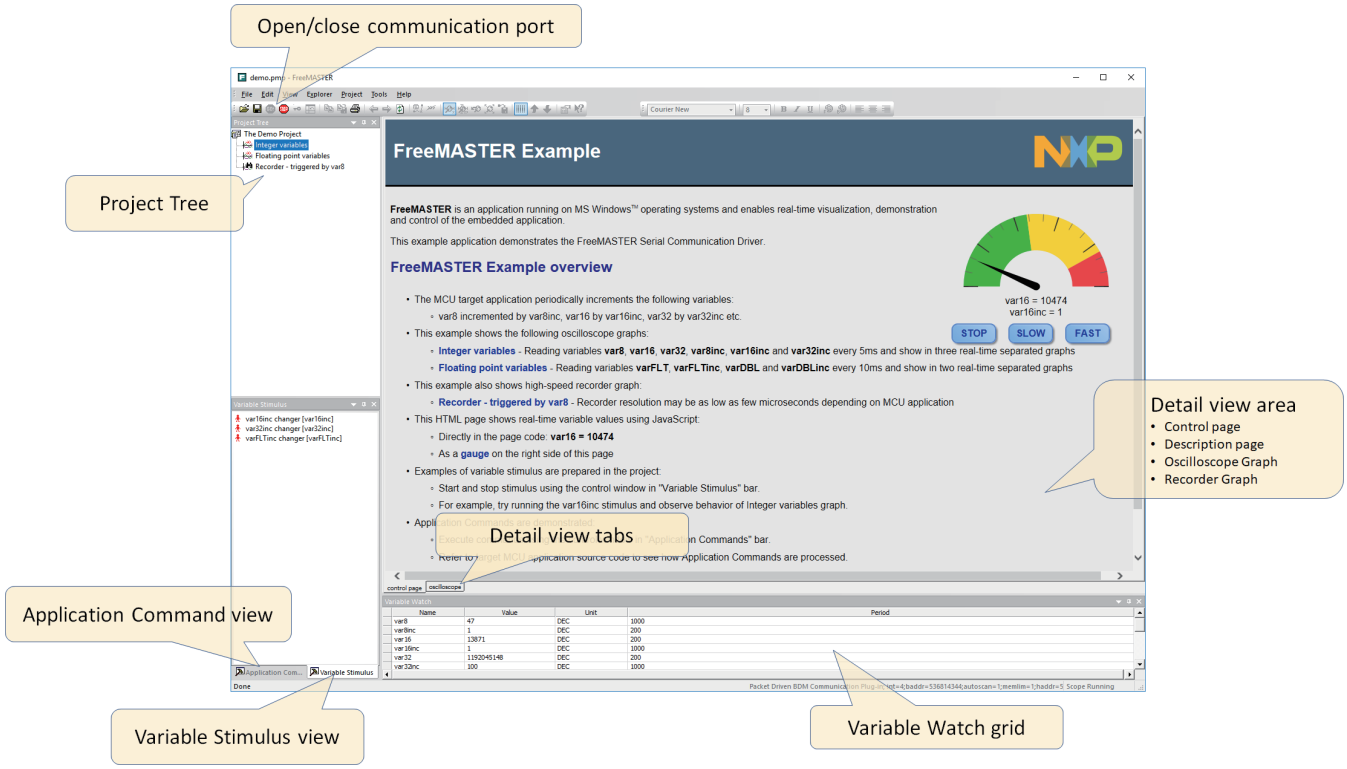

図 2. アプリケーション・ウィンドウ

［Project Tree（プロジェクト・ツリー）］部分には、監視または制御の対象となるアプリケーションの論理ツリー構造が表示されます。プロジェクトのサブブロック、オシロスコープ、レコーダ、パイプの定義を、論理構造内のプロジェクト･ブロックに追加することで、プロジェクト･ツリーを形成できます。この部分では、定義されているプロジェクト・ツリー要素をポイント・アンド・クリックで選択できます。

［Detail View（詳細ビュー）］部分に表示される内容は、プロジェクト・ツリー要素で選択された項目に応じて動的に変化します。ツリーで選択された項目の種類に応じて、この部分内のいくつかのタブにその項目に関連付けられた追加情報のサブ・ページが表示されます。

- ［Control page（制御ページ）］:ターゲット･システムを制御する目的で作成された HTML ページ。制御ページが定義されている場合、プロジェクト･ツリーで選択されている項目に関係なくすぐに使用でき、ユーザーはいつでもボードを制御することができます。
- ［Algorithm block description（アルゴリズム・ブロックの説明）］: 選択されたプロジェクト・ツリー項目のプロパティで URL が定義されている、HTML ページなどのドキュメント。プロジェクト・ツリーで選択されている項目が変わると、このビューの内容も変化します。
- ［Current item help（現在の項目のヘルプ）］:［Oscilloscope properties（オシロスコープのプロパティ）］または［Recorder properties（レコーダのプロパティ）］で URL が定義されている HTML ドキュメント。
- ［Oscilloscope（オシロスコープ）］:［Oscilloscope properties（オシロスコープのプロパティ）］に定義されているアプリケーションの変数を表すリアルタイム・グラフ。
- ［Recorder（レコーダ）］:［Recorder properties（レコーダのプロパティ）］の定義に従って、記録されたアプリケーションの変数を表示するグラフ。
- ［Pipe（パイプ）］: FreeMASTER のパイプ・データをテキストまたはグラフを用いて表示するビュー。

［Variable Watch（変数ウォッチ）］部分には、監視対象に設定されている変数が一覧表示されます。この部分には、変数のイミディエイト値が表示されるほか、ユーザーが変数を変更することができます（変数の定義で変更が有効になっている場合）。

アプリケーションに関連したすべての情報は、*.pmp 拡張子または新しい XML 形式の *.pmpx 拡張子で 1 つのプロジェクト・ファイルに格納されます。このプロジェクト･ファイルには、通信の設定やオプション、プロジェクト･ツリー、HTMLページ、リアルタイム･チャートの定義、変数ウォッチのインターフェース設定、変数、コマンド、スティミュレータなど、使用されるオブジェクトがすべて含まれています。

FreeMASTER アプリケーションのあらゆるオブジェクトや設定を復元するために、プロジェクト・ファイルは随時開きます。また、プロジェクトで使用されているすべてのビュー部分のレイアウトおよびウィンドウ･レイアウトは、新しい *.pmpx 形式で保存することもできます。これを有効にするには、［View（表示）］メニューの［Store Layout in Project File（プロジェクト・ファイルにレイアウトを保存）］項目を選択します。

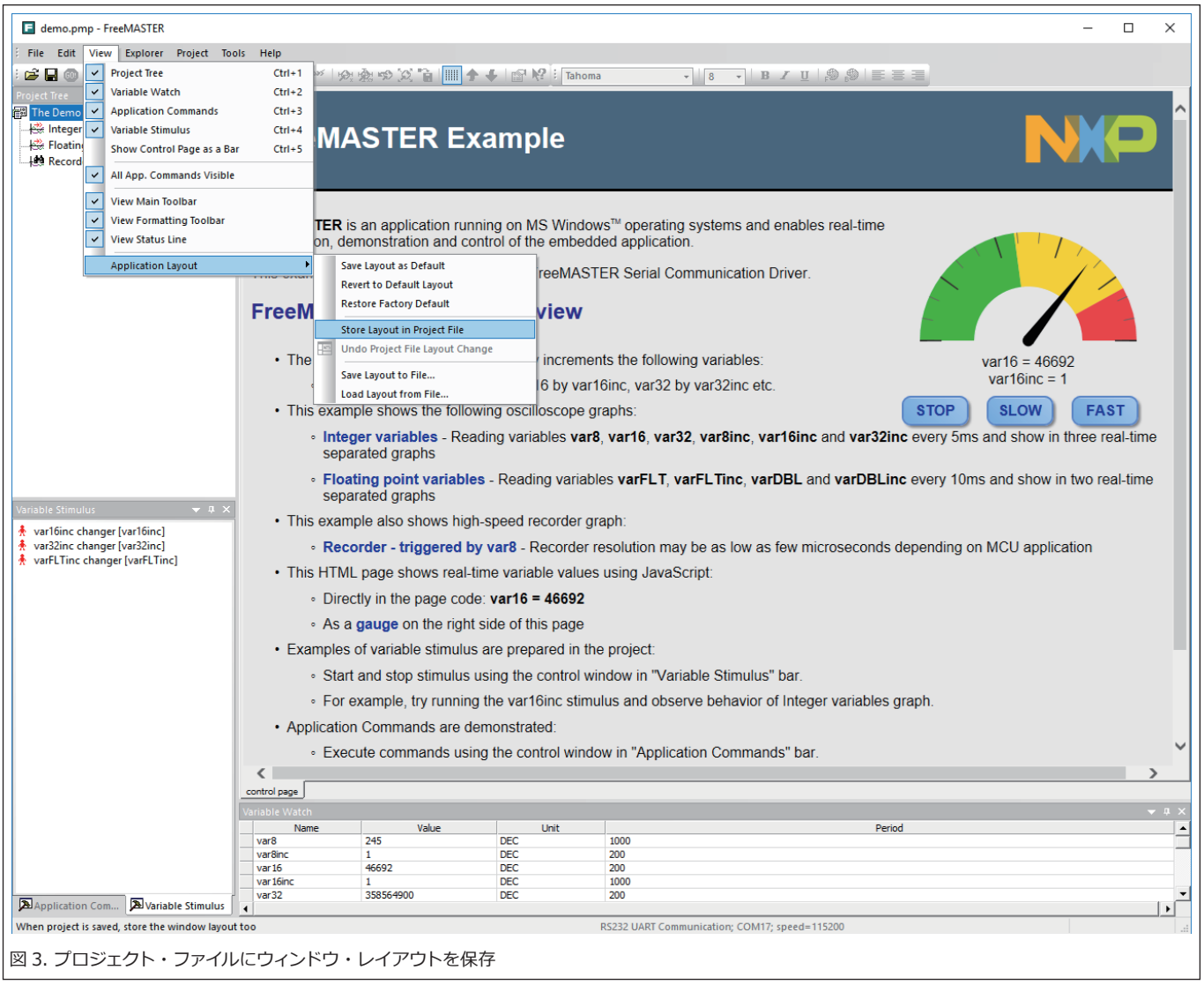

図 3. プロジェクト・ファイルにウィンドウ・レイアウトを保存

4.1.1 プロジェクト・ツリー

新しいプロジェクトを作成すると、[Project Tree(プロジェクト・ツリー)]ウィンドウには「New Project(新しいプロジェクト)」というルート・プロジェクト・ブロックが 1 つしかない空の構造が表示されます。ユーザーは、このブロックのプロパティに変更を加えたり、サブブロック、オシロスコープ、レコーダ、パイプなどを構造に追加したりすることができます。

変更の対象となるツリー項目を右クリックして表示されるコンテキスト・メニューから、プロパティに変更を加えたり、新しいプロジェクト・ツリー項目を追加したりできます。

4.1.1.1 プロジェクト・ブロックとサブブロック

プロジェクト・ブロックには通常、FreeMASTER で実際に動かすアプリケーションまたはアルゴリズムに不可欠なコンポーネントが含まれます。アルゴリズムを複数のブロックに分割する場合は、サブブロックを追加できます。それぞれのブロックには個別の [algorithm block description (アルゴリズム・ブロックの説明)] ページがあり、監視対象の変数とコマンドが設定されています。そのすべてを、以下の図に示した [Project Block Properties (プロジェクト・ブロックのプロパティ)] ダイアログで定義できます。

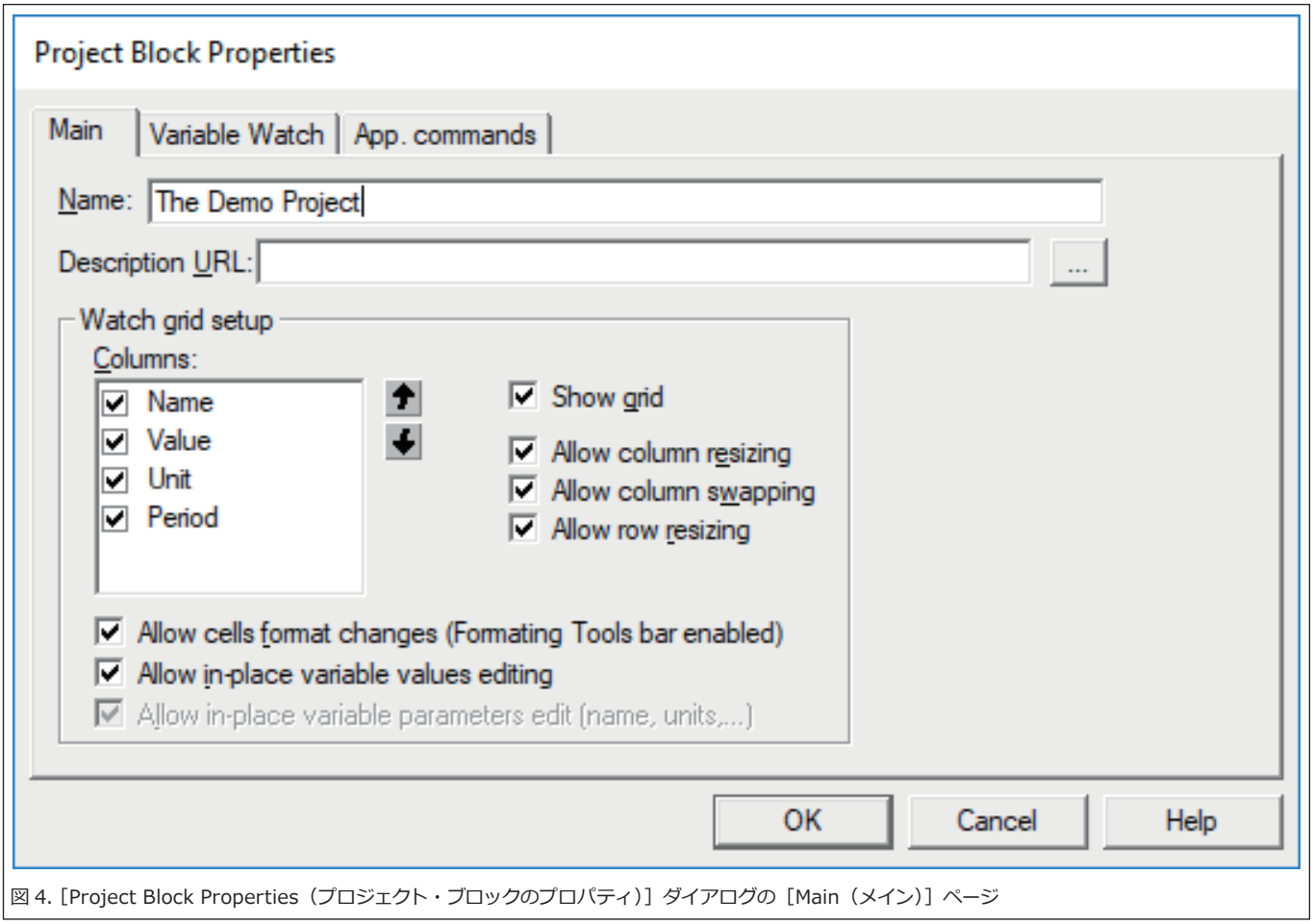

図 4. [Project Block Properties（プロジェクト・ブロックのプロパティ）] ダイアログの［Main（メイン）］ページ

［Main（メイン）］ページには、以下のユーザー設定項目が表示されます。

- ［Name（名前）］: プロジェクト・ツリーに表示されるプロジェクト・ブロックの名前。
- ［Description URL（説明 URL）］: 説明用 URL、または［Detail View（詳細ビュー）］部分の［Algorithm Block Description（アルゴリズム・ブロックの説明）］タブに表示される *.htm ファイルまたは *.html ファイルへのパスを選択します。例として使用されるデモ・アプリケーションでは説明ページが空のままであるため、［Algorithm Block Description（アルゴリズム・ブロックの説明）］タブは表示されません。詳細については、「詳細ビュー」を参照してください。
- ［Watch-grid setup（ウォッチ・グリッドの設定）］: 以降の各オプション・ボックスのオン / オフによって、ウォッチ・グリッドに表示する列（［Name（名前）］、［Value（値）］、［Unit（単位）］、［Period（期間）］）の選択、上下矢印ボタンによる列順の変更、グリッド動作オプション（列のサイズ変更、入れ替え、行のサイズ変更）の選択、ツールバーを使ったグリッド・セル形式の変更の許可（「書式設定バー」を参照）、インプレース変数値の編集を行うことができます。

下図に示した［Watch（ウォッチ）］ページで、FreeMASTER プロジェクトの変数の中からこのプロジェクト・ブロックのコンテキストで監視する変数を選択します。

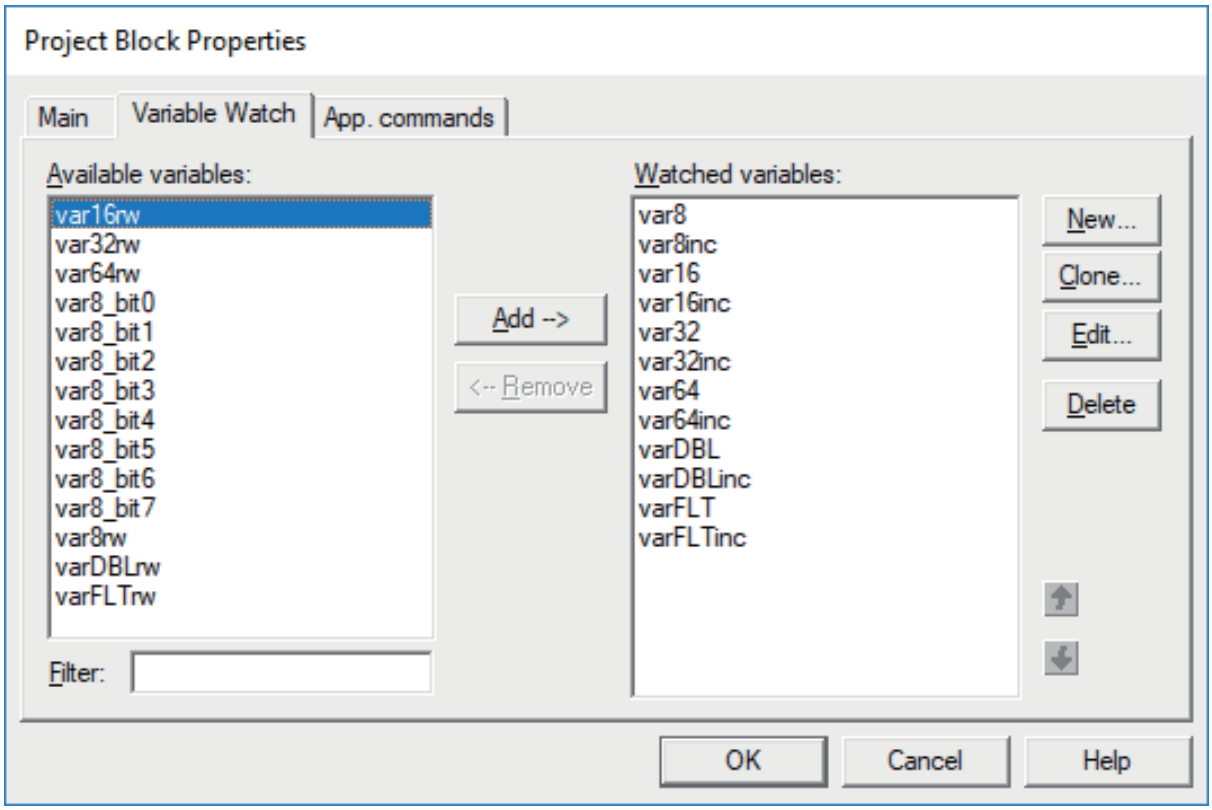

図 5. [Project Block Properties（プロジェクト・ブロックのプロパティ）] ウィンドウの [Variable Watch（変数ウォッチ）] ページ

[Watched variables（監視対象の変数）] リストにある変数が、ウォッチ・グリッドでの監視対象として現在選択されているプロジェクト変数です。現在ツリーで選択されているブロック項目に関して、監視対象としてまだ選択されていないプロジェクト変数は、[Available variables（選択可能な変数）] リストに表示されます。以下のボタンを使用します。

- [Add/Remove（追加 / 削除）]:[Watched variables（監視対象の変数）] ウィンドウに変数を移動したり、ウィンドウから変数を移動したりします。
- [New（新規作成）]：新しい変数を作成します（「変数」を参照）。
- [Clone（複製）]：選択された変数のコピーを使用して新しい変数を作成します。
- [Edit（編集）]：選択された変数のプロパティに変更を加えます。
- [Delete（削除）]：選択された変数をプロジェクトから削除します。
- 上下矢印：ウォッチ・グリッドにおける監視対象の変数の表示順を設定します。

FreeMASTER は、変数の読み取り / 書き込みによって、また、いわゆる「アプリケーション・コマンド」（「コマンド」を参照）の送信によってボード・アプリケーションと通信を行います。ウォッチ・グリッドに表示される変数は [Project Tree（プロジェクト・ツリー）] 部分で選択されたブロックによって異なるため、利用できるアプリケーション・コマンドも、選択されたブロックによって異なる場合があります。[App. commands（アプリケーション・コマンド）] ページ（下図参照）では、このプロジェクト・ブロックのコンテキストで [Commands（コマンド）] 部分から利用できるコマンドを設定したり、アプリケーション・コマンドを管理したりできます。

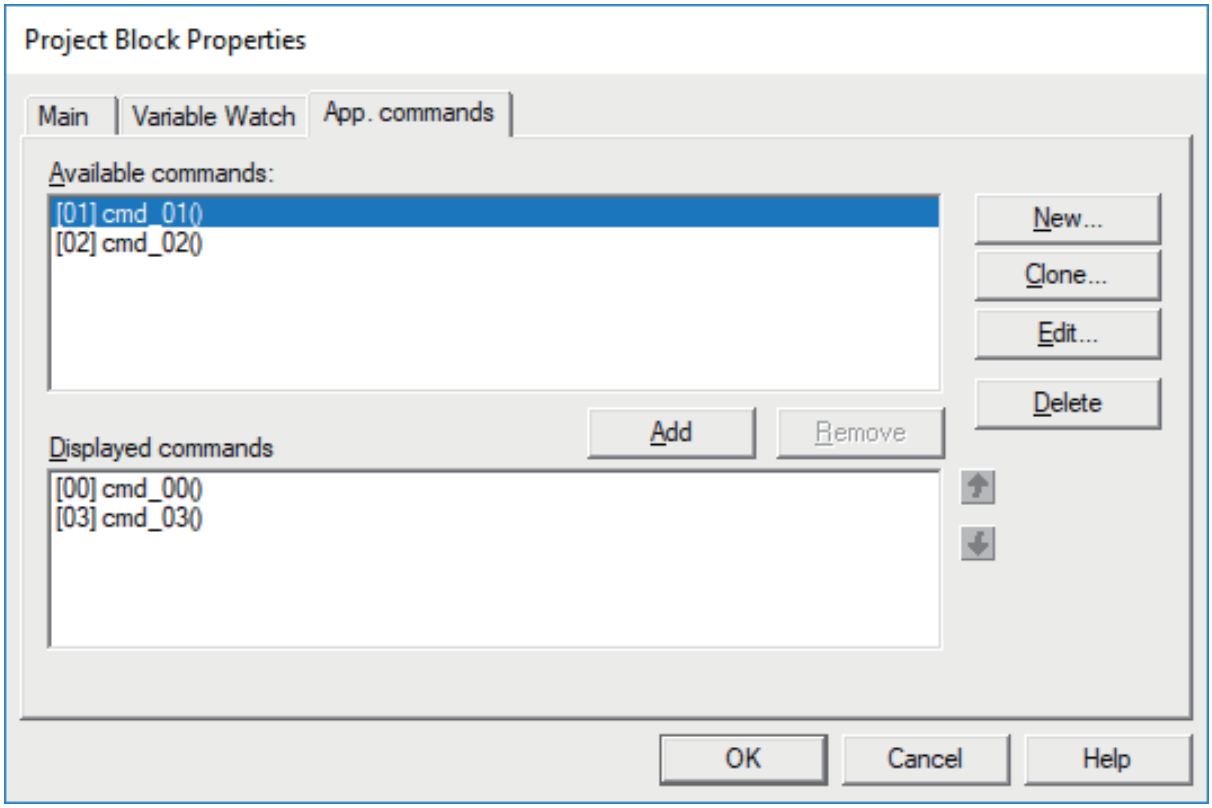

図 6. [Project Block Properties（プロジェクト・ブロックのプロパティ）] ウィンドウの [App. commands（アプリケーション・コマンド）] ページ

コマンドは、[Available commands（利用可能なコマンド）] ウィンドウに一覧表示されます。[Add（追加）] ボタンを使用して [Displayed commands（表示されるコマンド）] ウィンドウにコマンドを移動すると、そのコマンドが [Commands（コマンド）] 部分から利用できるようになります。逆の操作を行うには、[Remove（削除）] ボタンを使用します。以下のボタンを使用します。

- [New（新規作成）]：新しいコマンドを作成します（「コマンド」を参照）。
- [Clone（複製）]：選択されたコマンドのコピーを使用して新しいコマンドを作成します。
- [Edit（編集）]：選択されたコマンドのプロパティに変更を加えます。
- [Delete（削除）]：選択されたコマンドを削除します。
- 上下矢印：[Fast Access（高速アクセス）] 部分におけるコマンドの表示順を設定します。

メモ：多くの場合、[Commands（コマンド）] 部分には、プロジェクト・ツリーで選択されている項目に関係なくすべてのアプリケーション・コマンドを表示した方が便利です。[View（表示）] メニューには [All App.Commands Visible（すべてのアプリケーション・コマンドを表示する）] という項目があります。これを選択すると、すべてのコマンドが常時、無条件に表示されます。

4.1.1.2 オシロスコープ

[Detail View（詳細ビュー）] 部分に表示されるリアルタイムのオシロスコープ・チャートは、プロジェクト・ツリー構造の [Oscilloscope（オシロスコープ）] 項目によって定義されます。プロパティ・ウィンドウ（下図）で、オシロスコープ・チャートの外観や特性を設定できます。

[Main（メイン）] ページには、以下のユーザー設定項目が表示されます。

- [Name（名前）]：プロジェクト・ツリーに表示されるオシロスコープ項目の名前。
- [Description URL（説明 URL）]：[Detail View（詳細ビュー）] 部分の [current item help（現在の項目のヘルプ）] タブに表示されるファイルのローカル・パスまたはドキュメントの URL を指定します。このドキュメントで、チャートの変数や設定をユーザーに説明します。
- [Oscilloscope properties（オシロスコープのプロパティ）]：すべてのオシロスコープ変数に使用される共通のプロパティ。
- [Period（周期）]：オシロスコープのサンプリング周期。
- [Buffer（バッファ）]：チャートのローカル・バッファに保持されるサンプルの数。この値を大きくすることで、最近のデータ・ポイントをより多くさかのぼってチャート内で表示できるようになります。
- [Legend location（凡例の場所）]：チャートの凡例の表示と非表示および場所を設定します。

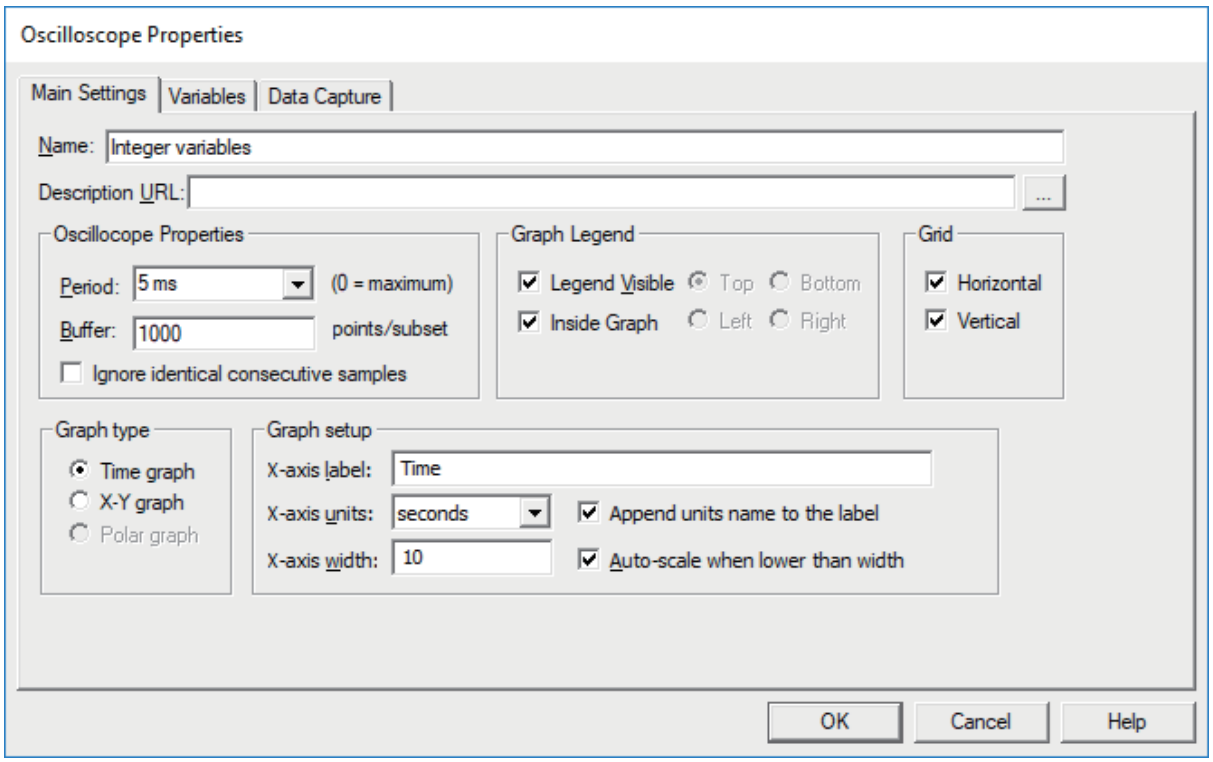

図 7. [Oscilloscope Properties（オシロスコープのプロパティ）] ウィンドウのメイン・ページ

- [Grid（グリッド）]：チャートに表示する水平 / 垂直グリッド線を選択します。
- [Graph type（グラフの種類）]：オシロスコープの動作モードを選択します。
- [Time graph（時間グラフ）]：チャートに変数（値と時間の対比）が表示されます。
- [X-Y graph（X-Y グラフ）]：変数間の依存関係（値と値の対比）がチャートに表示されます。
- [Graph setup（グラフの設定）]（時間グラフ用）：
  - [X-axis label（X 軸ラベル）]：X 軸に表示される名前を指定します。
  - [X-axis units（X 軸の単位）]：軸の単位を選択します。
  - [X-axis width（X 軸の幅）]：X 軸の範囲を指定します。
  - [Auto-scale X-axis until width is reached（軸幅に到達するまで X 軸を自動スケーリング）]：オシロスコープの動作開始後、サブセットの長さが X 軸の幅よりも短い場合に軸幅をスケーリングします。
- [Graph setup（グラフの設定）]（X-Y グラフ用）：
  - [X-variable（X 変数）]：X 軸の値に使用される変数を選択します。
  - [X-axis min（X 軸最小）]：X 軸の下限値を設定します。
  - [X-axis max（X 軸最大）]：X 軸の上限値を設定します。

[Variables(変数)]ページは、オシロスコープ・チャートに表示される変数の割り当てに使用します。表示される変数の数は、ターゲット・マイクロコントローラ・アプリケーションにおける FreeMASTER ドライバの設定によって制限されます。[Add Variable（変数の追加）] ボタンを使用してチャートに新しいスロットを追加し、ドロップダウン・リストを使用してスロットに変数を割り当てます。

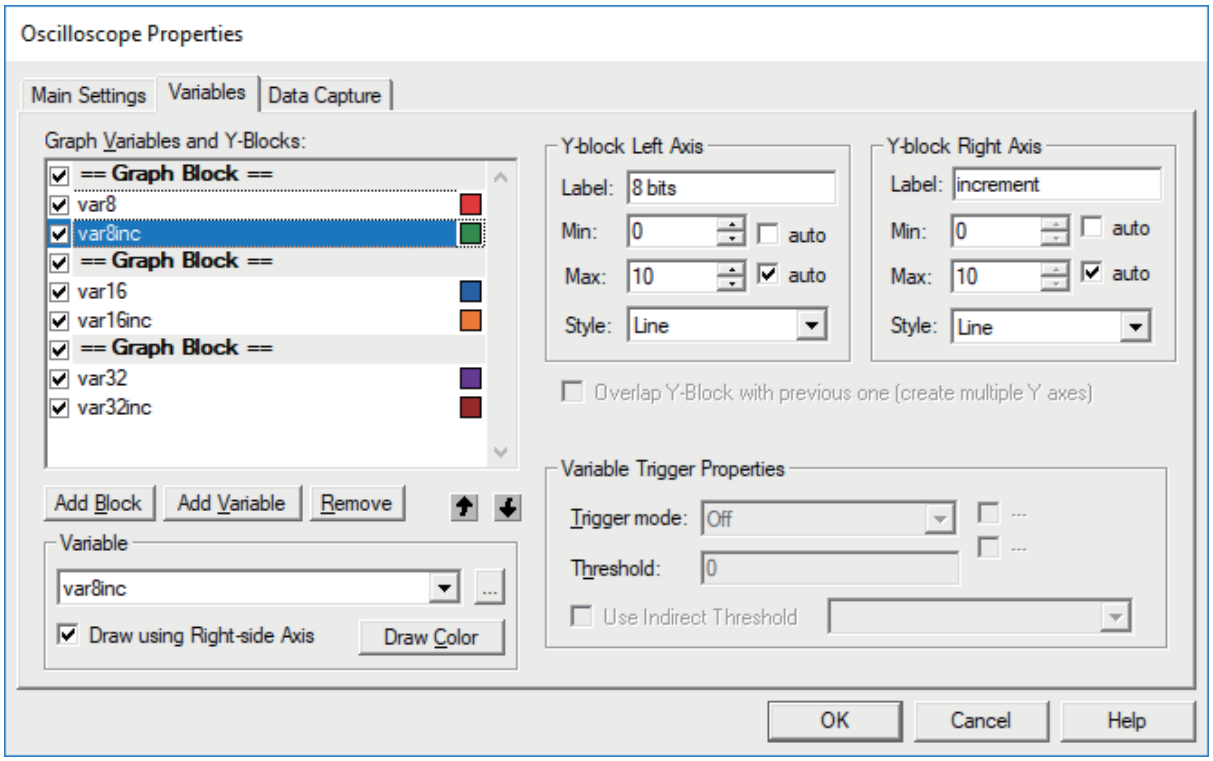

図 8. [Oscilloscope Properties（オシロスコープのプロパティ）] ウィンドウの [Variables（変数）] ページ

Y ブロックは、1 つの左 Y 軸（場合によってはさらに 1 つの右 Y 軸）で表されるグラフ要素です。グラフには Y ブロックを個別に描画することも、重ねて描画することもできます。[Add Block（ブロックの追加）] ボタンを使用して新しい Y ブロック・セパレータを作成し、それをマウスでドラッグして変数リスト内にドロップします。グラフ内の共通の Y ブロックに置かれる一連の変数が、セパレータ項目によってグループ化されます。

- [Y-block Left Axis（Y ブロック左軸）] 領域では、軸の名前を設定するとともに、軸の最小値と最大値を指定して軸の範囲を設定します。[auto（自動）] チェック・ボックスをオンにすることで、最小と最大を自動的に追跡することもできます。[Style（スタイル）] では、データ・サブセットの描画スタイルをドロップダウン・リストのボックスから選択します。
- Y ブロックの右軸に変数を割り当てるには、選択する変数の [Draw using Right-side Axis（右側の軸を使用して描画する）] オプションをオンにします。このオプションをオンにした状態で、[Y-block Right Axis（Y ブロックの右軸）] 領域で右軸のプロパティを編集します。

この設定で表示されるオシロスコープ・チャートを示したのが以下の図です。右軸に、var8inc、var16inc、var32inc の各変数が割り当てられていることに注目してください。

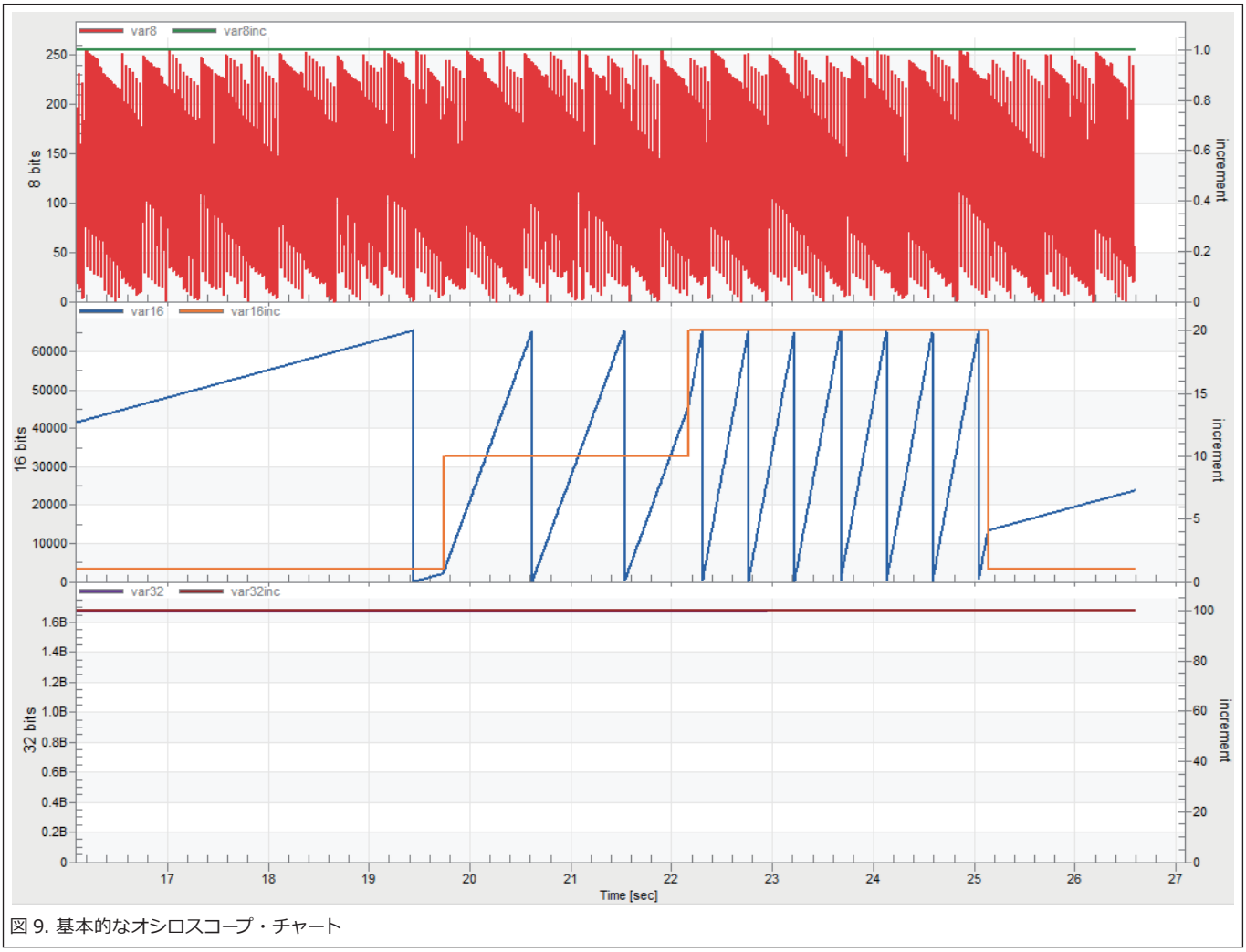

図 9. 基本的なオシロスコープ・チャート

後続の Y ブロックをグラフ内で重ね合わせて、複数の Y 軸を表示することができます。次のように、[Overlap Y-block with previous one（Y ブロックを先行する Y ブロックに重ね合わせる）] オプションを使用します。

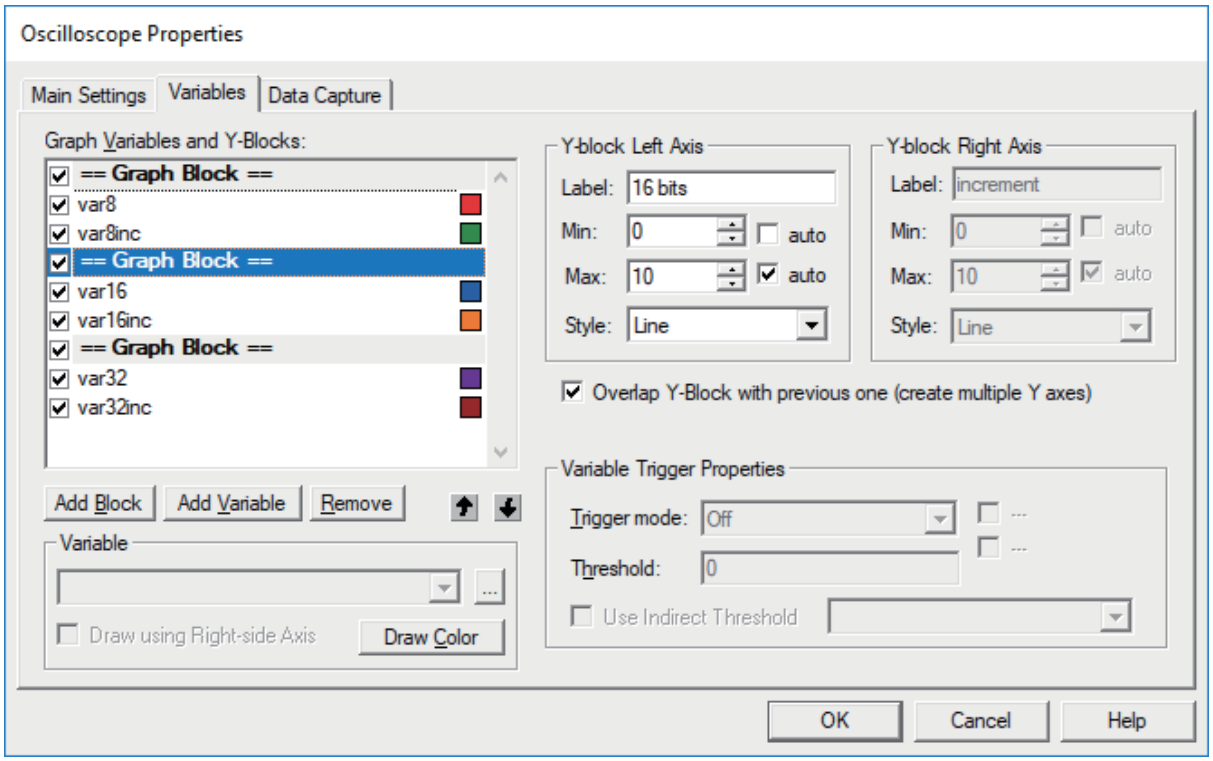

図 10. 2 つの Y ブロックの結合

この設定で表示されるチャートを示したのが以下の図です。

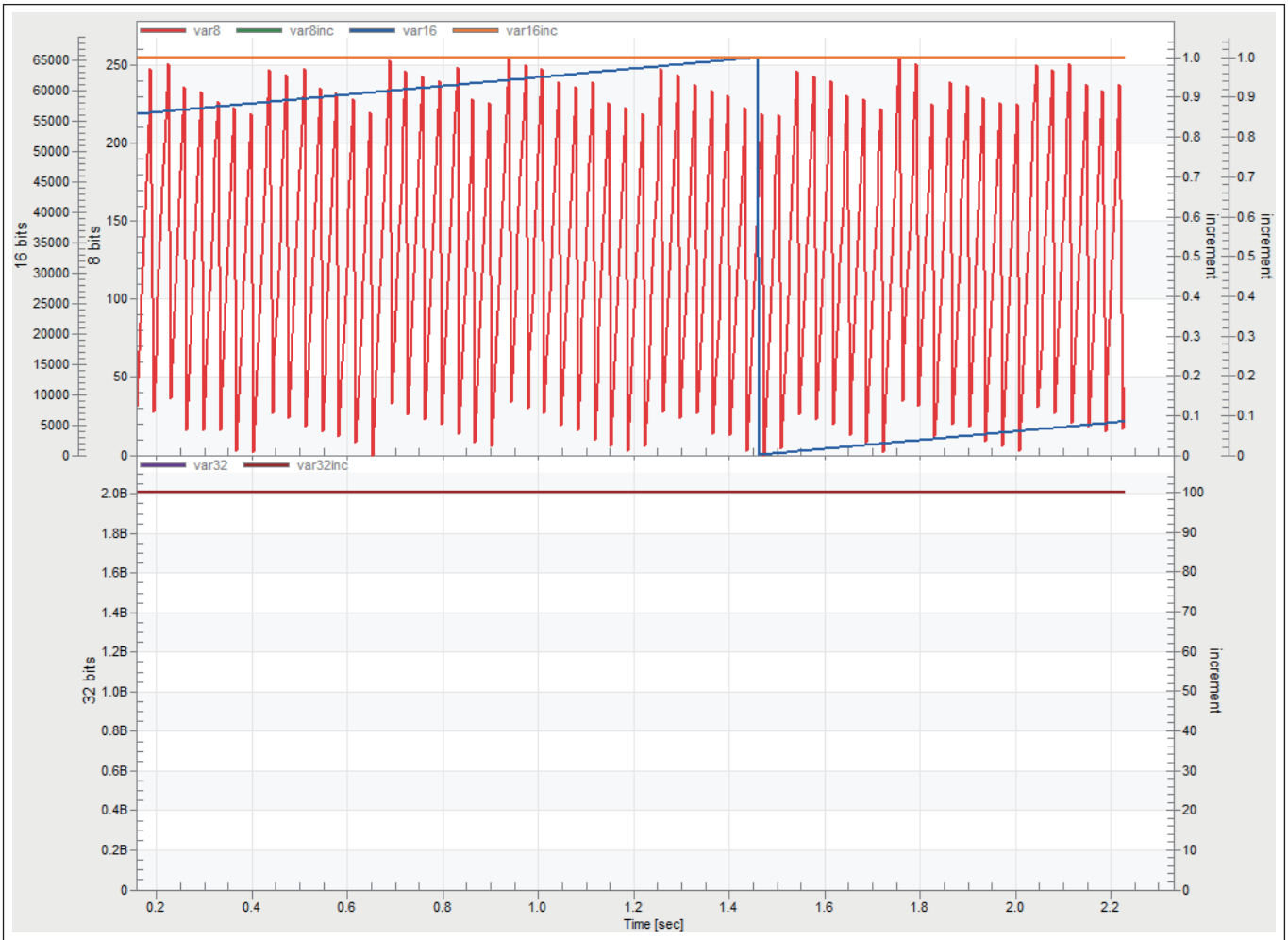

図 11. グラフ内で 2 つの Y ブロックを結合

4.1.1.3 レコーダ

［Detail View（詳細ビュー）］部分に表示されるリアルタイムのレコーダ・チャートは、プロジェクト・ツリー構造の［Recorder（レコーダ）］項目によって定義されます。オシロスコープが定期的に変数値を読み取ってリアルタイムでプロットするのに対し、レコーダはターゲット・ボード上で動作し、アプリケーションの変数を読み取って、それらをバースト・モードでFreeMASTERツールに送信します。レコーダの変数は継続的にサンプリングされ、ターゲット・ボード・アプリケーション内の循環バッファに格納されます。ターゲットでトリガ・イベントが検出されると、指定のレコーダ・サンプル数に達するまでデータ・サンプルがカウントされます。指定されたサンプル数に達した時点で、データが FreeMASTER アプリケーションに送信されます。このメカニズムの方が、オシロスコープよりもはるかに短いサンプリング周期を使用でき、極めて高速なアクションのサンプリングとプロットが可能となります。

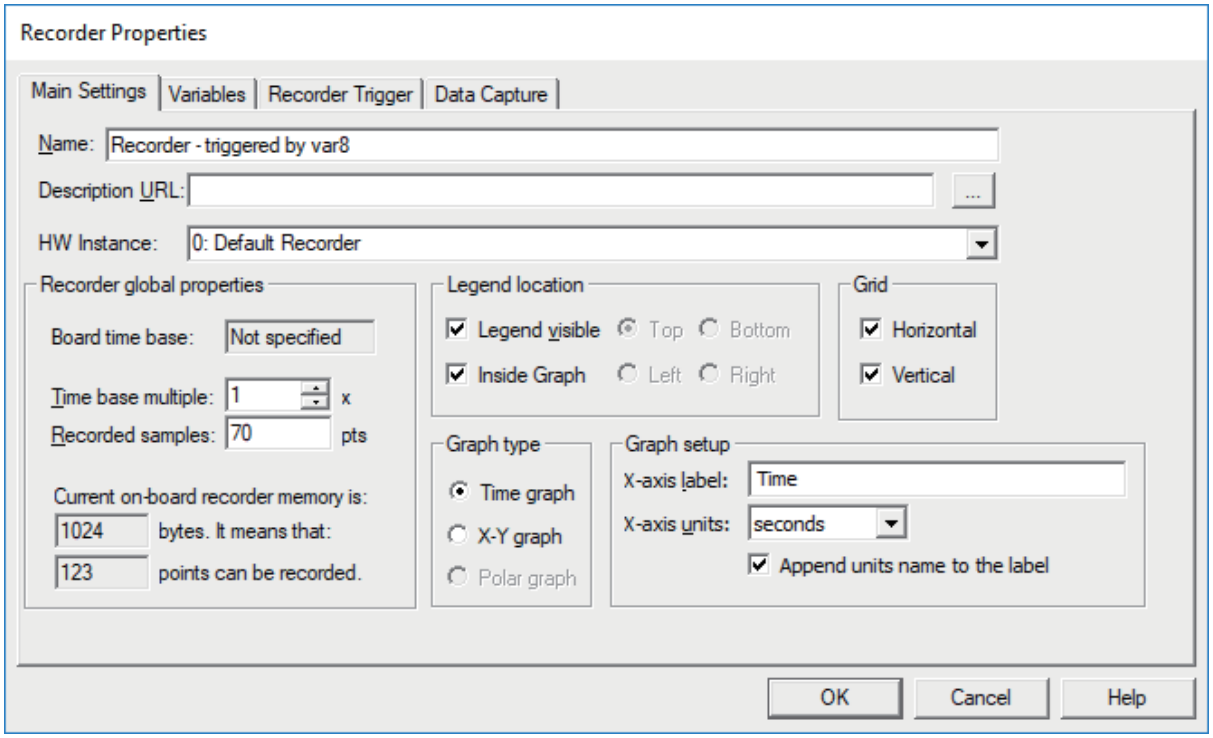

図 12. [Recorder Properties（レコーダのプロパティ）] ウィンドウのメイン・ページ

[Main（メイン）] ページには、以下のユーザー設定項目が表示されます。

- [Name（名前）]：プロジェクト・ツリーに表示されるレコーダ項目の名前。
- [Description URL（説明 URL）]：[Detail View（詳細ビュー）]部分の[current item help（現在の項目のヘルプ）]タブに表示されるファイルのローカル・パスまたはドキュメントの URL を指定します。
- [Hardware Instance（ハードウェア・インスタンス）]：ターゲット・マイクロコントローラ・アプリケーションに定義されているレコーダ・インスタンスの識別子。アプリケーションには複数のレコーダを定義し、その各レコーダを異なる機能でサンプリングすることができます。また、それぞれのレコーダ・インスタンスには名前を割り当てることもできます（Periodic-Timer Recorder、PWM-Interrupt Recorder、ADC Sampler Recorder など）。異なるハードウェア・インスタンスの複数のレコーダを FreeMASTER で同時に実行することができます。
- [Recorder properties（レコーダのプロパティ）]：すべてのレコーダ変数に使用される共通のプロパティ。
  - [Board timebase（ボードのタイム・ベース）]：ボード・アプリケーションによって事前に設定されたサンプリング周期。ターゲット・マイクロコントローラ・アプリケーションの FMSTR_REC_TIMEBASE 設定オプションを使用して記録周期を定義します。記録ベースが変化する場合や記録ベースを指定できない場合、このオプションは 0 のままにしてください。
  - [timebase multiple（タイム・ベースの倍数）]：レコーダの動作に使用されるサンプリング周期を延長する際のタイム・ベースの整数の倍数を設定します。
  - [Recorded samples（記録サンプル数）]：記録されるサブセット 1 つあたりの、バッファに格納されるサンプルの数。
  - [onboard recorder memory（オンボード・レコーダ・メモリ）]：レコーダの動作用に割り当てられるオンボード・アプリケーション・メモリ量が表示されます。メモリ・サイズ、記録される変数の形式、記録される変数の数から、レコーダのメモリに格納されるデータ・ポイントの最大数が計算されて表示されます。[Recorded samples（記録サンプル数）] に設定する値は、この結果よりも小さくなければなりません。
- [Legend location（凡例の場所）]：チャートの凡例の表示と非表示および場所を設定します。
- [Grid（グリッド）]：チャートに表示する水平 / 垂直グリッド線を選択します。
- [Graph type（グラフの種類）]：レコーダの動作モードを選択します。
  - [Time graph（時間グラフ）]：チャートに変数（値と時間の対比）が表示されます。
  - [X-Y graph（X-Y グラフ）]：変数間の依存関係（値と値の対比）がチャートに表示されます。

- [Graph setup（グラフの設定）]（時間グラフ用）：
  — [X-axis label（X 軸ラベル）]：X 軸に表示される名前を指定します。
  — [X-axis units（X 軸の単位）]：軸の単位を選択します。
- [Graph setup（グラフの設定）]（X-Y グラフ用）：
  — [X-variable（X 変数）]：X 軸の値に使用される変数を選択します。
  — [X-axis min（X 軸最小）]：X 軸の下限値を設定します。
  — [X-axis max（X 軸最大）]：X 軸の上限値を設定します。

[Recorder properties（レコーダのプロパティ）] ダイアログの [Variables（変数）] ページ（下図）の外観は、[Oscilloscope properties（オシロスコープのプロパティ）] ダイアログの対応するページとまったく同じです。レコーダ・チャートに変数を追加する方法およびチャート自体の設定方法について詳しくは、「オシロスコープ」を参照してください。

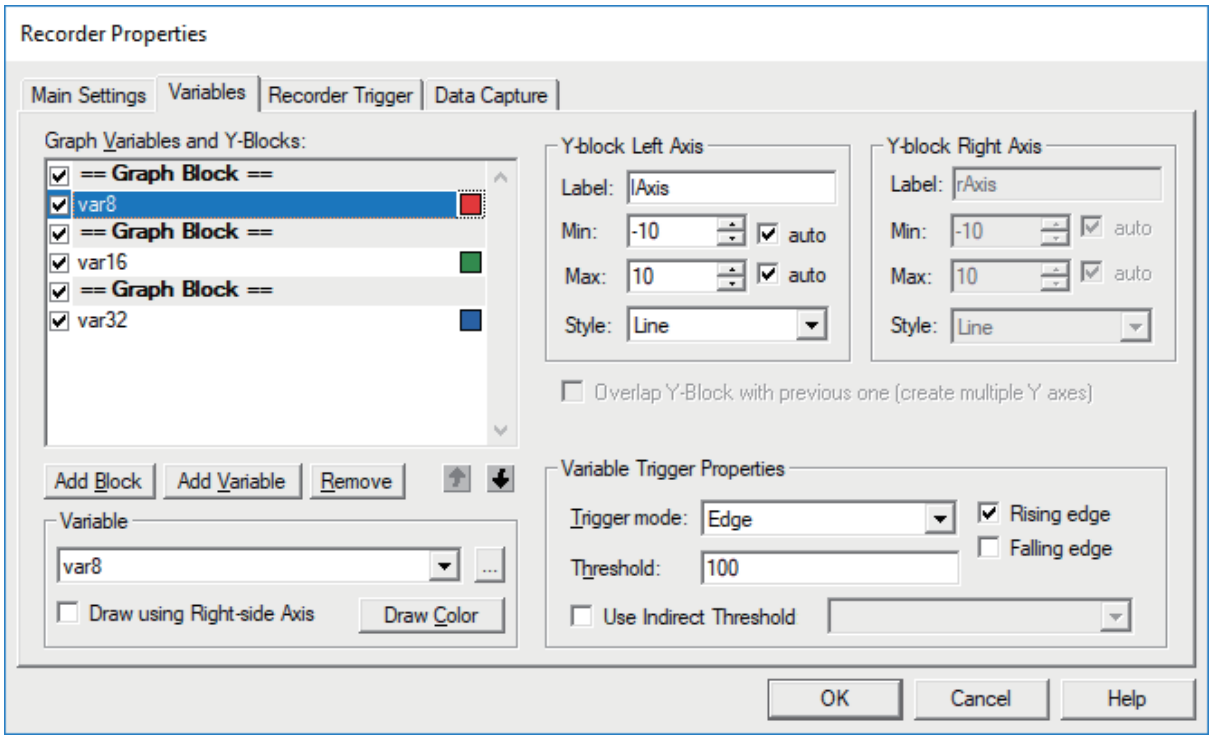

図 13. [Recorder Properties（レコーダのプロパティ）] ウィンドウの [Variables（変数）] ページ

レコーダの設定には、変数リストの一般的な設定に加え、[Variable Trigger Properties（変数トリガ・プロパティ）] のオプションがあります。リスト内の変数（複数可）は、いわゆる「トリガ」変数として設定することができます。そのようなトリガ変数の値が定義された閾値を超えたとき、所定の時間の経過後にレコーダの記録が停止されます。この「停止」期間中、トリガ・イベント後の残りのデータ・ポイントが記録されます。また、トリガ・イベントの前にサンプリングされたデータ・ポイント（「プリトリガ」サンプル）も所定の数だけ保持されます。

トリガの設定は、[Recorder Trigger（レコーダ・トリガ）] タブで行います。

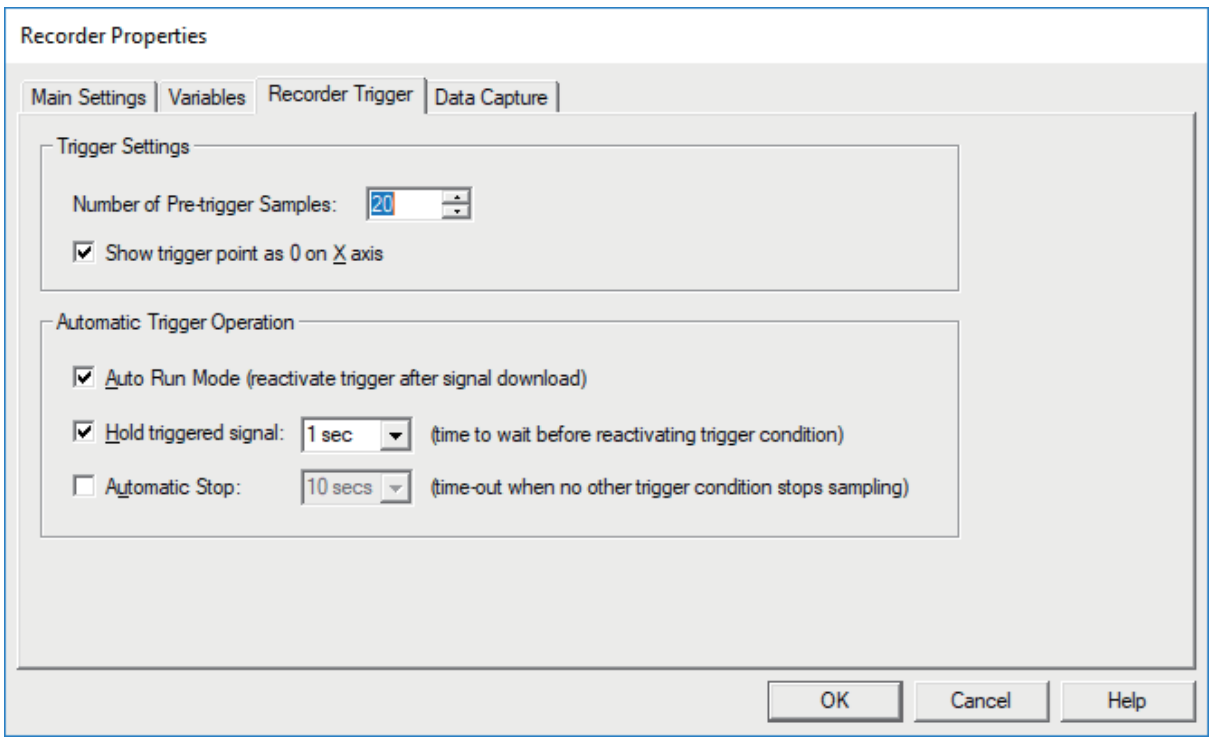

図 14. [Recorder Properties（レコーダのプロパティ）] ウィンドウのトリガ・ページ

- [Number of Pre-trigger Samples（プリトリガ・サンプル数）]：トリガ・イベントの前に保存して表示するサンプルの数を指定します。
- [Show trigger point as 0 on X axis（トリガ・ポイントを X 軸の 0 として表示する）]：オンにした場合、トリガ・イベントの時刻に X 軸の原点が置かれます。
- [Automatic Trigger Operation（自動トリガ動作）]：トリガを再開して繰り返しの記録を行う条件を指定します。
  - [Auto run mode(自動実行モード)]:繰り返しの記録を有効にします。トリガ・イベントが検出されて、バッファにデータが格納され、バッファ・データが FreeMASTER にダウンロードされた後、トリガが自動的に再開され、次のトリガ・イベントの発生後すぐに新しいデータがダウンロードされます。
  - [Hold triggered signal（トリガされた信号を保持）]：このチェック・ボックスをオンにした場合、チャートに信号が表示されてからトリガを再開するまでの待機時間を指定します。
  - [Automatic Stop（自動停止）]：このチェック・ボックスをオンにした場合、トリガ・イベントを待ち受ける時間の最大値を指定します。指定した時間内にトリガ・イベントが検出されなかった場合、サンプリングは無条件に停止され、実際のバッファ・データがダウンロードされます。

4.1.1.4 Array Viewer

前のセクションで述べたように、レコーダには極めて高度なプロトコル・ロジックが要求されます。ターゲット・アプリケーションの一部として実行されるドライバが、それに対応していなければなりません。このため、JTAG インターフェースや BDM インターフェースを介して FreeMASTER が「パッシブ」に接続する場合、レコーダを使用することはできません。このようなインターフェースは、メモリの直接読み取り / 書き込み操作にしか対応していないためです。

Array Viewer は、レコーダをよりシンプルで軽量にしたものと考えることができます。Array Viewer はターゲット・メモリから配列を読み取り、その値をグラフに表示することができます。別のスカラ変数がトリガとして使用され、その値が変化するたびにグラフが更新されます。FreeMASTER は必要に応じて、グラフの表示後にトリガ変数を自動的にクリアし、新しいデータを準備してよいという肯定応答をターゲットに返すことができます。

Array Viewer の主な利点は、メモリの読み取り機能さえあればよい点です（トリガの肯定応答の場合は書き込み機能が必要）。そのため、直接アクセス型の JTAG インターフェースや BDM インターフェースを含め、どのような通信インターフェースでも動作することができます。多くの場合、スカラー変数の経時的遷移を収集することが主な役割であるレコーダを使用するよりも、ターゲット・メモリに存在する値の配列全体を表示する方が適切と言えます。

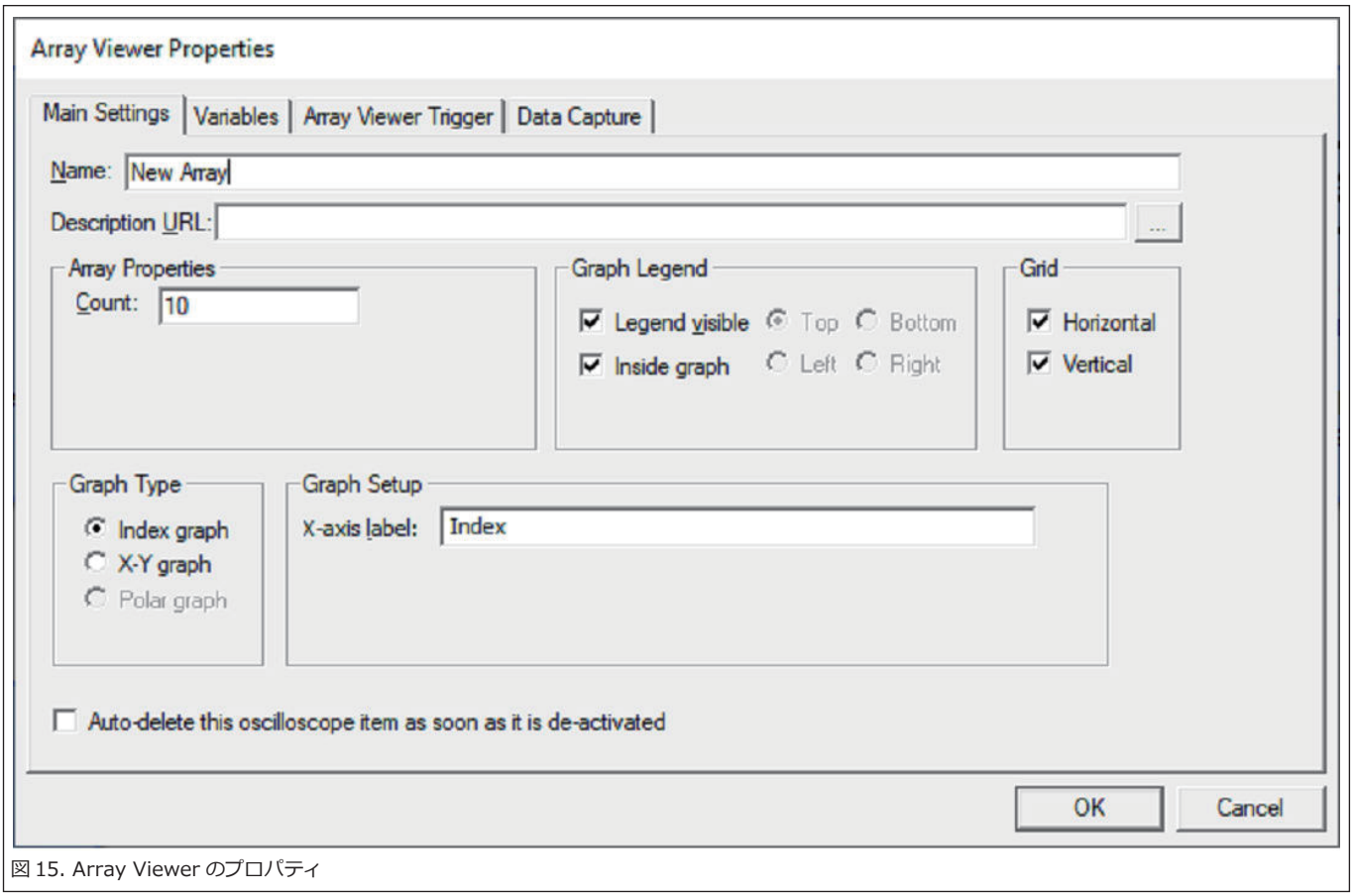

図 15. Array Viewer のプロパティ

[Main Settings（メイン設定）] タブでは、Array Viewer の全体的なパラメータを定義します。

- [Name（名前）] : プロジェクト・ツリーに表示されるレコーダ項目の名前。
- [Description URL（説明 URL）]:[Detail View（詳細ビュー）]部分の[current item help（現在の項目のヘルプ）]タブに表示されるファイルのローカル・パスまたはドキュメントの URL を指定します。
- [Array Length（配列長）] : 各配列からフェッチする要素の数を指定します（表示される配列は [Variables（変数）] タブに一覧表示されます）。
- [Graph Legend（グラフの凡例）]、[Grid（グリッド）]、[Graph Type（グラフの種類）]、[Graph Setup（グラフの設定）] : 前セクションで説明したレコーダの場合と同様にグラフのスタイルを定義します。

[Array Viewer properties（Array Viewer のプロパティ）] ダイアログの [Variables（変数）] タブは、レコーダやオシロスコープで使用されているものと同じです。可視化する配列のリストを定義する際に使用します。FreeMASTER では、ターゲット変数を配列の基底として識別するために特殊なフラグやオプションは使用されません。Array Viewer は単に、配列の要素のように見えるすべての変数（名前が角括弧で終わるもの）を配列全体を表す要素として提示します。たとえば、arr16[0] という名前の変数を、arr16 配列を表す基底の要素として使用できます。Array Viewer は、そのような代表的要素のアドレスを、配列全体を読み取るためのメモリの基底アドレスとして使用します。配列の長さは、[Main Settings（メイン設定）] タブで指定した[Array Length（配列長）]オプションから取得されます。また、配列から読み取られたすべての要素には、配列を表す変数の定義で指定されたサイズ、型、形式、ビット・マスク、リアルタイム後処理変換式が適用されます。その後で、それぞれの値が最終的なグラフに出力されます。

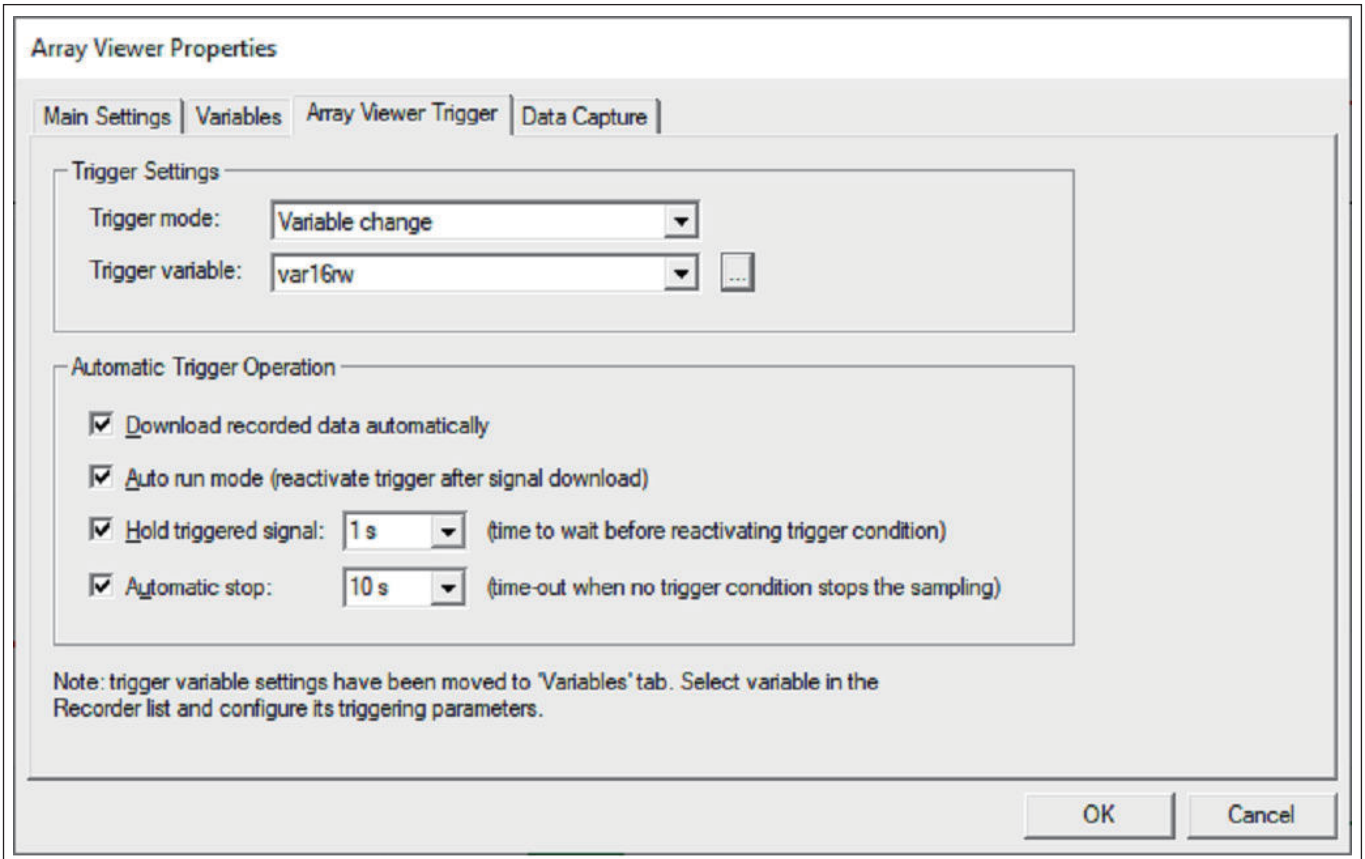

図 16. Array Viewer のプロパティ

[Array Viewer Trigger（Array Viewer のトリガ）] タブでは、ビューアのトリガ変数を始めとする各種オプションを定義します（これもレコーダ・オブジェクトとよく似ています）。

- [Trigger mode（トリガ・モード）] : Array Viewer が起動し、すべての配列を読み取って出力結果のグラフに表示するトリガ条件を選択します。次のいずれかを選択できます。
  - [Off（オフ）]: トリガは使用されません。手動でのボタン押下、または [Automatic stop（自動停止）] に指定された時間の経過によってのみ、ビューアによるデータの表示がトリガされます。
  - [Variable change（変数の変化）] : 下で選択された変数を監視し、変数の値が変化したときに随時ビューアを起動します。
  - [Variable change with acknowledge（変数の変化と肯定応答）] : 変数の変化を監視します。グラフの更新後、FreeMASTER によって変数が 0 に設定されます。
- [Trigger variable（トリガ変数）] : トリガ（選択したトリガ・モードによる）として使用する変数。
- [Download recorded data automatically（記録されたデータを自動的にダウンロードする）] : 通常は有効にします。トリガが検出された後、自動的にグラフ・データがダウンロードされて表示されます。オフにした場合、ボタンが押下されるのを待って処理が実行されます。
- [Auto run mode（自動実行モード）] : オンにした場合、グラフの更新後にトリガ条件が自動的に再開されます。これによって、トリガ変数が変化するたびにグラフが更新されます。
- [Hold triggered signal（トリガされた信号を保持）] : 自動実行モードで、グラフを画面に表示し続ける最小時間。この時間を経過すると、トリガが再開されます。
- [Automatic stop（自動停止）] : この時間が経過した場合、トリガが発生していなくてもビューアが起動してグラフが更新されます。

[Data Capture（データ・キャプチャ）] タブでは、配列データを 1 つのファイルまたは一連のファイルに保存できます。これも、レコーダの場合とほぼ同じです。

4.1.2 詳細ビュー

詳細ビューは、複数のページから成る部分です。詳細ビューに表示される各種ページは、プロジェクト・ツリーで選択した項目の種類によって異なります。

プロジェクトの［Options（オプション）］（「HTML ページ」）で制御ページが定義されている場合、すべてのプロジェクト・ツリー項目からいつでもボードを制御することができます。［algorithm block description（アルゴリズム・ブロックの説明）］ページの内容は、プロジェクト・ツリーで選択した項目によって変わります。プロジェクト・ツリーで［Oscilloscope（オシロスコープ）］項目または［Recorder（レコーダ）］項目を選択した場合、［current item help（現在の項目のヘルプ）］ページとオシロスコープ / レコーダ・チャート・ページも表示されます。

4.1.2.1 制御ページ

制御ページは、ボード・アプリケーションを制御する目的で作成された HTML ページです。一般に、スクリプトを活用したフォームや、組み込みアプリケーションを簡単に制御できる機能を備えたフォームが含まれています。このページの URL、つまりページのソース・コードが記述された HTML ファイルのパスは、プロジェクトの［Options（オプション）］ダイアログで指定できます（「HTML ページ」を参照）。

下図に示したのは、例として使用されるデモ・アプリケーションの制御ページです。このシンプルなアプリケーションには、その名前に反して、「制御」機能がほとんどありません。var16inc（var16 変数の増分量を制御する変数）の値を設定するボタンが 3 つあるだけです。また、このページでは、JavaScript のスクリプティング手法を使用して、HTML でコーディングされたページとより高度なゲージ・オブジェクトに直接変数の値を表示しています。HTML のコーディングとスクリプティングの詳細については、「FreeMASTER ActiveX インターフェース」を参照してください。

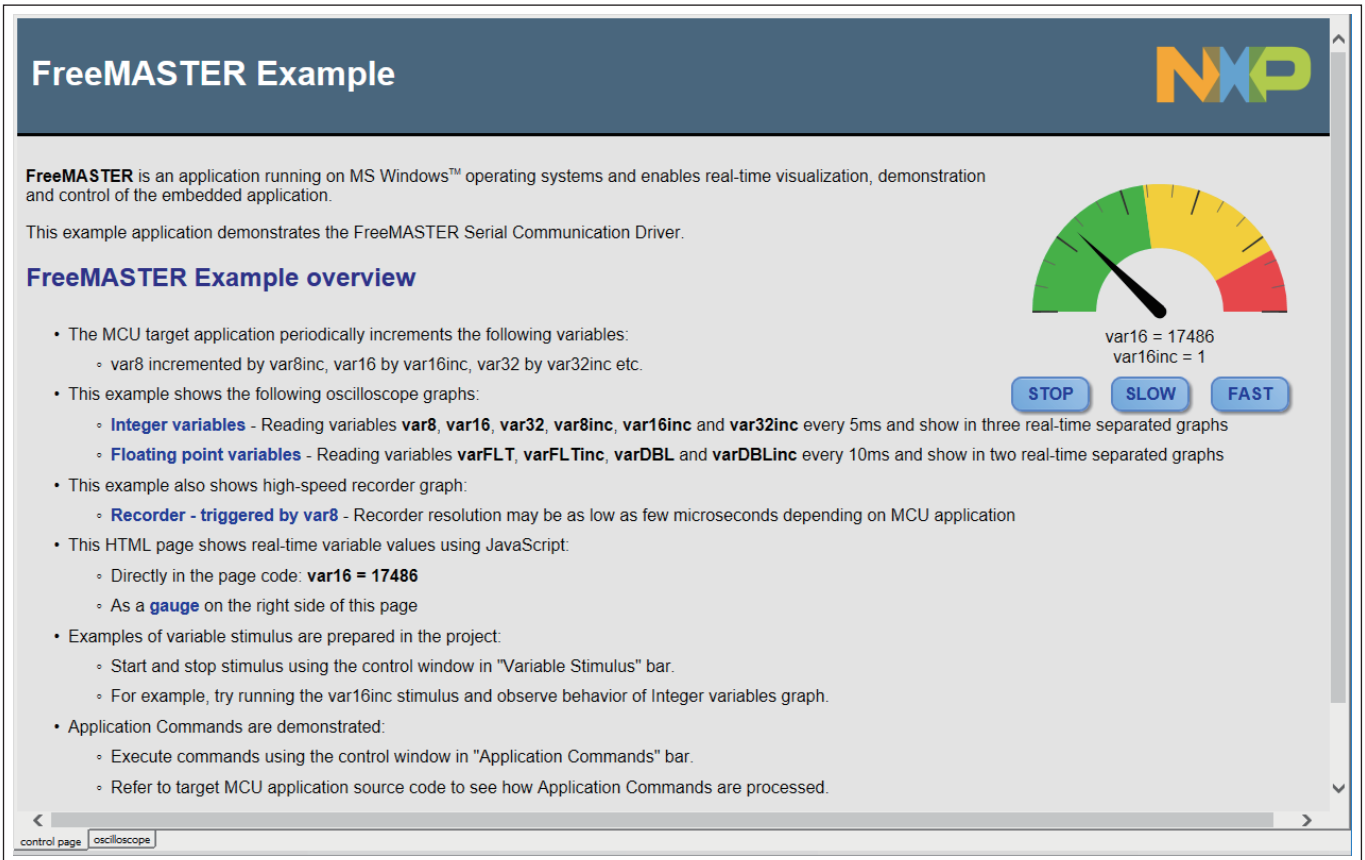

図 17. デモ・アプリケーションの制御ページ

デモンストレーションの制御ページには、デフォルトで Internet Explorer および ActiveX インターフェースとの下位互換性オプションが使用されているため、このプロジェクトは FreeMASTER の全バージョン（2.x および 3.x）で正しく動作します。バージョン 3.0 では、Chromium ウェブ・ブラウザ・エンジンによってレンダリングされるページ上の非同期 JSON-RPC インターフェースで JavaScript の最新手法を使用できるようになりました。同じページをスタンドアロンの Chrome ブラウザでも使用でき、スクリプト・コードの印刷、編集、デバッグなどで利便性がアップします。

FreeMASTER の制御ページを新しく設計する際は、Chromium レンダリングと JSON-RPC インターフェースを併用することをお勧めします。こうして作成されたページは、FreeMASTER Lite サービスに接続されたスタンドアロンの Chrome ブラウザでも適切に動作します。

その他、高度な制御ページのスクリーンショットを以下にいくつか紹介します。ここで紹介するアプリケーションでは、サード・パーティの計測コンポーネントが埋め込み ActiveX オブジェクトとして HTML コードに挿入されています。

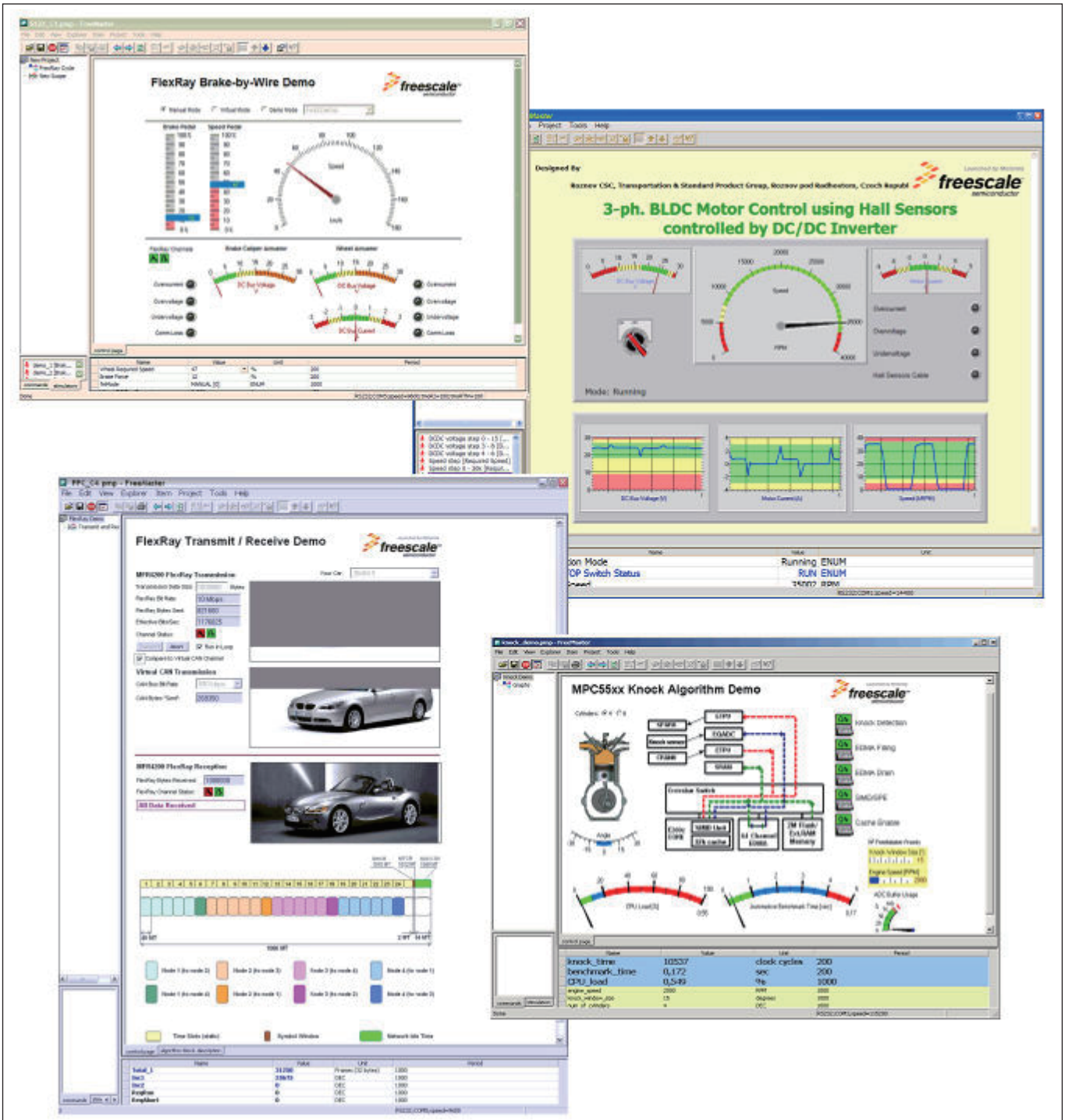

図 18. HTML 制御ページの例

4.1.2.2 アルゴリズム・ブロックの説明

［Detail View（詳細ビュー）］部分の［Algorithm Block Description（アルゴリズム・ブロックの説明）］ページには、選択したブロックの機能について説明するHTMLページが表示されます。ソース・ファイルのURLまたはローカル・パスは、ブロック項目のプロパティ・ダイアログで指定します（「プロジェクト・ブロックとサブブロック」を参照）。このページは、このページが定義済みで、なおかつ該当するブロック項目またはそのいずれかの子オシロスコープ / レコーダ項目を選択した場合に表示されます。

標準的な HTML ページと同様、このページにスクリプトや他のコントロールを記述することもできますが、あまり一般的ではありません。制御周りの機能はすべて制御ページにまとめることをお勧めします。

4.1.2.3 現在の項目のヘルプ

[Current item help (現在の項目のヘルプ)] タブには、選択したオシロスコープまたはレコーダについて説明する HTML ページが表示されます。このページは通常、定義や使用手順などの情報を含んでいます。説明 URL として指定されます。このページでスクリプトなどの制御機能を使用することも、やはり推奨されません。

4.1.2.4 オシロスコープ / レコーダ

[Detail View (詳細ビュー)] 部分の [Oscilloscope (オシロスコープ)] ページには、追跡された変数を表すリアルタイムのチャートが表示されます (「オシロスコープ」を参照)。同様に、[Recorder (レコーダ)] ページには、記録されたデータから作成されたチャートが表示されます (「レコーダ」を参照)。

4.1.3 ウォッチ・グリッド

アプリケーション・ウィンドウの一番下にある [Watch-Grid (ウォッチ・グリッド)] 部分には、ウォッチ変数が一覧表示されます。対象となるウォッチ変数とそのグラフィカル・プロパティは、プロジェクト・ブロックごとに個別に定義します。そのため、別のプロジェクト・ブロックを選択すれば、[Watch-Grid (ウォッチ・グリッド)] 部分の内容も変化します。

変数を定義する際は、変数の名前、単位、数値形式を指定します。さらに、書式設定バーを使用して、フォントの種類とサイズ、前景色と背景色、位置揃えなど、変数の外観を変更することもできます。詳細については、「書式設定バー」を参照してください。

監視できるのは、読み取り専用の変数だけです。変更が許可されている変数は、[Watch-Grid (ウォッチ・グリッド)] 部分から変更できます。変数の詳細については、「変数」を参照してください。

4.2 変数

FreeMASTER は、明確に規定された通信プロトコルを介してボード・アプリケーションとやり取りします。このプロトコルが、PC アプリケーションからターゲット・ボード・アプリケーションへのコマンド送信とその変数の読み取り / 書き込みをサポートします。FreeMASTER プロジェクトで使用されるすべてのコマンドおよび変数は、プロジェクト内で指定されている必要があります。

[Variables List(変数リスト)]ダイアログ・ボックス(下図)は、[Project(プロジェクト)]メニューの[Variables(変数)]項目を選択して開くことができます。また、プロジェクト開発における他の段階、たとえば[Oscilloscope Properties(オシロスコープのプロパティ)]ダイアログの[Variables(変数)]タブでも、このダイアログを開いて変数を管理できます。

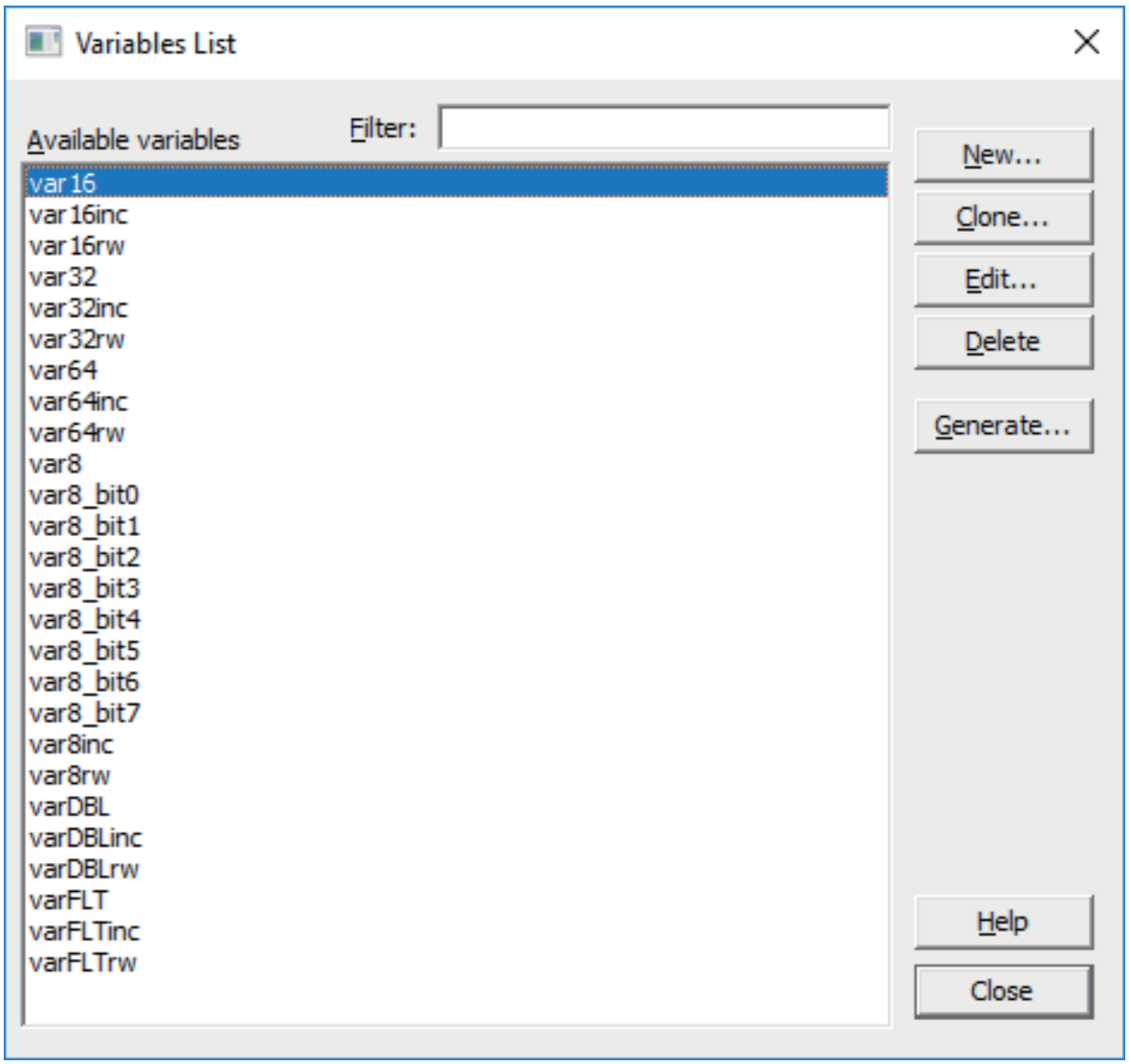

図 19. [Variables list（変数リスト）] ダイアログ・ボックス

新しい変数を定義するには、[New（新規作成）] ボタンを使用します。[Variable（変数）] ダイアログ・ボックスが開いて、変数のプロパティを設定することができます。既存の変数のコピーを作成したい場合は、元の変数を選択し、[Clone（複製）] ボタンを押します。

[Edit（編集）] ボタンをクリックすると、同じ [Variable（変数）] ダイアログ・ボックスが開いて、選択した変数のプロパティに変更を加えることができます。[Delete（削除）] ボタンを押すと、削除の確認後、選択した変数が削除されます。

[Generate（生成）] ボタンを押すと、組み込みアプリケーションの実行ファイルからロードされたシンボルに基づいて変数オブジェクトを一括作成するためのインターフェースが開きます。この点については、「変数の生成」で説明しています。シンボル・ファイルのロードについては、「シンボル・ファイル」で説明しています。

変数の設定

[Variable definition（変数の定義）] ダイアログには、図 20 に示す [Definition（定義）] と、図 21 に示す [Modifying（変更）] の 2 つのタブがあります。

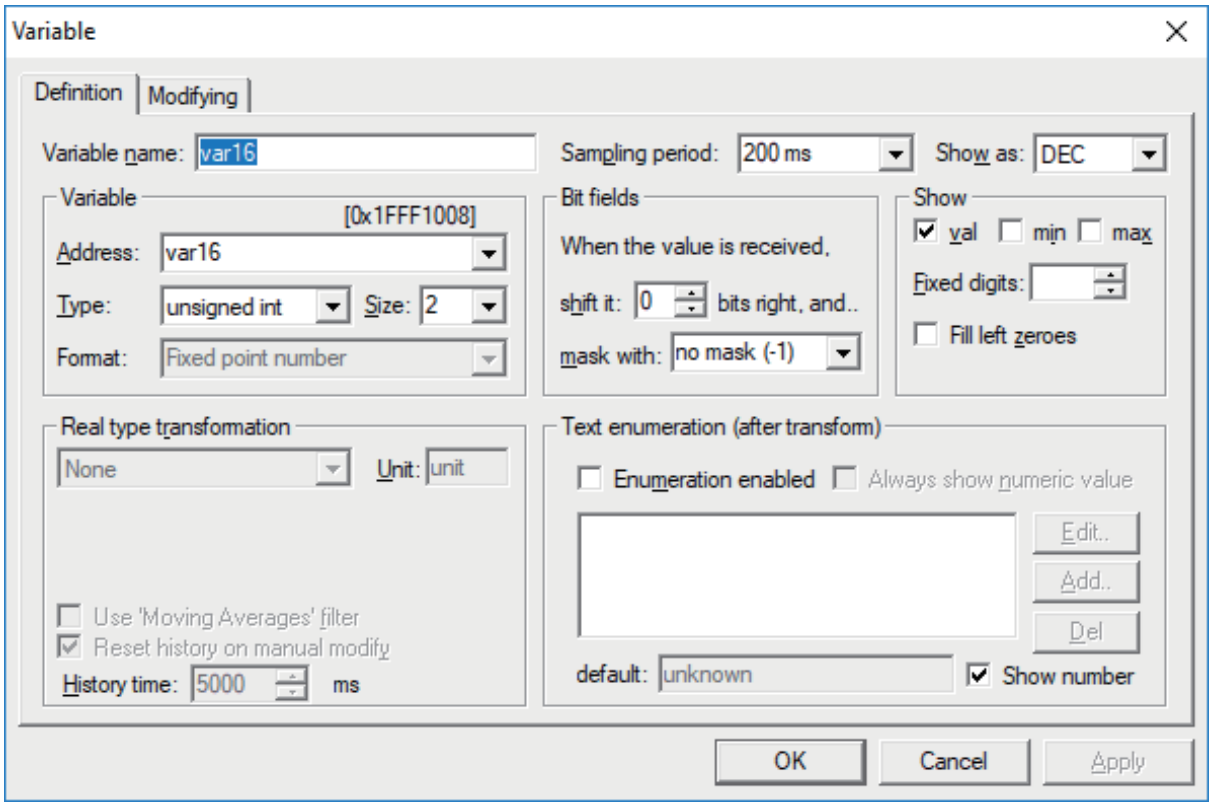

図 20. [Variable（変数）] ダイアログ・ボックスの [Definition（定義）] タブ

[Definition（定義）] タブでは、変数の一般的なプロパティを指定します。

- [Variable name（変数名）] : プロジェクトにおける変数の識別子として変数名を指定します。
- [Sampling period（サンプリング周期）] : 変数ウォッチに変数が表示されているときにその値をボードから読み取る時間周期。
- [Shows as（表示形式）] : 変数の値をウォッチ・ウィンドウに出力する際の形式。ドロップダウン・リストから適切な形式（DEC、HEX、BIN、ASCII、REAL のいずれか）を選択します。
- [Variable（変数）] パネル : 組み込みアプリケーションに定義されている変数についての情報。
  - [Address（アドレス）] : ターゲット・アプリケーションのメモリにおける変数の物理アドレス。直接 16 進数の値を入力することもできますが、アプリケーションの変数のシンボル名をドロップダウン・リストから選択することをお勧めします。シンボル・テーブルは、ターゲット・アプリケーションから直接（TSA 機能が有効になっている場合）または組み込みアプリケーションの実行ファイルから ELF 形式でロードできます。数値演算、sizeof() 演算子、配列デリファレンス演算子 [] を含んだ式をアドレス・フィールドに指定することもできます。
  - [Type（型）] : ターゲット・アプリケーションに定義されている変数の型（符号なし固定小数点、符号あり固定小数点、浮動小数点 IEEE、小数、符号なし小数、文字列）を選択します。
  - [Size（サイズ）] : ターゲット・アプリケーションに定義されている変数のサイズを指定します。
  - [Format（形式）] : 小数型が適切にデコードされるように、拡張バイナリ形式のパラメータを指定する必要があります。
- [Bit fields（ビット・フィールド）] : 特定の変数から単一のビットまたは一連のビットを抽出する操作に関するパラメータ。
  - [Shift（シフト）] : 受け取った値をマスク処理の前に右シフトする際のビット数を指定します。
  - [Mask（マスク）] : シフト済みの値との間で AND 演算を行うマスク値を選択または指定します。
  - [Shift（シフト）] フィールドと [Mask（マスク）] フィールドを使用することで、受け取った変数から任意のビット・フィールドを抽出できます。たとえば、16 ビットの整数値から最上位ビットを抽出したければ、15 ビット・シフトと 1 ビット・マスク（0x1）を指定することになります。
- [Show（表示）]:変数の値が、[Show as（表示形式）] フィールドで選択した値の表示形式に従って実際にどう出力されるかが、次の一連のパラメータによって制御されます。
  - [val（値）]、[min（最小）]、[max（最大）]:変数ウォッチで変数のイミディエイト値や検出されたピーク値を表示したい場合は、これらのチェック・ボックスをオンにします。ピーク値は、ウォッチ・ウィンドウで変数のエントリを右クリックし、メニュー・コマンド [Reset MIN/MAX（最小 / 最大のリセット）] を選択することでリセットできます。

— [Fixed digits（固定桁数）]（表示形式を DEC、HEX、BIN のいずれかに設定した場合）：数値の左側をゼロまたは空白で埋めて特定の桁数にして出力します。
— [Fixed digits（固定桁数）]（表示形式を REAL に設定した場合）：浮動小数点型の数値を、小数点以下の桁数を固定して出力します。
— [Zero terminated（ゼロ終端）]（表示形式を文字列値に設定した場合）：文字列値を、ゼロ文字が最初に出現する位置まで出力します。文字列値の場合、印字できない文字を 16 進数（または疑問符）として表示するかどうかや、その他いくつかの文字列固有の設定を選択することもできます。

- [Real type transformation（実数型変換）]：表示形式を REAL に設定した場合、追加的な後処理として変数の値に適用される数値変換を定義できます。
  — 変換の種類

    [linear: ax + b（線形:ax + b）]:線形変換 y = ax + b の定数「a」と「b」を指定します。パラメータ「a」と「b」は、数値として指定できるほか、プロジェクトの変数の名前で指定することができます。後者の場合は、イミディエイト値（直近の有効な値）がパラメータとして使用されます。

    [linear two points（線形 2 座標点）]:パラメータ「a」と「b」を使用するよりも、2 つの座標点を使用した方が線形変換を指定しやすければ、（x1, y1）と（x2, y2）の 2 つの座標点を入力します。この場合も、パラメータの値として数値または変数名を指定できます。

    [hyp: d/(ax+b) + c（双曲型：d/(ax+b) + c）]：双曲型変換関数のパラメータ「a」、「b」、「c」、「d」を指定します。

  — [Unit（単位）] = 変数ウォッチに表示される単位の名前。
  — [Use “Moving Averages” filter（「移動平均」フィルタを使用する）]：ノイズの多いアクションを監視するときは、イミディエイト値ではなく平均値を表示した方がよい場合があります。
  — [History time（履歴時間）]：平均値の計算に使用される時間間隔。
- [Text enumeration（テキスト列挙）]:特定の変数の値の意味を説明し、そのそれぞれにテキスト・ラベルを割り当てることができます。変数ウォッチには、割り当てたラベルが数値とともに、または数値の代わりに表示されます。テキスト・ラベルに対する値の割り当てを含んだルックアップ・テーブルは、[Edit（編集）]、[Add（追加）]、[Del（削除）] の各ボタンを使用して管理します。
  — [Default（デフォルト）]：一致するテキストがルックアップ・テーブルに見つからなかった場合に表示されるデフォルトのテキスト・ラベルを指定します。

[Modifying（変更）] ページ（下図）には、変数の値に変更を加えるうえでの設定や制限が表示されます。

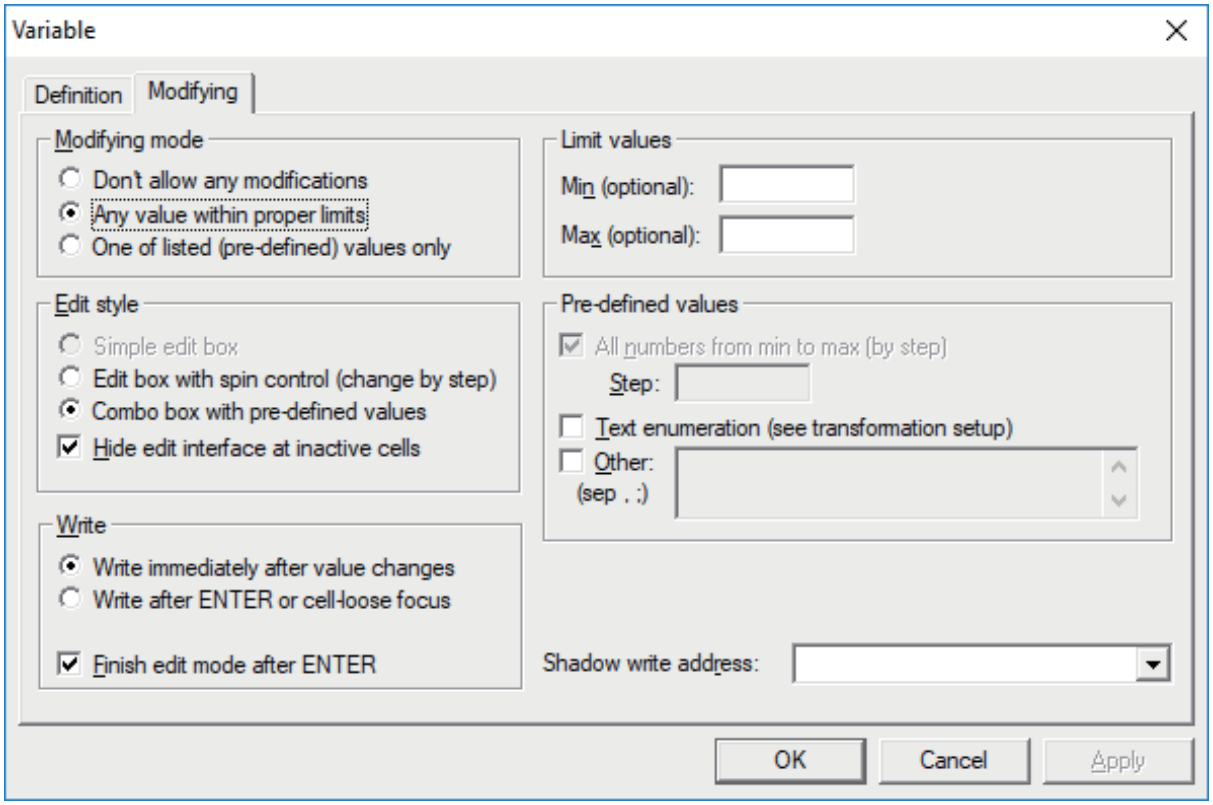

図 21. [Variable（変数）] ダイアログ・ボックスの [Modifying（変更）] タブ

- [Modifying mode（変更モード）]
  - [Don’ t allow any modifications（変更を一切許可しない）]：変数は読み取り専用です。このページに表示される他の設定はすべて無効になります。
  - [Any value within proper limits（適切な範囲内の任意の値）]:最小値と最大値、またはそのどちらか一方を指定できます。この場合、ウォッチ・ウィンドウに入力した値は、指定した制限に基づいて検証されます。
  - [One of listed values only（リストにあるいずれかの値のみ）]：一連の値を指定した場合、それらの値に限り、ウォッチ・ウィンドウに書き込むことができます。入力できる値は、[Pre-defined values（定義済みの値）] グループで指定できます。

  [All numbers from min to max（最小から最大までのすべての数値）]：[Min（最小）] から [Max（最大）] の範囲の [Step（ステップ）] 刻みのすべての数値が定義済みの値として扱われます。

  [Text enumeration（テキスト列挙）]：テキスト列挙ルックアップ・テーブルにあるすべての値が定義済みとして扱われます。

  [Other（その他）]：定義済みの値が他にもあれば、コンマ区切りまたはセミコロン区切りで指定します。
- [Edit style（編集スタイル）]：ウォッチ・ウィンドウ・グリッドの該当するセルに表示される、特定の変数を編集するためのインターフェースの外観を選択します。
  - [Edit box with spin control（スピン・コントロール付きの編集ボックス）]：変数の値を編集するためのインターフェースとして、値の増減に使用する 2 つのスピン矢印を備えた編集ボックスが表示されます。
  - [Combo box with pre-defined values（コンボ・ボックスと定義済みの値）]：変数の値を編集するためのインターフェースがドロップダウン・リスト・ボックスの形式で表示されます。リストから定義済みの値を選択できます。
  - [Hide edit interface at inactive cells（アクティブでないセルでは編集インターフェースを非表示にする）]：ウォッチ・グリッド内の該当するセルがキーボード・フォーカスを失ったとき、変数の編集インターフェースが非表示になります。
- [Write style（書き込みスタイル）]：変数の新しい値が実際にどのようなタイミングでボード・アプリケーションに送信されるかを指定します。
  - [Write immediately after each value changes（各値の変更後すぐに書き込む）]：スピン矢印ボタンを押すたびに、またはドロップダウン・リスト・ボックスから新しい値を選択するたびに、変更後の変数値が組み込みアプリケーションに送信されます。
  - [Write after ENTER or kill focus only（ENTER が押された後またはフォーカスを失った後にのみ書き込む）]：変更後の変数値は、Enter キーが押されたときに初めて組み込みアプリケーションに送信されます。

4.2.1 変数の生成

[Variable List（変数リスト）]ダイアログ（図 19）の[Generate（生成）]ボタンを押すと、下図に示す[Generate variables（変数の生成）]ダイアログ・ボックスが表示されます。

このダイアログでは、接続先の組み込みアプリケーション（TSA）やその実行ファイル（ELF）、またはリンカ MAP ファイルからロードされたシンボルの変数オブジェクトを自動的に生成できます（シンボル・テーブルの詳細については「シンボル・ファイル」を参照してください）。

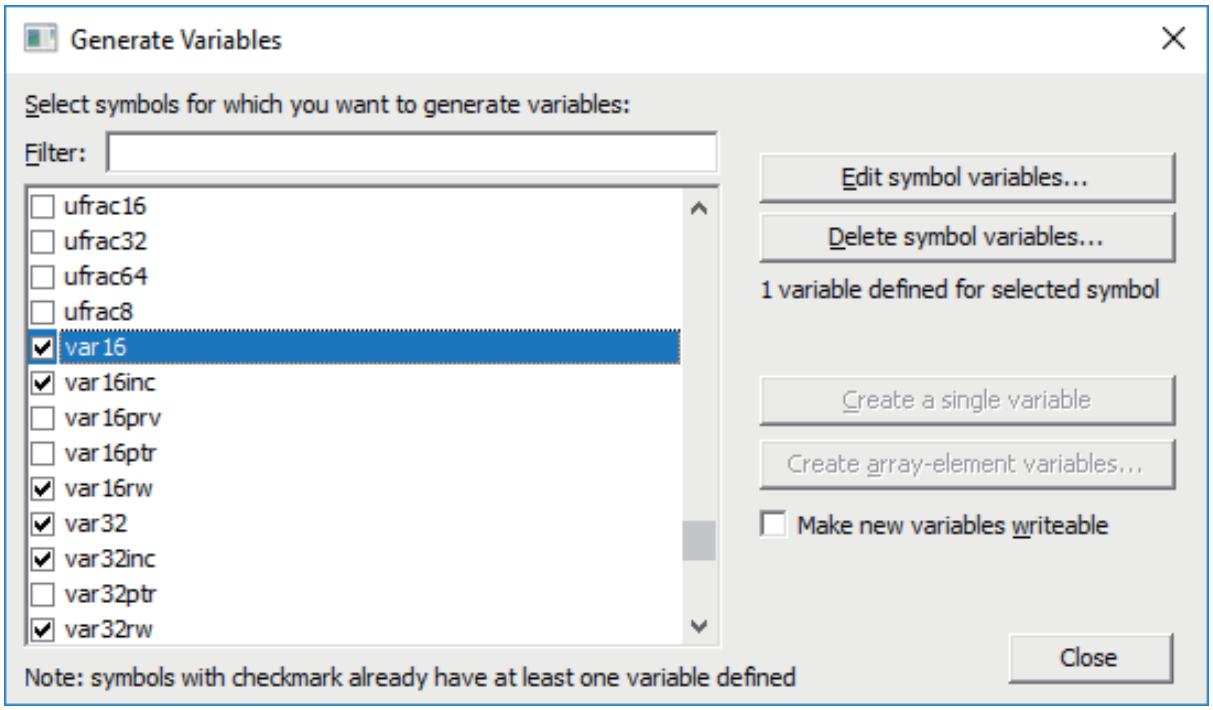

図 22. 変数の生成

このダイアログには、プロジェクトで利用できるすべてのシンボルが、現在のシンボル・ファイルから読み取られて一覧表示されます。既に変数が定義されているシンボルにはチェックマークが付きます。通常、変数は、チェックマークが付いていないシンボルに対して作成することになります。

- [Edit symbol variable（シンボル変数の編集）]：選択したシンボルにバインドされている変数がある場合に、その変数を編集するには、このボタンをクリックします。
- [Delete symbol variable（シンボル変数の削除）]：選択したシンボルにバインドされている変数を削除するには、このボタンをクリックします。
- [Create a single variable（単一の変数の作成）]:選択したすべてのシンボルを対象に、シンボルと同じ名前、適切なアドレス、型、サイズ設定を持った新しい変数を生成（作成）します。作成後、[Edit symbol variable（シンボル変数の編集）]を使用して、新しく作成した変数オブジェクトのパラメータを表示したり変更したりすることができます。
- [Generate array-element variables…（配列要素変数の生成…）]：配列の個々の要素をカプセル化する一連の変数を生成できます。

4.2.1.1 配列要素変数の生成

FreeMASTER のシンボル・リストには、配列を表すシンボルが 2 つあります。1 つは配列全体を表すもの、もう 1 つは先頭の（またはそれ以外の）配列要素を表すものです。

デモ・マイクロコントローラ・アプリケーションでは、短整数型の配列が次のように宣言されています。

```
short int arr16[10];
```

FreeMASTER では、`arr16` シンボルが、配列全体に使用される 20 バイトのメモリ領域を表します。加えて、`arr16[0]` という名前のシンボルは、配列の先頭 2 バイトの要素を表します。

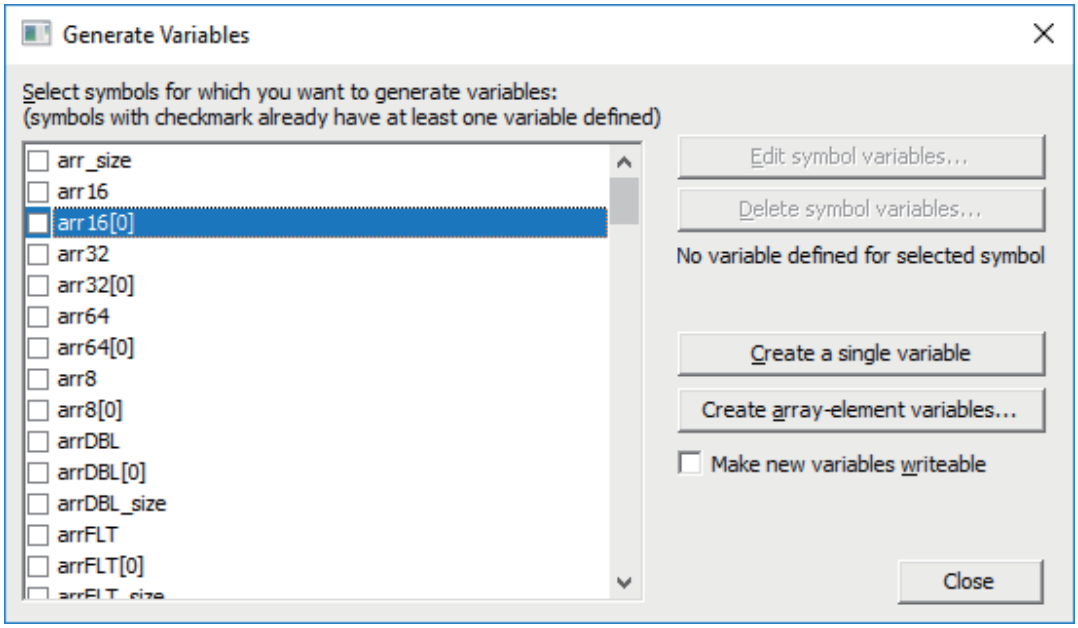

図 23. 配列要素変数の生成

1 つの変数を定義するときは、変数のアドレスとして `arr16[0]` とそれ以外のインデックス付き要素（`arr16[1]`、`arr16[2]` など）を使用できます。複数の配列要素変数をまとめて作成する場合は、前セクションで説明した［Generate Variables（変数の生成）］ダイアログで自動化できます。配列要素シンボル（`arr16[0]` など）を選択して、［Create array-element variables（配列要素変数の作成）］ボタンをクリックします。

簡単なダイアログが開いて、変数を生成する配列インデックスの範囲を指定できます。

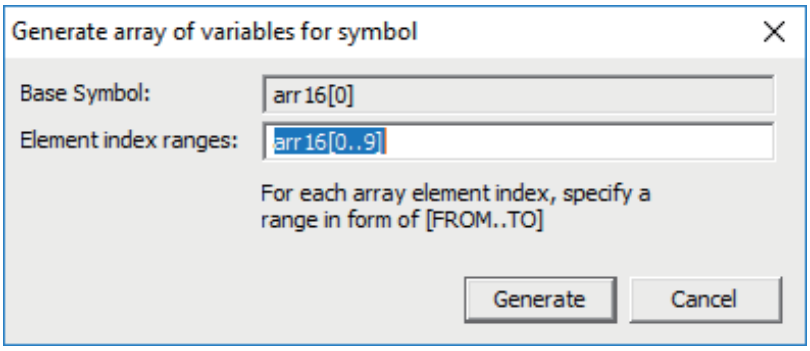

図 24. シンボルの変数の配列を生成

4.3 コマンド

プロジェクトで定義されたアプリケーション・コマンドのリストは、［Project（プロジェクト）］/［Commands（コマンド）］メニューを選択して表示できます（下図）。リストの管理は、［New（新規作成）］、［Clone（複製）］、［Edit（編集）］、［Delete（削除）］の各ボタンで行います。これは変数の管理とよく似ています。

- ［New（新規作成）］ボタン：新しいアプリケーション・コマンドを作成します。
- ［Edit（編集）］ボタン：選択されたアプリケーション・コマンドのプロパティを編集します。
- ［Clone（複製）］ボタン：選択されたコマンドのコピーを使用して新しいコマンドを作成します。
- ［Delete（削除）］ボタン：選択されたコマンドを削除します。
- ［Send（送信）］ボタン：組み込みアプリケーションにコマンドを送信するためのインターフェースを開きます。

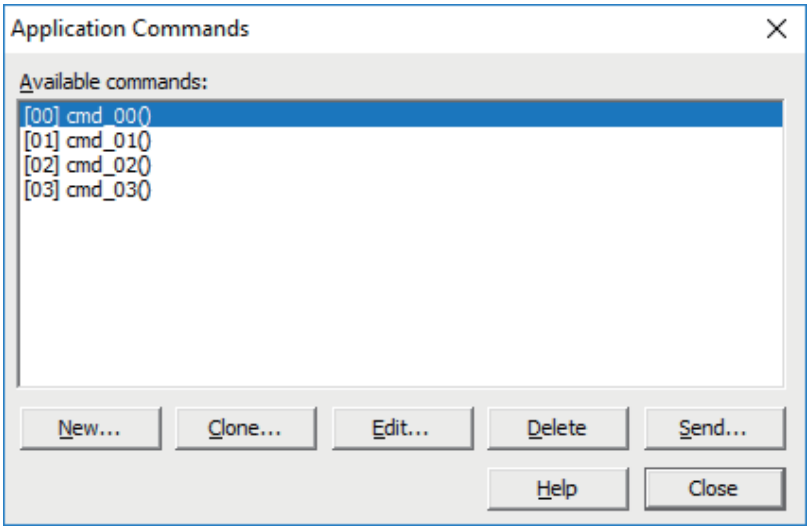

図 25. プロジェクトのアプリケーション・コマンド

[Send（送信）] ボタンを押して表示されるアプリケーション・コマンドの送信ダイアログ・ボックスでは、パラメータを必要に応じて指定して、組み込みアプリケーションにコマンドを送信することができます。このダイアログを図 26 に示します。図 27 に示したように、それぞれの引数にはヘルプ・メッセージを定義できます。定義したメッセージは、このダイアログで引数の値を入力するときに表示されます。

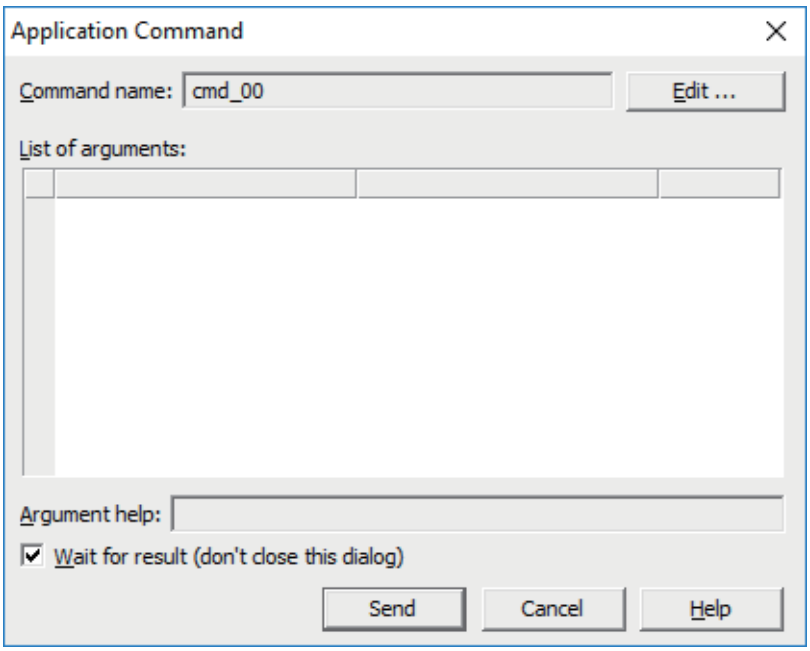

図 26. アプリケーション・コマンドの送信

ボードからデータが返されるまでダイアログを閉じずに（コマンドの結果を）待機したい場合は、[Wait for result（結果を待機する）] チェック・ボックスをオンにします。コマンドを送信する前に、コマンドの定義を確認または編集できます。

コマンドを定義したり編集したりする際は、[Application Command（アプリケーション・コマンド）] ダイアログ・ボックスが表示されます。下図は、このダイアログに 3 つあるページの 1 つ目を示したものです。

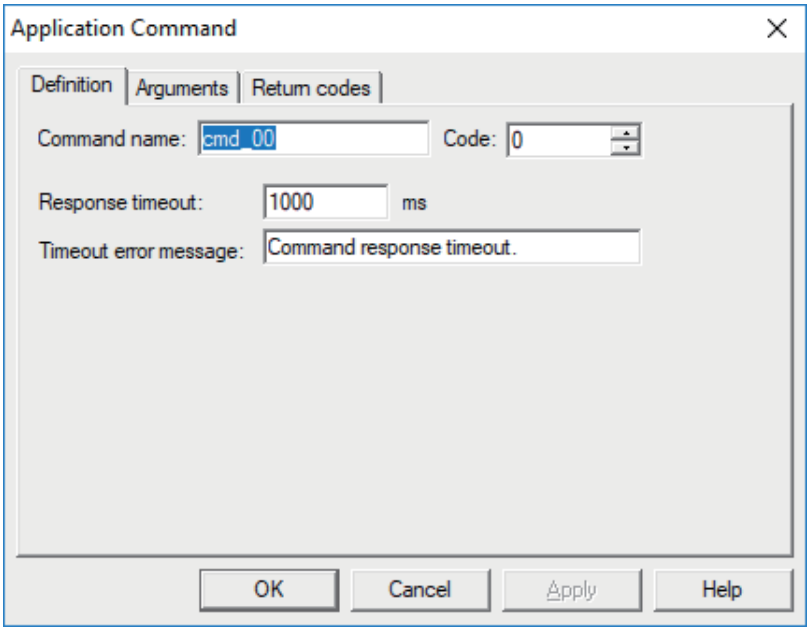

図 27. [Application Command（アプリケーション・コマンド）] ウィンドウの［Definition（定義）］タブ

［Definition（定義）］タブでは、プロジェクトで使用されているコマンド名を入力し、コード（ターゲット・ボード・アプリケーション内のコマンドを識別する 1 バイト・コマンド）を指定します。コマンドのコードとその目的、およびコマンドのリターン・コードとその目的は、ボード・アプリケーションの開発者から入手します。

［Response time-out（応答タイムアウト）］は、FreeMASTER がボード・アプリケーションの応答を待機する最大時間（ミリ秒）です。組み込みアプリケーションがコマンドを認識せず、なおかつ、コマンドに応答しないままこのタイムアウトに達した場合、［Timeout error message（タイムアウト・エラー・メッセージ）］フィールドに入力したテキストが警告ウィンドウに表示されます。

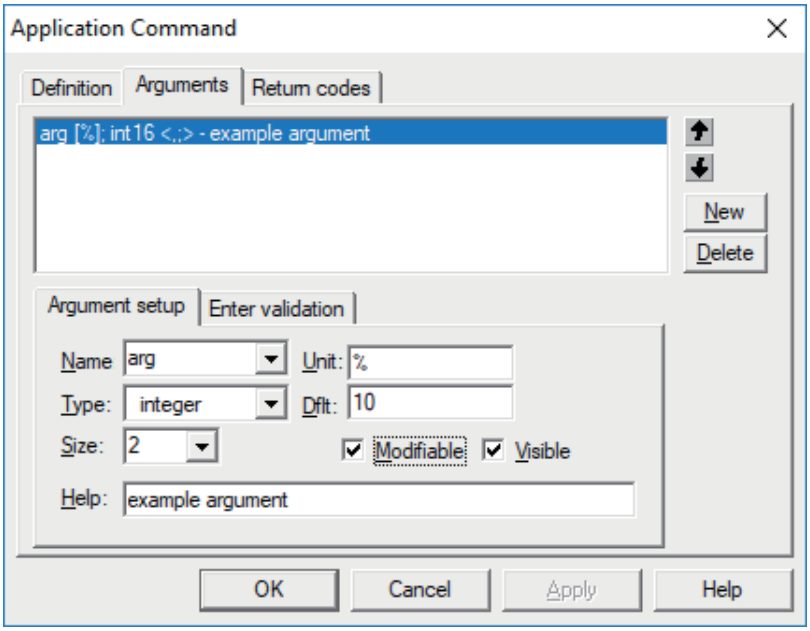

図 28. [Application Command（アプリケーション・コマンド）] ウィンドウの［Arguments（引数）］タブ（1 ページ）

［Arguments（引数）］タブ（上図）は、コマンドの引数を定義する際に使用します。引数のないコマンドの場合、引数リストには何も表示されません。コマンドには、コマンド・コードとともに値をターゲット・ボード・アプリケーションに渡すための引数を追加できます。

新しく引数を作成するには［New（新規作成）］ボタンを、リストで選択した引数を削除するには［Delete（削除）］ボタンを、引数の順序を変更するには上下の矢印を使用します。

[Argument setup（引数の設定）] サブ・ページでは、選択した引数のパラメータを定義します。

- [Name（名前）] : リストに表示される、または [Send application command（アプリケーション・コマンドの送信）] ダイアログで引数の値を求められたときに表示される引数の名前を指定します。既存の引数名をドロップダウン・リスト・ボックスから選択することもできます。
- [Type（型）] : 引数の数値（整数または浮動小数点）を指定します。
- [Size（サイズ）] : 引数の値のサイズ（バイト単位）を指定します。
- [Unit（単位）] : 引数の単位として表示するテキストを指定します。ターゲット・アプリケーションにこのテキストは送信されません。
- [Dflt（デフォルト）] : 引数のデフォルト値を入力します。この値は、[Send application command（アプリケーション・コマンドの送信）] ダイアログの引数リストに対して設定されます。空にした場合は、コマンドを送信する際に都度、値を入力します。
- [Modifiable（変更可能）] : このチェック・ボックスがオフの場合、[Send application command（アプリケーション・コマンドの送信）] ダイアログの引数リストでデフォルトの引数値を変更することはできません。
- [Visible（表示）] : このチェック・ボックスがオフの場合、引数リストに引数は表示されず、常にデフォルト値がターゲット・アプリケーションに送信されます。
- [Help（ヘルプ）] : [Send application command（アプリケーション・コマンドの送信）] ダイアログで引数の値を求められたときに表示するテキスト情報を作成します。

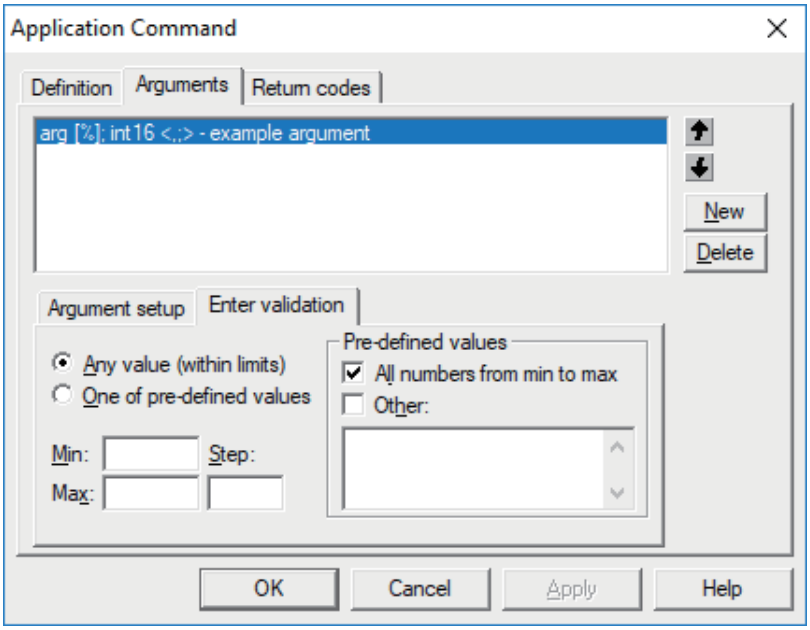

図 29. [Application Command（アプリケーション・コマンド）] ウィンドウの [Arguments（引数）] タブ（2 ページ）

上図の [Enter validation（検証条件の入力）] サブ・ページでは、引数の値に対する検証条件を定義します。

- 引数に指定できる値の指定 :
  - [Any value（任意の値）] : 引数の値として任意の数値を指定できます。[Min（最小）] と [Max（最大）] が設定されている場合、その制限の範囲内の値である必要があります。
  - [One of predefined values（定義済みのいずれかの値）] : [Pre-defined values（定義済みの値）] フィールドに定義されているいずれかの値のみ、引数の値として指定できます。
- [Pre-defined values（定義済みの値）] :
  - [All numbers from min to max（最小から最大までのすべての数値）] : このチェック・ボックスがオンになっている場合、[Min（最小）] から [Max（最大）] の範囲の [Step（ステップ）] 刻みのすべての数値が、有効な引数の値として扱われます。
  - [Other（その他）] : このチェック・ボックスをオンにした場合、その他、引数の値として有効な一連の値（コンマ区切り）を指定します。

[Return codes（リターン・コード）]ページ（下図）は、コマンドのリターン・コード・メッセージを指定する際に使用します。リターン・コードを作成するには、ページ左下のコード・フィールドに、16 進数（0x00）または 10 進数の形式でリターン・コードの値を入力し、その横のフィールドにリターン・コード・メッセージを入力して、[New（新規作成）] ボタンをクリックします。リターン・コード項目がリストに表示されます。この手順を繰り返して、必要なリターン・コードをすべて作成します。それぞれのリターン・コード・メッセージには、ページ右下のパネルからメッセージ・アイコンを割り当てることができます。割り当てたアイコンは、メッセージのテキストとともにメッセージ・ダイアログに表示されます。

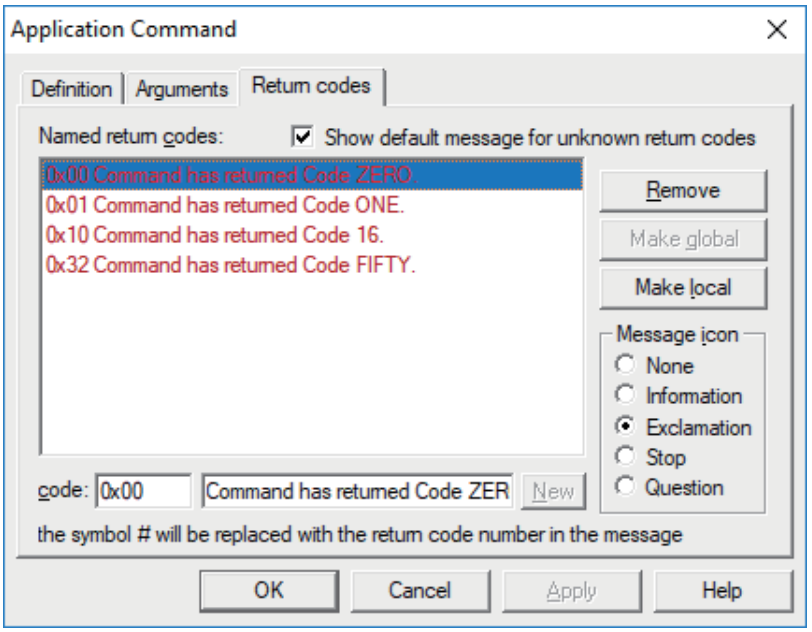

図 30. [Application Command（アプリケーション・コマンド）] ウィンドウの [Return codes（リターン・コード）] タブ

リターン・コードは、ローカルにすることもグローバルにすることもできます。ローカル・リターン・コードは 1 つのコマンドが対象であるのに対し、グローバル・コマンドのリターン・コードは、プロジェクトのすべてのコマンドで有効になります。ローカルとグローバルの有効範囲は、[Make local（ローカルにする）] ボタンと [Make global（グローバルにする）] ボタンで切り替えます。

ページ上部の [Show default messages for unknown return codes（未知のリターン・コードに対してはデフォルトのメッセージを表示する）] ボックスをオンにすると、ボード・アプリケーションからリストにないコードが返された場合に、そのリターン・コードを含んだ標準的なメッセージ・ボックスがポップアップ表示されます。

4.4 プロジェクト・ファイルのインポート

プロジェクトを作成する際は、変数、コマンド、オシロスコープ、レコーダの定義を再利用したり、以前のプロジェクトで作成した定義を観察したりすることができます。[File（ファイル）] / [Import（インポート）] メニュー・コマンドを選択して表示されるダイアログから、さまざまなプロジェクトで定義したオブジェクトを選択し、それらを現在のプロジェクトにインポートすることができます。

インポート手順の最初のダイアログを示したのが図 31 です。元のプロジェクト・ファイルの名前を指定したら、現在のプロジェクトにインポートするプロジェクト・ツリー項目を選択してチェックボックスをオンにします。インポートした項目の作成先となるブロック項目を選択することもできます。

インポートされたツリー項目とともに、変数やアプリケーション・コマンドなど、参照先のオブジェクトもすべて自動的にインポートされます。参照先オブジェクトの作成方法は、各リストの下にあるラジオ・ボタンのオン / オフで指定します。

- [Overwrite existing（既存のオブジェクトを上書きする）] : ツリー項目（オシロスコープなど）に伴ってインポートされるオブジェクト（変数など）がある場合、現在のプロジェクトに同じ名前を持った同じ型のオブジェクト（変数）がないか検索されます。見つかった場合は、インポートしたオブジェクトで上書きされます。
- [Bind to existing（既存のオブジェクトにバインドする）] : 同じ名前のオブジェクトが見つかった場合、上書きせずに、インポートされたツリー項目がそこにバインドされます。
- [Always create new（常に新規作成する）] : 参照先オブジェクトは、現在のドキュメントに既に存在していてもすべて作成されます（その場合、名前が複製されます）。
- [Merge imported root item（インポートされたルート項目をマージ）]:ルート項目をインポートする際、その変数ウォッチの定義を、現在のプロジェクトに含まれているルート項目のウォッチとマージすることができます。このオプションをオフにした場合、ルート項目がインポートされ、標準的なブロック項目として挿入されます。

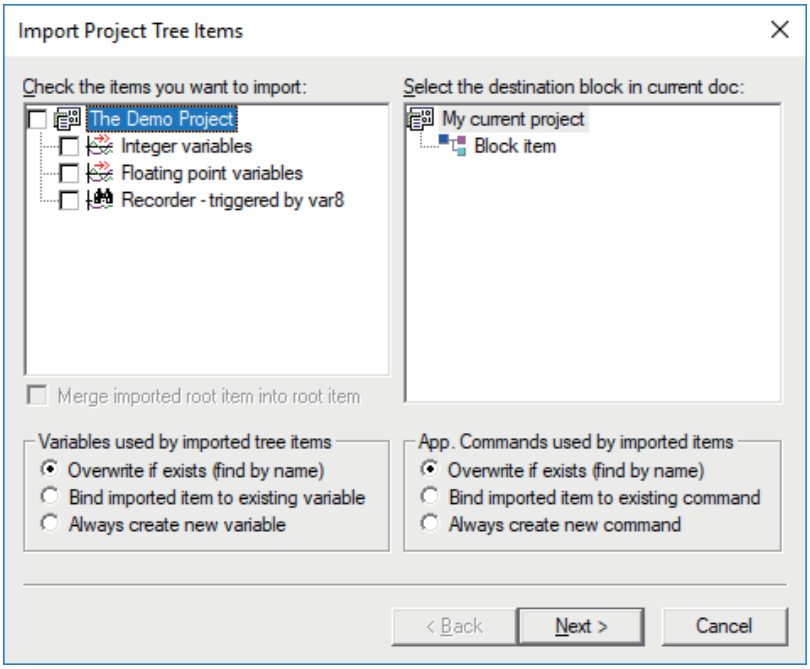

図 31. [Import Project Tree Items（プロジェクト・ツリー項目のインポート)] ウィンドウ

[Next（次へ)] ボタンを押すと、インポート手順の 2 つ目のダイアログが表示され、そこで別途インポートするオブジェクトを選択できます。この 2 つ目のダイアログを示したのが図 32 です。3 つのチェックボックス・リストには、インポート元のプロジェクト・ファイルに見つかったオブジェクトが表示されます。

- 変数:インポートする各変数のチェック･ボックスをオンにします。前出のダイアログ(図31)で選択したツリー項目から参照されている変数をインポートする必要はありません。そのような変数は常に無条件でインポートされます。
- アプリケーション・コマンド：インポートする各アプリケーション・コマンド定義のチェック・ボックスをオンにします。変数オブジェクトと同様、前出のダイアログで選択したブロック・ツリー項目から参照されているコマンドのチェック・ボックスをオンにする必要はありません。
  - [Global return messages（グローバル・リターン・メッセージ)]：このチェック・ボックスをオンにした場合、インポート元のドキュメントからアプリケーション・コマンドのリターン・コードおよびメッセージがインポートされます。
  - [Overwrite existing（既存のオブジェクトを上書きする)]：インポートしたリターン・コードで既存のリターン・コードが上書きされます。
- スティミュレータ：インポートする各変数スティミュレータの横のチェック・ボックスをオンにします。

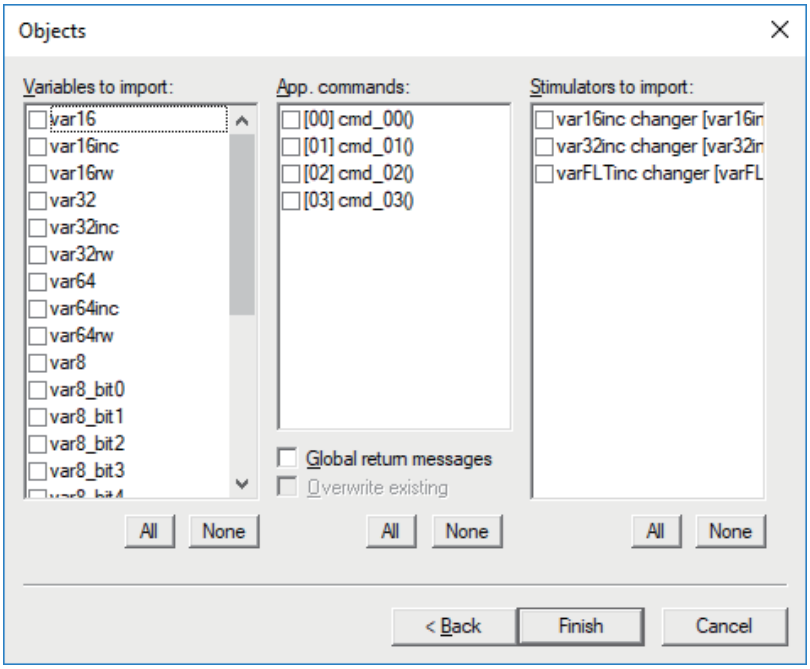

図 32. プロジェクト・オブジェクトのインポート

4.5 メニューの説明

以降のセクションでは、FreeMASTER のメイン・アプリケーション・メニューから利用できるコマンドについて説明します。アイコンが割り当てられている各メニュー・コマンドは、ツールバーからも利用できることに注目してください。ツールバーの方が素早く簡単にアクセスできます。

4.5.1 [File（ファイル）] メニュー

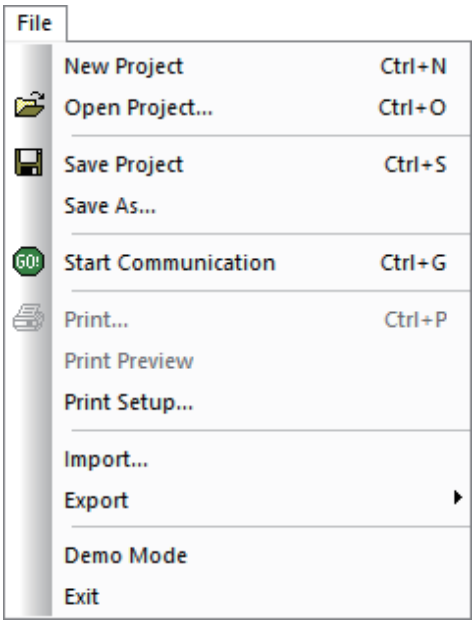

図 33. [File（ファイル）] メニュー

[New Project（新しいプロジェクト）] を選択すると新しい空のプロジェクトが作成されます。[Open Project（プロジェクトを開く）] は、既存のプロジェクト・ファイル（拡張子 *.pmp）が開くときに使用します。

[New Project（新しいプロジェクト）] を選択すると新しい空のプロジェクトが作成されます。[Open Project（プロジェクトを開く）] は、既存のプロジェクト・ファイル（拡張子 *.pmp）が開くときに使用します。

[Save Project（プロジェクトの保存）] では、開いているプロジェクトが現在のファイルに保存されるのに対し、[Save As（名前を付けて保存）] では、現在のプロジェクトに使用する新しいファイル名を指定できます。

通信を開始するには [Start Communication（通信の開始）] を、通信を一時停止するには [Stop Communication（通信の停止）] を選択します。開始時には、TSA シンボル・テーブルとシンボル・ファイルに変更がないか自動的にチェックされます。差違が見つかった場合、新しいアドレス変数を適用するかどうかをユーザーが選択できます。

現在のウィンドウの内容は [Print（印刷）] で出力できます（印刷可能な場合）。現在、印刷がサポートされるのは、Internet Explorer の HTML 説明ページと制御ページのみです。[Print Setup（印刷設定）] を選択すると、標準の印刷設定ダイアログが表示されます。

[Import（インポート）] では、既存のプロジェクト・ファイル（*.pmp または *.pmpx ファイル）から選択したオブジェクト（プロジェクト・ツリーの項目、変数、コマンド、スティミュレータ）を現在のプロジェクトにインポートできます。インポートは、FreeMASTER Lite の設定ファイル（拡張子 *.fmcfg）から行うこともできます。

[Export（エクスポート）] では、現在のプロジェクト設定と変数オブジェクトの定義を FreeMASTER Lite サービスの設定ファイルに保存できます。エクスポート先のファイルが既に存在する場合、エクスポートで変更されるのは、JSON 出力ファイル内の該当するエントリのみです。その他のエントリはそのまま維持されます。

[Demo Mode（デモ・モード）] は、アプリケーションのデモンストレーション・モードのオンとオフを切り替えます。デモ・モードでは、重要なプロジェクト設定を一切変更できません。

アプリケーションを終了するには、[Exit（終了）] を選択します。

4.5.2 [Edit（編集）] メニュー

[Edit（編集）] メニューには、標準的なクリップボード操作コマンド（切り取り、コピー、貼り付け）が表示されます。

オシロスコープ・グラフまたはレコーダ・グラフがアクティブである場合、[Copy Special（拡張コピー）] が有効になります。このコマンドを使用すると、グラフの画像をさまざまな形式やサイズでクリップボードまたはファイルに保存できます。その設定を行うためのダイアログを下図に示します。

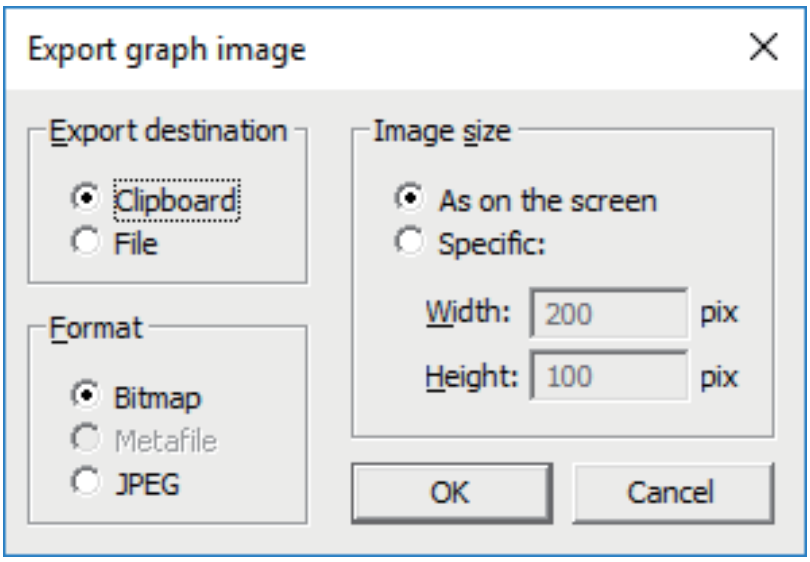

図 34. [Export graph image（グラフ画像のエクスポート）] ダイアログ

4.5.3 [View（表示）] メニュー

[View（表示）] メニューでは、ツールバー、書式設定バー、ステータス・ラインの表示と非表示を選択できます。

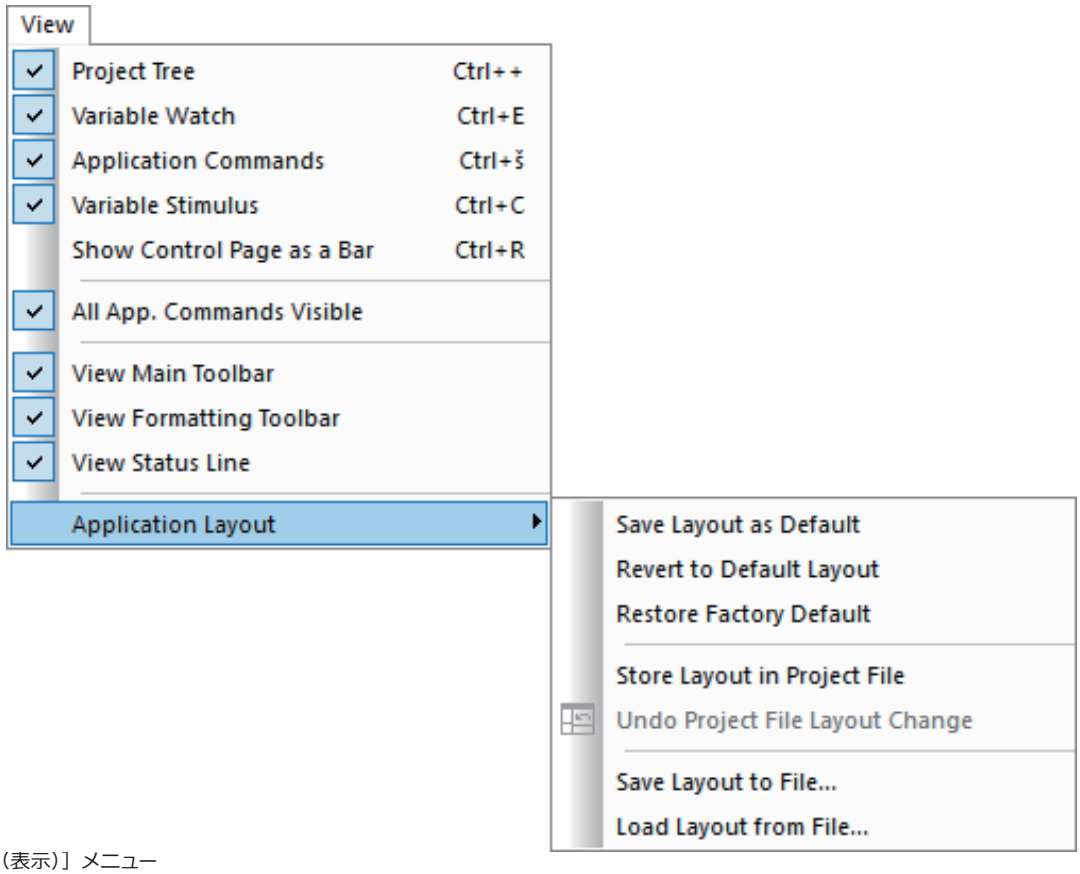

図 35. [View（表示）] メニュー

- [Show Control Page as a Bar（制御ページをバーとして表示する）] では、アプリケーション・ウィンドウのタブ付きのメイン領域から制御ページ・ビューのドッキングを解除し、フローティング・ウィンドウにすることができます。プロジェクト・ツリーや変数ウォッチなどのビューと同様、フローティ
  ング・ウィンドウは、必要に応じてドッキングすることができます。
- [All App.Commands Visible（すべてのアプリケーション・コマンドを表示する）] を選択すると、プロジェクトに定義されているすべてのアプリケーション・コマンドが常時 [Commands（コマンド）] ビューに表示されます。これをオフにした場合は、プロジェクト・ツリーで特定のブロック項目を選択したときに、[Project Block Properties（プロジェクト・ブロックのプロパティ）] で選択したコマンドのみが表示されます。
- [Application Layout（アプリケーション・レイアウト）] では、各種バーやビュー部分の直近のレイアウトを保存したり、復元したりすることができます。
  - [Store Layout in Project File（プロジェクト・ファイルにレイアウトを保存）] を使用すると、ウィンドウ・レイアウトを毎回プロジェクト・ファイルに保存することができます。レイアウトを含んだプロジェクト・ファイルをロードすると、保存済みのウィンドウ・レイアウトが適用されます。
  - [Undo Project File Layout Change（プロジェクト・ファイルのレイアウト変更を取り消す）] は、ウィンドウ・レイアウトが保存されているプロジェクトをロードするユーザーが利用できます。このオプションによってユーザーは、復元されたレイアウトを、最近使用したレイアウトに戻すことができます。
    他のユーザー向けに FreeMASTER プロジェクトを作成する際には、自分がプロジェクトに保存した特定のウィンドウ・レイアウトが確実に適用されるとは限らないことにご注意ください。

4.5.4 [Explorer] メニュー

[Explorer] サブメニューは、詳細ビューに HTML ページ（制御ページなど）が表示されているときに利用できます。このメニューは、Internet Explorer と Chromium レンダリング・モードに共通です。

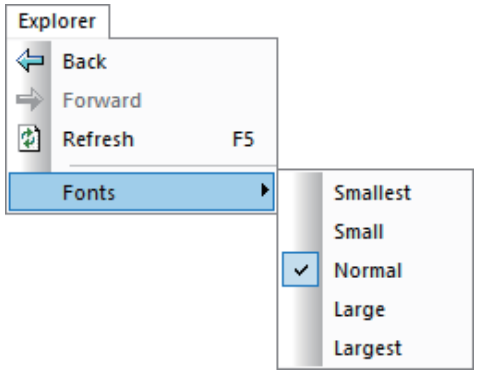

図 36. [Explorer] メニュー

[Back（戻る）]、[Forward（進む）]、[Refresh（最新の情報に更新）] の各項目は、組込みウェブ・ブラウザ・ウィンドウのコマンドを表します。過去にアクセスしたページをたどって移動したり、ページの内容を最新の情報に更新したりする際に使用します。

現在のウィンドウのフォント・サイズは [Fonts（フォント）] で設定します。

4.5.5 [Oscilloscope（オシロスコープ）] メニュー

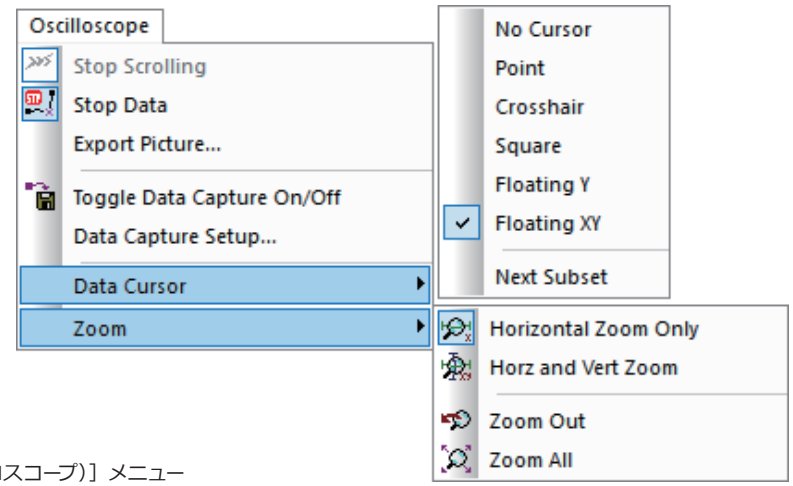

図 37. [Oscilloscope（オシロスコープ）] メニュー

- [Stop Scrolling（スクロールの停止）] と [Stop Data（データの停止）] は、FreeMASTER のオシロスコープ・チャートの変動（スクロール）を停止して、チャートを拡大 / 縮小したりデータ・カーソルでデータ系列を調査したりできるモードに移行するコマンドです。
- 2 つのコマンドの違いは、[Stop Scrolling（スクロールの停止）] では画像が停止されても受信データはチャートの最後に追加されるのに対し、[Stop Data（データの停止）] では受信データも停止される点です。
- [Export Picture（画像のエクスポート）] を選択すると、[Export graph image（グラフ画像のエクスポート）] ダイアログが開くので、そこで形式を指定して画像をクリップボードやファイルにエクスポートすることができます。
- [Data cursor（データ・カーソル）] では、チャートの値を観察する際のデータ・カーソルのスタイルを選択できます。その下位項目である [Next Subset（次のサブセット）] は、次に観察するチャートの線を選択するものです。データ・カーソルの制御にはキーボードの矢印キーを使用することに注意してください。カーソルを配置するには、(カーソルが「手」の形に変化した状態で) グラフの任意のデータ・ポイントをマウスでクリックします。
- [Zoom（ズーム）] サブメニューでは、ズームのモードとコマンドを設定できます。[Horizontal Zoom Only（水平方向ズームのみ）] モードでは、X 軸のみのズームが可能です。[Horzizontal and Vertical Zoom（水平 / 垂直方向ズーム）] では、自由に矩形範囲を選択してズームすることができます。

4.5.6 [Project（プロジェクト）] メニュー

[Project（プロジェクト）] メニューの項目は、プロジェクトの設定にアクセスして変数オブジェクトやコマンド・オブジェクトを定義したり、他のプロジェクト・リソースを管理したりする際に使用します。

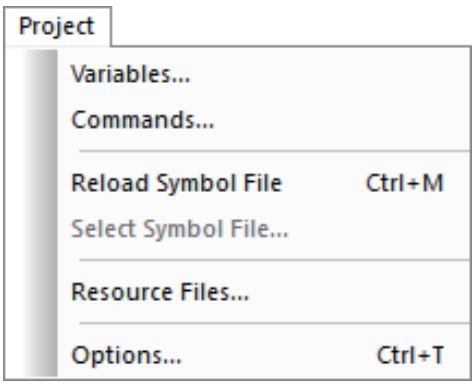

図 38. [Project（プロジェクト）] メニュー

- [Variables（変数）] を選択すると [Variables List（変数リスト）] ダイアログ・ボックスが表示され、そこでプロジェクトのすべての変数を管理したり、新しい変数を作成したり、ロードされたシンボルから変数を一括生成したりすることができます。
- [Commands（コマンド）] を選択すると [Application Commands（アプリケーション・コマンド）] ダイアログ・ボックスが表示され、そこでプロジェクトのコマンドを管理することができます。
- [Reload Map File（マップ・ファイルのリロード）] は、現在選択されている ELF ファイルまたは MAP ファイルから、ボード・アプリケーションの変数の物理アドレスを更新するコマンドです。TSA によってロードされるシンボルは、ボードが接続されるたび自動的に更新されることに注意してください。
- [Select Symbol File（シンボル・ファイルの選択）] は、[Project（プロジェクト）] の [Options（オプション）] で複数の ELF ファイルまたは MAP ファイルを指定した場合に利用できます。どのファイルからシンボルをロードするかを選択できます。
- [Resource Files（リソース・ファイル）] では、プロジェクトで参照されている HTML、ELF、MAP ファイルを確認できます。これは、プロジェクト・ファイルを他のユーザーに配布する際、リソースへのアクセスに使用されているのが相対パスのみであるかどうかを事前に確認する目的で利用できます。絶対パスは、別のホスト・コンピュータでプロジェクトを開いた場合に無効になるので決して使用しないでください。
- [Options（オプション）] を選択すると、プロジェクトのオプションと設定をすべて含んだメイン・ダイアログ・ボックスが表示されます。

4.6 ツールバー

4.6.1 メイン・ツールバー

よく使用するメニュー・コマンドには、メイン・ツールバーから素早くアクセスできます。

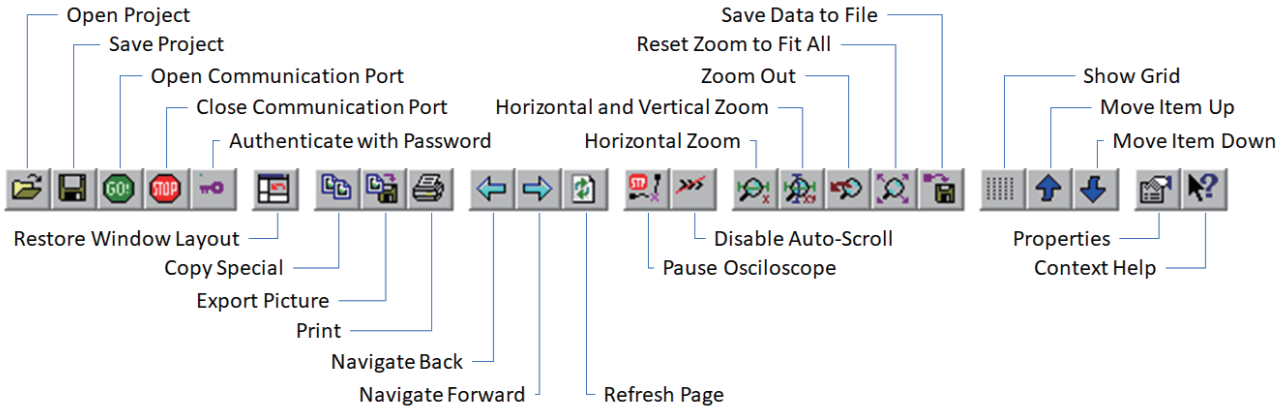

図 39. メイン・ツールバー

4.6.2 書式設定バー

[Watch-Grid（ウォッチ・グリッド）] 部分や [Pipe（パイプ）] 部分など、フォントや色を設定できるビューに入力フォーカスがある場合は書式設定バーを利用できます。

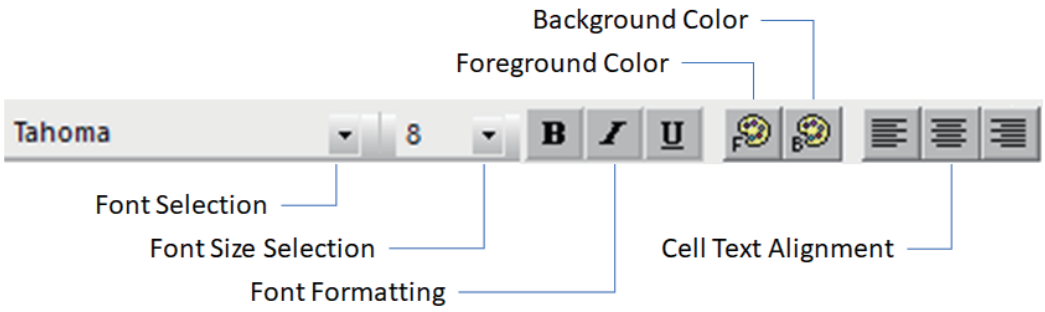
Background Color
Foreground Color
Tahoma
8
B
I
U
Font Selection
Font Size Selection
Font Formatting
Cell Text Alignment

図 40. 書式設定バー

第 5 章
プロジェクトのオプション

アプリケーションとプロジェクトのオプションを設定するには、[Options（オプション）] ダイアログを使用します。[Project（プロジェクト）] メニューの [Options（オプション）] を選択することでアクティブにできます。このダイアログはいくつかのページに分かれていて、それぞれ特定の目的を持った設定が集められています。

5.1 通信

FreeMASTER アプリケーションとターゲット・ボードとの間の通信に関連したパラメータを設定するには、1 つ目のタブを設定します（図 41）。

- [Communication（通信）]
  —[RS-232] : ターゲット・アプリケーションとの接続には、標準のシリアル・インターフェース（3 線式 RS-232 ケーブル）が使用されます。
  — ボードへのアクセスに使用するシリアル・ポートを [Port（ポート）] で選択します。ポートのシステム名（「COM1」など）を指定するか、選択ドロップダウン・リストの横に表示される関連する説明テキストから任意の単語を使用します。一部の USB シリアル・ケーブルでは、割り当て先の COM ポートが接続ごとに変わる可能性があります。説明的な名前を使用すると、接続の手順がわかりやすくなります。
  —[Speed（速度）] : 通信のボーレートを選択または指定します。この速度は、組み込みアプリケーションの設定と一致している必要があります。
  — ホスト PC 上のシリアル通信に関連したタイムアウト値を設定するには、[Timeouts（タイムアウト）] ボタンを押します。
  —[Plug-in Module Interface（プラグイン・モジュール・インターフェース）]：組込みデバイスとの通信は、独立した（カスタム）プラグイン・モジュールによって処理されます。このモジュールは Microsoft COM+ 仕様に準拠していること、また、システムのレジストリ・データベースに登録されていることが必要です。詳細については、プロトコルと通信ライブラリのドキュメントを参照してください。DCOM（または他のネットワーク・プロトコル）、CAN、BDM/JTAG デバッガ・モジュールを介した通信を可能にする一連のプラグインが、FreeMASTER アプリケーションとともにインストールされます。

  ドロップダウン・リストには、ローカル・システムに登録されているプラグイン・モジュールがすべて表示されます。
  —[Connect string（接続文字列）] : プラグイン・モジュールの設定は、内部的に「接続文字列」と呼ばれる文字列形式で保存されます。この文字列は直接指定できるほか、[Build（作成）] ボタンをクリックするとモジュール固有の設定ダイアログが表示されます。
  —[Save settings to project file（プロジェクト・ファイルに設定を保存する）] : このチェック・ボックスをオンにした場合、次回プロジェクトを保存するときに、プロジェクト・ファイル内に通信パラメータが保存されます。次回プロジェクトをロードすると、その通信設定が復元されてデフォルトの代わりに使用されます。

  ご利用のコンピュータ上で有効な通信ポートとオプションが他のコンピュータでも有効であるとは限らないので注意してください。このオプションは慎重に使用する必要があります。

  —[Save settings to registry（設定をレジストリに保存する）]：このチェック・ボックスをオンにした場合、ローカル・コンピュータのレジストリ・データベースに設定が保存されます。次回アプリケーションを起動したときには、これらの設定がデフォルトで使用されます。

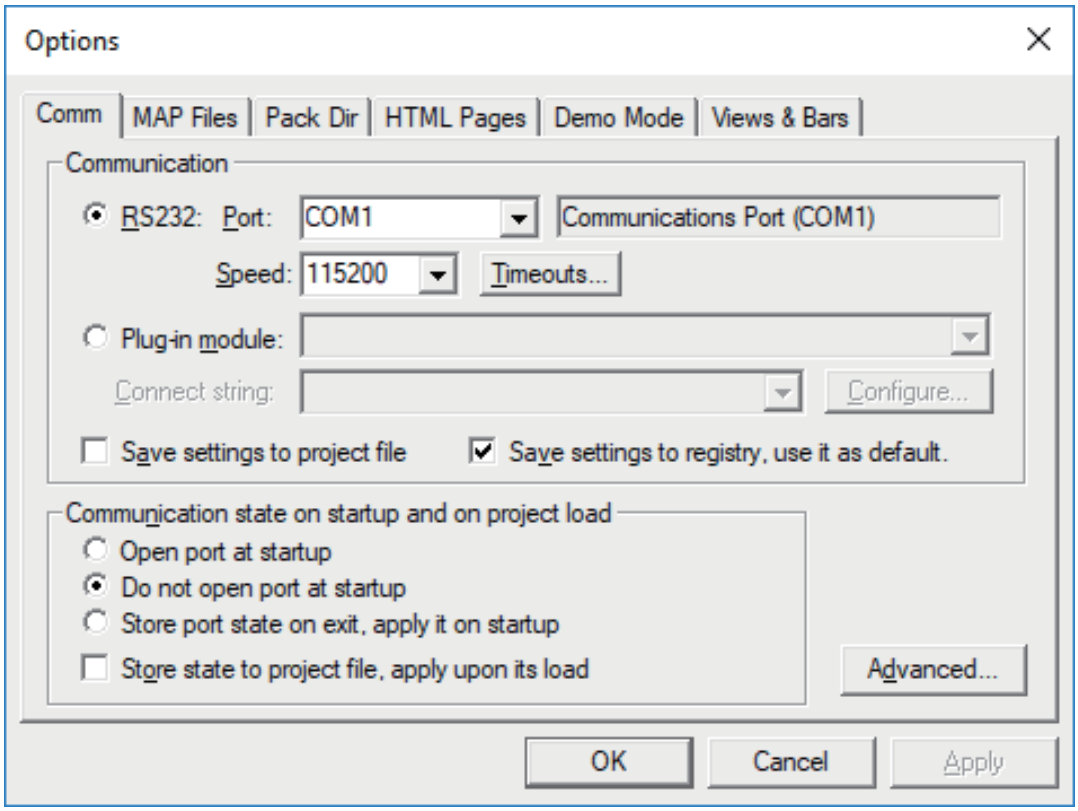

図 41. 通信オプション

注意

FreeMASTER のインストール・パックでは、FreeMASTER Remote Server アプリケーションとの通信を処理する 2 つの通信用プラグイン・モジュールが配布されます。両者の違いは、サーバへの接続に使用されるプロトコルにあります。1 つは Microsoft DCOM リモート・プロシージャ・コール・インターフェースを使用するもので、もう 1 つは標準の HTTP テキスト・ベースのプロトコルを使用するものです。リモート・サーバとの間で DCOM ベースの通信を使用する場合、ネットワーク経由での接続時にセキュリティの問題を考慮する必要があります。ローカル側とリモート側の双方で DCOMCNFG ユーティリティを実行して、システム・レベルで DCOM を適切に設定する必要があります。リモート側にもリモート・サーバが適切にインストールされている必要があります。リモート・コンピュータとの接続に問題がある場合は、ローカル・ネットワークの管理者にお問い合わせください。

- [Communication state（通信の状態）]：アプリケーションの起動時に通信ポートを開くかどうかは、3 つのオプションのいずれかを選択することで設定できます。
  - [Open port at start up（起動時にポートを開く）]
  - [Do not open port at start up（起動時にポートを開かない）]
  - [Store port status on exit, apply it on start up（終了時にポートの状態を保存し、起動時にそれを適用する）]
  - [Store status to a project file, apply it on its load（プロジェクト・ファイルに状態を保存してそのロード時に状態を適用する）]：このチェック・ボックスをオンにした場合、次回プロジェクトを保存するときに、通信ポートの状態がプロジェクト・ファイルに保存されます。その状態が、次回プロジェクトをロードしたときに復元されます。
- 通信スレッドの優先順位設定を含んだダイアログを開くには、[Advanced（詳細設定）] ボタンを押します。ただし通常、このような設定を変更する必要はありません。

5.2 シンボル・ファイル

プロジェクトの変数を定義する際に、ターゲット・メモリ・ロケーションの物理アドレスを直接的な 16 進数値ではなくシンボル名で指定した方が都合がよいケースがよくあります。シンボル・テーブルは、標準の ELF デバッグ形式であれば、組み込みアプリケーションの実行ファイルから直接ロードすることができます。他のケースでは、リンカによって生成されたテキスト MAP ファイルをロードし、解析してシンボル情報を得ることもできます。

プロジェクトでは、シンボル・テーブルを含んだ複数のファイルを指定できます。後から、[Project（プロジェクト）] メニューの [Select Symbol File（シンボル・ファイルの選択）] を選択することで、別のファイルに基づく異なるシンボル・テーブルに切り替えることができます。たとえば、評価ボードでテストする組み込みアプリケーションに関して、RAM メモリから実行するコード用とフラッシュ・メモリから実行するコード用の 2 つのシンボル・ファイルをプロジェクトで指定できます。

注意

アプリケーションの実行ファイルまたはシンボル・ファイルをロードすることに加えて、TSA（Target-Side Addressing）と呼ばれる機能をターゲット MCU ドライバで有効にすることができます。アプリケーション・プログラマは、TSA テーブルという、FreeMASTER に表示されるデータ型と変数を記述するテーブルを定義することができます。マイクロコントローラ・ボードが接続されると、FreeMASTER は自動的に TSA テーブルを読み取ってその情報を使用します。TSA アプリケーションとシンボル・ファイルの両方が使用されている場合、TSA 情報が優先されます。TSA を使用する大きな利点として、安全性とセキュリティ上の理由が挙げられます。TSA テーブルによって記述された変数は読み取り専用にできるため、FreeMASTER が変数の書き込みを試みたとしても、その値はターゲット MCU 側で能動的に拒否されます。また、どの TSA テーブルにも記述されていない変数は非表示となり、読み取り専用アクセスに対しても保護されます。

下図は、プロジェクトの [Options（オプション）] ダイアログの [MAP Files（MAP ファイル）] タブを示しています。

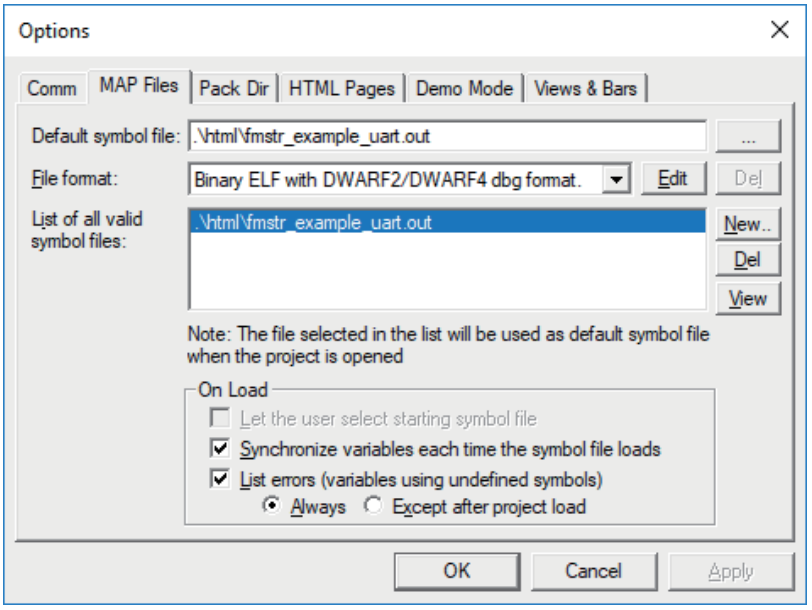

図 42. シンボル・ファイルのオプション

- [Default symbol file（デフォルトのシンボル・ファイル）]:デフォルトでロードするシンボル・ファイルの名前を指定します。参照ボタン（...）を押し、ファイルを開くための標準のダイアログで目的のファイルを選択してください。
- [File format（ファイル形式）] : ファイル形式を選択します。
  - [Standard binary ELF（標準バイナリ ELF）] : ELF ファイルの場合は、このオプションを選択します。
  - [Hiware MAP File 509（Hiware MAP ファイル 509）] : HiWare Smart Linker v5.0.9 の場合は、このオプションを選択します。
  - [Define new Regular Expression-based parser（新しい正規表現ベースのパーサーを定義する）] : 正規表現パターン・マッチングを利用して新しいテキスト・ファイル・パーサーを定義するには、このオプションを選択します。「正規表現ベースの MAP ファイル・パーサー」を参照してください。
- [List of valid symbol files（有効なシンボル・ファイルのリスト）] : プロジェクトで定義されているシンボル・ファイルのリスト。リストの管理は、[New（新規作成）] ボタンと [Del（削除）] ボタンで行います。
- [View（表示）] : 選択ファイルから解析されたシンボル・テーブルを表示します。

注意

シンボル・ファイルへのパスは、いくつかの形式で指定できます。ローカル・ディスク上の絶対パス、プロジェクト・ファイルが格納されているディレクトリへの相対パス、現在の「パック」ディレクトリへの相対パスを使用できます。他のコンピュータにプロジェクトを展開する場合は、後者が最も適しています。詳細については、「プロジェクト・ファイルへのリソース・ファイルのパック」を参照してください。

- [On Load（ロード時）] パネルのオプションは、プロジェクトのロード後、シンボル・ファイルに関して実行されるアクションを制御します。
  - [Let the user select the symbol file（ユーザーがシンボル・ファイルを選択できるようにする）]：このチェック・ボックスをオンにした場合、プロジェクトのロード時に最初のシンボル・ファイルを選択するように求められます。
  - [Synchronize variables each time the map file loads（MAP ファイルがロードされるたびに変数を同期する）]：このチェック・ボックスをオンにした場合、プロジェクトのロード後に新しいシンボル・アドレスで変数が自動的に更新されます。
  - [List errors（エラーをリストする）]：このチェック・ボックスをオンにした場合、ロード済みのシンボル・テーブルに存在しないシンボルを使用している変数がすべて専用ダイアログ・ボックスにリストされます。破損した変数を編集するか、または別のシンボル・ファイルを選択できます。

5.2.1 正規表現ベースの MAP ファイル・パーサー

コンパイラまたはリンカが ELF 出力形式をサポートしておらず、ビルトインの HiWare パーサーが MAP ファイルを解析できない場合、正規表現パターンを使用して内部 MAP ファイルの構造を記述します。

正規表現パターン・マッチングの理論の解説は、本書の範囲を超えています。ただし、このテクノロジの利点をわかりやすく説明するために簡単な例を紹介します。この例では、NXP DSC56F800E プロセッサ・ファミリ用のコンパイラおよびリンカによって生成された xMap ファイルのパーサーを定義します。

以下に示したのは、グローバル・シンボルの長さを表す xMap 行の例です。

```
00002320 00000001 .bss Flength (bsp.lib pcmasterdrv.o )
```

```
00002321 00000001 .bss Fpos (bsp.lib pcmasterdrv.o )
```

まず、行の形式を正規表現パターンで記述します。パターン

```
[0-9a-fA-F]+\s+[0-9a-fA-F]+\s+\S+\s+F\w+
```

は、以下のように解釈できます。

- 1 桁以上の 16 進数：[0-9a-fA-F]+
  - その後に続く 1 つ以上のスペース（または他の空白文字）：¥s+
  - その後に続く 1 桁以上の 16 進数：[0-9a-fA-F]+
  - その後に続く 1 つ以上の空白文字：¥s+
  - その後に続く一連の非空白文字：¥S+
  - その後に続く 1 つ以上の空白文字：¥s+
  - その後に続く文字 F
  - その後に続く 1 つ以上の英数字（単語）文字：¥w+

シンボル・パラメータを表すサブ・パターンを識別するには、丸括弧で囲んで指定します。

```
([0-9a-fA-F]+)\s+([0-9a-fA-F]+)\s+\S+\s+F(\w+)
```

次に、各シンボル値のサブ・パターンのインデックスを識別します。

- 1 つ目のサブ・パターンは、シンボルのアドレス（インデックス 1）に対応します。
- その次のサブ・パターンは、シンボルのサイズ（インデックス 2）に対応します。
- その次のサブ・パターンは、シンボルの名前（インデックス 3）に対応します。

ここまでに準備したすべてのパラメータを正規表現ダイアログに入力します（下図）。

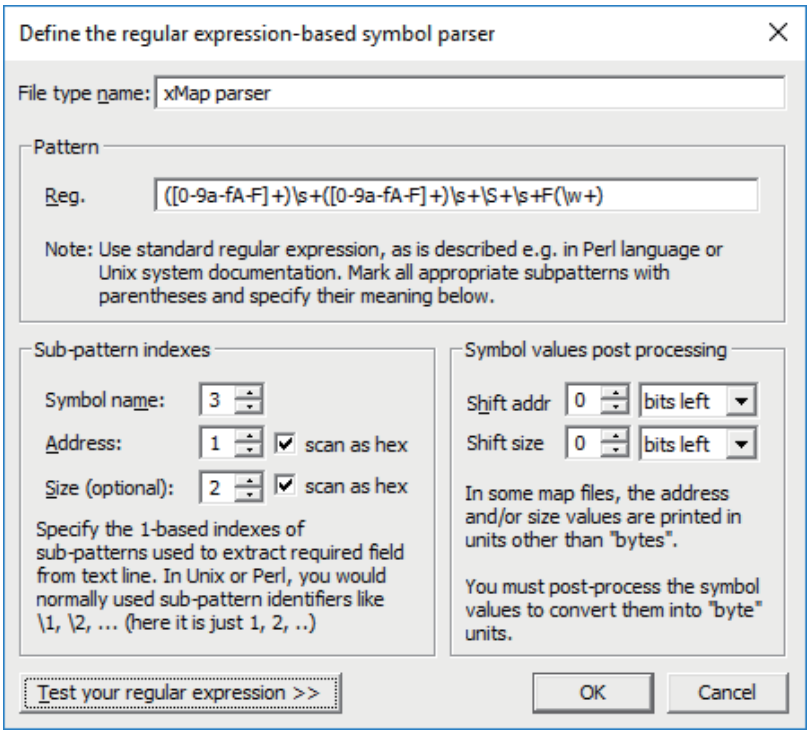

図 43. 正規表現を利用した xMap ファイル・パーサー

[Test your regular expression（正規表現のテスト）] ボタンを押すと、ダイアログが縦に展開してテスト・パネルが表示されます（下図）。xMap ファイルから 1 行を切り取って入力行フィールドに貼り付け、[Execute reg. expression（正規表現の実行）] ボタンを押します。

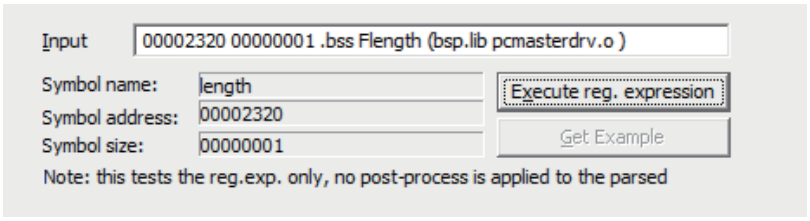

図 44. 正規表現のテスト

上の図のように、[Symbol name（シンボル名）]、[Symbol address（シンボルのアドレス）]、[Symbol size（シンボルのサイズ）] の各出力フィールドが表示されるはずです。エラーが発生した場合は、正規表現の機能に必要な Internet Explorer 5.5（以上）がコンピュータにインストールされていることを確認してください。

最後に、シンボルの後処理パラメータでサイズ値の 1 ビット・シフトを設定します。これを行う理由は、xMap ファイルにはシンボルのサイズがワード単位（16 ビット）で出力されますが、この値はバイト単位である必要があるためです。

注意

作成されたすべての正規表現パーサーは、後から使用できるよう、自動的にプロジェクト・ファイルとローカル・コンピュータのレジストリ・データベースに保存されます。

5.3 プロジェクト・ファイルへのリソース・ファイルのパック

プロジェクトには、往々にして多くの異なるファイルのデータが使用されます。たとえば、プロジェクト・ツリー項目に対して表示される各 HTML ページとそのページに使用されている各画像は、別々のファイルに格納されます。そのことが原因で、プロジェクトを別のコンピュータに展開または配布する際に問題が生じる場合があります。[Pack Dir（パック・ディレクトリ）] のオプション（下図）を使用すると、プロジェクト・ファイルの保存時に、必要に応じてリソース・ファイルをプロジェクト・ファイルにパックすることができます。

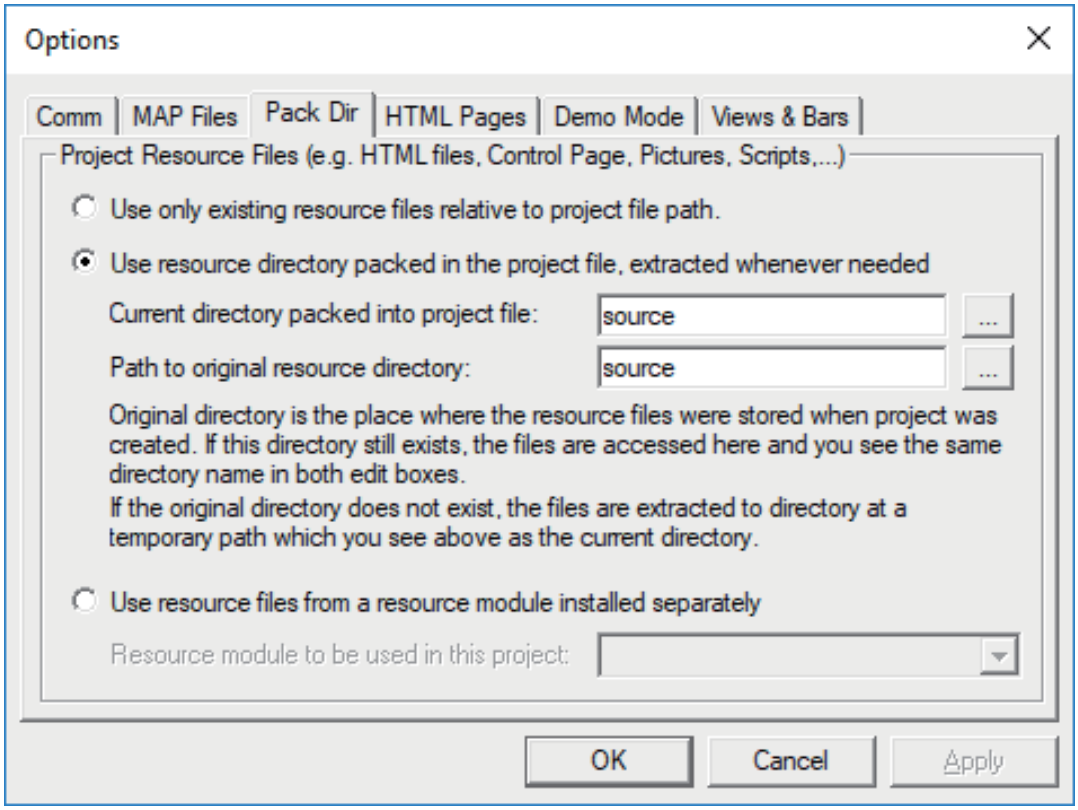

図 45. パック・ディレクトリのオプション

リソース・ファイルをプロジェクト・ファイルにパックするには、これらすべてのファイルを 1 つのディレクトリまたはディレクトリ構造に入れて、そのディレクトリのパスを、[Pack Dir（パック・ディレクトリ）] ページの両方のフィールドに指定します。参照ボタン（...）を押すと、ディレクトリを開くための標準のダイアログでディレクトリを探して選択できます。ディレクトリのパスは、プロジェクト・ファイルの保存先ディレクトリを基準とした相対パスで指定する必要があります。

パックされたファイルを含んだプロジェクトが次回アプリケーションによってロードされるときは、元のパス、つまり [Pack Dir（パック・ディレクトリ）] ページの 2 つ目の項目がチェックされます。元のパスが見つからない場合、またはその中のいずれかのファイルが、当初プロジェクトにパックされていたファイルと異なる場合、システムの一時領域に一時ディレクトリが作成されて、パックされたファイルがそのディレクトリに抽出されます。以後、その一時ディレクトリが、残りのリソース・ファイル用のデフォルトのディレクトリとして設定されます。FreeMASTER の HTML ファイルやシンボル・ファイルに対するパスは、[Pack Dir（パック・ディレクトリ）] ダイアログで指定したディレクトリを基準とする相対パスで指定することが極めて重要です。

現在一時ディレクトリが使用されている場合、[Pack Dir（パック・ディレクトリ）] ページには一時ディレクトリのパスが表示されます。ただし、元のリソース・ディレクトリのパスは引き続き記憶され、将来プロジェクトを開くたびにその存在がチェックされます。

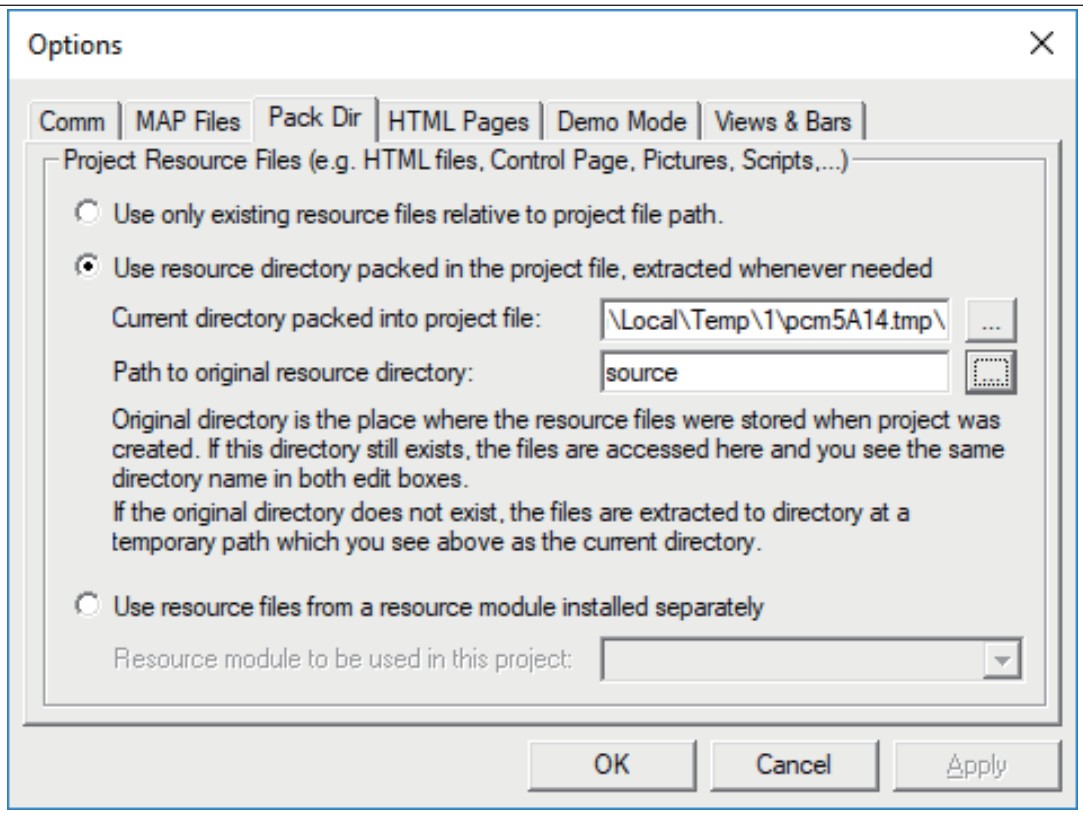

図 46. パック・ディレクトリのオプション（使用中）

5.3.1 リソース・ファイル・マネージャ

外部リソースやシンボル・ファイルを多く使用した複雑なプロジェクトを展開する際は、すべてのファイル・パスが正しいこと、そして適切な相対形式になっていることを事前に確認することをお勧めします。そして異なるホスト上の一時ディレクトリにファイルが抽出される際は、そのパスの相対形式が適切に評価されなければなりません。

リソース・ファイル・マネージャは、［Project（プロジェクト）］/［Resource Files（リソース・ファイル）］メニューから起動できます。このウィンドウの一般的な外観を下図に示します。

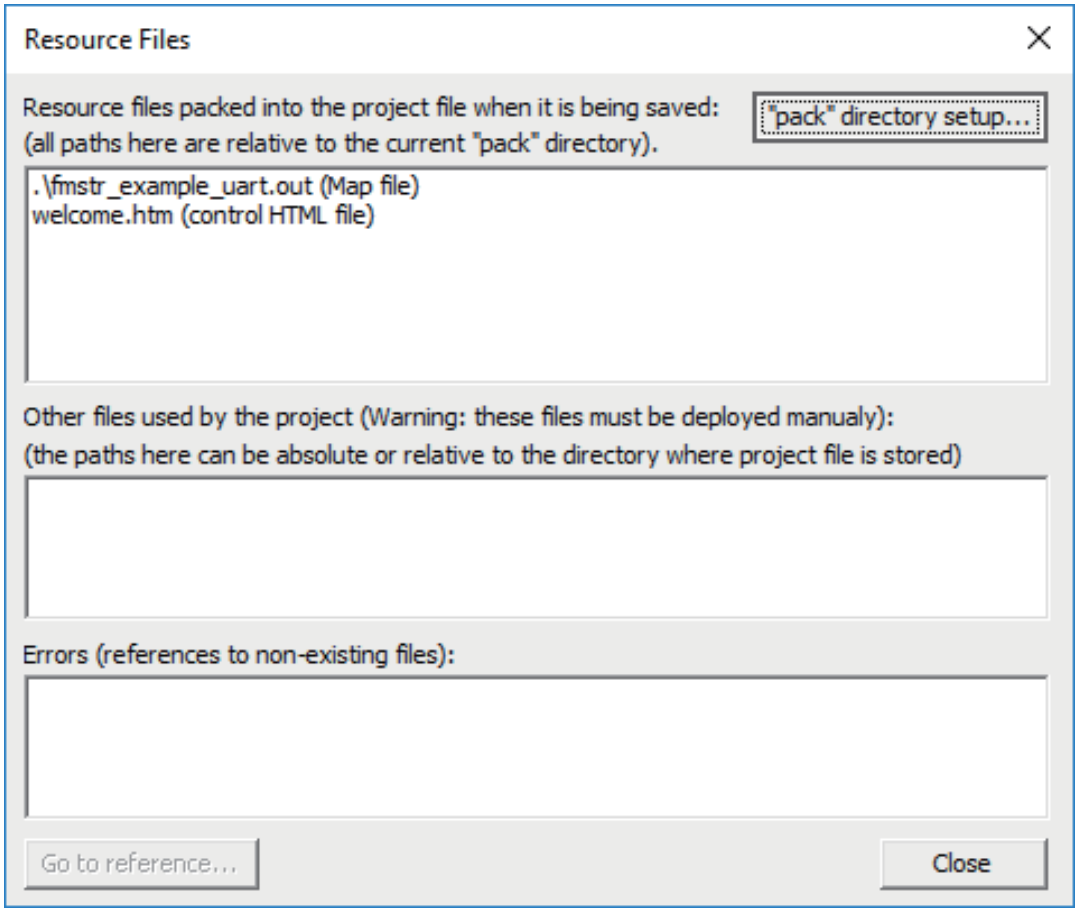

図 47. リソース・ファイル・マネージャ

このダイアログには、プロジェクトによって直接参照されるすべての外部ファイルが表示されます。HTML ページから間接的に参照される他のファイル（画像など）も存在する場合もあることに注意してください。ファイルがパック・ディレクトリに存在するかどうか、また、それらが HTML ページから相対パスで参照されているかどうかを確認する必要があります。

- このダイアログの1つ目のリストには、現在のパック・ディレクトリに置かれているファイルが表示されます。プロジェクト・ファイルにはその保存時に、これらのファイルが適切に格納されます。他のホストでは、ファイルが一時領域に抽出されます（必要な場合）。
- このダイアログの 2 つ目のリストには、プロジェクトで正しく使用されているものの、パック・ディレクトリの外部に置かれているファイルが表示されます。これらのファイル名は絶対パスで指定されるか、プロジェクトの格納先ディレクトリを基準とした相対パスで指定されます。ただし、ファイルは自動的にプロジェクト・ファイルに格納されるわけではありません。手動で展開する必要があります。
- 3 つ目のリストには、存在しないファイルへの参照が表示されます。
- ["Pack" directory setup（「パック」ディレクトリの設定）]：このボタンを押すと、プロジェクトの［Options（オプション）］ダイアログが開き、パック・ディレクトリの場所やその他の設定を再定義することができます。
- ［Go to reference（参照に移動）］：選択したファイルの参照に使用されるダイアログを開きます。これは、プロジェクト・ツリー項目の場合は［Properties（プロパティ）］ダイアログに、共有 HTML ファイルやシンボル・ファイルの場合はプロジェクトの［Options（オプション）］ダイアログになります。

5.4 HTML ページ

プロジェクトでは、プロジェクト・ツリーで選択した項目に関連したコントロールや説明が、HTML ページを使用して表示されます。これらのページのパスまたは URL は、各ツリー項目のプロパティ・ダイアログで指定されます。［HTML Pages（HTML ページ）］オプション・タブ（下図）では、アプリケーションのメイン・ウィンドウに表示される一般的な HTML ファイルのパスを指定できます。

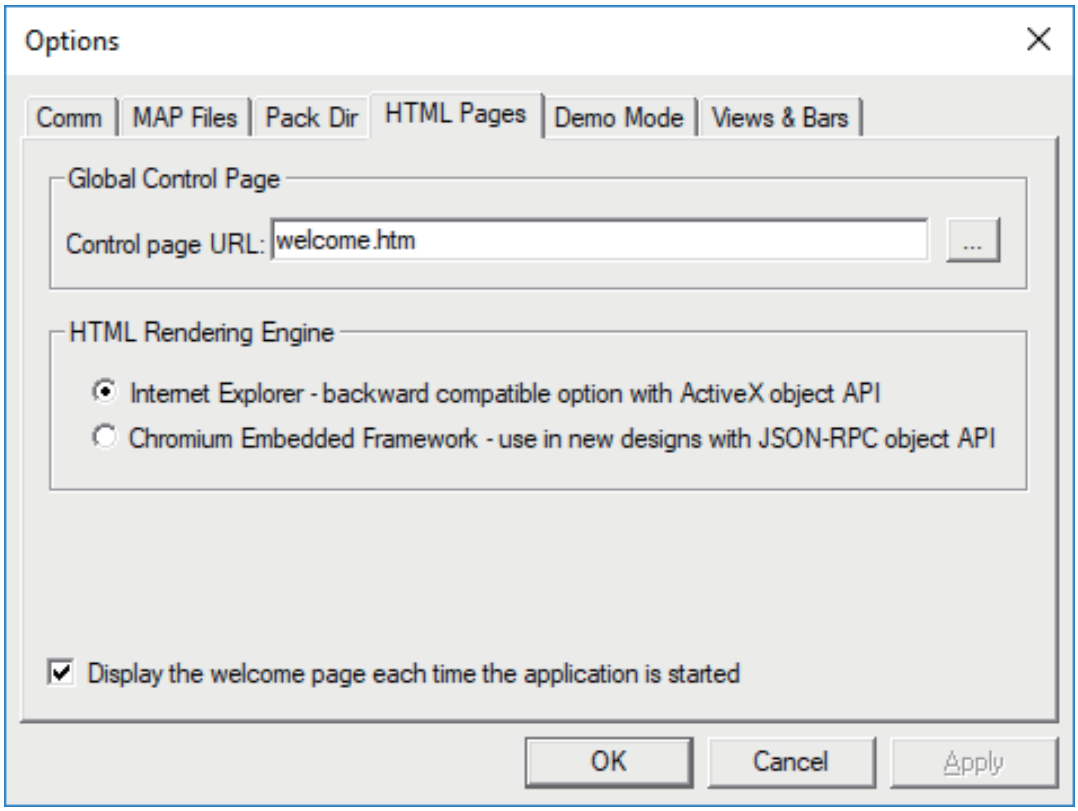

図 48. [HTML pages（HTML ページ）] のオプション

[Control page URL（制御ページの URL）] に、FreeMASTER プロジェクトの全体的な制御ページとして使用する HTML ファイルの名前を設定します。制御ページは、FreeMASTER のメイン・ウィンドウからいつでもアクセスできます。通常、ターゲット・マイクロコントローラ・アプリケーションに関連したカスタム・ユーザー・インターフェースが表示される場所として使用されます。

通常、制御ページにはグラフィカルな制御要素（プッシュ・ボタン、ゲージなど）が含まれ、その各要素のロジックや振る舞いを実装する JavaScript コードが埋め込まれています。FreeMASTER では、以下の 2 つの HTML レンダリング・エンジンがサポートされています。

- [Internet Explorer]：このオプションは、以前の FreeMASTER リリースとの下位互換性を確保します。FreeMASTER アプリケーション・オブジェクトと通信してターゲット・マイクロコントローラのリソースにアクセスするためには、スクリプトで ActiveX テクノロジを使用する必要があります。
- [Chromium Embedded Framework]：このオプションは FreeMASTER 3.0 以降で提供され、HTML ページのレンダリングと JavaScript コードの実行に組込みの Chromium ビューが使用されます。FreeMASTER アプリケーション・オブジェクトとの通信には JSON-RPC プロトコルが使用されます。

「制御ページ」と「HTML とスクリプティング」の関連情報を参照してください。

5.5 デモ・モード

FreeMASTER の機能の重要な役割は、ターゲット・ボード・アプリケーションのデモンストレーションと説明です。作成したデモンストレーション・プロジェクトが不注意で変更されることは絶対に避けなければなりません。改変を防ぐために、プロジェクトの作成者は、プロジェクトをデモ・モードに切り替えることで変更を禁止できます。[Demo Mode（デモ・モード）] タブの詳細については、下図を参照してください。

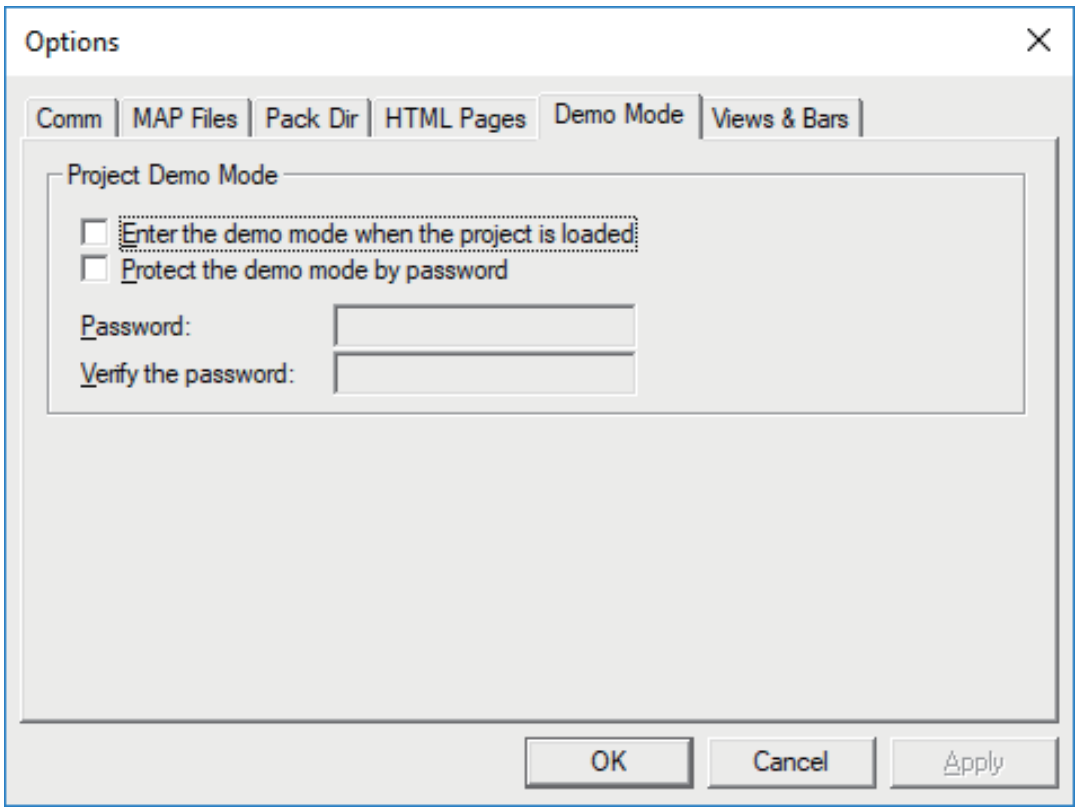

図 49. [Demo mode（デモ・モード）] のオプション

[Enter the Demo Mode (デモ・モードに移行する)] チェック・ボックスをオンにした場合は、プロジェクトのロード後、自動的にデモ・モードが作動します。デモ・モードは、[File (ファイル)] メニューから [Demo Mode (デモ・モード)] を選択することにより手動で開始できます。デモ・モードの終了は、パスワードで保護することができます。

デモ・モードでは、プロジェクト・ツリー項目のプロパティを変更したり、ツリー項目を追加 / 削除したり、プロジェクトのオプション（通信ページのオプションを除く）を変更したりすることはできません。

デモ・モードを終了しようとすると、警告メッセージ（下図）が表示され、デモ・モードがパスワードで保護されている場合はパスワードの入力が求められます。

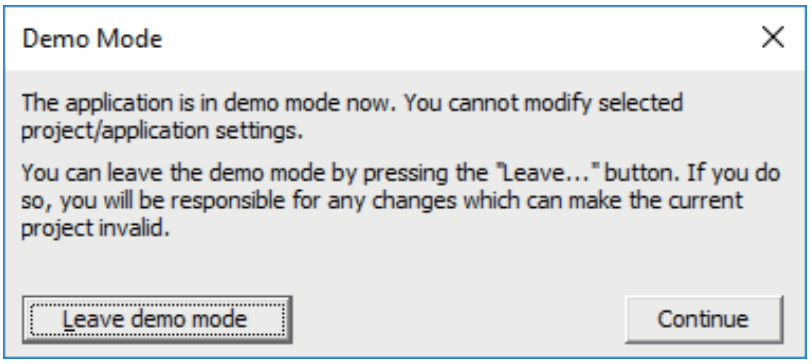

図 50. デモ・モード終了を確認するダイアログ

第 6 章
HTML とスクリプティング

FreeMASTER バージョン 2.x では、すべての HTML ページのレンダリングに、標準の Microsoft Internet Explorer コンポーネントが使用されます。FreeMASTER 3.0 では、レンダリング・エンジンを Internet Explorer から Chromium に切り替える選択肢が加わりました。

どちらの方法も、ユーザーは JavaScript コードで拡張された HTML ページを作成して、JavaScript コードから FreeMASTER の機能にアクセスすることができます。このアプローチを使用することで、カスタムのロジック、データ処理、グラフィカル・コントロールを備えた強力なユーザー・インターフェースを作成できます。このユーザー・インターフェースは FreeMASTER のメイン・ウィンドウに直接置くことができるので、エンド・ユーザーからは FreeMASTER アプリケーションに一体化しているように見えます。

Internet Explorer の利点：

- 下位互換性があるため、以前のバージョン（2.x）の FreeMASTER でも正しく動作します。
- ActiveX オブジェクトがサポートされるため、お使いのグラフィカル・デザインにサード・パーティのオブジェクトが利用されている場合には唯一の選択肢となります。
- FreeMASTER ActiveX インターフェースとのやり取りは同期通信であるため、スクリプトの設計が単純で済みます。一方、ActiveX メソッドの完了を同期的に待機することが原因で、ユーザー・インターフェースの応答が鈍くなることがあります。

Chromium エンジンの利点：

- 非同期 JavaScript プログラミングなど、最新のスクリプティング技術に対応しています。
- FreeMASTER 3.0 と FreeMASTER「Lite」のどちらとも、JSON-RPC プロトコルおよび完全非同期の Promise ライクなインターフェースを使用して通信することができます。
- ページの開発とテストは、強力な JavaScript デバッガを備えたスタンドアロンの Chrome ブラウザで行うことができます。

以降の内容は主に、HTML ページのデザインとスクリプティングの経験がある制御ページの作成者を対象としています。

6.1 特殊な HTML ハイパーリンク

FreeMASTER 環境に表示される HTML ページを作成する際は、特殊なハイパーリンクを使用できます。それによってユーザーは、プロジェクト・ツリー内を移動したり、特定のアプリケーション・コマンドを呼び出したりすることができます。

- 「HREF=pcmaster:selitem:itemname:tabname」と指定することで、項目を手動で選択した場合とまったく同様に、「itemname」という名前のプロジェクト・ツリー項目を選択し、選択されたタブを詳細ビューに表示し、他のビューの内容を設定します。
- 「tabname」に使用できる識別子は以下のとおりです。
  — ctl：[Control Page（制御ページ）] タブ
  — blk：[Algorithm Block Description（アルゴリズム・ブロックの説明）] タブ
  — info：[Current Item Help（現在の項目のヘルプ）] タブ
  — osc：[Oscilloscope（オシロスコープ）] タブ
  — rec：[Recorder（レコーダ）] タブ

アプリケーション・コマンドを呼び出すハイパーリンクの例を以下に示します。

- 「HREF=pcmaster:cmdw:cmdname(arguments)」と指定すると、「cmdname」アプリケーション・コマンドを送信して、ターゲット・アプリケーションがそれを処理するまで待機します。
- 「HREF=pcmaster:cmddlg:cmdname(arguments)」と指定すると、「cmdname」アプリケーション・コマンドの[Send Application Command（アプリケーション・コマンドの送信）] ダイアログを表示します。指定した引数の値は、ダイアログの該当するフィールドに入力されます。
- 「HREF=pcmaster:cmd:cmdname(arguments)」と指定すると、「cmdname」アプリケーション・コマンドを送信します。コマンドの引数の値は、丸括弧で囲んで指定できます。デフォルト値が定義されている引数は省略できます。

6.2 FreeMASTER ActiveXインターフェース

FreeMASTER オブジェクトは、FreeMASTER アプリケーションを起動するたびにシステム・レジストリに登録されます。そのクラス ID（CLSID）は以下のとおりです。

```
{48A185F1-FFDB-11D3-80E3-00C04F176153}
```

レジストリ名は「MCB.PCM.1」です。バージョンに依存しない名前は「MCB.PCM」です。

FreeMASTER の関数は、任意の HTML コードから FreeMASTER ActiveX コントロール経由で呼び出すことができます。FreeMASTER ActiveX コントロールをクラス ID 番号（下の例を参照）で HTML コードに挿入し、サイズ（高さと幅）をゼロに設定してオブジェクトを非表示にします。

```
<object name="freemaster" width="0" height="0" classid="clsid:48A185F1-FFDB-11D3-80E3-00C04F176153">
```

FreeMASTER ActiveX コントロールには、以下の関数が用意されています。

- GetAppVersion：アプリケーションのバージョンを取得します。
- OpenProject：指定されたプロジェクト・ファイルを開きます。
- SaveProject：現在のプロジェクトを保存するか、[Save As（名前を付けて保存）] ダイアログを開きます。
- StartStopComm、StartComm、StopComm：通信ポートを開く、または閉じます。
- IsCommPortOpen および IsBoardDetected：接続状態を調べる際に使用できます。
- GetHtmlDocument：FreeMASTER ウィンドウから HTML DOM オブジェクトを取得します。
- SendCommand：FreeMASTER で定義されているコマンドを送信します。
- SendCommandDlg：コマンド・ダイアログを開いて、FreeMASTER で定義されているコマンドを送信します。
- ReadVariable および ReadMultipleVariables：FreeMASTER で定義されている変数から値を読み取ります。
- WriteVariable および WriteMultipleVariables：FreeMASTER で定義されている変数に値を書き込みます。
- ReadMemory および「ReadMemoryHex」：指定された場所からメモリ・ブロックを読み取ります。
- 「ReadIntArray」、「ReadUIntArray」、「ReadFloatArray」、「ReadDoubleArray」：指定された場所から数値の配列を読み取ります。
- 「WriteIntArray」、「WriteUIntArray」、「WriteFloatArray」、「WriteDoubleArray」：指定された場所に数値の配列を書き込みます。
- 「ReadIntVariable」、「ReadUIntVariable」、「ReadFloatVariable」、「ReadDoubleVariable」：指定された場所から 1 つの値を読み取ります。
- 「WriteIntVariable」、「WriteUIntVariable」、「WriteFloatVariable」、「WriteDoubleVariable」：指定された場所に 1 つの値を書き込みます。
- GetCurrentRecorderData：現在レコーダ・チャートに表示されているデータを取得します。
- GetCurrentRecorderSeries：現在表示されているレコーダ・チャートから 1 つのデータ系列を取得します。
- StartCurrentRecorder：現在表示されているレコーダを開始します。
- StopCurrentRecorder：現在表示されているレコーダを停止します。
- GetCurrentRecorderState：現在のレコーダの状態コードとテキストを取得します。
- LocalFileOpen、LocalFileClose：テキスト・ベースの一時ストレージに使用されるローカル・ファイルを開いたり閉じたりします。
- LocalFileWriteString：テキスト・ベースのデータをファイルに書き込みます。
- LocalFileReadString：テキスト・ベースのデータをファイルから読み取ります。
- GetSymbolInfo、GetStructMemberInfo、GetAddressInfo：ターゲット・アプリケーションからシンボルに関する情報を取得します。
- DefineSymbol、DeleteAllScriptSymbols：ターゲット・アプリケーションから得られるシンボルの情報に対して別途操作を実行します。
- SubscribeVariable、UnSubscribeVariable：変数の値が変化したときにスクリプトが通知を受け取るようにします。
- SelectItem：FreeMASTER のプロジェクト・ツリー・ビューからプロジェクト項目を選択します。
- DefineVariable、DefineOscilloscope、DefineRecorder、DefinePipe、DefineArrayViewer、DefineWatchBlock:FreeMASTER の変数オブジェクトやオシロスコープ / レコーダのグラフを動的に作成または変更する際に使用できる複雑な関数。
- 「GetVariableInfo」、「GetRecorderDefinition」、「GetOscilloscopeDefinition」、「GetPipeDefinition」、「GetArrayViewerDefinition」、「GetWatchBlockDefinition」：プロジェクト項目についての包括的な情報を取得する際に使用できます。
- IsBoardWithActiveContent、EnumHrefLinks、EnumProjectFiles：TSA Active Content 項目を列挙する際に使用できます。TSA および Active Content の詳細については、FreeMASTER シリアル・ドライバのドキュメントを参照してください。
- PipeOpen、PipeClose、「PipeWrite」など、各種の「Pipe」関数：ターゲット・ボードとのロスレス通信に使用します。
- EnumVariables、EnumSymbols：現在のプロジェクトにロードされている変数とシンボルを列挙します。

- GetDetectedBoardInfo：接続されているボードについての情報を取得します。
- GetCommPortInfo：現在の通信ポートの設定についての情報を取得します。
- FireCustomEvent：接続されているすべての ActiveX クライアントおよび JSON-RPC クライアントに通知イベントを送信します。

以下のコールバック・イベントが実装されています。

- OnRecorderDone：現在選択されているレコーダが新しいレコーダ・データのダウンロードを完了したときに呼び出されます。
- OnCommPortStateChanged：通信ポートの状態が変化したときに呼び出されます。
- OnBoardDetected：ターゲット・ボードへの有効な接続が確立されたときに呼び出されます。
- OnVariableChanged：サブスクライブしている変数が変化したときに呼び出されます。

6.3 FreeMASTER JSON-RPC インターフェース

FreeMASTER アプリケーションは起動時、ポート番号 41000 で待機している JSON-RPC サーバを初期化します。JSON-RPC サーバは、クライアントからのプレーンな TCP または WebSocket 接続を受け入れます。複数の FreeMASTER インスタンスを同時に起動した場合は、ポート 41001 ～ 41016 が自動的に使用されます（実行ファイルを実行する際に「/rpcs」コマンドライン・オプションにより手動でポートを指定した場合を除く）。

クライアント・スクリプトの接続方法および JSON-RPC のメソッドをリモートから呼び出す方法は数多くあります。デフォルトの FreeMASTER の例で推奨されて例示されている方法では、一般的な「simple-jsonrpc-js」（GitHub から入手可能）実装の機能をラップした JavaScript オブジェクトが使用されます。このラッパー・オブジェクトは（従来の慣習から）PCM という名前が付けられ、FreeMASTER のインストール・フォルダの「freemaster-client.js」ファイルに組込まれます。また、このファイルは、FreeMASTER Liteウェブ・サーバのどのクライアントからも簡単に利用できます。

PCM ラッパー・オブジェクトを使用してリモート JSON-RPC メソッドを呼び出すのは、ローカルから通常の非同期 JavaScript メソッドを標準的な引数で呼び出すのと同じぐらい簡単です。従来の ActiveX インターフェースから利用できるすべてのメソッドは、PCM オブジェクトからも利用できます。パラメータの順序も変わりません。

JSON-RPC 接続を、以下の呼び出しによって確立します。

```
rpcs_addr = "localhost:41000";
pcm = new PCM(rpcs_addr, on_connected, on_error, on_error);
```

コールバック・メソッドとして、接続が成功した場合とエラーが発生した場合の処理を行う「on_connected」と「on_error」を渡します。サーバ・ポートが「/rpcs」コマンドライン・オプションでオーバーライドされていない限り、「rpcs_addr」パラメータは「localhost:41000」に設定する必要があります。FreeMASTER バージョン 3.1.3 には、「FreeMASTER」という名前の JScript オブジェクトから有効なアドレスを取得する新しい機能が加わっています。これは、以下の特殊なインクルードを挿入することで定義します。

```
<!-- Get the information object named "FreeMASTER" -->
<script type="text/javascript" src="fmstr://localapp/info.js"></script>
```

このステートメントは、現在 Chromium HTML レンダリング・エンジンの「ホスト」となっている FreeMASTER インスタンスから仮想「info.js」ファイルをフェッチするものです。このファイルには、実行中の FreeMASTER アプリケーション内でしかアクセスできないことに注意してください。ウェブ・ブラウザから単体で HTML/JScript コードを実行する場合や、下位互換性を確保する必要がある場合は、必ずデフォルト値である「localhost:41000」を使用してください。

PCM オブジェクトにデフォルトでラップされているのは、FreeMASTER と FreeMASTER Lite の両方のサーバに共通のメソッドだけです。フルバージョンの FreeMASTER アプリケーションのみを対象とした「特別な」メソッド（ActivateWindow、SelectItem、Exit など）にアクセスするには、EnableExtraFeatures() メソッドを最初に呼び出します。通常、特別なメソッドは、FreeMASTER Lite サービスで使用したときに機能が破綻する原因となりますので注意してください。

JSON-RPC のメソッドから返される結果データには、応答オブジェクトの「data」メンバでアクセスできます。データ・メンバの形式は、呼び出したメソッドによって異なります。詳細については、「ActiveX と JSON-RPC のメソッド」を参照してください。FreeMASTER デスクトップ・アプリケーションから返された応答オブジェクトに、「xtra」オブジェクト・メンバが含まれていることがあります。ここには、FreeMASTER Lite の応答には存在しない特別な情報が格納されています。

たとえば、変数の値を取得するための基本的な JSON-RPC メソッドがあります。

```
promise = pcm.ReadVariable("variable_name");
```

読み取りに成功すると、FreeMASTER サーバと FreeMASTER Lite サーバから、どちらも応答オブジェクト（いわゆる「Promise」オブジェクト）が返され、このオブジェクトが、戻り値の変数を含んだ「data」メンバに解決されます。FreeMASTER デスクトップ・アプリケーションからは、さらに「xtra.formatted」値が返されます。この場合、格納されている値は同じですが、変数オブジェクトの設定に基づく書式が適用されています。JSON-RPC インターフェースの使用の詳細については、「HTML ページに埋め込まれた JavaScript と JSON-RPC」のコード例を参照してください。

6.4 FreeMASTER Lite

FreeMASTER Lite は、FreeMASTER 3.0 で初めて導入されました。ホスト・コンピュータ上で動作するサービス・ライクなアプリケーションであり、さまざまなクライアントがターゲット・マイクロコントローラ・ボードにリモートからアクセスすることを可能にします。クライアントは、JSON-RPC プロトコルと、デスクトップ FreeMASTER アプリケーションが備える API とほぼ同じ API を使用してサービスにアクセスします。デスクトップ・アプリケーションとの大きな違いは、FreeMASTER Lite にはユーザー・インターフェースがない点です。設定には、ローカルの JSON 設定ファイルが使用されます。このファイルに、物理的な接続パラメータが指定されているほか、ロードするシンボルを含んだ ELF ファイルが記述され、また、クライアントからアクセス可能な変数オブジェクトも定義されています。このサービスは開始後、ホスト・コンピュータ上でサイレントに動作し、受信クライアント接続を待機します。

FreeMASTER Lite では、HTML ページや他のローカル・ファイルをリモート・クライアントに返す従来型ウェブ・サーバも実行されます。スマートフォンやタブレットを始めとするモバイル端末へのグラフィカルな制御ページの展開は、これによって大幅に簡素化されます。モバイル端末がすべきことは、ウェブ・ブラウザを起動して、FreeMASTER Lite サービスが実行されているコンピュータのインターネット・アドレスを開くだけです。

以下、FreeMASTER Lite の代表的なユース・ケースを挙げます。

- リモートのモバイル端末（タブレットなど）上、または localhost 接続を使用する同じコンピュータ上のウェブ・ブラウザで単独で動作する FreeMASTER 制御ページ・アプリケーションのバックエンド・エンジン。
- スタンドアロン、カスタム、サード・パーティのアプリケーションからターゲット・マイクロコントローラ・アプリケーションに接続することを可能にするサービス。FreeMASTER Lite には、スクリプティングとネットワーク通信をサポートするすべてのアプリケーションから接続できます（Matlab、LabView など）。
- カスタムの JavaScript または Python スクリプトからマイクロコントローラ・ボードに接続することを可能にするサービス。スクリプトなどの自動化環境をテストする際に活用できます。
- Windows および Linux オペレーティング・システムにおける FreeMASTER のコネクティビティ。

6.4.1 FreeMASTER と FreeMASTER Lite の比較

以下、FreeMASTER Lite とデスクトップ FreeMASTER アプリケーションの主な違いを箇条書きで示しました。

FreeMASTER Lite はプラットフォームに依存しません。

- デスクトップ FreeMASTER は、Windows オペレーティング・システムでのみ動作します。
- FreeMASTER Lite は、Windows と Linux のオペレーティング・システムで動作します。

JSON-RPC API と似ていますが、まったく同一というわけではありません。

- どちらの API にも、JSON-RPC のリモート呼び出しをカプセル化した JavaScript PCM ラッパー・オブジェクトを使用してアクセスできます。実装の詳細については、FreeMASTER のインストール・フォルダにある「freemaster-client.js」ファイルを参照してください。このファイルは、FreeMASTER Lite のビルトイン・ウェブ・サーバからも入手できます。
- デスクトップ FreeMASTER には、このアプリケーションでしか利用できない特別なメソッドがあります。たとえば、SelectItem、OpenProject、Exit はユーザー・インターフェースを操作するメソッドですが、FreeMASTER Lite では利用できません。
- デスクトップ FreeMASTER には、ユーザーによって定義された、変数の「視覚的」側面についての情報（印刷の書式設定、値の変換、値からテキストへの変換の列挙など）も存在します。そのため、FreeMASTER Lite で変数を読み取った場合、変数の値しか返されません。一方、デスクトップ FreeMASTER からは、テキストに書式を適用した値など、追加の情報を含んだ「xtra」オブジェクトも返されます。
- デスクトップ FreeMASTER は Lite サービスとは異なり、独自のバックグラウンド処理を実行し、独自のタイミングで処理を行うことができます。そのため、JSON-RPC 呼び出しを使用して変数スティミュレータを起動した後、スティミュレータを独立して実行し続けることができます。FreeMASTER Lite では、周期的な処理や時間ベースの処理全体をクライアント・アプリケーションが担います。
- 同様に、デスクトップ FreeMASTER はターゲット・ボードへの接続を監視し、いわゆるイベントを生成します。イベントは、サーバによって生成されてクライアント側で処理される非同期の JSON-RPC 通知メッセージです。イベントには、OnBoardDetected、OnRecorderDone、OnVariableChanged（サブスクライブされた変数用）などがあります。
- PCM ラッパー・オブジェクトからデフォルトでエクスポートされるのは、FreeMASTER Lite と互換性のある API です。デスクトップ固有の関数は、 PCM ラッパー・オブジェクトの EnableExtraFeatures() メソッドを呼び出して明示的に有効にする必要があります。

6.5 ActiveX と JSON-RPC のメソッド

以降のセクションは、FreeMASTER の ActiveX のメソッドとそれに対応する JSON-RPC API のリファレンスとしてご利用ください。

FreeMASTER Lite サーバでサポートされる JSON-RPC のメソッドを詳細に解説したリファレンスを、スクリプトのソース・コードから自動的に生成されるハイパーテキスト・ドキュメントとして個別に入手できます。詳細については、FreeMASTER Lite のインストール・フォルダにある html/index.html ファイルを参照してください。インストール・フォルダまたは作成済みのショートカットからサービスの実行ファイルを起動し、システムのデフォルトのウェブ・ブラウザで前述のページを自動的に開くこともできます。

以下で説明するメソッド宣言には、戻り値に加えて入力パラメータと出力パラメータが使用されています。このような完全な宣言を使用できるのは、VBScriptやVisual Basic for Applicationsなど、参照渡しの「output」パラメータを返すことのできる言語に限られます。その他の言語（JavaScriptなど）で返されるのは通常の戻り値のみで、出力パラメータ値を取得することはできません。

出力値には、以下の 2 とおりの方法でアクセスできます。

- ActiveX インターフェースを使用している場合、呼び出しから制御が返された後、LastResult や LastRetMsg など、メイン・インターフェース・オブジェクトの LastXXX プロパティを使用して出力パラメータにアクセスできます。
- JSON-RPC インターフェースを使用している場合、応答オブジェクトには戻り値である「data」に加え、「xtra」オブジェクトが格納されます。この xtra オブジェクトに、他の出力値を保持するメンバが格納されています。ただし、FreeMASTER Lite サーバからは「xtra」オブジェクトは返されません。

> 注意
>
> JSON-RPC 呼び出しをラップする PCM オブジェクトのメソッドから返される真の値は、リモート呼び出しを表す標準的な Promise オブジェクトです。リモート呼び出しが成功した場合、「data」メンバと「xtra」メンバを含んだ応答オブジェクトが Promise オブジェクトによって解決（resolve）されます。実行が成功したことを伝えるために、応答オブジェクトには、値が「true」に設定された「succeeded」メンバも格納されます。通信の問題でリモート呼び出しが失敗した場合や、応答オブジェクトの「succeeded」が「false」に設定されていた場合、Promise オブジェクトは拒否（reject）されます。
> Promise インタフェースと非同期 JavaScript プログラミングの概念に慣れるために、インターネットの豊富な情報で知識を深めたうえで、JSON-RPC インタフェースを使用したスクリプトのコーディングを始めることを強くお勧めします。

以降の API リファレンス・ページでは、メソッド・プロトタイプ仕様で以下の記号を使用しています。

- [in] : 必須の入力パラメータを表します。
- [in, optional] : オプションの入力パラメータを表します。メソッドの呼び出し時に値を省略できます。
- [out] : 参照渡しの出力パラメータを表します。出力値にアクセスする方法については、後述の「出力」表を参照してください。出力パラメータはすべて省略可能です。

また各メソッドには、互換性を示す以下の記号が付いています。

- ActiveX：チェック・マーク（✓）が付いているメソッドは、ActiveX インターフェースで利用できます。
- JSON-RPC:チェック・マーク（✓）が付いているメソッドは、デスクトップ FreeMASTER アプリケーションによって実装される JSON-RPC インターフェースから利用できます。freemaster-client.js ファイルを使用して JSON-RPC リモート呼び出しをラップする PCM オブジェクトを使用します。
- Lite:チェック・マーク（✓）が付いているメソッドは、FreeMASTER Lite が実装する JSON-RPC インターフェースから利用できます。チェック・マークが付いていない（✗という記号が付いている）関数は、FreeMASTER Lite ではサポートされません。また、デスクトップ・アプリケーションを使用する場合、そのような関数にアクセスするためには、EnableExtraFeatures メソッドを呼び出す必要があります。

6.5.1 GetAppVersion

プロトタイプ：

```
GetAppVersion([out] retMsg)
```

説明：

FreeMASTER アプリケーションのバージョンを取得します。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite（API は完全互換ではありません）

入力：

なし

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値（ActiveX のみ）	LastResult	response.xtra.retval	バージョン番号（スカラ数値）。
retMsg	LastRetMsg	response.data	バージョン番号（テキスト形式）。 この値は、JSON-RPC メソッドから返されます。

6.5.2 OpenProject

プロトタイプ：

```
OpenProject ([in] project)
```

説明：

指定されたプロジェクト・ファイルを開きます。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
project	プロジェクト・ファイルのパスを含んだ文字列。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	ブール型の戻り値。プロジェクトが正常に開いた場合に true が返されます。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.3 SaveProject

プロトタイプ：

```
SaveProject([in] saveAs);
```

説明：

現在のプロジェクトを保存するか、[Save As（名前を付けて保存）] ダイアログを開きます。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
saveAs	false にした場合、直近の既知のパスに、プロジェクトがサイレントに保存されます。true にした場合、または新規プロジェクトの場合、[Save As（名前を付けて保存）] ダイアログが表示されます。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	プロジェクトが保存された場合、ブール値「true」が返されます。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.4 StartStopComm

プロトタイプ：

```
StartStopComm ([in] start)
```

説明：

現在ロードされているプロジェクトで選択された FreeMASTER の通信ポートを開く、または閉じます。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
start	ブール値 • true：通信を開始します。 • false：通信を停止します。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	ポートの操作に成功したかどうかのブール値が返されます。

備考：

FreeMASTER Lite のインターフェースには、StartComm と StopComm という異なるメソッドがあります。デスクトップ FreeMASTER アプリケーションで新たに設計する場合は、こちらを使用することをお勧めします。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.5 StartComm

プロトタイプ：

```
StartComm ([in] name)
```

説明：

名前または完全接続文字列で指定された、FreeMASTER の通信ポートを開きます。StartStopComm の代わりに、こちらの関数を使用することをお勧めします。FreeMASTER デスクトップ・アプリケーションを使用する際、FreeMASTER プロジェクト内で行った設定を使用する代わりに、新しい接続のすべてのパラメータを呼び出し元のスクリプトで指定する必要がある場合に役立ちます。

接続文字列は「RS232」または{CLSID}形式とし、中括弧で囲んで指定する必要があります。続けて、オプションの引数をセミコロンで区切って指定します。

完全なプラグイン CLSID 識別子の代わりに、以下に示した短い名前を使用できます。

```
        "DAP"      {1BB3C904-F1F4-4652-92EE-368716FE1D10}
        "EONCE"    {5A129680-897E-4014-916D-9B1FE13DE156}
        "PEMICRO"  {225E034C-1AA6-4EC5-9A6C-A9FA9E890373}
        "JLINK"    {8C575DB3-9769-4B66-AB60-C83CEA683E01}
        "ISYSTEM"  {80F17965-EDBD-41F1-9182-3C44E04E5794}
        "PDBDM"    {30AB7AA1-493A-4DD7-9509-C9C91EFFF0FD}
        "CAN"      {C10A92C3-7D47-4FDC-94B6-64B8E5C85E01}
        "LIN"      {C10A92C4-7D47-4FDC-94B6-64B8E5C85E01}
        "NET"      {6D13CD9D-9F2D-4066-B655-45B54CA7494B}
Examples of connection strings:
      RS232;port=COM1;speed=115200
      CAN;drv=kvaser;port=1;bitrate=500000;cmdid=0x7aa;rspid=0x7aa;tmo=500
      DAP;devid=0;dapid=0;DAP_Port=SW;DAP_SWJ=Y;DAP_Clock=1000000
      PEMICRO;drv=6;ptype=3;pnum=1;devid=;devlock=0;jtagspd=500;target=NXP_K6x_K64FN1M0M12
```

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
name	FreeMASTER Lite の設定ファイルにおける接続名を表す文字列。FreeMASTER デスクトップ・アプリケーションに組み込まれているデフォルトの接続を指定するには、特殊な名前である「preset」を使用します。FreeMASTER と FreeMASTER Lite のどちらのケースも、完全接続文字列で名前を指定することもできます。その場合は、設定パラメータをすべて使用して、シリアル接続またはプラグイン接続を開くことができます。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	ポートの操作に成功したかどうかのブール値が返されます。
retMsg	LastRetMsg	response.xtra.message	エラーが発生した場合、この値にはエラー・メッセージが格納されます。

6.5.6 StopComm

プロトタイプ：

```
StopComm ()
```

説明：

現在開いている FreeMASTER の通信ポートを閉じます。

互換性 :✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

なし

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	ポートの操作に成功したかどうかのブール値が返されます。

6.5.7 IsCommPortOpen

プロトタイプ：

```
IsCommPortOpen ()
```

説明：

通信ポートの現在の状態を返します。

互換性 :✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

なし

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	通信ポートが現在開いた状態かどうかを表すブール値。

6.5.8 IsBoardDetected

プロトタイプ：

```
IsBoardDetected ()
```

説明：

ターゲット・ボードとの接続の状態を返します。

互換性 :✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

なし

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	現時点でボードが検出された場合、ブール値の「true」が返されます。

6.5.9 IsBoardWithActiveContent

プロトタイプ：

```
IsBoardWithActiveContent ()
```

説明：

この関数は、現在接続されているターゲット MCU アプリケーションに Active Content が存在するかどうかを調べる際に使用できます。Active Content は、MCU アプリケーションの TSA（Target-Side Addressing）テーブルを使用して有効化され、HTML ページ、プロジェクト・ファイル、ハイパーリンクなど、ファイル関連のリソースを含んでいます。TSA の詳細については、FreeMASTER シリアル・ドライバのドキュメントを参照してください。

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

なし

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	現時点でボードが検出され、なおかつ TSA Active Content が利用可能な場合、ブール値の「true」が返されます。

備考：

この関数の JavaScript 使用例は、FreeMASTER の起動直後に表示される FreeMASTER のようこそページのソース・コードでご覧いただけます。ようこそページでは、ターゲット MCU アプリケーションのコードに埋め込まれたハイパーリンクとプロジェクト・ファイルが検出されるとそれが表示されます。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.10 GetHtmlDocument

プロトタイプ：

```
GetHtmlDocument ([in] winId, [in] activate)
```

説明：

Internet Explorer レンダリング・エンジンの HTML ドキュメントを表す DOM（Document Object Model）オブジェクトのハンドルを取得します。

互換性：

✓ ActiveX、✗ JSON-RPC、✗ Lite

入力：

引数	説明
winId	アクセスする HTML ページを識別する数値。以下のいずれかの値を使用します。 • 0：制御ページ • 1：ブロックの説明ページ • 2：詳細な説明ページ（オシロスコープおよびレコーダ項目用）
activate	選択された HTML ビューをアクティブにするかどうかを選択するブール値。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
オブジェクトのディスパッチ・ハンドル	LastResult	該当なし	返されたハンドルをスクリプトから使用して、HTML ドキュメント・オブジェクトにアクセスできます。

6.5.11 SendCommand

プロトタイプ：

```
SendCommand ([in] cmd, [out] retMsg)
```

説明：

FreeMASTER アプリケーション・コマンドを送信します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
cmd	コマンド名および送信する引数を含んだ文字列値。コマンドは、現在開いている FreeMASTER プロジェクトで定義されている必要があります。コマンドを送信する際は、関数の呼び出しに似た構文を使用します。たとえば、コマンドに 3 つのパラメータがある場合、「command_name(1, 2, 3)」のようにします。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	エラーが発生することなくコマンドが送信された場合、ブール値「true」が返されます。FreeMASTER プロジェクトにコマンドが見つからなかった場合は「False」が返されます。
retMsg	LastRetMsg	response.xtra.message	コマンドの呼び出し後に返されるテキスト。エラーが発生した場合、この値にはエラー・メッセージが格納されます。
	LastAppCommand _retCode	response.xtra.retCode	コマンドのリターン・コード。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.12 SendCommandDlg

プロトタイプ：

```
SendCommandDlg ([in] cmd, [out] retMsg)
```

説明：

FreeMASTER の［Send Application Command（アプリケーション・コマンドの送信）］ダイアログを呼び出します。

互換性：

✓ ActiveX、✗ JSON-RPC、✗ Lite

入力：

引数	説明
cmd	コマンド名およびダイアログに表示する引数を含んだ文字列値。コマンドは、現在開いている FreeMASTER プロジェクトで定義されている必要があります。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	該当なし	エラーが発生することなくコマンドが送信された場合、ブール値「true」が返されます。FreeMASTER プロジェクトにコマンドが見つからなかった場合は「False」が返されます。
retMsg	LastRetMsg	該当なし	コマンドの呼び出し後に返されるテキスト。エラーが発生した場合、この値にはエラー・メッセージが格納されます。

6.5.13 ReadVariable

プロトタイプ：

```
ReadVariable ([in] var, [out] numValue, [out] textValue, [out] retMsg)
```

説明：

FreeMASTER の変数の値を読み取ります。現在の FreeMASTER プロジェクト、または FreeMASTER Lite の設定ファイルに変数が定義されている必要があります。

デスクトップ FreeMASTER アプリケーションの場合のみ：定義されたサンプリング時間より変数の値が新しい場合、このメソッドからは「キャッシュ」された値が返されます。たとえば、変数のサンプリング時間が1秒の場合に、それよりも高頻度でスクリプトがReadVariable関数を呼び出しても、ターゲットからはリアルタイムの値が返されません。このキャッシュ機構を無効にするには、varパラメータで変数名の前に感嘆符（「!」）を使用してください。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
var	読み取る変数の名前を含んだ文字列値。現在開いている FreeMASTER プロジェクト、または FreeMASTER Lite の設定ファイルに変数が定義されている必要があります。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	エラーが発生することなく変数が読み取られた場合、ブール値「true」が返されます。指定された変数がFreeMASTER プロジェクトに見つからなかった場合や、通信エラーが発生した場合は、「False」が返されます。
numValue	LastVariable_vValue	response.data	変数 var の数値表現を返します。
textValue	LastVariable_tValue	response.xtra.formatted	変数の値を表示するために必要な形式と単位を表す文字列値を返します。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

6.5.14 WriteVariable

プロトタイプ：

```
WriteVariable ([in] var, [in] value, [out] retMsg)
```

説明：

FreeMASTER の変数に値を書き込みます。現在の FreeMASTER プロジェクト、または FreeMASTER Lite の設定ファイルに変数が定義されている必要があります。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
var	書き込む変数の名前を含んだ文字列値。現在開いている FreeMASTER プロジェクト、または FreeMASTER Lite の設定ファイルに変数が定義されている必要があります。
value	変数に書き込む値。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	エラーが発生することなく変数が書き込まれた場合、ブール値「true」が返されます。指定された変数が FreeMASTER プロジェクトに見つからなかった場合や無効な値が渡された場合、または通信エラーが発生した場合は、「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

6.5.15 ReadMemory

プロトタイプ：

```
ReadMemory ([in] addr, [in] size, [out] data, [out] retMsg)
ReadMemoryHex ([in] addr, [in] size, [out] textData, [out] retMsg)
```

説明：

ターゲット・アプリケーションからメモリのブロックを読み取ります。

- ReadMemory は、スクリプティング環境に準拠したデータを返します。
  - ActiveX：データは、スクリプト言語（VBScript など）やコンパイル型言語（Visual Basic など）で使用できるバリアント型のセーフ配列として返されます。JavaScript の場合は、`toArray()` メソッドを使用して、セーフ配列を JavaScript 配列に変換する必要があります。
  - JSON-RPC：データは、ネイティブ JSON 配列として返されます。

ActiveX の場合のみ、各種のスクリプティング環境から任意で使用できる関数が他にも存在します。

- ReadMemory_t は、コンパイル型言語（Visual Basic など）で使用できる厳密に型指定された値のセーフ配列でデータを返します。バリアント型の配列を使用するよりも、型指定された配列の方が処理速度は圧倒的に速くなります。
- ReadMemoryHex は、文字列値でデータを返します。16 進数形式では、1 バイトが 2 文字で表現されます。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
addr	読み取るメモリ・ブロックのアドレス。これは、数値の絶対アドレス、現在の FreeMASTER プロジェクトにおける有効なシンボル名、複合式（シンボル名、オフセット値の加算、乗算を含む、または配列のデリファレンス演算子を使用）のいずれかになります。
size	読み取るメモリ・ブロックのサイズ。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	エラーが発生することなくメモリ・ブロックが読み取られた場合、ブール値「true」が返されます。無効なアドレスが指定された場合や通信エラーが発生した場合は「False」が返されます。
data	LastMemory_data	response.data	戻り値の配列。
textData（ReadMemoryHex のみ）	LastMemory_hexstr	response.xtra.hex	文字列形式の戻りデータ。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

備考：

この関数は、下位互換性を確保する目的でのみ提供されます。要素サイズを 1 バイトに設定した ReadUIntArray を代わりに使用してください。

6.5.16 WriteMemory

プロトタイプ：

```
WriteMemory ([in] addr, [in] size, [in] data, [out] retMsg)
```

説明：

バイトのブロックをターゲット・アプリケーションのメモリに書き込みます。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
addr	書き込むメモリ領域のアドレス。これは、数値の絶対アドレス、現在の FreeMASTER プロジェクトにおける有効なシンボル名、複合式（シンボル名、オフセット値の加算、乗算を含む、または配列のデリファレンス演算子を使用）のいずれかになります。
size	書き込むメモリ・ブロックのサイズ。
data	書き込むバイトのセーフ配列。少なくとも size 個の要素を含んでいる必要があります。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	エラーが発生することなくメモリ・ブロックが読み取られた場合、ブール値「true」が返されます。無効なアドレスが指定された場合や通信エラーが発生した場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

備考：

この関数は、下位互換性を確保する目的でのみ提供されます。要素サイズを 1 バイトに設定した WriteUIntArray を代わりに使用してください。

6.5.17 ReadXxxArray

プロトタイプ：

1、2、または 4 バイトの符号付き整数の配列にアクセスする：

```
ReadIntArray ([in] addr, [in] size, [in] elemSize, [out] data, [out] retMsg)
```

1、2、または 4 バイトの符号なし整数の配列にアクセスする：

```
ReadUIntArray ([in] addr, [in] size, [in] elemSize, [out] data, [out] retMsg)
```

4 バイトの浮動小数点数の配列にアクセスする：

```
ReadFloatArray ([in] addr, [in] size, [out] data, [out] retMsg)
```

8 バイトの浮動小数点数の配列にアクセスする：

```
ReadDoubleArray ([in] addr, [in] size, [out] data, [out] retMsg)
```

説明：

ターゲット・アプリケーションからメモリのブロックを読み取って、整数または浮動小数点数の配列として呼び出し元に返します。

- ReadXxxArray は、スクリプティング環境に準拠したデータを返します。
  - ActiveX：データは、スクリプト言語（VBScript など）やコンパイル型言語（Visual Basic など）で使用できるバリアント型のセーフ配列として返されます。2 バイトおよび 4 バイトの符号なし整数型をカプセル化するバリアント型は VBScript エンジンでは処理されません。このメソッドは、そのような値を浮動小数点形式に変換して返します。JavaScript の場合は、toArray() メソッドを使用して、セーフ配列を JavaScript 配列に変換する必要があります。
  - JSON-RPC：データは、ネイティブ JSON 配列として返されます。

ActiveX の場合のみ、各種のスクリプティング環境から任意で使用できる関数が他にも存在します。

- ReadXxxArray_v は基本的に ReadXxxArray と同じですが、2 バイトと 4 バイトの符号なし整数型に対する VBScript の変換は実行しません。
- ReadXxxArray_t は、コンパイル型言語（Visual Basic など）で使用できる厳密に型指定された値のセーフ配列でデータを返します。バリアント型の配列を使用するよりも、型指定された配列の方が処理速度は圧倒的に速くなります。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
addr	読み取るメモリ・ブロックのアドレス。これは、数値の絶対アドレス、現在の FreeMASTER プロジェクトにおける有効なシンボル名、複合式（シンボル名、オフセット値の加算、乗算を含む、または配列のデリファレンス演算子を使用）のいずれかになります。
size	配列から読み取る要素の数。
elemSize	配列要素のサイズ（バイト単位）。ターゲットから読み取られたバイト数の合計は、size * elemSizeで計算できます。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	エラーが発生することなくメモリ・ブロックが読み取られた場合、ブール値「true」が返されます。無効なアドレスが指定された場合や通信エラーが発生した場合は「False」が返されます。
data	LastMemory_data	response.data	戻り値の配列。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

6.5.18 WriteXxxArray

ActiveX のプロトタイプ：

1、2、または 4 バイトの符号付き整数の配列にアクセスする：

```
WriteIntArray ([in] addr, [in] size, [in] elemSize, [in] data, [out] retMsg)
```

1、2、または 4 バイトの符号なし整数の配列にアクセスする：

```
WriteUIntArray ([in] addr, [in] size, [in] elemSize, [in] data, [out] retMsg)
```

4 バイトの浮動小数点数の配列にアクセスする：

```
WriteFloatArray ([in] addr, [in] size, [in] data, [out] retMsg)
```

8 バイトの浮動小数点数の配列にアクセスする：

```
WriteDoubleArray ([in] addr, [in] size, [in] data, [out] retMsg)
```

JSON-RPC のプロトタイプ：

JSON-RPC の関数には、「size」引数がありません。配列サイズはデータから自動的に推測されます。

```
WriteIntArray ([in] addr, [in] elemSize, [in] data, [out] retMsg)
WriteUIntArray ([in] addr, [in] elemSize, [in] data, [out] retMsg)
WriteFloatArray ([in] addr, [in] data, [out] retMsg)
WriteDoubleArray ([in] addr, [in] data, [out] retMsg)
```

説明：

呼び出し元から渡された数値の配列を、ターゲット CPU に適したバイトのブロックに変換してターゲット・メモリに書き込みます。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite（API は完全互換ではありません）

入力：

引数	説明
addr	書き込むメモリ・ブロックのアドレス。これは、数値の絶対アドレス、現在の FreeMASTER プロジェクトにおける有効なシンボル名、複合式（シンボル名、オフセット値の加算、乗算を含む、または配列のデリファレンス演算子を使用）のいずれかになります。
size（ActiveX のみ）	ターゲット・メモリに書き込む要素の数。data パラメータで指定したすべての配列要素を書き込む場合は 0 に設定します。
elemSize	配列要素のサイズ（バイト単位）。ターゲットに書き込むバイト数の合計は、size * elemSizeで計算できます。
data	書き込む値の配列。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	エラーが発生することなくメモリ・ブロックが書き込まれた場合、ブール値「true」が返されます。無効なアドレスが指定された場合や通信エラーが発生した場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

6.5.19 ReadXxxVariable

プロトタイプ：

1、2、または 4 バイトの符号付き整数にアクセスする：

```
ReadIntVariable ([in] addr, [in] size, [out] value, [out] retMsg)
```

1、2、または 4 バイトの符号なし変数にアクセスする：

```
ReadUIntVariable ([in] addr, [in] size, [out] value, [out] retMsg)
```

4 バイトの浮動小数点型変数にアクセスする：

```
ReadFloatVariable ([in] addr, [out] value, [out] retMsg)
```

8 バイトの浮動小数点型変数にアクセスする：

```
ReadDoubleVariable ([in] addr, [out] value, [out] retMsg)
```

説明：

ターゲット・アプリケーションからメモリを読み取って、整数または浮動小数点数の形式で呼び出し元に返します。ReadVariable 関数とは異なり、これらの関数はメモリに直接アクセスします。FreeMASTER で定義されている変数オブジェクトは使用されません。各呼び出しには、異なるスクリプティング環境で使用できるオプションの関数が存在します。

- ReadXxxVariable は、スクリプティング環境に準拠したデータを返します。
  - ー ActiveX：スクリプト言語（VBScript など）やコンパイル型言語（Visual Basic など）で使用できるバリアント型として値が返されます。2 バイトおよび 4 バイトの符号なし整数型をカプセル化するバリアント型は VBScript エンジンでは処理されません。このメソッドは、そのような値を浮動小数点形式に変換して返します。
  - ー JSON-RPC：値は、ネイティブ JSON 値として返されます。

ActiveX の場合のみ、各種のスクリプティング環境から任意で使用できる関数がもう 1 つ存在します。

- ReadXxxVariable_v は基本的に ReadXxxVariable と同じですが、2 バイトと 4 バイトの符号なし整数型に対する VBScript の変換は実行しません。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
addr	読み取るメモリ・ブロックのアドレス。これは、数値の絶対アドレス、現在の FreeMASTER プロジェクトにおける有効なシンボル名、複合式（シンボル名、オフセット値の加算、乗算を含む、または配列のデリファレンス演算子を使用）のいずれかになります。
size	読み取る変数のサイズ。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	エラーが発生することなくメモリが読み取られた場合、ブール値「true」が返されます。無効なアドレスが指定された場合や通信エラーが発生した場合は「False」が返されます。
value	LastVariable_vValue	response.data	戻り値。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

6.5.20 WriteXxxVariable

プロトタイプ：

1、2、または 4 バイトの符号付き整数を書き込む：

```
WriteIntVariable ([in] addr, [in] size, [in] value, [out] retMsg)
```

1、2、または 4 バイトの符号なし整数を書き込む：

```
WriteUIntVariable ([in] addr, [in] size, [in] value, [out] retMsg)
```

4 バイトの浮動小数点数を書き込む：

```
WriteFloatVariable ([in] addr, [in] size, [in] value, [out] retMsg)
```

8 バイトの浮動小数点数を書き込む：

```
WriteDoubleVariable ([in] addr, [in] size, [in] value, [out] retMsg)
```

説明：

ターゲット・メモリに変数を 1 つ書き込みます。WriteVariable 関数とは異なり、これらの関数はメモリに直接アクセスします。FreeMASTER で定義されている変数オブジェクトは使用されません。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
addr	書き込むメモリ・ブロックのアドレス。これは、数値の絶対アドレス、現在の FreeMASTER プロジェクトにおける有効なシンボル名、複合式（シンボル名、オフセット値の加算、乗算を含む、または配列のデリファレンス演算子を使用）のいずれかになります。
size	ターゲット変数のサイズ。
value	書き込む値。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	エラーが発生することなくメモリ・ブロックが書き込まれた場合、ブール値「true」が返されます。無効なアドレスが指定された場合や通信エラーが発生した場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

6.5.21 ReadMultipleVariables

プロトタイプ：

```
ReadMultipleVariables ([in] arrNames, [out] outputJSON)
```

説明

FreeMASTER で定義された複数の変数を 1 回の呼び出しで読み取るには、この関数を使用します。この呼び出しは機能上、複数の ReadVariable 呼び出しの順次実行に相当します。これを使用することで、ActiveX/JSON-RPC の通信や他のフレームワーク関連の処理によって消費される、場合によっては数ミリ秒にも達するオーバーヘッド時間を最小限に抑えることができます。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
arrNames	読み取る変数名の配列。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	呼び出し中に正常に読み取られた変数の数。
outputJSON	LastMultipleVariables_json	response.data	ActiveX：結果オブジェクトの JSON 形式の配列を含む文字列。 JSON-RPC：結果オブジェクトの配列。 配列要素は、関数の呼び出し時に入力として渡された各変数名につき 1 つ存在します。 各配列要素は、以下のプロパティを持つオブジェクトです。 • name：変数名 • status：読み取り操作の結果を表すブール値 • value：数値変数 • formatted：書式設定された文字列の値 • retMsg：エラー・メッセージ

6.5.22 WriteMultipleVariables

プロトタイプ：

```
WriteMultipleVariables ([in] arrValues, [out] outputJSON)
```

説明

FreeMASTER で定義された複数の変数を 1 回の呼び出しで書き込むには、この関数を使用します。この呼び出しは機能上、複数の WriteVariable 呼び出しの順次実行に相当します。これを使用することで、ActiveX/JSON-RPC の通信や他のフレームワーク関連の処理によって消費される、場合によっては数ミリ秒にも達するオーバーヘッド時間を最小限に抑えることができます。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
arrValues	変数名と書き込む値の配列。各配列要素は、以下のプロパティを持つオブジェクトである必要があります。 • name：変数名 • value：書き込む変数の値

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	呼び出し中に正常に書き込まれた変数の数。
outputJSON	LastMultipleVariables_json	response.data	ActiveX：結果オブジェクトの JSON 形式の配列を含む文字列。 JSON-RPC：結果オブジェクトの配列。 配列要素は、関数の呼び出し時に入力として渡された各変数値につき 1 つ存在します。 各配列要素は、以下のプロパティを持つオブジェクトです。 • name：変数名 • status：書き込み操作の結果を表すブール値 • retMsg：エラー・メッセージ

6.5.23 GetCurrentRecorderData

プロトタイプ：

```
GetCurrentRecorderData ([out] data, [out] serieNames, [out] timeBaseSec, [out] retMsg)
```

説明：

現在レコーダ・チャートに表示されているデータを取得します。

- GetCurrentRecorderData は、スクリプティング環境に準拠した 2 次元配列データを返します。最初の次元は系列インデックス、2 番目の次元は値インデックスです。
  - ー ActiveX：データは、スクリプト言語（VBScript など）やコンパイル型言語（Visual Basic など）で使用できるバリアント型の 2 次元セーフ配列として返されます。2 バイトおよび 4 バイトの符号なし整数型をカプセル化するバリアント型は VBScript エンジンでは処理されません。このメソッドは、そのような値を浮動小数点形式に変換して返します。JavaScript の場合は、toArray() メソッドを使用して、セーフ配列を JavaScript 配列に変換する必要があります。
  - ー JSON-RPC：データは、配列（レコーダの各データ系列につき 1 つ）のネイティブ JSON 配列として返されます。

ActiveX の場合のみ、各種のスクリプティング環境から任意で使用できる関数が他にも存在します。

- GetCurrentRecorderData_t は、コンパイル型言語（Visual Basic など）で使用できる「倍精度浮動小数点」型のセーフ配列でデータを返します。バリアント型の配列を使用するよりも、型指定された配列の方が処理速度は速くなります。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

なし。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	レコーダ項目からデータが正常に取得された場合、ブール値「true」が返されます。レコーダに有効なデータが存在しない場合や、現在アクティブなレコーダが存在しない場合は「False」が返されます。
data	LastRecorder_data	response.xtra.data	値の 2 次元配列を返します。最初の次元は系列インデックス、2 番目の次元は値インデックスです。
serieNames	LastRecorder _serieNames	response.xtra.names	該当するインデックスの系列の名前を含んだ配列を返します。
timeBaseSec	LastRecorder _timeBaseSec	response.xtra. baseRateSec	各記録サンプルの間隔（秒）を返します。そのような値がターゲットから提供されない場合、この値は 0 になります。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

備考：

このメソッドは、FreeMASTER Lite ではサポートされません。デスクトップ・アプリケーションで一般的な「現在選択されている」レコーダ項目という概念が FreeMASTER Lite にはないためです。FreeMASTER Lite には、レコーダを直接制御して読み取るためのメソッドが用意されています。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.24 GetCurrentRecorderSeries

プロトタイプ：

```
GetCurrentRecorderSeries ([in] serieName, [out] data, [out] timeBaseSec, [out] retMsg)
```

説明：

現在表示されているレコーダ・チャートから 1 つのデータ系列を取得します。

- GetCurrentRecorderSeries は、スクリプティング環境に準拠した 1 次元配列データを返します。
  - ActiveX：データは、スクリプト言語（VBScript など）やコンパイル型言語（Visual Basic など）で使用できるバリアント型のセーフ配列として返されます。2 バイトおよび 4 バイトの符号なし整数型をカプセル化するバリアント型は VBScript エンジンでは処理されません。このメソッドは、そのような値を浮動小数点形式に変換して返します。JavaScript の場合は、toArray() メソッドを使用して、セーフ配列を JavaScript 配列に変換する必要があります。
  - JSON-RPC：データは、ネイティブ JSON 配列として返されます。

ActiveX の場合のみ、各種のスクリプティング環境から任意で使用できる関数が他にも存在します。

- GetCurrentRecorderSeries_t は、コンパイル型言語（Visual Basic など）で使用できる「倍精度浮動小数点」型のセーフ配列でデータを返します。バリアント型の配列を使用するよりも、型指定された配列の方が処理速度は速くなります。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
serieName	取得するデータがあるデータ系列の名前（記録された変数名）を含んだ文字列。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	レコーダ項目からデータが正常に取得された場合、ブール値「true」が返されます。レコーダに有効なデータが存在しない場合や、現在アクティブなレコーダが存在しない場合は「False」が返されます。
data	LastRecorder_data	response.xtra.data	戻り値の配列。
timeBaseSec	LastRecorder _timeBaseSec	response.xtra. baseRateSec	取得された各記録サンプルの間隔（秒）。そのような値がターゲットから提供されない場合、この値は 0 になります。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

備考：

このメソッドは、FreeMASTER Lite ではサポートされません。デスクトップ・アプリケーションで一般的な「現在選択されている」レコーダ項目という概念が FreeMASTER Lite にはないためです。FreeMASTER Lite には、レコーダを直接制御して読み取るためのメソッドが用意されています。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.25 StartCurrentRecorder

プロトタイプ：

```
StartCurrentRecorder ([out] retMsg)
```

説明：

現在 FreeMASTER ウィンドウに表示されているレコーダを開始します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

なし

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	レコーダが正常に開始された場合、ブール値「true」が返されます。現在アクティブなレコーダが存在しない場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

備考：

このメソッドは、FreeMASTER Lite ではサポートされません。デスクトップ・アプリケーションで一般的な「現在選択されている」レコーダ項目という概念が FreeMASTER Lite にはないためです。FreeMASTER Lite には、レコーダを直接制御して読み取るためのメソッドが用意されています。

6.5.26 StopCurrentRecorder

プロトタイプ：

```
StopCurrentRecorder ([out] retMsg)
```

説明：

現在 FreeMASTER ウィンドウに表示されているレコーダを手動で停止します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

なし

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	レコーダが正常に停止された場合、ブール値「true」が返されます。現在アクティブなレコーダが存在しない場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

備考：

このメソッドは、FreeMASTER Lite ではサポートされません。デスクトップ・アプリケーションで一般的な「現在選択されている」レコーダ項目という概念が FreeMASTER Lite にはないためです。FreeMASTER Lite には、レコーダを直接制御して読み取るためのメソッドが用意されています。

6.5.27 GetCurrentRecorderState

プロトタイプ：

```
GetCurrentRecorderState ([out] state, [out] retMsg)
```

説明：

現在のレコーダの状態コードと割り当てられている状態テキストを取得します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

なし

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	レコーダの状態が正常に読み取られた場合、ブール値「true」が返されます。現在アクティブなレコーダが存在しない場合は「False」が返されます。
state	LastRecorder_state	response.xtra.state	現在のレコーダの状態を表す数値を返します。有効な状態コードは以下のとおりです。 • 0：アイドル • 1：開始中 • 2：実行中 • 3：結果のダウンロード中 • 4：信号の受信を一時停止中 • 5：エラー • 6：手動停止中 • 7：初期化されていない • 8：エラーで一時停止中 • 9：ダウンロード準備完了
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は、現在のレコーダ状態の説明テキストが格納されます。

備考：

このメソッドは、FreeMASTER Lite ではサポートされません。デスクトップ・アプリケーションで一般的な「現在選択されている」レコーダ項目という概念が FreeMASTER Lite にはないためです。FreeMASTER Lite には、レコーダを直接制御して読み取るためのメソッドが用意されています。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.28 RunStimulators

プロトタイプ：

```
RunStimulators ([in] stimNames)
```

説明：

1つ（または複数）の FreeMASTER 変数スティミュレータを起動します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
stimNames	起動するスティミュレータのセミコロン区切りリスト。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	実際に起動されたスティミュレータの数。この数は、stimNames パラメータにセミコロン区切りで指定されたスティミュレータ名の数と等しいか、それよりも小さくなります。名前で検出されなかったスティミュレータや既に実行されているスティミュレータは、このカウントに含まれません。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.29 StopStimulators

プロトタイプ：

```
StopStimulators ([in] stimNames)
```

説明：

1つ（または複数）の FreeMASTER 変数スティミュレータを停止します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
stimNames	停止するスティミュレータのセミコロン区切りリスト。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	実際に停止されたスティミュレータの数。この数は、stimNames パラメータにセミコロン区切りで指定されたスティミュレータ名の数と等しいか、それよりも小さくなります。名前で検出されなかったスティミュレータや実行されていないスティミュレータは、このカウントに含まれません。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.30 LocalFileOpen

プロトタイプ：

```
LocalFileOpen ([in] fileName, [in] openMode)
```

説明：

プロジェクト領域のローカルに格納されたファイルを開き、後続の操作に使用する、ファイルのハンドルを返します。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
fileName	開くファイルの名前。必要に応じて相対パスを指定できます。セキュリティ上の理由から、絶対パスで指定されたファイルや安全でない拡張子のファイルをこの関数で開くことはできません。許可されている拡張子は以下のとおりです。 `.txt、.xml、.htm、.html、.c、.cpp、.asm、.h、.hpp` 相対パスは、以下のいずれかの仮想フォルダ名で始めることができます。 • `FMSTR_PROJECT_PATH`：現在のプロジェクト・ファイルの場所。 • `FMSTR_PACKDIR_PATH`：現在のリソース・モジュール・フォルダの場所。
openMode	開いているファイルのアクセス・モードを指定します。 • 'r'：読み取り用にファイルを開きます（デフォルト）。 • 'w'：書き込み用にファイルを作成するかファイルを開きます。既存のファイルが存在する場合、元のファイルは長さゼロに切り詰められます。 • 'a'：追記モードで書き込み用にファイルを作成するかファイルを開きます。既存のファイルが存在していても、元のファイルは切り詰められません。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	ファイルの数値ハンドル。ファイルを開くことができなかった場合は、ブール値「false」が返されます。

6.5.31 LocalFileClose

プロトタイプ：

```
LocalFileClose ([in] handle)
```

説明：

直前の LocalFileOpen 呼び出しで返された handle パラメータによって識別されるファイルを閉じます。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
handle	閉じるファイルのハンドル。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	ファイルが正常に閉じられた場合、ブール値「true」が返されます。ファイル・ハンドルが有効でない場合は、「False」が返されます。

6.5.32 LocalFileWriteString

ActiveX のプロトタイプ：

```
LocalFileWriteString ([in] handle, [in] data, [in, optional] charsToWrite, [in, optional] unicode)
```

JSON-RPC のプロトタイプ：

JSON-RPC の関数には charsToWrite 引数がありません。文字列サイズはデータから自動的に推測されます。

```
LocalFileWriteString ([in] handle, [in] data, [in, optional] unicode)
```

説明：

直前の LocalFileOpen 呼び出しで返された handle パラメータによって識別されるファイルにデータを書き込みます。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite（API は完全互換ではありません）

入力：

引数	説明
handle	ファイルを開くためのハンドル。
data	ファイルに書き込むテキスト。
charsToWrite （ActiveX のみ）	実際に書き込む文字数。このパラメータを省略するか 0 に設定した場合、data パラメータに渡したテキスト全体が書き込まれます。
unicode	「True」：データを Unicode 形式で書き込みます。 「False」：データを ASCII 形式で書き込みます。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	ファイルに書き込まれた文字数を返します。

6.5.33 LocalFileReadString

プロトタイプ：

```
LocalFileReadString ([in] handle, [in] charsToRead, [in, optional] unicode, [out] retString)
```

説明：

直前の LocalFileOpen 呼び出しで返された handle パラメータによって識別されるファイルからデータを読み取ります。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
handle	ファイルを開くためのハンドル。
charsToRead	読み取る文字数。
unicode	「True」：データを Unicode 形式で読み取ります。 「False」：データを ASCII 形式で読み取ります。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	実際にファイルから読み取られた文字の数を返します。
retString	LastLocalFile_string	response.data	ファイルから読み取られた文字を含む戻りデータ。

6.5.34 GetSymbolInfo

ActiveX のプロトタイプ：

```
GetSymbolInfo ([in] symbol, [out] symAddr, [out] symSize, [out] retMsg, [in] useVBScriptTypes)
```

JSON-RPC のプロトタイプ：

```
GetSymbolInfo ([in] symbol)
```

説明：

アプリケーションの ELF または MAP ファイルから解析されたシンボル、またはランタイム中に TSA 機能を使用して取得されたシンボルについての情報を返します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
symbol	シンボルの名前。
useVBScriptTypes（ActiveX のみ）	VBScript スクリプト言語と互換性のある浮動小数点数としてアドレスとサイズを取得するには、「true」を使用します。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	シンボルが存在していてその情報が取得された場合、ブール値「true」が返されます。シンボルが見つからなかった場合は、「False」が返されます。
symAddr	LastSymbolInfo_addr	response.xtra.addr	シンボルのアドレス。
symSize	LastSymbolInfo_size	response.xtra.size	シンボルのサイズ。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.35 GetStructMemberInfo

ActiveX のプロトタイプ：

```
GetStructMemberInfo ([in] typeName, [in] member, [out] membOff, [out] membSize, [out] retMsg,
[in] useVBScriptTypes)
```

JSON-RPC のプロトタイプ：

```
GetStructMemberInfo ([in] typeName, [in] member)
```

説明：

アプリケーションの ELF ファイルから解析された、またはランタイム内で TSA 機能を使用して取得された構造体データ型メンバについての情報を返します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
typeName	構造体または共用体データ型の名前。
member	構造体または共用体メンバの名前。
useVBScriptTypes（ActiveX のみ）	VBScript スクリプト言語と互換性のある浮動小数点数としてアドレスとサイズを取得するには、「true」を使用します。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	該当する型が存在していてその情報が取得された場合、ブール値「true」が返されます。構造体型が見つからなかった場合は、「False」が返されます。
membOff	LastMemberInfo_offset	response.xtra.offset	構造体メンバのオフセットを返します。
membSize	LastMemberInfo_size	response.xtra.size	構造体メンバのサイズを返します。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.36 GetAddressInfo

プロトタイプ：

```
GetAddressInfo ([in] addr, [in] size, [out] isExactMatch, [out] symbolName)
```

説明：

アプリケーションの ELF ファイルから解析された、またはランタイム内で TSA 機能を使用して取得されたメモリ・ロケーションについての情報を返します。ターゲットのグローバル・シンボルおよびスタティック・シンボルをすべて評価し、指定されたメモリ領域と一部でも重なっているかどうかを確認します。このアルゴリズムは、構造体のメンバや配列の要素も含め、最も合致するメモリ領域の名前を検出しようと試みます。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
addr	メモリ・アドレス。
size	調査するメモリ領域のサイズ。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	メモリ領域がグローバル・シンボルまたはスタティック・シンボルにマッピングされていた場合、ブール型の値「true」が返されます。メモリ領域がいずれのシンボルにもマッピングされていなかった場合は「false」値が返されます。
isExactMatch	LastAddressInfo_exact	response.xtra. isExactMatch	返されたシンボルがメモリのアドレスおよびサイズと完全に一致する場合、「true」値が返されます。
symbolName	LastAddressInfo_name	response.xtra. symbolName	メモリ領域の最も合致する名前を返します。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.37 GetXxxDefinition

プロトタイプ：

```
GetOscilloscopeDefinition([in] name, [out] defString, [out] retMsg)
GetRecorderDefinition([in] name, [out] defString, [out] retMsg)
GetArrayViewerDefinition([in] name, [out] defString, [out] retMsg)
GetPipeDefinition([in] name, [out] defString, [out] retMsg)
GetWatchBlockDefinition([in] name, [out] defString, [out] retMsg)
```

説明：

この関数を使用して、対応する各項目の JSON 定義レコードを取得できます。その後、その JSON レコードを前述の DefineOscilloscope、DefineRecorder、DefineArrayViewer、DefinePipe、DefineWatchBlock の各メソッドから使用して、プロパティや設定に変更を加えた新しいプロジェクト項目を作成できます。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
name	定義オブジェクトを取得する項目の名前。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	オブジェクトが見つかって定義レコードが返された場合、ブール値「true」が返されます。
defString	LastDefinition_string	response.xtra.def	ActiveX：JSON 形式の定義を含む文字列。 JSON-RPC：定義オブジェクト自体。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.38 DefineSymbol

プロトタイプ：

```
DefineSymbol ([in] symbol, [in] addr, [in] typeName, [in] size, [out] retMsg)
```

説明：

この関数を使用して、通常はターゲット・アプリケーションの実行ファイル（ELF または MAP ファイル）を解析することによって取得されるシンボル情報、またはランタイム中にTSA機能を使用して取得されるシンボル情報を拡張できます。動的に割り当てられた変数など、アドレスと型が分かっている非グローバル・オブジェクトのシンボルを定義する際に活用できます。シンボルが定義されていれば、デスクトップ・アプリケーションのインターフェースから、または DefineVariable 関数を使用して、そのシンボルの FreeMASTER 変数を定義できます。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
symbol	シンボル名。構造体のメンバをマッピングしながら複雑なシンボルを定義する際は、ドット（.）、->、括弧（[]）などの特殊文字を含めることもできます。
addr	シンボルのアドレス。
typeName	シンボルが持つ型の名前。通常は、アプリケーションの ELF ファイルのデバッグ情報からロードされた既存の構造体型の名前になります。指定しなかった場合や、空の文字列を渡した場合、シンボルに型は割り当てられません。
size	シンボルのサイズ。また、`sizeof(TYPE)` や数式など、特殊な式を使用することもできます。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	指定した型情報でシンボルが定義されている場合、ブール値「true」が返されます。型名が指定されているものの、アプリケーションの実行ファイルからロードされた型情報の中にその型がない場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.39 DeleteAllScriptSymbols

プロトタイプ：

```
DeleteAllScriptSymbols()
```

説明：

この関数は、過去に DefineSymbol 呼び出しによって定義されたシンボルをすべて削除します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

なし。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	このメソッドは「true」値を返します。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.40 SubscribeVariable

プロトタイプ：

```
SubscribeVariable ([in] varName, [in] interval, [out] retMsg)
```

説明：

変数の値の変化に対するイベント駆動型の処理を可能にします。変数を「サブスクライブ」すると、

変数の値の変化が FreeMASTER によって検出されるたびに、OnVariableChanged 通知イベントがスクリプトに送られます。FreeMASTER は、サブスクライブされたすべての変数を、指定された間隔で定期的にサンプリングします。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
varName	変数名。単なるシンボル名ではなく、有効な FreeMASTER 変数名にする必要があります。
interval	変数の変化を判定する間隔。OnVariableChanged イベントが呼び出されるのは、この間隔（interval 値）ごとに 1 回だけです。変数がもっと短時間で変化する場合でも同じです。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.subscriptionId	ゼロ以外の値が、変数のサブスクライブを解除する際に使用できるサブスクリプション ID です。OnVariableChanged イベント・コールバックにおける変数も、この ID によって識別されます。 変数をサブスクライブできなかった場合、値はゼロになります。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

備考：

JSON-RPC インターフェースの場合は、EnableEvents 関数を呼び出すことで通知の送信が有効になります。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.41 UnSubscribeVariable

プロトタイプ：

```
UnSubscribeVariable ([in] nameOrId, [out] retMsg)
```

説明：

第 1 パラメータで識別された変数について、変数のサブスクリプションをキャンセルし、OnVariableChanged イベント呼び出しを停止します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
nameOrId	変数のサブスクライブに使用された変数名（または、SubscribeVariable 呼び出しから返されたサブスクリプション ID）。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	変数のサブスクライブが解除された場合、ブール値「true」が返されます。エラーが発生した場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.42 SelectItem

プロトタイプ：

```
SelectItem ([in] name, [in] tabPage)
```

説明：

FreeMASTER ウィンドウでプロジェクト・ツリー項目を選択し、項目に割り当てられたいずれかのビューのタブをアクティブにします。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
name	FreeMASTER の「プロジェクト・ツリー」ビューに表示される項目の名前。名前自体を指定するか、またはすべての親ブロック名の完全修飾パスと項目名をスラッシュ文字で区切って指定できます。
tabPage	ツリー項目の選択後にアクティブにする FreeMASTER ビュー・タブの名前。 以下のいずれかの値を使用します。 • control：[Control Page（制御ページ）] タブ • blkinfo：[Parent Block description（親ブロックの説明）] ページ • iteminfo：[Item description（項目の説明）] ページ • scope：[Oscilloscope graph（オシロスコープ・グラフ）] タブ • recorder：[Recorder graph（レコーダ・グラフ）] タブ

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	項目が見つかって選択された場合、ブール値「true」が返されます。それ以外の場合は「False」が返されます。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.43 DeleteItem

プロトタイプ：

```
DeleteItem([in] name, [out] retMsg)
```

説明

プロジェクト・ツリーから項目を削除します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
name	FreeMASTER の「プロジェクト・ツリー」ビューに表示される項目の名前。 名前自体を指定するか、またはすべての親ブロック名の完全修飾パスと項目名をスラッシュ文字で区切って指定できます。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	項目が見つかって削除された場合、ブール値「true」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.44 FireCustomEvent

プロトタイプ：

```
FireCustomEvent([in] arg)
```

説明

このサーバ・インスタンスに接続されたすべてのクライアントで通知イベント（OnCustomEvent）を発生させるには、この関数を使用します。イベントは、ActiveX と JSON-RPC の両方のタイプのすべてのクライアントで発生します。JSON-RPC クライアントは、EnableExtraFeatures と EnableEvents を呼び出してイベントをサブスクライブする必要があります。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
arg	OnCustomEvent ハンドラに渡すバリアント型（スクリプトの任意の型）のイベント引数。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	イベントが生成された場合、ブール値「true」が返されます。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.45 DefineVariable

ActiveX のプロトタイプ：

```
DefineVariable ([in] name, [in] defString, [out] retMsg)
```

JSON-RPC のプロトタイプ：

```
DefineVariable ([in] defString)
```

説明：

FreeMASTER の変数オブジェクトとそのすべてのプロパティを定義します。スクリプトには、オブジェクトのすべてのプロパティが JSON 表記で記述されている必要があります。変数オブジェクトは新たに作成されるか、既存の変数オブジェクトがある場合は、そのオブジェクトに変更が加えられます。JSON で定義されていないすべてのプロパティには、新しいオブジェクトの作成時にデフォルト値が割り当てられます。

以下の JSON サンプルは、参考のために定義レコードを簡潔に示したものです。GetVariableInfo() を呼び出すことで取得することもできます。[Variable Watch（変数ウォッチ）] ビューで変数をコピーする際、クリップボード・テキスト形式内では、同じレコードが「var_def」オブジェクトとして使用されることに注意してください。

```
{
    // general information
    "name": "var16",                // variable name
    "period": 200,                  // sampling period in milliseconds
    "comment": "",                  // comment visible in Watch view
    "description": "",              // internal description
    "unit": "unit",                 // physical units of REAL mode

    // variable type
    "treat_as": 0,                  // 0=uint, 1=int, 2=float, 3=fract, 4=ufract, 5=string
    "type_fmt": "",                 // fract/ufract type description (e.g. UQ1.15)

    // physical location and type in embedded application
    "address": "var16",             // C symbol name or an address expression
    "byte_size": 2,                 // size of the variable
    "bit_shift": 0,                 // bit-field shift
    "bit_mask": "-1",               // bit-field mask

    // virtual variable special settings
    "virtual": 0,                   // non-zero to set as virtual
    "virtual_default": "0",         // default virtual variable value
```

```
    // enumeration type
    "enum_enabled": 0,               // non-zero to enable enumerations
    "enum": "",                      // name of the enum type
    "enum_num_always": 0,            // display numeric value always
    "enum_default_val": "unk",       // default text label
    "enum_num_with_default": 1,      // numeric value visible at default text

    // visualization in the Watch view
    "show_as": 0,                    // 0=DEC,1=HEX,2=BIN,3=ASCII,4=REAL,
    "show_val": 1,                   // value visible
    "show_max": 0,                   // maximum visible
    "show_min": 0,                   // minimum visible
    "show_num_fixed_digs": 0,        // fixed number of digits
    "show_num_lzero_fill": 0,        // use 0 to fill fixed number of digits
    "show_num_afp_digs": 0,          // number of digits after decimal point
    "show_num_exp": 0,               // non-zero to show in exponential format
    "show_ascii_zterm": 0,           // display as zero-terminated string (ASCII)
    "show_ascii_hex": 1,             // display non-printable chars as HEX
    "show_ascii_chwdt": 1,           // character width in bytes

    // numeric filter
    "filt_enabled": 0,               // non-zero to enable averaging filter
    "filt_reset_onmod": 1,           // reset filter when value is modified
    "filt_time": 5000,               // filter period in milliseconds

    // variable modification properties
    "modif_mode": 1,                 // 0=disabled, 1=any value, 2=pre-defined values
    "modif_edit_mode": 2,            // 1=edit box, 2=drop-down list
    "modif_set_mode": 0,             // 0=write after any change, 1=wait for Enter key
    "modif_auto_fin": 1,             // finish edit mode after Enter key
    "modif_auto_hide": 1,            // hide edit interface if cell is not focused
    "modif_predefs_modes": 1,        // predefined values: 0x1=range, 0x2=enum, 0x4=other
    "modif_predefs_other": "",       // "other" pre-defined values (when 0x4 flag is used)
    "modif_set_min": "",             // minimum value for range-based predef. values
    "modif_set_max": "",             // maximum value for range-based predef. values
    "modif_set_step": "",            // iteration step for range-based predef. values

    // advanced parameters
    "modif_address": "",             // shadow write address
}
```

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite（API は完全互換ではありません）

入力：

引数	説明
name（ActiveX のみ）	作成または変更する FreeMASTER 変数の名前。
defString	ActiveX：JSON 形式の定義を含む文字列。 JSON-RPC：定義オブジェクト自体。この定義オブジェクトの「name」値として新しい変数名を指定することもできます。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	オブジェクトが作成または変更された場合、ブール値「true」が返されます。エラーが発生した場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

JavaScript の例：

```
function define_variable(name, symbol, size, period)
{
    var def = {
      "address" : symbol,
      "byte_size" : size,
      "period" : period,
    };
    ok = pcm.DefineVariable(name, JSON.stringify(def));
}
```

6.5.46 DefineOscilloscope

JSON-RPC のプロトタイプ：

```
 DefineOscilloscope ([in] name, [in] defString, [out] retMsg)
```

説明：

FreeMASTER のオシロスコープ・オブジェクトとそのすべてのプロパティを定義します。スクリプトには、オブジェクトのすべてのプロパティが JSON 表記で記述されている必要があります。オシロスコープ・オブジェクトは新たに作成されるか、既存のオシロスコープ・オブジェクトがある場合は、そのオブジェクトに変更が加えられます。JSON で定義されていないすべてのプロパティには、新しいオブジェクトの作成時にデフォルト値が割り当てられます。

以下の JSON サンプルは、参考のために定義レコードを簡潔に示したものです。GetXxxDefinition を呼び出すことで取得することもできます。[Project Tree（プロジェクト・ツリー）] ビューでオシロスコープ項目をコピーする場合には、クリップボード内のテキスト形式で同じレコードが「tree_item」オブジェクトとして使用されることに注意してください。

```
{
    "name": "var16 scope",           // oscilloscope item name
    "href": "",                      // description page URL

    // oscilloscope settings
    "scope_period": 2.5e-002,        // sampling period in seconds
    "scope_ignore_duplicity": 0,     // ignore samples which are identical to previous
    "scope_x_width": 5,              // x-axis total width
    "scope_x_autoscale": 0,          // autoscale x-axis until width is reached
```

```
// graph settings
"graph_buffer": 2000,            // graph buffer size in points
"graph_type": 0,                 // graph type: 0=Time, 1=XY
"graph_legend_loc": 0,           // nonzero if legend is visible
"graph_legend_vis": 1,           // nonzero to put legend to graph interior
"graph_legend_inside": 1,        // exterior legend location: 0=top, 1=bottom, 2=left, 3=right
"graph_grid_horz": 1,            // show graph horizontal grid
"graph_grid_vert": 1,            // show graph vertical grid
"graph_x_units": 2,              // X-axis time units: 0=us, 1=ms, 2=sec, 3=min (Time graph only)
"graph_x_label": "Time",         // X-axis label (Time graph only)
"graph_x_addunits": 1,           // non-zero to add time unit name to axis label (Time graph only)
"graph_x_min": 0,                // nonzero to enable automatic X axis minimum value (XY graph only)
"graph_x_max": 100,              // nonzero to enable automatic X axis maximum value (XY graph only)
"graph_x_min_auto": 1,           // manual X axis minimum value (XY graph only)
"graph_x_max_auto": 1,           // manual X axis maximum value (XY graph only)
"auto_delete": 0,                // nonzero to set the Oscilloscope item to auto-delete mode

// oscilloscope variables
"var_info": [                    // array of graph variables
   { "var_def": {                // variable definition record
         "name": "var16",        // specifying 'name' is enough to locate existing variable
   },
   "visible": 1,                 // variable visible in graph
   "y_block": 0,                 // assign to y-block 0..4
   "color": 3026413              // subset color
    },
    {
 "var_def": {
        "name": "var16inc",
 },
    "variable": "var16inc",
    "visible": 1,
    "y_block": 1,
    "color": 4688896
  }
],

// graphs details for each Y-axis block
"yblock_info": [                 // array of Y-block definitions
  {
     "join_class": 0,            // blocks with equal value will be joined

     // left-axis settings
     "laxis_label": "16 bits",   // label
     "laxis_style": 0,           // style
     "laxis_min_auto": 0,        // use automatic minimum scaling
     "laxis_max_auto": 1,        // use automatic maximum scaling
     "laxis_min": 0,             // minimum value (when auto=0)
     "laxis_max": 10,            // maximum value (when auto=0)

     // right-axis settings
     "raxis_label": "Increment", // label
     "raxis_style": 0,           // style
     "raxis_min": 0,             // use automatic minimum scaling
     "raxis_max": 10,            // use automatic maximum scaling
     "raxis_min_auto": 0,        // minimum value (when auto=0)
     "raxis_max_auto": 1,        // maximum value (when auto=0)
     "raxis_subsets": 1          // number of subsets to assign to right axis
```

```
        },                              // (counted from the last subset)
        {
             "join_class": 1,           // a different class means to separate this block
             "laxis_label": "lAxis",
             "laxis_style": 0,
             "laxis_min": -10,
             "laxis_max": 10,
             "laxis_min_auto": 1,
             "laxis_max_auto": 1,
             "raxis_label": "rAxis",
             "raxis_style": 0,
             "raxis_min": -10,
             "raxis_max": 10,
             "raxis_min_auto": 1,
             "raxis_max_auto": 1,
             "raxis_subsets": 0
        },
    ],
}
```

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
name	プロジェクト・ツリーで作成または変更する FreeMASTER オシロスコープ項目の名前。
defString	ActiveX：JSON 形式の定義を含む文字列 JSON-RPC：定義オブジェクト自体

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	オブジェクトが作成または変更された場合、ブール値「true」が返されます。エラーが発生した場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

備考：

ActiveX インターフェースでは、下位互換性を確保するために、この関数が DefineScope としても提供されています。

JavaScript の例：

```
function define_my_scope()
{
     // array of oscilloscope variables, my_variable_x should already be valid FreeMASTER
     // variable objects, first two variables will share the Y graph block
     var vars = [
      { "variable":” my_variable_1” , "visible":true, "y_block":0 },
```

```
      { "variable":" my_variable_2" , "visible":true, "y_block":0 },
      { "variable":" my_variable_3" , "visible":true, "y_block":1 },
    ];
    // array of scope Y-blocks, we have two, not joined
     var yblocks = [
      { "laxis_label":"raw signal", "join_class":0,
        "laxis_min_auto":true, "laxis_max_auto":true },
      { "laxis_label":"touch status", "join_class":1,
        "laxis_min_auto":true, "laxis_max_auto":true },
     ];
    // scope definition
    var def = {};
    def["var_info"] = vars;
    def["yblock_info"] = yblocks;
    def["scope_period"] = 0.025; // 25ms
    def["href"] = "my_description_page.htm";

    pcm.DefineScope( "my oscilloscope" , JSON.strinigy(def));
}
```

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.47 DefinePipe

プロトタイプ：

```
DefineRecorder ([in] name, [in] defString, [out] retMsg)
```

説明：

FreeMASTER のパイプ・オブジェクトとそのすべてのプロパティを定義します。スクリプトには、オブジェクトのすべてのプロパティが JSON 表記で記述されている必要があります。パイプ・オブジェクトは新たに作成されるか、既存のパイプ・オブジェクトがある場合は、そのオブジェクトに変更が加えられます。JSON で定義されていないすべてのプロパティには、新しいオブジェクトの作成時にデフォルト値が割り当てられます。

以下の JSON サンプルは、参考のために定義レコードを簡潔に示したものです。また、これは GetXxxDefinition を呼び出すことでも取得できます。[Project Tree（プロジェクト・ツリー）]ビューでパイプ項目をコピーする場合には、クリップボード内のテキスト形式で同じレコードが「tree_item」オブジェクトとして使用されることに注意してください。

```
{
    "name": "Terminal Pipe",          // array viewer item name
    "href": "",                       // description page URL

    "pipe_port": 1,                   // target pipe port number
    "pipe_refresh_period": 100,       // pipe update period
    "pipe_element_size": 1,           // character or pipe element size
    "pipe_wrap_lines": 1,             // wrap console lines

    "font_name": "Courier New",       // font used in view
    "font_size": 8,                   // font size
}
```

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.48 DefineRecorder

プロトタイプ：

```
DefineRecorder ([in] name, [in] defString, [out] retMsg)
```

説明：

FreeMASTER のレコーダ・オブジェクトとそのすべてのプロパティを定義します。スクリプトには、オブジェクトのすべてのプロパティが JSON 表記で記述されている必要があります。レコーダ・オブジェクトは新たに作成されるか、既存のレコーダ・オブジェクトがある場合は、そのオブジェクトに変更が加えられます。JSON で定義されていないすべてのプロパティには、新しいオブジェクトの作成時にデフォルト値が割り当てられます。

以下の JSON サンプルは、参考のために定義レコードを簡潔に示したものです。GetXxxDefinition を呼び出すことで取得することもできます。[Project Tree（プロジェクト・ツリー）] ビューでレコーダ項目をコピーする場合には、クリップボード内のテキスト形式で同じレコードが「tree_item」オブジェクトとして使用されることに注意してください。

```
{
    "name": "var16 recorder",         // recorder item name
    "href": "",                       // description page URL

    // recorder settings
    "rec_id": 0                       // target recorder instance to use
    "rec_time_base": 0,               // recorder time base multiplier (one less)
    "rec_total_samples": 70,          // total number of recorded samples
    "trig_pretrigger": 20,            // number of pre-trigger samples
    "trig_astop_ena": 0,              // stop automatically when timeout expires
    "trig_astop_delay": 10,           // automatic stop timeout
    "trig_aload": 1,                  // load data automatically when stopped
    "trig_hold_ena": 1,               // hold loaded data in graph
    "trig_hold_delay": 1,             // minimum hold time
    "trig_arun": 1,                   // run automatically after hold time
    "trig_zero_x": 1,                 // show trigger point as 0 on X axis

    // graph settings
    "graph_buffer": 2000,             // graph buffer size in points
    "graph_type": 0,                  // graph type: 0=Time, 1=XY
    "graph_legend_loc": 0,            // nonzero if legend is visible
    "graph_legend_vis": 1,            // nonzero to put legend to graph interior
    "graph_legend_inside": 1,         // exterior legend location: 0=top, 1=bottom, 2=left, 3=right
    "graph_grid_horz": 1,             // show graph horizontal grid
    "graph_grid_vert": 1,             // show graph vertical grid
    "graph_x_units": 2,               // X-axis time units: 0=us, 1=ms, 2=sec, 3=min (Time graph only)
    "graph_x_label": "Time",          // X-axis label (Time graph only)
    "graph_x_addunits": 1,            // non-zero to add time unit name to axis label (Time graph only)
    "graph_x_min": 0,                 // nonzero to enable automatic X axis minimum value (XY graph only)
    "graph_x_max": 100,               // nonzero to enable automatic X axis maximum value (XY graph only)
    "graph_x_min_auto": 1,            // manual X axis minimum value (XY graph only)
    "graph_x_max_auto": 1,            // manual X axis maximum value (XY graph only)
    "auto_delete": 0,                 // nonzero to set the Oscilloscope item to auto-delete mode

    // recorder variables
    "var_info": [                     // array of graph variables
        {
            "var_def": {              // variable definition record
                "name": "var16",      // specifying 'name' is enough to locate existing variable
            },
            "visible": 1,             // variable visible in graph
            "y_block": 0,             // assign to y-block 0..4
            "color": 3026413          // subset color
            "trg_mode": 1,            // trigger mode 0=off, 1=edge, 2=level
```

```
            "trg_rising": 1,        // trigger on rising edge or level high
            "trg_falling": 0,       // trigger on falling edge or level low
            "trg_threshold": 100    // trigger threshold value
        },
        {
            "var_def": {
                    "name": "var16inc",
            },
            "variable": "var16inc",
            "visible": 1,
            "y_block": 1,
            "color": 4688896
            "trg_mode": 0,
        }
    ],

    // graphs details for each Y-axis block
    "yblock_info": [                        // array of Y-block definitions
        {
            "join_class": 0,                // blocks with equal value will be joined

            // left-axis settings
            "laxis_label": "16 bits",       // label
            "laxis_style": 0,               // style
            "laxis_min_auto": 0,            // use automatic minimum scaling
            "laxis_max_auto": 1,            // use automatic maximum scaling
            "laxis_min": 0,                 // minimum value (when auto=0)
            "laxis_max": 10,                // maximum value (when auto=0)

            // right-axis settings
            "raxis_label": "Increment",     // label
            "raxis_style": 0,               // style
            "raxis_min": 0,                 // use automatic minimum scaling
            "raxis_max": 10,                // use automatic maximum scaling
            "raxis_min_auto": 0,            // minimum value (when auto=0)
            "raxis_max_auto": 1,            // maximum value (when auto=0)
            "raxis_subsets": 1              // number of subsets to assign to right axis
        },                                  // (counted from the last subset)
        {
            "join_class": 1,                // a different class means to separate this block

            "laxis_label": "lAxis",
            "laxis_style": 0,
            "laxis_min": -10,
            "laxis_max": 10,
            "laxis_min_auto": 1,
            "laxis_max_auto": 1,

            "raxis_label": "rAxis",
            "raxis_style": 0,
            "raxis_min": -10,
            "raxis_max": 10,
            "raxis_min_auto": 1,
            "raxis_max_auto": 1,
            "raxis_subsets": 0
        },
    ],
}
```

互換性 :✓ ActiveX、✓ JSON-RPC、✗ Lite

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.49 DefineWatchBlock

JSON-RPC のプロトタイプ：

```
DefineWatchBlock ([in] name, [in] defString, [out] retMsg)
```

説明：

[Variable Watch（変数ウォッチ）] ビューの独自の定義を含んだ FreeMASTER のブロック・オブジェクトをプロジェクト・ツリーに定義します。スクリプトには、オブジェクトのすべてのプロパティが JSON 表記で記述されている必要があります。ブロック・オブジェクトは新たに作成されるか、既存のブロック・オブジェクトがある場合は、そのオブジェクトに変更が加えられます。JSON で定義されていないすべてのプロパティには、新しいオブジェクトの作成時にデフォルト値が割り当てられます。

以下の JSON サンプルは、参考のために定義レコードを簡潔に示したものです。GetXxxDefinition を呼び出すことで取得することもできます。[Project Tree（プロジェクト・ツリー）] ビューでブロックをコピーする場合には、クリップボード内のテキスト形式で同じレコードが「tree_item」オブジェクトとして使用されることに注意してください。

```
{
    "name": "New Block",                        // block item name
    "href": "",                                 // description page URL

    // Variable Watch settings
    "show_grid": 1,                             // show grid
    "ena_row_sizing": 1,                        // enable row sizing
    "ena_col_sizing": 1,                        // enable column sizing
    "ena_col_swapping": 1,                      // enable column swapping
    "ena_inplace_values": 1                     // enable variable modifications

    // watched variables
    "watch_variables": [
        {
            "var_def": {                        // variable definition record
                "name": "var16",                // specifying 'name' is enough to locate existing variable
            },
            "format": {                         // format of the watched variable
                "row_height": 20                // grid row size
                "cells": [                      // array of up to 5 cells of the row
                    {                           //
                        "font_name": "Tahoma",  // font name
                        "font_size": 12,        // font size
                        "font_bold": 0,         // bold font
                        "font_italic": 0,       // italic font
                        "font_underline": 0,    // underlined font
                        "alignment": 33,        // cell alignment 33=left,34=right,36=center
                        "backcolor": 16777215,  // cell background color as hex BBGGRR
                        "forecolor": 255        // cell foreground color
                    },                          //
                ],                              //
            }                                   //
        },                                      //
        {                                       // next watched variable
            "var_def": {                        //
                "name": "var16inc",             //
            },                                  //
            "format": {                         //
                "row_height": 20                //
                "cells": []                     //
```

```
                }                                       //
            },                                          //
            {                                           // next watched variable
                "var_def": {                            //
                    "name": "var16rw",                  //
                },                                      //
                "format": {                             //
                    "row_height": 20                    //
                    "cells": [],                        //
                }
            }
        ],

        "column_info": [                                // array of variable watch columns
            {                                           // [0] = 'name' column
                "column_visible": 1,                    // column visible
                "column_width": 120,                    // column size
                "column_ix": 0                          // column placement (may be swapped)
            },                                          //
            {                                           // [1] = 'value' column
                "column_visible": 1,                    //
                "column_width": 120,                    //
                "column_ix": 1                          //
            },                                          //
            {                                           // [2] = 'unit' column
                "column_visible": 1,                    //
                "column_width": 120,                    //
                "column_ix": 2                          //
            },                                          //
            {                                           // [3] = 'period' column
                "column_visible": 1,                    //
                "column_width": 50,                     //
                "column_ix": 3                          //
            },                                          //
            {                                           // [4] = 'comment' column
                "column_visible": 0,                    //
                "column_width": 200,                    //
                "column_ix": 4                          //
            }
        ],
    }
}
```

互換性 :✓ ActiveX、✓ JSON-RPC、✗ Lite

入力 :

引数	説明
name	プロジェクト・ツリーで作成または変更する FreeMASTER オシロスコープ項目の名前。
defString	ActiveX : JSON 形式の定義を含む文字列 JSON-RPC : 定義オブジェクト自体

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	オブジェクトが作成または変更された場合、ブール値「true」が返されます。エラーが発生した場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.50 DefineArrayViewer

JSON-RPC のプロトタイプ：

```
DefineOscilloscope ([in] name, [in] defString, [out] retMsg)
```

説明：

FreeMASTER の Array Viewer オブジェクトとそのすべてのプロパティを定義します。スクリプトには、オブジェクトのすべてのプロパティが JSON 表記で記述されている必要があります。Array Viewer オブジェクトは新たに作成されるか、既存の Array Viewer オブジェクトがある場合は、そのオブジェクトに変更が加えられます。JSON で定義されていないすべてのプロパティには、新しいオブジェクトの作成時にデフォルト値が割り当てられます。

以下の JSON サンプルは、参考のために定義レコードを簡潔に示したものです。GetXxxDefinition を呼び出すことで取得することもできます。[Project Tree（プロジェクト・ツリー）] ビューで Array Viewer 項目をコピーする場合には、クリップボード内のテキスト形式で同じレコードが「tree_item」オブジェクトとして使用されることに注意してください。

```
{
    "name": "New Array Viewer",         // array viewer item name
    "href": "",                         // description page URL

    // array viewer settings
    "array_count": 10,                  // number of elements read from each array
    "trig_astop_ena": 0,                // stop automatically when timeout expires
    "trig_astop_delay": 10,             // automatic stop timeout
    "trig_aload": 1,                    // load data automatically when stopped
    "trig_hold_ena": 1,                 // hold loaded data in graph
    "trig_hold_delay": 1,               // minimum hold time
    "trig_arun": 1,                     // run automatically after hold time
    "trig_mode": 2,                     // 0=off,1=variable change,2=change&auto-clear

    "trig_var_def": {                   // trigger variable definition record
        "name": "var16trig",            // specifying 'name' is enough to locate existing variable
    },

    // graph settings
    "graph_buffer": 2000,               // graph buffer size in points
    "graph_type": 0,                    // graph type: 0=Time, 1=XY
    "graph_legend_loc": 0,              // nonzero if legend is visible
    "graph_legend_vis": 1,              // nonzero to put legend to graph interior
    "graph_legend_inside": 1,           // exterior legend location: 0=top, 1=bottom, 2=left, 3=right
    "graph_grid_horz": 1,               // show graph horizontal grid
    "graph_grid_vert": 1,               // show graph vertical grid
    "graph_x_units": 2,                 // X-axis time units: 0=us, 1=ms, 2=sec, 3=min (Time graph only)
    "graph_x_label": "Time",            // X-axis label (Time graph only)
    "graph_x_addunits": 1,              // non-zero to add time unit name to axis label (Time graph only)
    "graph_x_min": 0,                   // nonzero to enable automatic X axis minimum value (XY graph only)
    "graph_x_max": 100,                 // nonzero to enable automatic X axis maximum value (XY graph only)
```

```
    "graph_x_min_auto": 1,              // manual X axis minimum value (XY graph only)
    "graph_x_max_auto": 1,              // manual X axis maximum value (XY graph only)
    "auto_delete": 0,                   // nonzero to set the Oscilloscope item to auto-delete mode

    // viewed arrays
    "var_info": [                       //
        {
            "var_def": {                // variable definition record
                "name": "arr16[0]",     // specifying 'name' is enough to locate existing variable
            },
            "visible": 1,               // variable visible in graph
            "y_block": 0,               // assign to y-block 0..4
            "color": 3026413            // subset color
        }
    ],

    // graph details for each Y-axis block
    "yblock_info": [
        {
            "join_class": 0,            // blocks with equal value will be joined

            // left-axis settings
            "laxis_label": "lAxis",     // label
            "laxis_style": 0,           // style
            "laxis_min": -10,           // use automatic minimum scaling
            "laxis_max": 10,            // use automatic maximum scaling
            "laxis_min_auto": 1,        // minimum value (when auto=0)
            "laxis_max_auto": 1,        // maximum value (when auto=0)

            // right-axis settings
            "raxis_label": "rAxis",     // label
            "raxis_style": 0,           // style
            "raxis_min": -10,           // use automatic minimum scaling
            "raxis_max": 10,            // use automatic maximum scaling
            "raxis_min_auto": 1,        // minimum value (when auto=0)
            "raxis_max_auto": 1,        // maximum value (when auto=0)
            "raxis_subsets": 0          // number of subsets to assign to right axis
        }                               // (counted from the last subset)
    ],
}
```

互換性 :✓ ActiveX、✓ JSON-RPC、✗ Lite

入力 :

引数	説明
name	プロジェクト・ツリーで作成または変更する FreeMASTER オシロスコープ項目の名前。
defString	ActiveX : JSON 形式の定義を含む文字列 JSON-RPC : 定義オブジェクト自体

出力 :

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	オブジェクトが作成または変更された場合、ブール値 true」が返されます。エラーが発生した場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.51 EnumHrefLinks

プロトタイプ：

```
EnumHrefLinks ([in] index, [out] hrefName)
```

説明：

ターゲット MCU アプリケーションの Active Content によって参照されているハイパーリンクを列挙します。呼び出し元のスクリプトは、無効な「false」値が返されるまで「index」値を増やしながら、この関数を呼び出す必要があります。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
index	取得するハイパーリンクのインデックス。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	特定の「index」位置にハイパーリンクが存在した場合、ハイパーリンクの URL を文字列形式で表した値。 それ以外の場合は、ブール値「false」が返されます。
hrefName	LastLinkName	response.xtra.name	ハイパーリンク・テキスト名、つまりユーザーに対して表示される文字列。

JavaScript の例：

この関数のJavaScript使用例は、FreeMASTERの起動時に表示されるFreeMASTERのようこそページのソース・コードでご覧いただけます。ようこそページでは、ターゲット MCU の Active Content 機能が持っているハイパーリンクが検出されるとそれが表示されます。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.52 EnumProjectFiles

プロトタイプ：

```
EnumProjectFiles ([in] index, [out] prjName)
```

説明：

ターゲット MCU アプリケーションの Active Content によって参照されているプロジェクト・ファイルを列挙します。呼び出し元のスクリプトは、無効な「false」値が返されるまで「index」値を増やしながら、この関数を呼び出す必要があります。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
index	取得する埋め込みプロジェクト・ファイルのインデックス。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	特定の「index」位置にプロジェクトが存在するときに、プロジェクト・ファイルの URL を文字列形式で表した値。それ以外の場合は、「false」値が返されます。
prjName	LastLinkName	response.xtra.name	プロジェクト名、つまりユーザーに対して表示される文字列。

JavaScript の例：

この関数のJavaScript使用例は、FreeMASTERの起動時に表示されるFreeMASTERのようこそページのソース・コードでご覧いただけます。ようこそページでは、ターゲット MCU の Active Content 機能が持っているハイパーリンクが検出されるとそれが表示されます。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.53 PipeOpen

プロトタイプ：

```
PipeOpen ([in] port, [in, optional] txBufferSize, [in, optional] rxBufferSize, [out] retMsg)
```

説明：

ターゲット MCU との通信に使用される FreeMASTER のパイプ・オブジェクトを、ロスレス・ストリーム I/O チャネルを使用して初期化します。それぞれのパイプは、ポート番号によって完全に識別されます。通信を正しく行うには、ターゲット MCU 側でも、同じポートのパイプを初期化して定期的に処理する必要があります。詳細については、FreeMASTER シリアル・ドライバのドキュメントを参照してください。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。
txBufferSize	ローカル送信バッファのサイズ。このバッファは、PipeWrite 関数呼び出しによってターゲット MCU に送信されるデータを蓄積し、MCU がそれを受け取る準備ができるまで待機する目的で使用されます。 省略可能であり、デフォルトでは 0x1000 に設定されます。
rxBufferSize	ローカル受信バッファのサイズ。このバッファは、ターゲット MCU から受信するデータを蓄積し、スクリプトが PipeRead 関数を呼び出してそれを読み取るまで待機する目的で使用されます。 省略可能であり、デフォルトでは 0x1000 に設定されます。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	パイプが正常に初期化された場合、ブール値「true」が返されます。それ以外の場合は「False」が返されます。
retMsg	LastRetMsg	response.error.msg	エラーが発生した場合、この値にはエラー・メッセージが格納されます。それ以外の場合は空になります。

6.5.54 PipeClose

プロトタイプ：

```
PipeClose ([in] port)
```

説明：

パイプ・オブジェクトの初期化を解除します。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	パイプが正常に閉じられた場合、ブール値「true」が返されます。それ以外の場合は「False」が返されます。

6.5.55 PipeFlush

プロトタイプ：

```
PipeFlush ([in] port, [in] timeout_ms)
```

説明：

保留中のデータをターゲット MCU に送信することを試みます。また、MCU 側で送信待ちになっている新しいデータがあれば、そのデータも受信します。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。
timeout_ms	保留データの送信に費やす最大試行時間（ミリ秒）。 ゼロにした場合、データの送信が 1 回だけ試行されます。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	この関数は、フラッシュの試行後にローカル送信バッファに残っているバイト数を返します。

6.5.56 PipeSetDefaultTxMode

プロトタイプ：

```
PipeSetDefaultTxMode ([in] txAllOrNothing)
```

説明：

各種 PipeWrite 関数の後続の呼び出しに使用するデフォルトの送信モードを設定します。

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
txAllOrNothing	各種 PipeWrite 関数の後続の呼び出しでデフォルトとして使用するブール値。

出力：

なし。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.5.57 PipeSetDefaultRxMode

プロトタイプ：

```
PipeSetDefaultRxMode ([in] rxAllOrNothing, [in] rxTimeout_ms)
```

説明：

各種 PipeRead 関数の後続の呼び出しに使用するデフォルトの受信モードを設定します。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
rxAllOrNothing	各種 PipeRead 関数の後続の呼び出しでデフォルトとして使用するブール値。
rxTimeout_ms	後続の PipeRead 呼び出しで使用するデフォルトのタイムアウト（ミリ秒）。

出力：

なし。

6.5.58 PipeSetDefaultStringMode

プロトタイプ：

```
PipeSetDefaultStringMode ([in] unicode)
```

説明：

PipeWriteString 関数と PipeReadString 関数で使用するデフォルトの文字列エンコーディングを設定します。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
unicode	PipeWriteString 関数と PipeReadString 関数の後続の呼び出しでデフォルトとして使用するブール値。

出力：

なし。

6.5.59 PipeGetRxBytes

プロトタイプ：

```
PipeGetRxBytes ([in] port)
```

説明：

各種 PipeRead 関数で読み取るローカル受信バッファの利用可能なバイト数を返します。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	ローカル受信バッファで使用可能な状態にあるバイト数。

6.5.60 PipeGetTxBytes

プロトタイプ：

```
PipeGetTxBytes ([in] port)
```

説明：

ローカル送信バッファで保留状態になっているバイト数を返します。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	ローカル送信バッファで、ターゲット MCU への送信待ち状態になっている保留中のバイト数。

6.5.61 PipeGetTxFree

プロトタイプ：

```
PipeGetTxFree ([in] port)
```

説明：

ローカル送信バッファの空き容量を返します。この関数は、各種 PipeWrite 関数を呼び出す前に、正常に書き込めるかどうかを確認する目的で使用します。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	ローカル送信バッファの空き容量。

6.5.62 PipeGetRxBufferSize

プロトタイプ：

```
PipeGetRxBufferSize ([in] port)
```

説明：

ローカル受信バッファのサイズを返します。これは、PipeOpen 呼び出しでパイプ・オブジェクトを初期化するときに指定されるサイズです。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	ローカル受信バッファのサイズ。

6.5.63 PipeGetTxBufferSize

プロトタイプ：

```
PipeGetTxBufferSize ([in] port)
```

説明：

ローカル送信バッファのサイズを返します。これは、PipeOpen 呼び出しでパイプ・オブジェクトを初期化するときに指定されるサイズです。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	ローカル送信バッファのサイズ。

6.5.64 PipeWriteString

ActiveX のプロトタイプ：

```
PipeWriteString ([in] port, [in] string, [in] charsToWrite, [in] allOrNothing, [in] unicode)
```

JSON-RPC のプロトタイプ：

```
PipeWriteString ([in] port, [in] string, [in] allOrNothing, [in] unicode) 説明：
```

選択されたパイプに対し、入力文字列からの文字を書き込みます。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite（API は完全互換ではありません）

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。
string	書き込む文字列。
charsToWrite（ActiveX のみ）	パイプに書き込む文字数。このパラメータは省略可能です。省略した場合は文字列全体が書き込まれます。
allOrNothing	ゼロ以外を指定した場合、PipeWrite 関数は、ローカル送信バッファに対してすべてのデータを一気に書き込むことを試みます。データがバッファに収まりきらない場合、データは一切送信されません。 このパラメータは省略可能です。デフォルト値は、PipeSetDefaultTxMode 関数によって決まります。
unicode	ゼロ以外を指定した場合、各文字が Unicode エンコーディング（1 文字あたり 2 バイト）で書き込まれます。Unicode 文字は常に不可分の処理として書き込まれます。断片的な値が送信される心配はありません。 このパラメータは省略可能です。デフォルト値は、PipeSetDefaultStringMode 関数によって決まります。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	この関数は、書き込まれた文字数を返します。

6.5.65 PipeWriteXxxArray

ActiveX のプロトタイプ：

1、2、または 4 バイトの符号付き整数の配列を書き込む：

```
PipeWriteIntArray ([in] port, [in] elemSize, [in] data, [in, optional] count, [in, optional] allOrNothing)
```

1、2、または 4 バイトの符号なし整数の配列を書き込む：

```
PipeWriteUIntArray ([in] port, [in] elemSize, [in] data, [in, optional] count, [in, optional] allOrNothing)
```

4 バイトの浮動小数点数の配列を書き込む：

```
PipeWriteFloatArray ([in] port, [in] data, [in, optional] count, [in, optional] allOrNothing)
```

8 バイトの浮動小数点数の配列を書き込む：

```
PipeWriteDoubleArray ([in] port, [in] data, [in, optional] count, [in, optional] allOrNothing)
```

JSON-RPC のプロトタイプ：

JSON-RPC の関数には、「count」引数がありません。配列サイズはデータから自動的に推測されます。

```
PipeWriteIntArray ([in] port, [in] elemSize, [in] data, [in, optional] allOrNothing)
PipeWriteUIntArray ([in] port, [in] elemSize, [in] data, [in, optional] allOrNothing)
PipeWriteFloatArray ([in] port, [in] data, [in, optional] allOrNothing)
PipeWriteDoubleArray ([in] port, [in] data, [in, optional] allOrNothing)
```

説明：

呼び出し元から渡された数値の配列をバイトのブロックに変換し、そのデータを選択されたパイプに書き込みます。各配列メンバは、送信バッファに対して常に不可分の処理として書き込まれます。断片的に値が送信される心配はありません。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite（API は完全互換ではありません）

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。
elemSize	配列要素のサイズ（バイト単位）。ターゲットに書き込まれるバイト数の合計は、count * elemSizeで計算できます。
data	書き込む値の配列。
count（ActiveX のみ）	パイプに書き込む要素数。このパラメータは省略可能です。省略した場合は配列全体が書き込まれます。
allOrNothing	ゼロ以外を指定した場合、PipeWrite 関数は、ローカル送信バッファに対してすべてのデータを一気に書き込むことを試みます。データがバッファに収まりきらない場合、データは一切送信されません。 このパラメータは省略可能です。デフォルト値は、PipeSetDefaultTxMode 関数によって決まります。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.data	この関数は、正常に書き込まれた配列要素数を返します。

6.5.66 PipeReadString

プロトタイプ：

```
PipeReadString ([in] port, [in] rxTimeout_ms, [in] charsToRead, [in, optional] allOrNothing, [in, optional]
unicode, [out] retString)
```

説明：

選択されたパイプから文字を読み取ります。

互換性：✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。
rxTimeout_ms	要求された文字数の受信を待機する時間（ミリ秒）。 このパラメータは省略可能です。デフォルト値は、PipeSetDefaultRxMode によって決まります。
charsToRead	パイプから読み取る文字数。このパラメータは省略可能です。省略した場合、タイムアウトに達するまでに処理可能な最大文字数が読み取られます。
allOrNothing	ゼロ以外を指定した場合で、なおかつ「charsToRead」のカウントがゼロ以外である場合、この関数は、必要なすべてのデータを一気に読み取ることを試みます。利用可能なデータがないままタイムアウトに達した場合、データは一切受信されません。 このパラメータは省略可能です。デフォルト値は、PipeSetDefaultRxMode 関数によって決まります。
unicode	ゼロ以外を指定した場合、各文字が Unicode エンコーディング（1 文字あたり 2 バイト）で読み取られます。 Unicode 文字は常に不可分の処理として読み取られます。断片的な値を受信する心配はありません。 このパラメータは省略可能です。デフォルト値は、PipeSetDefaultStringMode 関数によって決まります。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	この関数は、読み取られた文字数を返します。
retString	LastPipe_data	response.data	このパラメータ変数には文字列が格納されます。

6.5.67 PipeReadXxxArray

プロトタイプ：

1、2、または 4 バイトの符号付き整数を読み取る：

```
PipeReadIntArray ([in] port, [in] elemSize, [in] rxTimeout_ms, [in] count, [in] allOrNothing, [out] data)
```

1、2、または 4 バイトの符号なし変数を読み取る：

```
PipeReadUIntArray ([in] port, [in] elemSize, [in] rxTimeout_ms, [in] count, [in] allOrNothing, [out] data)
```

4 バイトの浮動小数点型変数にアクセスする：

```
PipeReadFloatArray ([in] port, [in] rxTimeout_ms, [in] count, [in] allOrNothing, [out] data)
```

8 バイトの浮動小数点型変数にアクセスする：

```
PipeReadDoubleArray ([in] port, [in] rxTimeout_ms, [in] count, [in] allOrNothing, [out] data)
```

説明：

要求された数の整数または浮動小数点数をパイプから読み取り、整数または浮動小数点数の配列として呼び出し元に返します。

各呼び出しには、異なるスクリプティング環境で使用できるオプションの関数が存在します。

- PipeReadXxxArray は、スクリプティング環境に準拠したデータを返します。
  - ActiveX：データは、スクリプト言語（VBScript など）やコンパイル型言語（Visual Basic など）で使用できるバリアント型のセーフ配列として返されます。2バイトおよび4バイトの符号なし整数型をカプセル化するバリアント型はVBScriptエンジンでは処理されません。このメソッドは、そのような値を浮動小数点形式に変換して返します。JavaScript の場合は、toArray() メソッドを使用して、セーフ配列を JavaScript 配列に変換する必要があります。
  - JSON-RPC：データは、ネイティブ JSON 配列として返されます。

ActiveX の場合のみ、各種のスクリプティング環境から任意で使用できる関数が他にも存在します。

- PipeReadXxxArray_v は基本的に PipeReadXxxArray と同じですが、2 バイトと 4 バイトの符号なし整数型に対する VBScript の変換は実行しません。
- PipeReadXxxArray_t は、コンパイル型言語（Visual Basic など）で使用できる厳密に型指定された値の VB 配列でデータを返します。バリアント型の配列を使用するよりも、型指定された配列の方が処理速度は圧倒的に速くなります。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
port	パイプ・チャネルを識別するポート番号。使用されるのは、値の下位 16 ビットのみです。
elemSize	配列要素のサイズ（バイト単位）。パイプから読み取られるバイト数の合計は、count * elemSize で計算できます。
rxTimeout_ms	要求された文字数の受信を待機する時間（ミリ秒）。 このパラメータは省略可能です。デフォルト値は、PipeSetDefaultRxMode によって決まります。
count	パイプに書き込む要素数。このパラメータは省略可能です。省略した場合、タイムアウトに達するまでに処理可能な最大数の値が読み取られます。
allOrNothing	true を指定した場合で、なおかつ「count」がゼロ以外である場合、この関数は、必要なすべてのデータを一気に読み取ることを試みます。利用可能なデータがないままタイムアウトに達した場合、データは一切受信されません。 このパラメータは省略可能です。デフォルト値は、PipeSetDefaultRxMode 関数によって決まります。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	この関数は、読み取られた配列要素数を返します。
data	LastPipe_data	response.data	このパラメータ変数には結果の配列が格納されます。

6.5.68 EnumVariables

プロトタイプ：

```
EnumVariables ([in] index, [out] variableName)
```

説明：

現在のプロジェクトから FreeMASTER の変数を列挙します。呼び出し元のスクリプトは、無効な「false」値が返されるまで「index」値を増やしながら、この関数を呼び出す必要があります。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
index	取得する変数のインデックス。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	有効な値が返されているときは、ブール値「true」が返されます。「index」に対応する有効な変数オブジェクトが存在しない場合は「False」となり、列挙を停止する必要があります。
variableName	LastEnum_name	response.data	変数名。

6.5.69 EnumSymbols

プロトタイプ：

```
EnumSymbols ([in] index, [out] symbolName)
```

説明：

現在のプロジェクトでロードされているターゲット・アプリケーションのシンボルを列挙します。呼び出し元のスクリプトは、無効な「false」値が返されるまで「index」値を増やしながら、この関数を呼び出す必要があります。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
index	取得するシンボルのインデックス。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	有効な値が返されているときは、ブール値「true」が返されます。「index」に対応する有効なシンボル・オブジェクトが存在しない場合は「False」となり、列挙を停止する必要があります。
symbolName	LastEnum_name	response.data	シンボル名。

6.5.70 GetDetectedBoardInfo

プロトタイプ：

```
GetDetectedBoardInfo ()
```

説明：

ホスト PC とターゲット MCU アプリケーションとの間の最初の FreeMASTER プロトコル・ハンドシェイク中に得られた情報を取得します。FreeMASTER 通信プロトコル・バージョン 4 の導入に伴い、この関数および返されるパラメータは廃止されます。プロトコル・バージョン 4 では、オシロスコープ、レコーダ、パイプ・オブジェクトを複数定義できるようになりましたが、この従来のデータ構造ではそれを記述できません。

プロトコル・バージョン 4 の設定パラメータを取得するには、GetConfigParamXxx のいずれかの関数を使用してください。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

なし

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	ボード情報が取得された場合、ブール値「true」が返されます。それ以外の場合は「False」になります。
	LastBoardInfo _protVer	response.data. protVer	プロトコル・バージョンのバイト。返された値が 4（またはそれ以上）である場合、新たな設定値を取得するために GetConfigParamXxx を使用します。
	LastBoardInfo _dataBusWidth	response.data. dataBusWdt	データのバス幅。通常、ほとんどの MCU プラットフォームで 1 になります。一部の DSC プラットフォームでは 2 の場合もあります。
	LastBoardInfo _versionWord	response.data. globVer	FreeMASTER ドライバ・バージョンのワード。
	LastBoardInfo _cmdBuffSize	response.data. cmdBuffSize	FreeMASTER シリアル通信バッファのサイズ。

（次のページに続く）

（前のページからの続き）

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
	LastBoardInfo _recBuffSize	response.data. recBuffSize	レコーダ用のデータ・バッファのサイズ（プロトコル v4：レコーダ #0 の情報のみ）。
	LastBoardInfo _recTimeBase	response.data. recTimeBase	レコーダのベース・サンプル・レートを、プロトコル v3 用シリアル・ドライバのドキュメントに記載されている単位でエンコードしたもの。
	LastBoardInfo _description	response.data. descr	ボードまたは FreeMASTER のドライバ記述文字列。

6.5.71 GetConfigParamXxx

プロトタイプ：

```
GetConfigParamU8 ([in] name, [out] paramValue) GetConfigParamULEB ([in] name, [out] paramValue)
GetConfigParamString ([in] name, [out] paramValue)
```

説明：

ターゲット MCU アプリケーションから設定パラメータを読み取ります。FreeMASTER 通信プロトコル・バージョン 4 の導入に伴い、GetDetectedBoardInfo 関数はこの関数に置き換えられます。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite

入力：

引数	説明
name	設定パラメータの名前を含む文字列。以下に示したのは、デフォルトのパラメータ名の一覧です。アプリケーションでカスタム・パラメータを定義することもできます。 U8 のパラメータ： • F1：フラグ • RC：ターゲットに実装されているレコーダの数 • SC：ターゲットに実装されているオシロスコープの数 • PC：ターゲットに実装されているパイプの数 ULEB のパラメータ（ULEB は一般的な符号なし数値）： • MTU：コマンドや応答フレームの処理に使用される内部通信バッファのサイズ • BA：最適化されたメモリ読み取り / 書き込みコマンドによって使用されるベース・アドレス String のパラメータ： • VS：バージョン文字列 • NM：アプリケーション名文字列 • DS：説明文字列 • BD：ビルド日時文字列

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	設定値の取得に成功した場合、ブール値「true」が返されます。それ以外の場合は「False」になります。
paramValue	LastConfigParam_value	response.data	実際のパラメータ値（数値または文字列形式）。

6.5.72 GetCommPortInfo

プロトタイプ：

```
GetCommPortInfo ([in, optional] name)
```

説明

この関数を使用して、通信ポートの設定についての情報を取得できます。FreeMASTER Lite のクライアントが、設定されている ELF ファイルについての情報を調べる際にもこのメソッドを使用できます。

互換性：

✓ ActiveX、✓ JSON-RPC、✓ Lite（API は完全互換ではありません）

入力：

引数	説明
name	接続を識別する名前。デスクトップ FreeMASTER では、有効な接続が現時点で 1 つしか存在しないため、このパラメータは省略可能です。このビルトインの接続は、特殊な名前「preset」で識別することもできます。

出力：

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
戻り値	LastResult	response.xtra.retval	ボードが検出されて情報が取得された場合、True が返されます。それ以外の場合は False になります。
		response.data.name	接続の名前。
	LastCommPortInfo _description	response.data.description	接続の説明テキスト。
	LastCommPortInfo _connectString	response.data. connectString	FreeMASTER がポートを開く際に使用する完全接続文字列。
		response.data.elf	シンボル情報をロードする際に使用する ELF ファイル名。

（次のページに続く）

（前のページからの続き）

パラメータ	ActiveX アクセス	JSON-RPC アクセス	説明
	LastCommPortInfo _isPlugin	response.xtra.isPlugin	通信用プラグインが使用されている場合、True が返されます。
	LastCommPortInfo _isRS232	response.xtra.isRS232	ネイティブのシリアル・ポートまたは USB シリアル・ポートが使用されている場合、True が返されます。
	LastCommPortInfo _speed	response.xtra.speed	使用されるシリアル・ボーレート。isRS232 メンバが true の場合のみ有効です。
	LastCommPortInfo _portNum	response.xtra.portHint	COM ポート番号。isRS232 メンバが true の場合のみ有効です。
	LastCommPortInfo _portHint	response.xtra.portNum	シリアル・ポートを検索する際に使用される連想ワード。isRS232 メンバが true の場合のみ有効です。連想ワードは、シリアル・ポートの説明に含まれている単語であれば何でもよく、ユーザーが直接の COM ポート番号の代わりに指定できます。

6.5.73 EnableExtraFeatures

プロトタイプ：

```
EnableExtraFeatures([in] enable)
```

説明

FreeMASTER Lite とは互換性がないデスクトップ・アプリケーション「特別な」JSON-RPC メソッドを使用できるようにするには、FreeMASTER デスクトップ・アプリケーションの JSON-RPC インターフェースでこの関数を呼び出す必要があります。この関数を呼び出すことで実質的に、制御ページまたはクライアント・スクリプトが、FreeMASTER デスクトップ・アプリケーションでの使用のみを意図したものであると宣言したことになります。

最もよく使用される「特別な」メソッドとしては、EnableEvents、SubscribeVariable、UnSubscribeVariable、DefineSymbol、RunStimulators、SelectItem、OpenProject などがあります。

互換性：

✗ ActiveX、✓ JSON-RPC、✗ Lite（API は完全互換ではありません）

入力：

引数	説明
enable	特別な機能を有効にする場合は True。特別な機能を明示的に有効にする働きしかないため、False 値を使用することはできません。

出力：

なし

6.5.74 EnableEvents

プロトタイプ：

```
EnableEvents([in] enable)
```

説明：

JSON-RPC イベントの生成をアクティブにするには、FreeMASTER デスクトップ・アプリケーションの JSON-RPC インターフェースでこの関数を呼び出す必要があります。この関数を呼び出すことで、OnRecorderDone、OnCommPortStateChanged、OnBoardDetected、OnVariableChanged の各イベントを処理できるようになります。

互換性：

✗ ActiveX、✓ JSON-RPC、✗ Lite

入力：

引数	説明
enable	JSON-RPC サーバによるイベントの生成を有効にする場合は True。無効にする場合は False を指定します。

出力：

Promise オブジェクトは解決のみを行い、値を持ちません。

重要：この関数を使用するには、先に EnableExtraFeatures メソッドを呼び出しておく必要があります。

6.6 ActiveX のプロパティ

FreeMASTER の ActiveX メソッドには、入力パラメータと出力パラメータがあります。JavaScript などの言語では、入力パラメータの指定に関しては特に難しいことはありませんが、出力値の扱いについては注意が必要な場合があります。このような言語は出力値の受け取りをサポートしていないため、メソッドの呼び出し時には出力値を省略する必要があります。呼び出し後、すべての出力値は各種 ActiveX プロパティ（下表を参照）を使用してフェッチできます。

FreeMASTER の ActiveX インターフェースの各プロパティでは「GetLast...」関数も使用できるため、プロパティ値自体の読み取り時と同じ機能を利用できます。ActiveX オブジェクトのプロパティにアクセスすることのできないスクリプティング環境で、このような関数を使用できます。たとえば、メソッドの呼び出しにしか対応していない言語で「LastRetMsg」プロパティを読み取るには、GetLastRetMsg() 関数を使用します。

注意

JSON-RPC インターフェースを使用している場合、このようなプロパティは意味を持たなくなります。JSON-RPC サーバはすべての出力データを応答オブジェクトで返すため、クライアントはごく簡単に処理することができます。

表 1.FreeMASTER ActiveX オブジェクトのプロパティ

プロパティ名	説明
LastResult	最後に呼び出した ActiveX 関数の戻り値。
LastRetMsg	最後に呼び出した ActiveX 関数から返されたエラー・メッセージ（「retMsg」出力パラメータの値）。
LastVariable_vValue	最後に呼び出した ReadVariable メソッドの「numValue」出力パラメータで返された値。
LastVariable_tValue	最後に呼び出した ReadVariable メソッドの「textValue」出力パラメータで返された値。
LastMemory_data	最後に呼び出した各種 ReadMemory メソッドまたは各種 ReadXxxArray メソッドの「arrData」出力パラメータで返された値の配列。
LastRecorder_data	最後に呼び出した GetCurrentRecorderData メソッドまたは GetCurrentRecorderSeries メソッドの「arrData」出力パラメータで返された値の配列。
LastRecorder_serieNames	最後に呼び出した GetCurrentRecorderData メソッドの「arrSerieNames」出力パラメータで返された値の配列。
LastRecorder_timeBaseSec	最後に呼び出した GetCurrentRecorderData メソッドの「timeBaseSec」出力パラメータで返された値。

（次のページに続く）

表 1.FreeMASTER ActiveX オブジェクトのプロパティ（続き）

プロパティ名	説明
LastRecorder_state	最後に呼び出した GetCurrentRecorderState メソッドの「state」出力パラメータで返された値。
LastLocalFile_string	最後に呼び出した LocalFileReadString メソッドから返された「retString」テキスト・バッファ。
LastSymbolInfo_size	最後に呼び出した GetSymbolInfo メソッドから返された「symSize」値。
LastSymbolInfo_addr	最後に呼び出した GetSymbolInfo メソッドから返された「symAddr」値。
LastMemberInfo_size	最後に呼び出した GetStructMember メソッドから返された「membSize」値。
LastMemberInfo_offset	最後に呼び出した GetStructMember メソッドから返された「membOff」値。
LastAddressInfo_name	最後に呼び出した GetAddressInfo メソッドから返された「symbolName」値。
LastAddressInfo_exact	最後に呼び出した GetAddressInfo メソッドから返された「isExactMatch」値。
LastLinkName	最後に呼び出した EnumHrefLinks メソッドまたは EnumProjectFiles メソッドから返された名前。
LastPipe_data	最後に呼び出したいずれかの PipeRead メソッドから返されたパイプ・データ。
LastEnum_name	最後に呼び出した EnumVariables メソッドまたは EnumSymbols メソッドから返された列挙された名前。
LastBoardInfo_xxx	GetDetectedBoardInfo メソッドから返された複数のプロパティ。
LastCommPortInfo_xxx	GetCommPortInfo メソッドから返された複数のプロパティ。
LastMultipleVariables_json	ReadMultipleVariables または WriteMultipleVariables から返された JSON 文字列。

6.7 ActiveX と JSON-RPC のイベント

イベントは、非同期で発生する状態の変化などの事象をクライアントに伝える目的で FreeMASTER が使用する通知メッセージです。ActiveX インターフェースと JSON-RPC インターフェースにおけるイベントの扱いは極めて単純です。

Internet Explorer と ActiveX に関して、イベント・ハンドラを導入する最も確実かつ簡潔な方法は、VBScript のサブルーチンを使用して、実行中の処理を他の VBScript 関数または JavaScript 関数に転送するというものです。たとえば、「pcm」という FreeMASTER ActiveX オブジェクトの「OnBoardDetected」イベントを処理するには、以下のサブルーチンを使用します。

```
<script language="VBScript">
Sub pcm_OnBoardDetected()
    ' call JavaScript or VBScript handler function
    my_handler_code()
End Sub
</script>
```

JSON-RPC では、イベントが通知メッセージとしてサーバからクライアントに送信されます。JSON-RPC クライアントの実装方法に応じて、イベント処理の方法も異なる場合があります。FreeMASTER のインストールで配布される PCM ラッパー・オブジェクトでは「simple-jsonrpc-js」実装が使用されており、あらゆるサーバ・イベントをごく簡単に処理することができます。PCM コードによって、イベント・ハンドラの導入が一層簡素化されます。ユーザー・コードでは、イベント自体の名前を基に命名された PCM オブジェクトのプロパティに対してハンドラ関数を割り当てます。注意すべき点は、JSON-RPC のイベントを受信する前に、EnableExtraFeatures メソッドと EnableEvents メソッドを呼び出しておく必要があることです。OnBoardDetected イベントのコード例を以下に示します。

```
function my_handler_code()
{
    console.log("event received");
}
```

```
// enable FreeMASTER desktop application features
pcm.EnableExtraFeatures();

// enable events to be received
pcm.EnableEvents().then(() => {
    // install the handler
    pcm.OnBoardDetected = my_handler_code;
});
```

以降のセクションでは、FreeMASTER デスクトップ・アプリケーションによって生成されるイベントについて説明します。

6.7.1 OnRecorderDone

プロトタイプ：

```
OnRecorderDone()
```

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

説明：

このイベントは、アクティブなレコーダが新しいデータのダウンロードを完了したときに呼び出されます。その後、GetCurrentRecorderData メソッドまたは GetCurrentRecorderSeries メソッドを使用してデータを取得できます。

重要:JSONRPC インターフェースでイベントを受信する前に、EnableExtraFeatures メソッドと EnableEvents メソッドを呼び出しておく必要があります。

6.7.2 OnCommPortStateChanged

プロトタイプ：

```
OnCommPortStateChanged(portOpen)
```

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

説明：

このイベントは、通信ポートを開いた（portOpen が true）とき、または閉じた（portOpen が false）ときに呼び出されます。

重要:JSONRPC インターフェースでイベントを受信する前に、EnableExtraFeatures メソッドと EnableEvents メソッドを呼び出しておく必要があります。

6.7.3 OnBoardDetected

プロトタイプ：

```
OnBoardDetected()
```

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

説明：

このイベントは、FreeMASTER が有効なターゲット・ボードを検出し、そこから最初のハンドシェイク情報を取得したときに呼び出されます。

重要:JSONRPC インターフェースでイベントを受信する前に、EnableExtraFeatures メソッドと EnableEvents メソッドを呼び出しておく必要があります。

6.7.4 OnVariableChanged

プロトタイプ：

```
OnVariableChanged(varName, subscriptionId)
```

✓ ActiveX、✓ JSON-RPC、✗ Lite 説明：

このイベントは、サブスクライブしている変数の値が変化したときに呼び出されます。詳細については、SubscribeVariable メソッドの説明を参照してください。

重要:JSONRPC インターフェースでイベントを受信する前に、EnableExtraFeatures メソッドと EnableEvents メソッドを呼び出しておく必要があります。

6.7.5 OnCustomEvent

プロトタイプ：

```
OnCustomEvent(arg)
```

互換性：

✓ ActiveX、✓ JSON-RPC、✗ Lite

説明

このイベントは、同じ FreeMASTER ActiveX サーバまたは JSON-RPC サーバに接続しているクライアントが FireCustomEvent メソッドを呼び出した後に発生します。このイベントからは、呼び出し元によって指定された引数値が得られます。

重要:JSONRPC インターフェースでイベントを受信する前に、EnableExtraFeatures メソッドと EnableEvents メソッドを呼び出しておく必要があります。

6.8 スクリプト例

このセクションのサンプル・アプリケーションでは、さまざまなスクリプティング環境で ActiveX または JSON-RPC の関数を初期化して使用する方法を紹介しています。どのサンプルも、あらかじめターゲット・ボードを接続し、FreeMASTER シリアル・ドライバ・パッケージに同梱されるデフォルトのデモ・アプリケーションを実行しておく必要があります。

JSON-RPC の使用法に関する他の JS サンプル（NodeJS や Python の例を含んだ Jupyter Notebooks など）については、FreeMASTER Lite パッケージを参照してください。

6.8.1 HTML ページに埋め込まれた VisualBasic スクリプト

VisualBasic スクリプト（VBScript）は、かつて Internet Explorer が標準のブラウザであった頃に、ウェブ・ページで使用されるスクリプト言語として広く利用されていました。このテクノロジは他のウェブ・ブラウザではサポートされないため、今では当時ほどポピュラーではありませんが、ローカルでレンダリングされたページの背後に能動的なコードを与える目的で、依然として多くのユーザーに利用されています。VBScript は、ActiveX のメソッドから返される参照渡し（output）パラメータをネイティブでサポート数少ないスクリプト言語の 1 つです。VBScript では、FreeMASTER オブジェクトをごく簡単な方法で使用できます。

```
<html>
<head>
    <script language="VBScript">

    Function onError(err)
        'Erors are reported in the status field.
        document.getElementById("status").innerHTML = err
    End Function

    Function read_variable(name, span_id)
        'ReadVariable uses FreeMASTER variable object from current project. Use
        'ReadUIntVariable to access the memory directly using a symbol name.
        bSucc = pcm.ReadVariable(name, vValue, tValue, bsRetMsg)

        If bSucc then
            document.getElementById(span_id).innerHTML = tValue
```

```
        else
            onError("Error when reading variable " & name & ". " + bsRetMsg)
        End If
    End Function

    Function read_array(name, elemSize, length, span_id)
        'Arrays are accessed in memory directly, using a symbol name and element size.
        bSucc = pcm.ReadUIntArray(name, length, elemSize, arr, bsRetMsg)

        If bSucc then
            document.getElementById(span_id).innerHTML = ""
            For i = 0 to uBound(arr)
                document.getElementById(span_id).innerHTML =
                    document.getElementById(span_id).innerHTML & arr(i) & ", "
            Next
        else
            onError("Error when reading variable " & name & ". " + bsRetMsg)
        End If
    End Function

    Function write_variable(name, input_id)
        val = document.getElementById(input_id).value

        'WriteVariable uses FreeMASTER variable object from current project. Use
        'WriteUIntVariable to access the memory directly using a symbol name.
        bSucc = pcm.WriteVariable(name, val, bsRetMsg)

        If bSucc then
            document.getElementById("status").innerHTML = "Write of the " & name & " succeeded."
        else
            onError("Error when reading variable " & name & ". " + bsRetMsg)
        End If
    End Function

    </script>
</head>
<body>
    <!-- The main FreeMASTER ActiveX communication object -->
    <object id="pcm" height="0" width="0" classid="clsid:48A185F1-FFDB-11D3-80E3-00C04F176153">
    </object>

    <!-- User form -->
    Read var16 = <span id="var16_read">N/A</span> <input type="button" value="Read var16"
        onclick="call read_variable('var16', 'var16_read')" /> <br />
    Read var16inc = <span id="var16inc_read">N/A</span> <input type="button" value="Read var16inc"
        onclick="call read_variable('var16inc', 'var16inc_read')" /> <br />
    Read arr16 = <span id="arr16_read">N/A</span> <input type="button" value="Read arr16"
        onclick="call read_array('arr16', 2, 10, 'arr16_read')" /> <br />

    Write var16inc: <input type="text" id="var16inc_val" value="10" />
    <input type="button" value="Write var16inc"
        onclick="call write_variable('var16inc', 'var16inc_val')" /> <br />

    Status: <span id="status">No errors.</span> <br />
</body>
</html>
```

6.8.2 HTML ページに埋め込まれた JavaScript と JSON-RPC

この例では、FreeMASTER 3.0で導入された、JavaScriptでのJSON-RPC通信の使用法を紹介しています。このサンプルは、FreeMASTERのメイン・ウィンドウに埋め込まれた Chromium ビューのほか、外部で実行される Chrome ウェブ・ブラウザでも動作します。

```
<html>
<head>
    <!-- Get the information object named "FreeMASTER" (needs version 3.1.3 and above) -->
    <script type="text/javascript" src="fmstr://localapp/info.js"></script>
    <!-- load JSON-RPC and FreeMASTER wrapper object -->
    <script type="text/javascript" src="./simple-jsonrpc-js.js"></script>
    <script type="text/javascript" src="./freemaster-client.js"></script>

    <script type="text/javascript">
    var pcm;    // the main FreeMASTER communication object

    function main()
    {
        /* Desktop FreeMASTER listens on port 41000 by default, unless this is
         * overridden on command line using /rpcs option. FreeMASTER Lite
         is configurable. */
        // default address
        var rpcs_addr = "localhost:41000";

        // freemaster 3.1.3 and above provides the info about itself:
        if(typeof(FreeMASTER) != "undefined")
        rpcs_addr = FreeMASTER.rpcs_addr;

        pcm = new PCM(rpcs_addr, on_connected, on_error, on_error);
        pcm.OnServerError = on_error;
        pcm.OnSocketError = on_error;
    }

    function on_connected()
    {
        /* Typically, you want to enable extra features to make use of the full API
         * provided by desktop application. Leave this disabled and avoid any extra
         * features when creating pages compatible with FreeMASTER Lite. */
        //pcm.EnableExtraFeatures(true);
    }

    function on_error(err)
    {
        /* Erors are reported in the status field. */
        document.getElementById("status").innerHTML = err;
    }

    function read_variable(name, span_id)
    {
        /* ReadVariable uses FreeMASTER variable object from current project. Use
         * ReadUIntVariable to access the memory directly using a symbol name. */
        return pcm.ReadVariable(name)
            .then((value) => {
                document.getElementById(span_id).innerHTML = value.data;
            })
            .catch((err) => {
                on_error(err.msg);
            });
    }
```

```
    function read_array(name, elemSize, length, span_id)
    {
        /* Arrays are accessed in memory directly, using a symbol name and element size. */
        pcm.ReadUIntArray(name, length, elemSize)
            .then((value) => {
                document.getElementById(span_id).innerHTML = "";
                for(i=0; i<value.data.length; i++)
                    document.getElementById(span_id).innerHTML += value.data[i] + ", ";
            })
            .catch((err) => {
                on_error(err.msg);
            });
    }

    function write_variable(name, input_id)
    {
        var val = document.getElementById(input_id).value;

        /* WriteVariable uses FreeMASTER variable object from current project. Use
         * WriteUIntVariable to access the memory directly using a symbol name. */
        pcm.WriteVariable(name, val)
            .then(() => {
                document.getElementById("status").innerHTML = "Write of the " + name + " succeeded.";
            })
            .catch((err) => {
                on_error(err.msg);
            });
    }

    </script>
</head>
<body onload="main()">
    <!-- User form -->
    Read var16 = <span id="var16_read">N/A</span> <input type="button" value="Read"
        onclick="read_variable('var16', 'var16_read')" /> <br />
    Read var16inc = <span id="var16inc_read">N/A</span> <input type="button" value="Read"
        onclick="read_variable('var16inc', 'var16inc_read')" /> <br />
    Read arr16 = <span id="arr16_read">N/A</span> <input type="button" value="Read"
        onclick="read_array('arr16', 2, 10, 'arr16_read')" /> <br />
    Write var16inc: <input type="text" id="var16inc_val" value="10" />
    <input type="button" value="Write"
        onclick="write_variable('var16inc', 'var16inc_val')" /> <br />

    Status: <span id="status">No errors.</span> <br />
</body>
</html>
```

6.8.3 HTML ページに埋め込まれた JavaScript と ActiveX

JavaScript は、動的 HTML ページを作成するための最も一般的なスクリプト言語の 1 つです。FreeMASTER ActiveX コントロールで使用するためには、いくつかの特殊なテクニックが必要となります。

- JavaScript は、参照渡し（output）パラメータをネイティブでサポートしていません。このようなパラメータの使用を避け、出力値は Last で始まる各種プロパティ（LastResult など）から取得する必要があります。詳細については、「ActiveX のプロパティ」を参照してください。
- JavaScript では、ActiveXインターフェースで使用される「セーフ配列」形式とは互換性のない異なる形式の配列が使用されます。以下のサンプル・コードに示したような特殊な変換ルーチンを使用する必要があります。

```
<html>
<head>
    <script type="text/javascript">

    function on_error(err)
    {
        /* Erors are reported in the status field. */
        document.getElementById("status").innerHTML = err;
    }

    function read_variable(name, span_id)
    {
        /* ReadVariable uses FreeMASTER variable object from current project. Use
         * ReadUIntVariable to access the memory directly using a symbol name. */
        if(pcm.ReadVariable(name))
            document.getElementById(span_id).innerHTML = pcm.LastVariable_vValue;
        else
            on_error("Error when reading variable " + name + ". " + pcm.LastRetMsg);
    }

    function read_array(name, elemSize, length, span_id)
    {
        /* Arrays are accessed in memory directly, using a symbol name and element size. */
        if(pcm.ReadUIntArray(name, length, elemSize))
        {
            var rarr = pcm.LastResult ? pcm.LastMemory_data.toArray() : new Array();
            document.getElementById(span_id).innerHTML = "";
            for(i=0; i<rarr.length; i++)
                document.getElementById(span_id).innerHTML += rarr[i] + ", ";
        }
        else
        {
            on_error("Error when reading array " + name + ". " + pcm.LastRetMsg);
        }
    }

    function write_variable(name, input_id)
    {
        var val = document.getElementById(input_id).value;
        /* WriteVariable uses FreeMASTER variable object from current project. Use
         * WriteUIntVariable to access the memory directly using a symbol name. */
        if(pcm.WriteVariable(name, val))
            document.getElementById("status").innerHTML = "Write of the " + name + " succeeded.";
        else
            on_error("Error when writting variable " + name + ". " + pcm.LastRetMsg);
    }
    </script>
</head>
<body>
    <!-- The main FreeMASTER ActiveX communication object -->
    <object id="pcm" height="0" width="0" classid="clsid:48A185F1-FFDB-11D3-80E3-00C04F176153">
    </object>

    <!-- User form -->
    Read var16 = <span id="var16_read">N/A</span> <input type="button" value="Read var16"
```

```
        onclick="read_variable('var16', 'var16_read')" /> <br />
    Read var16inc = <span id="var16inc_read">N/A</span> <input type="button" value="Read var16inc"
        onclick="read_variable('var16inc', 'var16inc_read')" /> <br />
    Read arr16 = <span id="arr16_read">N/A</span> <input type="button" value="Read arr16"
        onclick="read_array('arr16', 2, 10, 'arr16_read')" /> <br />

    Write var16inc: <input type="text" id="var16inc_val" value="10" /> <input type="button"
        value="Write var16inc" onclick="write_variable('var16inc', 'var16inc_val')" /> <br />

    Status: <span id="status">No errors.</span> <br />
</body>
</html>
```

6.8.4 Excel の VisualBasic for Applications

FreeMASTER オブジェクトは、Excel の Visual Basic for Applications（VBA）プロジェクトから使用することもできます。コードを記述する前に、VisualBasic 開発環境の「ツール」の「参照設定」ダイアログで「FreeMASTER ActiveX Type Library」への参照を登録します。

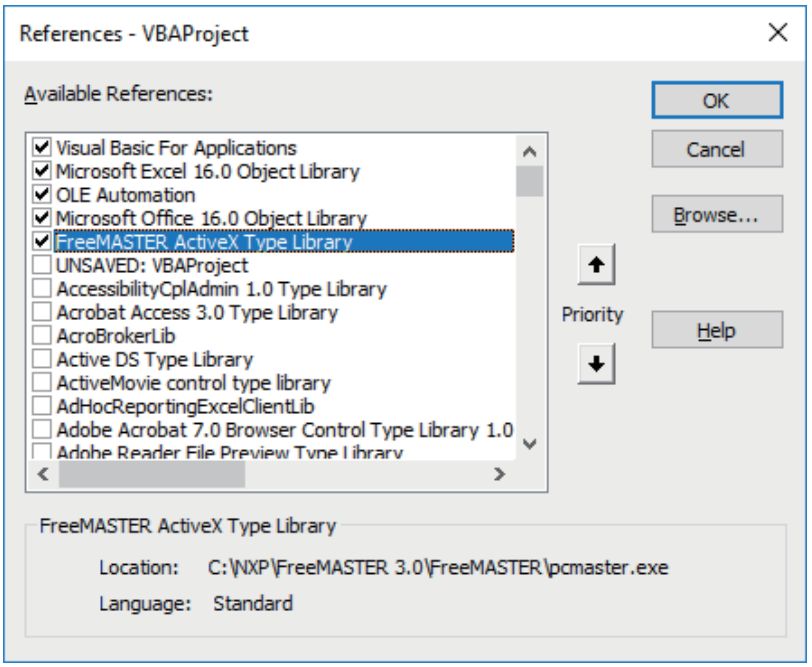

図 51. VBA で FreeMASTER のタイプ・ライブラリを登録する

VB コードで宣言する FreeMASTER ActiveX オブジェクトの名前は McbPcm です。タイプ・ライブラリが適切に登録された場合、FreeMASTER オブジェクトがエディタによって自動的に検出され、コードの入力中、オブジェクトのメソッドをインテリジェントに検索できるようになります。

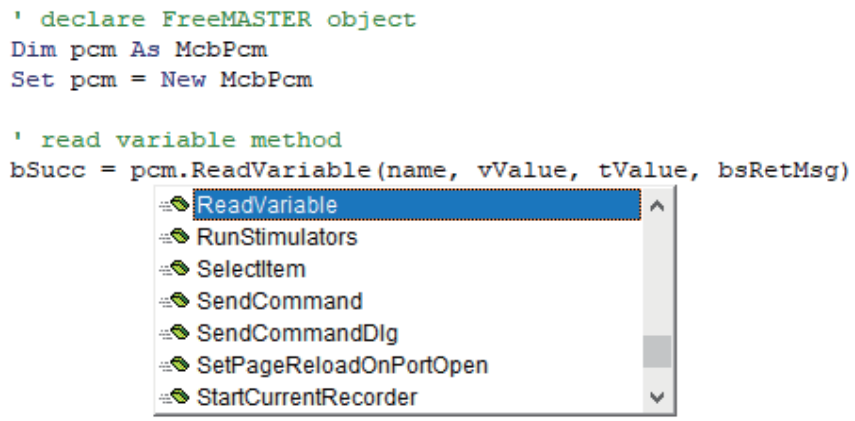

図 52. VBA 環境でのコードの編集

```
Dim pcm As McbPcm
Dim sht

Private Sub Form_Load()
    'initialize FreeMASTER object
    Set pcm = New McbPcm

    Set sht = Worksheets("sheet1")
End Sub

Sub OnError(err)
    'Show error in status cell
    sht.Range("B7").Value = "ERROR: " & err
End Sub

Sub ShowStatus(text)
    'Show status in status cell
    sht.Range("B7").Value = text
End Sub

Sub ReadVar16()
    'Macro read var16
    Call ReadVariable("var16", "B1")
End Sub

Sub ReadVar16inc()
    'Macro read var16inc
    Call ReadVariable("var16inc", "B2")
End Sub

Sub ReadArr16()
    'Macro read arr16
    Call ReadArray("arr16", 2, 10, "B3:K3")
End Sub

Sub WriteVar16inc()
    'Macro write var16
    Call WriteVariable("var16inc", "B5")
End Sub

Sub ClearArray()
    'Macro clear cells for array
```

```
   sht.Range("B3:K3").Value = ""
    ShowStatus ("Array was cleared")
End Sub

Sub ReadVariable(name, cell)
    Dim bSucc As Boolean

    If pcm Is Nothing Then
        Call Form_Load
    End If

    'ReadVariable uses FreeMASTER variable object from current project. Use
    'ReadUIntVariable to access the memory directly using a symbol name.
    bSucc = pcm.ReadVariable(name, vValue, tValue, bsRetMsg)
    If bSucc Then
        sht.Range(cell).Value = tValue
        ShowStatus ("Readvariable OK")
    Else
        ' something failed
        OnError (bsRetMsg)
    End If
End Sub

Sub ReadArray(name, elemSize, length, cell)
    Dim bSucc As Boolean

    If pcm Is Nothing Then
        Call Form_Load
    End If

    'Arrays are accessed in memory directly, using a symbol name and element size.
    bSucc = pcm.ReadUIntArray(name, length, elemSize, arr, bsRetMsg)
    If bSucc Then
        sht.Range(cell).Value = arr
        ShowStatus ("ReadUIntArray OK")
    Else
        ' something failed
        OnError (bsRetMsg)
    End If
End Sub

Sub WriteVariable(name, cell)
    Dim bSucc As Boolean
    Dim val As String

    If pcm Is Nothing Then
        Call Form_Load
    End If

    val = sht.Range(cell).Value

    'WriteVariable uses FreeMASTER variable object from current project. Use
    'WriteUIntVariable to access the memory directly using a symbol name.
    bSucc = pcm.WriteVariable(name, val, bsRetMsg)
    If bSucc Then
        ShowStatus ("Write of " & name & " was successful.")
    Else
        ' something failed OnError (bsRetMsg)
        End
    If
End Sub
```

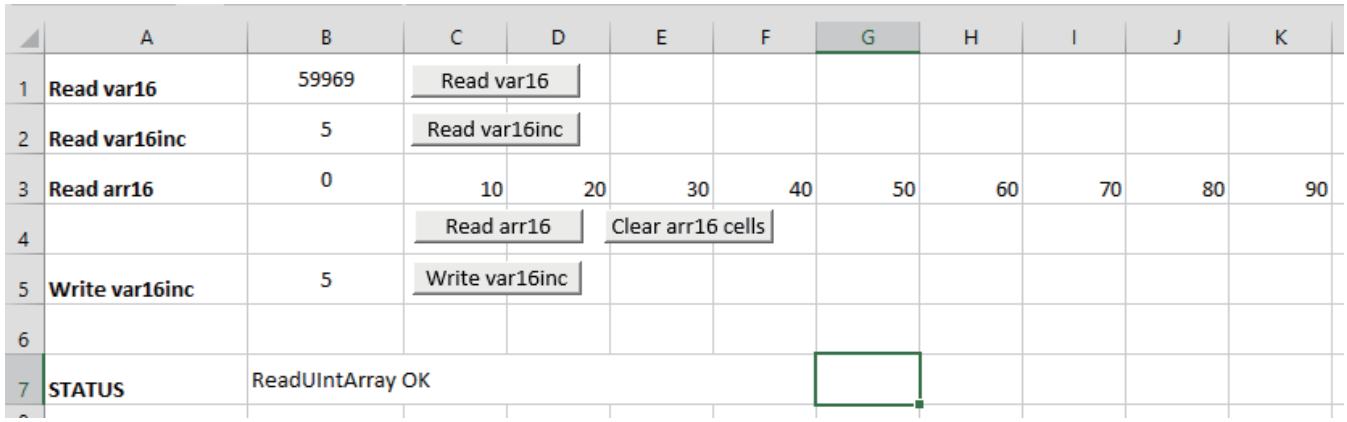

図 53. Excel フォーム

6.8.5 VBA と Excel セル関数

前のセクションでは、スプレッドシートに挿入されたアクティブ・ボタンを使用して Excel のマクロを呼び出しましたが、これは最適な方法ではないこともあります。このセクションでは、セル内のExcel関数を使用する方法を紹介します。そのまま利用できるサンプルが「example_cell_functions.xlsm」ファイルに用意されています。

読み取りまたは書き込みの対象となる変数の名前と値は、カスタム関数（読み取り用の PCMVAR、書き込み用の PCMWR、カスタム・イベント処理用の PCMFIRE）とともにスプレッドシートのセルに入力されています。

このスプレッドシートは、デモ・プロジェクトを実行する FreeMASTER をターゲット・アプリケーションに接続した状態で使用します。プロジェクトには、「var8」、「var16」、「var32」、「var16inc」、「var32inc」の各変数が定義されている必要があります。関数を実行するには、緑色のセル内の値に変更を加えます。また、「F9」（[Formulas/Calculate Now（数式 / 今すぐ計算）] を押すと、PCMVAR 関数が更新されて変数のイミディエイト値がターゲットから読み取られます。

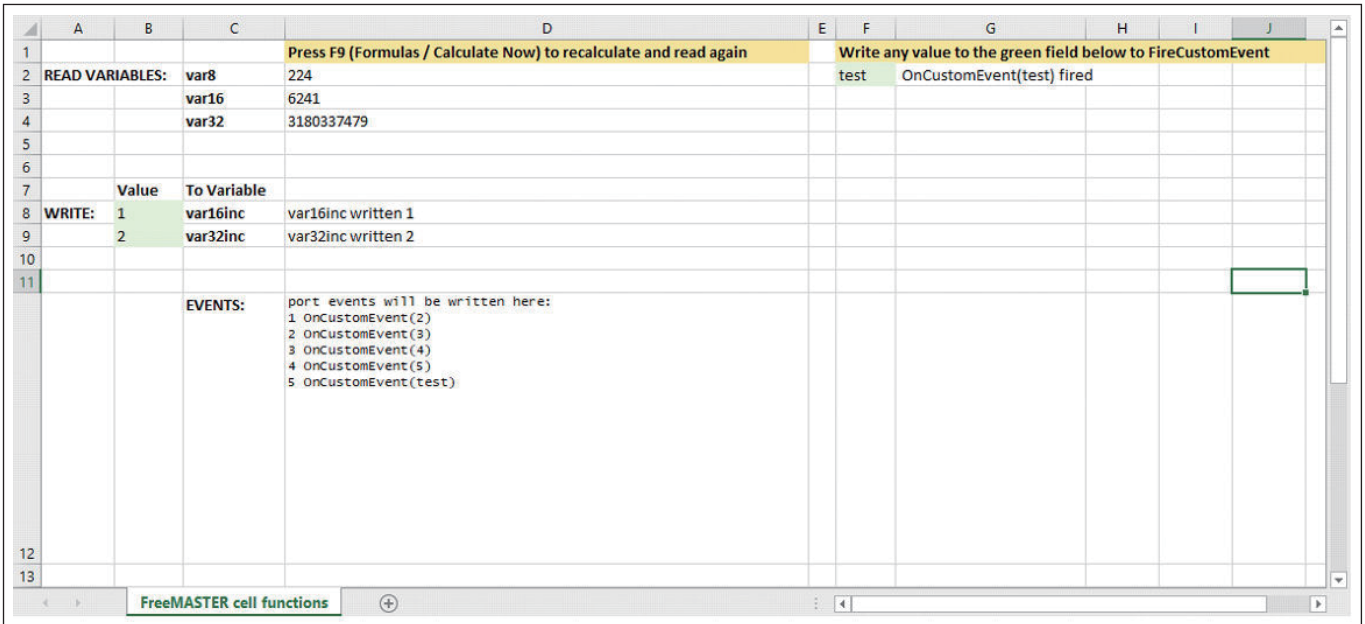

図 54. FreeMASTER のセル関数

「ThisWorkbook」セクションのコードは、FreeMASTER の「pcm」オブジェクトへの安全なアクセスを提供し、VBA 標準に従ってイベント・ハンドラを導入しています。

```
Public WithEvents pcm As McbPcm
Private evCounter As Integer
```

```
Public Function GetPCM(reset As Boolean)
    If reset Then
        Set ThisWorkbook.pcm = Nothing
    End If

    If pcm Is Nothing Then
        Set ThisWorkbook.pcm = New McbPcm
    End If

    Set GetPCM = ThisWorkbook.pcm
End Function

Private Sub pcm_OnCustomEvent(ByVal vArg As Variant)
    evCounter = evCounter + 1
    Set r = Sheet1.Range("events")
    r.value = r.value & CStr(evCounter) & " OnCustomEvent(" & CStr(vArg) & ")" & vbNewLine
End Sub

Private Sub pcm_OnCommPortStateChanged(ByVal vPortOpen As Variant)
    evCounter = evCounter + 1
    Set r = Sheet1.Range("events")
    r.value = r.value & CStr(evCounter) & " OnCommPortStateChanged(" & CStr(vPortOpen) & ")" & vbNewLine
End Sub

Private Sub pcm_OnBoardDetected()
    evCounter = evCounter + 1
    Set r = Sheet1.Range("events")
    r.value = r.value & CStr(evCounter) & " OnBoardDetected" & vbNewLine
End Sub
```

セルの関数は、スプレッドシートの Module1 のコードで定義します。

```
Function PCMRESET()
    ThisWorkbook.GetPCM (True)
End Function

' call ReadVariable
Function PCMVAR(name)
    Dim pcm As McbPcm
    Set pcm = ThisWorkbook.GetPCM(False)

    Application.Volatile

    ok = pcm.ReadVariable(name, vval, tval, msg)

    If (ok) Then
        PCMVAR = tval
    Else
        PCMVAR = msg
    End If

End Function

' call WriteVariable
Function PCMWR(name, value)
    Dim pcm As McbPcm
```

```
    Set pcm = ThisWorkbook.GetPCM(False)

    ok = pcm.WriteVariable(name, value, msg)

    If (ok) Then
        PCMWR = name + " written " + CStr(value)
    Else
        PCMWR = msg
    End If

End Function

' call FireCustomEvent with an argument
Function PCMFIRE(arg)
    Dim pcm As McbPcm
    Set pcm = ThisWorkbook.GetPCM(False)

    ok = pcm.FireCustomEvent(arg)

    If (ok) Then
        PCMFIRE = " OnCustomEvent(" & CStr(arg) & ") fired"
    Else
        PCMFIRE = "Error when firing OnCustomEvent"
    End If

End Function
```

6.8.6 Matlab m-script

このコードは、Matlab コンソールでテストします。

```
% create FreeMASTER ActiveX object
pcm = actxserver ('MCB.PCM');

% write 10 value into "var16" variable
bSucc = pcm.WriteVariable('var16inc', 10);
var16 = 0;

% read the "var16" variable as defined in FreeMASTER project
bSucc = pcm.ReadVariable('var16');
if bSucc
    var16 = pcm.LastVariable_vValue;
end
% show the value
var16

% configure matlab to accept safe array as single dimension
feature('COM_SafeArraySingleDim', 1) ;
% write 4 bytes to 'arr8' array. WriteMemory function expects SafeArray as input data
bSucc = pcm.WriteMemory('arr8', 4, {11;22;33;44})
% reads 4 bytes from memory at address of var32inc variable
bSucc = pcm.ReadMemory('var32inc', 4);
if bSucc
    % reads data of last call the ReadMemory()function.
    readMemResult = pcm.LastMemory_data
end
```

6.9 実行中の複数の FreeMASTER インスタンスにアクセスするスクリプト

複雑なスクリプトや制御ページから、実行中の複数の FreeMASTER インスタンスにアクセスすることができます。これは、コンピュータに接続されるターゲット・ボードが増え、そのすべてのデータをスクリプトで収集しなければならない場合に活用できます。

COM+/ActiveX インターフェースの場合：

- FreeMASTER アプリケーションの複数のインスタンスを、「Multi-FreeMASTER Class」（識別子は MCB.MULT）という名前の特殊な ActiveX オブジェクトで実行、停止、制御することができます。
- このオブジェクトは、Internet Explorer の制御ページ「<OBJECT id=" mult" name=" mult" height=" 1" width=" 1" classid=" clsid:A8E26DF5-85A1-4552-AD0E-6C8AA10641EA" ></OBJECT>」でインスタンス化できます。
- このオブジェクトにはメソッドが 1 つだけあります。「GetAppObject(instanceName, runMode, openProject)」というメソッドです。先ほどこのセクションの中で説明した API を使用して制御される共通の FreeMASTER オブジェクト・インスタンスを返します。
- パラメータ：
  - 「instanceName」：実行中のインスタンスの文字列識別子。「/sharex NAME」コマンドライン・パラメータを使用してインスタンスに名前を付けることができます。目的のインスタンスが見つからない場合は、新しい FreeMASTER インスタンスが起動されます。
  - 「runMode」：数値（1 = 通常、2 = 最小）。0x10（ポート閉）、0x20（ポート開）、0x30（ポート開、詳細出力）に追加できます。
  - 「openProject」：インスタンスが見つからず起動中である場合に開くプロジェクト。
- その場合、1 つの「MCB.MULT」オブジェクトを使用して、命名済みの複数の FreeMASTER インスタンスを実行、または接続することができます。これにより、すべてのインスタンスを 1 つのスクリプトから制御できます。
- FreeMASTER バージョン 3.1.3 では、MCB.MULT オブジェクトのサポートが 64 ビット・アプリケーションに拡張されているため、FreeMASTER 自体が 32 ビットの実行ファイルであったとしても、すべての 64 ビット・クライアントがその機能をフルに使用できます。

JSON-RPC インターフェースの場合：

- 各 FreeMASTER を異なる「/rpcs PORT」コマンドライン・オプションで実行し、特定のポートで JSON-RPC サーバを起動します。デフォルトのポートは 41000 です。
- 「/rpcs」オプションを使用しなかった場合、他の FreeMASTER インスタンスでは、41001 ～ 41016 の範囲のサーバ・ポートが選択されます。FreeMASTER 内の Chromium ビューをホストとする HTML コードでスクリプトが実行されている場合、ポート番号は、以下のように宣言された仮想的な JScript コードから検出されます。

```
<script type="text/javascript" src="fmstr://localapp/info.js"></script>
```

  詳細については、「FreeMASTER JSON-RPC インターフェース」および「HTML ページに埋め込まれた JavaScript と JSON-RPC」セクションのサンプル・コードを参照してください。
- JSON-RPC クライアントの機能を果たすスクリプトからは、必要な数のサーバ・インスタンスに適切なポートで接続し、接続インターフェースを使用して各インスタンスを制御することができます。

第 7 章
FreeMASTER の下層

FreeMASTERを使用して、ターゲット・マイクロコントローラ・アプリケーションの監視と制御を行うことができます。監視は、テキスト・ベースの変数ウォッチ・グリッドを使用して行うことも、オシロスコープとレコーダのチャートを使用してグラフィカルに行うこともできます。基本的な制御機能は、アプリケーション・コマンドを使用するか、ウォッチ・グリッドで変数に変更を加えることで実現されます。FreeMASTER のパイプを使用した双方向通信も利用できます。

この基本的な機能に加え、カスタム制御ページを HTML や JavaScript で作成して、追加のグラフィカル・オブジェクト（ボタン、ゲージ、スライダなど）を使用することもできます。ActiveX または JSON-RPC インターフェースを使用することで、制御ページから FreeMASTER のオブジェクトとそのプログラミング・インターフェースにアクセスできます。ソフトウェアのレイヤ構造としては、FreeMASTER の基盤となるフレームワークの上に、制御ページによって上位レイヤが実装されます。

制御ページ以外にも、FreeMASTER のテクノロジを使用してカスタム開発を行う方法はあります。FreeMASTER ソフトウェア自体も内部でいくつかのレイヤに分割されていますが、そのレイヤを相互に直接連携させることができます。そのようなカスタマイズは決して簡単ではなく、ある程度 NXP のサポートが必要になる場合があります。詳細については、FreeMASTER のコミュニティ・フォーラムでお問い合わせください。

7.1 通信ライブラリ

通信ライブラリは FreeMASTER デスクトップ・アプリケーションの下層にあり、ターゲット MCU アプリケーションとの通信全体を処理しています。動的にロードされるライブラリの形式になっているので、FreeMASTER 通信プロトコルを使用するカスタム・アプリケーションから使用することもできます。Windows におけるこのライブラリの名前は `mcbcXX.dll` です。

FreeMASTER バージョン 1.2 ～ 2.0 では、`mcbc12.dll` が使用されます。FreeMASTER 2.5 および 3.0 では `mcbc30.dll` が採用されています。FreeMASTER Lite でも、同じ API を備えた同様の通信ライブラリが使用されていますが、こちらはクロス・プラットフォームで動作するように更新されています。

この DLL は、Windows オペレーティング・システムのネイティブ・コードを生成する、ほぼすべてのプログラミング言語で利用できます。さらに、一部のスクリプト言語でも利用可能です。コンパイル時のライブラリのリンクに必要な *.lib ファイルおよびインターフェース・ヘッダ・ファイルは、FreeMASTER インストール済み環境の userdev ディレクトリにあります。

- mcbcom.h：メイン通信ライブラリ・ヘッダ・ファイル
- mcberr.h：エラー・コード定義ヘッダ・ファイル

注意

FreeMASTER プロトコルまたは FreeMASTER 通信ライブラリを使用したカスタム・アプリケーションを設計する際は、NXP のライセンス条項に従ってください。FreeMASTER ツールの使用に関して、このライセンスで許可されるのは、NXP Semiconductors のプロセッサを利用したアプリケーションとシステムのみです。

7.2 通信プラグイン

通信プラグインは、FreeMASTER で使用可能な動的にロードされるライブラリ（つまり FreeMASTER の通信ライブラリ）であり、ターゲットとの代替的な物理的通信を実現するために使用されます。FreeMASTER 通信ライブラリでは、低レベルの RS-232、またはオペレーティング・システムに用意されている USB 通信インターフェースを使用して、ネイティブのシリアル通信プロトコルを直接実装しています。プロトコル・フレームを送信したり応答フレームを受信したりするための一般的なルーチンが、このライブラリに含まれています。その他の物理通信手段もプラグインの形で用意できるため、特定の物理媒体に適した代替的な送受信実装を実現できます。

FreeMASTER にはいくつかの通信プラグインが付属しており、メイン・アプリケーションと併せて自動的にインストールされます。

- 「CAN 通信用プラグイン」は、CAN メッセージを使用してシリアル・プロトコル・フレームを送受信します。FreeMASTER ドライバも、ターゲット MCU アプリケーションの CAN インターフェースを使用するようにが設定する必要があります。
- 「BDM/JTAG 通信用プラグイン」は、シリアル・プロトコルをエミュレートし、変数アクセス・プロトコル・フレームを捕捉して、同じ動作をエミュレートします（JTAG インターフェースを介した直接メモリ・アクセスが使用されます）。このプラグインを使用する場合、ターゲット MCU アプリケーションでプロトコル・ドライバを使用する必要がありません。すべてのメモリ・アクセスは、MCU の CPU が介在することなくバックグラウンドで実行されます。

高度な機能（レコーダ、パイプなど）は、「連携」するドライバ・コードが MCU アプリケーションに存在しないため、このプラグインでは正しく機能しません。

- 「パケットドリブン BDM 通信用プラグイン」は、ターゲット・メモリに直接アクセスする代わりに、JTAG インターフェースを通信インターフェースとして使用します。この場合も、FreeMASTER ドライバがターゲット MCU アプリケーションに存在し、また、PD-BDM 通信用に設定されていることが必要となります。
- 「ネットワーク通信用プラグイン」は、ターゲット組み込みアプリケーションに対するTCPまたはUDPでの直接接続をサポートします。アプリケーションが TCP/IP スタック（lwIP など）を使用していること、また、FreeMASTER プロトコルをサポートしていることが必要です。MCUXpresso SDK v2.10（以降）に付属するデモ・アプリケーション「fmstr_net」および「fmstr_wifi」を参照してください。
- ネットワーク・プラグインは、ローカルまたはリモート・コンピュータで実行される RTT プロトコルを使用して、SEGGER J-Link デバッガ・プローブでの通信もサポートします。RTT 通信とアクティブなデバッガ・セッションは同じ J-Link デバイスを共有し、同時に使用することができます。

FreeMASTER デスクトップ・アプリケーションと FreeMASTER Lite は、どちらも通信プラグインをサポートしますが、そのテクノロジに以下の違いがあります。

- FreeMASTER は、Microsoft COM+ テクノロジを使用してプラグインを実装し、そのメソッドを呼び出します。カスタム通信プラグインを設計するために必要なインターフェース・ファイルは、FreeMASTER インストール済み環境の userdev ディレクトリにあります。
  — mcbcom_i.idl：メイン COM+ インターフェース定義ファイル
  — mcbcom_i.c and mcbcom_i.h：インターフェースおよび関連する GUID 識別子の定義
- FreeMASTER Liteでは、一般的な localhost RPC over TCP プロトコルが使用されます。

カスタム・プラグイン・モジュールの開発は難易度の高い作業です。NXP FreeMASTER コミュニティ・フォーラムを利用して協力を求めたり、開発計画について話し合うなどしてください。

注意

FreeMASTER プロトコルまたは FreeMASTER 通信ライブラリを使用したカスタム・アプリケーションを設計する際は、NXP のライセンス条項に従ってください。FreeMASTER ツールの使用に関して、このライセンスで許可されるのは、NXP Semiconductors のプロセッサを利用したアプリケーションとシステムのみです。

第 8 章
参照資料

- FreeMASTER の使用：シリアル・ドライバの実装（ドキュメント AN4752）
- FreeMASTER タイム・デバッグ・ツールとマイクロコントローラ用 CodeWarrior v10.X プロジェクトの統合（ドキュメントAN4771）
- MC56F847xx および MC56F827xx DSC ファミリ用フラッシュ・ドライバ・ライブラリ（ドキュメント AN4860）
- FreeMASTER ツール・ホーム：www.nxp.jp/freemaster
- FreeMASTER コミュニティ・エリア：community.nxp.com/community/freemaster
- MCUXpresso SDK ホーム：www.nxp.jp/mcuxpresso
- MCUXpresso SDK Builder：mcuxpresso.nxp.com/en

第 9 章
変更履歴

表 2 は、初期リリース以降、本書に加えられた変更をまとめたものです。

表 2. 変更履歴

リビジョン	日付	説明
1.0	03/2006	限定初期リリース
2.0	09/2007	FreeMASTER バージョンに合わせて更新。新しいフリースケール・ドキュメント・テンプレートを使用。
2.1	12/2007	新しい Fast Recorder 機能とその API に関する説明を追加。
2.2	04/2010	MPC56xx プラットフォーム用のサポートを追加。CAN インターフェース用の新しい API を追加。
2.3	04/2011	Kxx Kinetis プラットフォームと MQX オペレーティング・システムのサポートを追加。
2.4	06/2011	シリアル・ドライバの更新。USB CDC インターフェースのサポートを追加。
2.5	08/2011	パケット・ドリブン BDM インターフェースを追加。
2.7	12/2013	FLEXCAN32 インターフェース、バイト・アクセス、isr コールバック設定オプションを追加。
2.8	06/2014	古いライセンス・テキストを削除（最新のライセンスについてはソフトウェア・パッケージの内容を参照）。
2.9	03/2015	ドライバ・バージョン 1.8.2 および 1.9 に合わせて更新（FreeMASTER パイプ、TSA Active Content、LIN トランスポート・レイヤ・サポート、DEBUG-TX 通信のトラブルシューティング、Kinetis SDK サポート）
3.0	08/2016	ドライバ・バージョン 2.0 に合わせて更新。MPC56xx、MPC57xx、KEAxx、S32Kxx の各プラットフォームのサポートを追加。新しい NXP ドキュメント・テンプレートと新しいライセンス契約を使用。MCAN インターフェースを追加。インストール先のフォルダ構造を再編成。
4.0	04/2019	FreeMASTER v3.0 および MCUXpresso SDK 2.6 の一部としてリリースされたドライバに合わせて更新。新しい V4 シリアル通信プロトコルおよび新しい設定オプションに合わせて更新。このバージョンのドキュメントでは、S08、S12、ColdFire、Power など、レガシー・プラットフォームに関連した古い情報の多くが削除されています。
4.1	04/2020	MCUXpresso SDK 2.8 に同梱される FreeMASTER ドライバに合わせた軽微な更新。
4.2	09/2020	サンプル・アプリケーションの説明と MCUXpresso Config Tools に関する情報を追加。パイプに関連した API の説明を修正。
4.3	10/2021	RTT 通信、MCB.MULT オブジェクト、CEF の「info.js」仮想ファイルについての情報を追加。FreeMASTER 3.1.3 リリースに伴う。
4.4	03/2022	FreeMASTER バージョン 3.1.4 に伴う。「HTML とスクリプティング」に、新しい ActiveX/JSON-RPC API の説明を追加。

連絡先：

ホームページ：

nxp.jp

ウェブ・サポート：

nxp.jp/support

保証および責任の制限 ― 本文書内の情報は、システムおよびソフトウェアの実装者が NXP 製品を使用できるようにするためにのみ提供されます。本文書内の情報に基づき集積回路を設計または製造することを許諾する明示的または黙示的な著作権ライセンスが本文書に基づき付与されることは一切ありません。NXP は、新たに通知することなく、本文書に記載されている製品に変更を加える権利を留保します。

NXP は、自社製品の特定目的適合性に関するいかなる保証も表明も行わず、何らかの製品または回路の用途または使用に起因するいかなる責任も負いません。NXP は、付随的損害、派生的損害を限定することなく含む一切の責任を明確に否認します。NXP のデータシートおよび／または仕様において提供される場合のある「標準的」パラメータは、用途によって異なる可能性があり、実際にも異なっています。また、実際の性能は、時間とともに変化する場合があります。「標準的」パラメータを含むすべての動作パラメータは、お客様の専門技術者によって、お客様それぞれの用途に適したものであることを検証していただく必要があります。NXP は、自社の特許権や第三者の権利に基づき、いかなるライセンスも譲渡しません。NXP は、下記のアドレスでご確認いただける標準的な販売条件に従って製品を販売します。アドレス：nxp.jp/SalesTermsandConditions

変更を加える権利 - NXP Semiconductors は、本文書に記載されている情報に随時、予告なく変更を加える権利を留保します。それらの情報には仕様や製品の説明を含みますが、それに限定されません。本文書は、本文書の発行前に提供されたすべての情報に優先し、それらを置き換えます。

セキュリティ ― お客様は、文書化されていない未確認の脆弱性が NXP のすべての製品にある可能性があると理解しているものとします。お客様のアプリケーションおよび製品に対するそうした脆弱性の影響を低減することを目的として、お客様は自らのアプリケーションおよび製品のライフサイクル全体を通じて、その設計および運用に責任を負います。お客様の責任は、お客様のアプリケーションで使用される NXP 製品がサポートする他のオープン・テクノロジや独自テクノロジにも及びます。いかなる脆弱性に対しても NXP は一切責任を負いません。お客様は、NXP からのセキュリティ・アップデートを定期的に確認し、適切な対策を講じる必要があります。お客様は、意図する用途の規則、規制、標準に最も合ったセキュリティ機能を備えた製品を選択するものとし、NXP が提供する情報またはサポートにかかわらず、自らの製品に関して最終的な設計決定を行い、その製品についての法律、規制、およびセキュリティに関連する要件の順守につき全責任を負います。NXP の製品セキュリティ・インシデント・チーム（PSIRT）（お問い合わせ先：PSIRT@nxp.com）が、NXP 製品のセキュリティ脆弱性に関する調査と報告、そして解決策のリリースを管理しています。

詳細については、http://www.nxp.jp を参照してください。

営業所の住所については、salesaddresses@nxp.com 宛てにメールでお問い合わせください。

リリース日：03/2022

ドキュメント ID：FMSTERUG