日本語ユーザー・マニュアル

BeeHive204

USB インターフェースと ISP 機能付き
マルチプログラミング・システム

ELNEC s.r.o.
Presov, Slovakia
September 2018

このマニュアルの使い方

このマニュアルでは制御プログラムのインストール方法とプログラマの使用方法について説明しています。ユーザーにはPCやソフトウエアのインストールに関する経験があることが前提です。コントロール・プログラムをインストールしたら印刷されたユーザー・マニュアルではなく、コントロールプログラム内の状況依存ヘルプを参照することをお勧めします。リビジョンは印刷されたユーザー・マニュアルの前に状況依存ヘルプに実装されています。.

お買い上げのユーザー様

Elnec プログラマをお買い上げ頂きましてありがとうございます。

現在のバージョンが最新でない場合は、Elnec WEB サイト (www.elnec.com)のサポート/ダウンロード・セクションから マニュアルの最新のバージョンをダウンロードして下さい。

.

安全にご使用頂くための使用上の注意と免責事項:

保証期間は納入後 3 年です。但し、保証期間内においても天災，操作ミス、ZIF ソケット，アダプターの予期せぬ消耗や汚れによる接触不良による不具合には対しては保証しかねます。全ての操作、保守と修理サービスは以下の安全情報に従うことが要件となります。**機器の理解不足や間違った使用により起こる予期しない問題による直接・間接のトラブルの責任を取ることはありません。**

このマニュアルに記載の製品データやプログラム、又は、アルゴリズム(ソフトウエアのバグ)等の使用に起因する損害は責任を負いません。

1. このユーザー・マニュアルは英文マニュアルからの転載であり、且つ、ご使用頂くためのヘルプとして頂くための資料です。記述に誤りがあると思える場合は英文で再確認して下さい。誤記、又は、不十分な記述により損害が発生した場合は ELNEC s.r.o, 有限会社データダイナミクスはその責任を負いません。
2. 以下の注意事項を守ってご使用下さい。
   - 誤った操作でデバイスを破損させる可能性があります。
   - 静電気による破損を防止するために付属の静電防止様リストストラップを使用する等の対策を履行して下さい。
   - ソケット、アダプター、デバイスのクリーニングに常に注意して下さい。埃、汚れ等はプログラミング・エラーの原因になります。
   - 誤って落としたり異物や液体等が内部に入った場合は直ぐに AC アダプターを抜いて下さい。
   - 高温な場所での長時間なご使用は出来るだけ避けて下さい。
3. プログラマーは USB ケーブルを使用してホスト・コンピュータに接続する必要があります。両方の USB プラグがコンピューターに接続されていることを確認し、利用可能な USB 電源が正常にプログラマーを動作させるのに十分であることを確認して下さい。
4. 可燃性ガスの近くで製品を操作しないで下さい。高温と湿度も避けて下さい。
5. 製品のケースを外したり内部部品の交換をしないで下さい。
6. アダプターはユーザーの取扱いによるピンの損傷や不十分なメンテナンス等の理由から消耗品と考えられています。ソケットが製造業者によって指定された作動の定格数を超えて使用されている場合、ソケットは接触不良に起因するエラーが起こります。これは接触エラー数の増加につながり書き込みの失敗を起こします。
7. 本製品を大量生産やマスターの生産などに使用する場合は完成品を組み立てる前に適切なテスト対策を必ず行ってください。ソフトウェア製品を使用したり操作することに起因するすべてのリスクはユーザーが負うものとします。
8. このマニュアルは参考であり、製品の仕様とマニュアルは予告なく変更することがあります。
9. このマニュアルに記載されている会社、又は、商標はそれぞれの所有者に帰属します。

プログラマーを使用して IC を操作する場合には高周波信号を使用してデバイスにアクセスします。適切なプログラミングと高い歩留まりを確保するためには次のことを守って下さい:

コンピューターやソフトウエアの誤操作等により不注意でバッファ内容を変更している場合もありますので、作業開始時にバッファを空にしてからファイルをロードしたり、チェックサムを確認を必ず行って下さい。

このマニュアルで使用されている表現

コントロール・プログラムのファンクションは **Load**, **File**, **Device** などの太字で表示されています。**<F1>**,の様に< >内はコントロール・キーです。

このマニュアルで使用されている用語:

Device - デバイス

プログラマブル IC 又は、プログラマブル・デバイス

ZIF socket - ZIF ソケット

ターゲット・デバイスを装着するために使われる ZIF [Zero Insertion Force] ソケット

Buffer - バッファ

コントロール・ソフトウエアにロードされたデータが PC のメモリー。リードされたデータや読み込みデータがバッファ・メモリーに置かれます。ファイルの保存はバッファ内のデータが保存されます。

USB ポート - USB プリンター・ポート

PC の USB ポートに接続して使用されます。

HEX data format – ヘキサ・データ形式

標準のテキスト・ビューワーで読むことができる、データ・ファイルの形式の1つです。バイト 5AH は '5' と 'A'のキャラクターとしてストアされます。それは、バイト 35Hと41H を意味します。

この HEX ファイルの1つの行（1レコード）が開始アドレス、データ・バイトとチェックサムで安全に保たれるすべてのレコードを含んでいます。

最低必要な PC の環境

	2x BeeHive204	BeeHive204
OS - Windows	XP	XP
CPU	C2D 2,6GHz	C2D 2,6GHz
RAM [MB]	1000	1000
free disk space [MB]	500	500
USB 2.0 high speed	●	●
LPT	-	-
CDROM	●	●

推奨 PC 必要環境

	2x BeeHive204	BeeHive204
OS - Windows	Win 7-10	Win 7-10
CPU	Core i5 *1	Core i3 *1
RAM [MB]	4000	2000
free disk space [MB]	2000	1000
USB 2.0 high speed	-	●
2x USB 2.0 high speed controllers	●	-

モニタ解像度 1024 x 768 を推奨
*1 は、それ又は、それ以上を意味します。
上記は S/W バージョン 3.38(2/2018)以降のご使用を前提
もし、2 つのプログラマーを 1 台の PC で使用頂くためには，*各プログラマーを別々の USB2.0 High speed controller(USB EHCI)に接続してご使用されることを強くお薦めします。*

クイック・スタート

プログラマー・ハードウエアのインストール

- 付属のケーブルを使用してプログラマの USB ポートを PC の USB ポートに接続します。
- 電源アダプタ(電源コード)のコネクタをプログラマに接続しプログラマをスイッチでオンにします。

※重要:USB の場合はソフトウエアがインストールされてから USB を接続して下さい。

プログラマー・ソフトウエアのインストール

CD が付属しておりますが、ご購入時はご使用時には
http://www.elnec.com/downloads.php から最新のソフトウエアをダウンロードされることをお薦めします。

コントロール・プログラムの実行

コントロール・プログラマーを実行するために PG4UW.EXE を立ち上げて下さい.

即ち、  をダブルクリックして下さい。

コントロール・プログラム Pg4uw をスタートしますと自動的にすべてのポートと接続されている ELNEC プログラマーをスキャンします。プログラム Pg4uw は ELNEC の全てのプログラマに対して共通です。

メニュー**"File"**はソースファイルの操作、設定とディレクトリーを見たり、ドライブを変更、ロードと保存するファイルとプロジェクトのバッファの開始と終了アドレスを変更します。
メニュー **"Buffer"** はバッファ操作、ブロック操作、バッファの一部をストリングスでフィル、イレース、チェクサムと他の項目(ストリングの検索と置き換え、印刷..).の編集とビューのために使用します。
メニュー **"Device"**は選択されたプログラマブル・デバイスで動作させるために使用します。: 選択[select], read[読み込み], ブランクチェック[blank check], プログラム[program], ベリファイ[verify], イレース[erase]とプログラミング・プロセス、シリアライゼーションと関連ファイルのコントロールのセッティング に使用します。
メニュー **"Programmer"**はプログラマーで使用する機能のための使用されます。
メニュー **"Options"**は各種デフォルト設定の確認や変更のために使用されます。
メニュー **"Help"**はそのバージョンでサポートされているデバイスとプログラマーとプログラム・バージョンについての情報を見るために使用されます。

デバイスのプログラム(書込み)

1. デバイスを選択:  をクリック
2. データをバッファへロード:

   a) ファイルからの場合:  をクリック

b) デバイスからの場合：ZIF ソケットにデバイスを装着し、  をクリック

*読み取った後はデバイスは抜いて下さい。

3. ターゲット(書き込みたい)デバイスを ZIF ソケットに装着
4. デバイスがブランクかをチェックするために  をクリック
5. デバイスにプログラム（書込み）  をクリック
6. デバイスに書き込んだデータとバッファのデータを照合(Verify)するために  をクリック

イントロダクション

BeeHive204 は最少の操作で大量生産プログラミングのために設計された非常に高速なユニバーサル 4x 48-pin 駆動同時マルチプログラミング・システムです。チップは論理的に最大プログラミング速度でプログラムされます。ビルト-イン ISP コネクターを使用することでイン-サーキットでチップをプログラムすることも出来ます。BeeHive204 は BeeProg2 プログラマー・ハードウエアをベースとして 4 つの独立したユニバーサル・プログラミング・モジュールで構成されています。従って、各ソケットは同期(同時プログラミング・モード)で実行することが出来ます。他のソケット(*以下、サイトと呼びます)でプログラミング中に他のソケットでチップの交換を行うことが出来ます。

ハンズ-フリー操作: 同期と同時操作がチップを装着した途端にプログラミングが開始することが出来ます。オペレーターは単にプログラムされたチップを取り除き、そして、新しいチップを装着するだけで操作出来ます。

BeeHive204 は Windows XP/Vista/7/8(32bit と 64bit)ベースのプロフェッショナルなモバイル・アプリケーションのためにも使用できる比較的小さくてパワフルなユニバーサル・プログラマーです。さらに、BeeHive204 は特別なモジュールを使用することなくマイコン, フラッシュ, GAL 等を幅広くサポートしています。供給電源とプログラミング電源がデジタルに制御され、そして、H のレベルを制限することが出来ますので、プログラマーは 1.8V からの真の低電圧デバイスにも使用出来ます。

BeeHive204 のインターフェースは IBM 互換の PCT 又は、同等以上の USB(2.0)ポート経由のポータブル又は、デスクトップ PC で動作します。

BeeHive204 の **FPGA ベース**完全コンフィギャラブル 48 **パワフル TTL ピンドライバー**がソケットの各ピンに H/L/プルアップ/プルダウン と 読み取り機能を供給。**高品質、高速**回路を持った先進のピン・ドライバーが、サポートされた全デバイスのためにオーバーシュートなしで、又は、グラウンド・バウンス無しでシグナルを供給します。ピン・ドライバーは 1.8V まで操作できますので、すべての低電圧デバイスをプログラムすることが出来ます。

ビルト-インのプロテクション回路が主電源のエラー、通信エラー又は、PC のフリーズによるプログラムされるデバイスのダメージを軽減します。プログラマーのハードウエアはいつでもピン・ドライバー、すべての電圧状況、プログラマーと PC 間のタイミングと通信をコントロールし、セルフ・テストに十分なリソースを提供します。

プログラミングのベリファイは VCCP のマージナル・レベルにより行われますのでプログラミング不良をなくし、データ保持が保障されます。

BeeHive204 はプルダウン・メニュー、ホット・キーとオンライン・ヘルプを持った**使い易いコントロール・プログラム**によりデバイスをクラス、マニファクチャラー又は、マニファクチャラー名をパーツ番号により選択することが出来ます。標準のデバイス操作機能(読み出し, ブランク・チェック, プログラム, ベリファイ) はいくつかのテスト機能と一緒に完了されます。プログラムは自動ファイル・フォーマットの検知と変換を含む、バッファとファイルの使用機能があります。

ソフトウエアは **Auto-increment 機能** を提供していますので、プログラムされるデバイスにシリアル番号を個々に割り当てることが出来ます。この機能は単にバッファ内のシリアル番号をソケットに新しいデバイスが挿入される度にインクレメントして行きます。さらに、この機能によりユーザーはシリアル番号やファイルからプログラムされたデバイスの ID 署名を読むことが出来ます。

BeeHive204 は自動ファイル形式認識機能で入力ファイルを扱います。
Jam files(JEDEC standard JESD-71)は Jam プレイヤーによってインタープリットされます。Jam files は各プログラム・デバイスのメーカーから供給された設計ソフトウエアによって生成されます。

VME files は VME プレイヤーによってインタープリットされます。VME file は SVF ファイルの圧縮されたバイナリー・バリアントでありハイ-レベル
IEEE 1149.1 バス・オペレーションを含んでいます。
VME files は各プログラム・デバイスのメーカーから供給された設計ソフトウエアによって生成されます。

ファイルのローディング中に行われます。 ソフトウエアは全ての知られているデータ形式はサポートされています。
バイナリー(RAW), インテル(拡張)HEX(Intel, Intel EXT), モトローラ, MOS テクノロジー, Exormax, Tektronix, ASCII-SPACE-HEX, ASCII HEX Altera POF, JEDEC(ver. 3.0.A), eg. from ABEL, CUPL, PALASM, TANGO PLD, OrCAD PLD, PLD Designer ISDATA 等々. JAM(JEDEC STAPLE Format), JBC(Jam STAPL Byte Code), STAPLE(STAPL File)JEDEC standard JESD-71 VME(ispVME file VME2.0/VME3.0)

チップは ZIF 又は、ISP コネクター(IEEE 1149.1 Joint Test Action Group (JTAG)インターフェース)でプログラムされます。

JTAG チェイン経由で複数のデバイスのプログラムとテストが可能です : JTAG chain (ISP-Jam) 又は、JTAG chain (ISP-VME).

フリー・アディショナル・サービス:

何故PG4UWの最新バージョンを使うのが重要か?

- 半導体メーカーは継続的に製品開発設計と製造において柔軟性、品質とスピードの必要性をサポートするために新しい技術によって製造された新しいパッケージ・タイプを持つ新しいデバイスを紹介しています。これらのベースを維持しアップデートするために我々は年間5000以上の最新のデバイスを制御プログラム[PG4UWコントロール・ソフトウエア、又は、以後、プログラムとも表記]にインプリメントしています。
- さらに、既存のデバイスにもその努力維持するために、又は、その技術的特性とプロセスの歩留まりを向上させるためにいくつかの変更が行われています。これらの変更がしばしばプログラミング・アルゴリズムに影響しますので頻繁にアップグレードする必要があります(プログラミング・アルゴリズムは特定のターゲット・デバイスにデータをプログラムする方法をプログラマーに指示命令のセットです)。従って、**プログラミングのプロセスで最新のアルゴリズムを使用して高品質の結果を得るための重要なキーとなります。** 多くの場合、古いアルゴリズムでもデバイスをプログラム出来る一方、それらの場合は最適なアルゴリズムで可能なデータ保持のレベルを提供することが出来ません。最新のアルゴリズムを使用していないとプログラミングの歩留りの減少や、多くの場合、プログラミング時間を増加させたりプログラムされたデバイスの長期信頼性に影響を与える可能性があります。
- われわれもミスをしている場合もあります…
  ELNECは常にそれらの新しいデバイスに対応した最適なプログラミング・アルゴリズムを提供出来る様に更新とデバイス・サポートに努力しています。
- フリー・テクニカル・サポート (E-mail).
- Webサイトを経由してのフリー・ソフトウエア・アップデート.

フリー・ソフトウエア・アップデートは
www.elnec.comからダウンロード出来ます。

ELNECは下記のサービスも提供しております。

• **AlgOR** (アルゴリズム・オン・リクエスト) サービスによりサポートされていないプログラミング・デバイスのサポートをElnecソフトウエアにより受けることが出来ます。更に詳しくは **www.elnec.com**をご覧下さい。

BeeHive204 の構成

1. 48 ピン ZIF ソケット
2. LED ＊操作結果を表示
3. LED パワー/スリープ
4. YES! ボタン
*Automatic YES! は通常 Disable になっていますが、これを Enable に設定しますと Programmer - Automatic YES!を使用時にデバイスを交換した後このボタンで書き込み開始が出来る様になります。
5. ISP コネクター
6. 電源表示 LED
7. 電源コネクター

8. 電源スイッチ
9. GND コネクターと ESD リスト・ストラップ・コネクションのためのコネクター
10. 温度制御ファン
11.Type 2 USB コネクター PC⇔BeeHive204 通信用ケーブル

BeeHive204 プログラマーを PC に接続

USB ポートを使用

ソフトウエアが先にインストールされていますと USB ポートをハード的にスキャンします。

USB ケーブルと電源ケーブルの接続はどちらが先でもかまいません。

ノート: プログラマーの電気的プロテクション機能はショートや長時間の電源の問題やコントロール・プログラムの中断、又は、PC のスイッチオフに対してターゲット・デバイスとプログラマーを保護していますが、LED ランプがビジーなときは絶対に ZIF ソケットからデバイスを取らないで下さい。

BeeHive204 が使用出来る状態であることを確認してください。そして、PC の電源をオンにして、コントロール・プログラムを実行してください。プログラマーを機械的なプリンター切り替え器等を経由して接続しないで下さい。

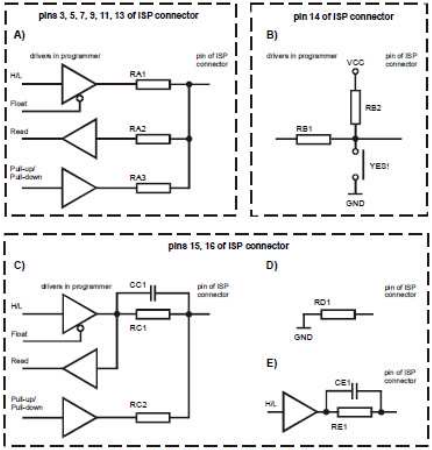
BeeHive204 によるイン-システム・シリアル・プログラミング

各デバイスを選択の上、Device Info[デバイス情報]でご確認下さい。デバイスのメーカーにより異なります。

プログラマーの ISP コネクターの正面

ISP コネクターは TE connetivity 社製 20pin コネクター 2-1634689-0 が使用されています。

ノート: *H/L/read ドライバー*



RA1 180R, RA2 1k3, RA3 22k,
RB1 10k, RB2 10k,
CC1 1n, RC1 1k3, RC2 22k,
RD1 22k, CE1 1n, RE1 1k3,

C) ISPプログラミングのために論理シグナルとして設定が必要な時はピン15と16を接続して下さい。
D) E) ピン15と16がLED OK と LED ERRORが状態として構成されている時
D) ISP動作の前のLED回路状態
E) ISP動作の後のLED回路状態

ノート: *LED OK又は、LED ERROR ON(輝いている)時, この状態は希望の*

ISPデバイスのHレベルによりロジカルH, Hのレベル1.8V – 5Vを表しています。
LED OK又は, LED ERROR OFF(輝いていない)時, この状態はロジカル, Lのレベルが0V – 4Vを表しています。この上記の値はプログラムされるチップとターゲット・システムを絶縁している抵抗の正確な値を示しています。
ISP コネクター・ピンのスペックはデバイスにより異なります。それはソフトウエア(Pg4uw)のメニュー**Device / Device Info (Ctrl+F1)**で見ることが出来ます。それぞれのデバイスの ISP プログラミングの方法が選択されいるか注意して下さい。選択されたデバイス名の後のサフィックにより表示されています。

ISP コネクターのピンのスペシフィケーションはプログラミング・デバイスに依存します。そして、**Device info window (Ctrl+F1)**に表示されます。関連するチップの ISP プログラミング方法が選択して下さい。プログラムされるチップ名の後の(ISP)サフィックスに示されております。これらのスペシフィケーションはデバイス・マニュファクチャラーのアプリケーション・ノートに対応しております。

ノート: *ISP ケーブル・コネクター上のピン番号 1 はトライアングル・スクラッチにより示されています。*
ISP ケーブル・コネクターは 20pin コネクターは Harting 09185207813 が使用されいます。
**ターゲット側は Harting 09 18 520 6324 をお薦めします。*

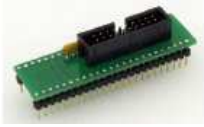
BeeHive204 ISP ケーブル

警告:

- *BeeHive204 をISP プログラマーとして使用する時は、ZIF ソケットにデバイスを装着しないで下さい。*
- *ZIF ソケットでデバイスをプログラミングする時は、ISP ケーブルを ISP コネクターに接続しないで下さい。*
- *付属の ISP ケーブルのみをご使用下さい。他の ISP ケーブルを使用されますと、プログラミングが上手くいかない可能性があります。*
- *BeeHive204 はプログラムされたデバイス(ISP コネクターのピン 1)とターゲット・システム(ISP コネクターのピン 5)を制限付きで供給することが出来ます。ターゲットから電源供給は出来ません。*
- *BeeHive204 はターゲット・デバイスにプログラミング電圧を印加し、そして、その値をチェックします。(ターゲット・システムはプログラミング電圧を修正することが出来ます。) もし、プログラミング電圧が期待したものと異なる場合は、ターゲット・デバイスは実行されません。*

プログラマーのセルフ・テスト
プログラマーが正常に動作していないと思える場合(但し、毎日使用しない場合でも最低3ヶ月毎)は,診断[Diagnostic]POD を 48 ピン ZIF ソケットに装着して BeeHive204 のセルフ・テストを行って下さい。ソフトウエアの Programmer/Selftest でセフルテストが行われます。

ISP コネクターのセルフ・テスト
ISP コネクターのための Diagnostic POD[#2]を ZIF ソケットに装着します。20 ピン・コネクターをプログラマーの ISP コネクターへケーブルで接続します。※20 ピンが内部的に正しく接続されていることを確認して下さい。(即ち、1-1, 2-2,…, 20-20)
ソフトウエアの Programmer/Selftest ISP connector でセフルテストが行われます。



カリブレーション・テスト・ポッド ※オプション
オプションですが、上記同様にソフトウエアの Programmer/Calibration test でカリブレーション・テストが行えます。

オプション: 注文番号 **70-0438**
48 Pins Calibration test POD, Type I

BeeHive204 スペシフィケーション

ベース・ユニット、DAC

- 4x ユニバーサル・プログラミング・モジュール(4x 48-pin DIL ソケット)
- USB 2.0 ハイ・スピード互換ポート 480 Mb/s 転送レート
- オン-ボード・インテリジェンス: パワフルなマイクロプロセッサと FPGA ベース・ステート・マシーン
- VCCP, VPP1と VPP2 のための 3 つの D/A コンバータで立ち上がりと下がり時間をコントロール
- VCCP 範囲 0..8V/1A
- VPP1, VPP2 範囲 0..26V/1A
- セルフ・テスト機能
- 電源供給入力とパラレル・ポート接続での静電気と ESD に対するプロテクション
- ESDリストストラップ接続のためのバナナ・ジャック
- グランドへ接続のためのバナナ・ジャック

ソケット, ピンドライバー

- 48 ピン DIL ZIF (Zero Insertion Force) ソケットが 48 ピンまでの 300/600 mil デバイスを受け付けます。
- ピンドライバー: 48 ユニバーサル
- VCCP / VPP1 / VPP2 は各ピンへ接続出来ます。
- 各ピンに対する完全グラウンド
- FPGA ベースの TTL ドライバーがすべてのピンドライバー・ピン上で H, L, CLK, プル-アップ, プル-ダウンを供給
- アナログ・ピンドライバー出力レベル選択可 : 1.8V～26V 迄
- 電流制限、過電流シャットダウン、電圧フェイル・シャットダウン
- ソケットの各ピン(IEC1000-4-2: 15kV 空中放電, 8kV 接触) 上での ESD プロテクション
- 各ピンの連続テストは毎プログラミング操作前にテストされます。

ISP コネクター

- 20-ピン・メス・タイプ *装着ミス・ロック
- 6 TTL ピンドライバーが H, L, CLK, プル-アップ, プル-ダウン, 1.8V～5V までのレベル H 選択可能な低電圧を含むすべてのデバイスをサポート
- 1x VCCP 電圧(範囲 2V..7V/100mA)
- ソース/シンク機能でのプログラム・チップ電圧(VCCP) と電圧感知
- 1x VPP 電圧(範囲 2V..25V/50mA)
- ターゲット・システム供給電圧(範囲 2V..6V/250mA)
- ISP コネクター(IEC1000-4-2: 15Kv 空中放電, 8kV 接触) の各ピン上での ESD プロテクション
- ISP デバイスのためのみ:2 出力信号が動作結果を表示 = LED OK と LED Error(アクティブ・レベル:最少 1.8V)
- 入力信号によるスイッチ YES!(アクティブ・レベル:最大 0.8V)

I.C. テスター

- TTL type: 54,74 S/LS/ALS/H/HC/HCT シリーズ
- CMOS タイプ: 4000, 4500 シリーズ

- static RAM: 6116.. 624000
- ユーザー定義パターン生成

ソフトウエア

- **アルゴリズム:** マニュファクチャラー承認、又は、認定されたアルゴリズムのみを使用。追加費用でカスタム・アルゴリズムも利用出来ます。
- **アルゴリズム・アップデート:** ソフトウエアのアップデートはレギュラーに行っております。約 4 週間に 1 度。**OnDeman[オンデマンド]バージョン**はユーザーの要求に応じるため、また、バグ・フィックスのためにほぼ毎日行っております。
- **メイン・フィーチャー:** 改訂履歴、セッション・ログ、オンライン・ヘルプ、デバイスとアルゴリズム情報

デバイス操作

- **標準:**
  - デバイス・タイプ、マニュファクチャラー、又は、パーツ名の一部を入力するだけで選択出来るインテリジェント・デバイス選択
  - EPROM/Flash EPROM の自動 ID ベースでの選択
  - ブランク・チェック, 読み込み[Read], ベリファイ
  - プログラム
  - イレース
  - コンフィギュレーションとセキュリティ・ビットのプログラム
  - 不正ビット・テスト
  - チェックサム
  - Jam 標準テストとプログラミング言語(STAPLE), JEDEC 標準"JESD-71 をインタープリット
  - SVF ファイルの圧縮バイナリー・バリエーション VME ファイルをインタープリット
  - SVF ファイル(シリアル・ベクトル・フォーマット)をインタープリット
  - Actel STAPL Player ファイルをインタープリット
- **セキュリティ**
  - インサーション・テスト
  - 接触チェック
  - ID バイト・チェック
- **スペシャル**
  - プロダクション・モード(デバイス装着後すぐに自動開始)
  - マルチ-プロジェクト・モード
  - 多くのシリアライゼーション・モード(インクリメンタル・モード、ファイルからのモード、カスタム・ジェネレータ・モード)
  - スタティスティクス[統計]
  - カウント-ダウン・モード

バッファ操作

- ビュー/編集, 検索/置き換え
- フィル/コピー, 移動, byte スワップ, word/dword スプリット
- チェックサム(バイト, ワード)
- 印刷

ファイル・ロード/セーブ

- 自動ファイル形式認識

サポート・ファイル形式(フォーマット)

- アンフォーマット(raw)バイナリー
- HEX: Intel, Intel EXT, Motorola S-record, MOS, Exormax, Tektronix, ASCII-SPCE-HEX, ASCII HEX
- Altera POF, JEDEC(ver. 3.0.A), 例えば、ABEL, CUPL, PALASM, TANGO PLD, OrCAD PLD, PLD Designer ISDATA 等々から
- JAM(JEDEC STAPL 形式), JBC(Jam STAPL バイト・コード)、STAPL(STAPL ファイル) JEDEC standard JESD-71
- VME(ispVME ファイル VME2.0/VME3.0)
- SVF(シリアル・ベクター・フォーマット revision E)
- STP(Actl STAPL file)

一般仕様

- 操作電圧 110-250V AC、90-264 VAC 最大, 47-63 HZ
- 消費電力 最大 60W(アクティブ)
- サイズ 361x234x56 [mm] (14.2x9.2x2.2 [インチ])
- 重さ 3.5kg(7.72lb) *アダプター類を除く
- 操作温度 5° ÷ 40°C
- 保管湿度 20%..80% 非結露

セットアップ

プログラマー・ソフトウエアのインストール

プログラマーのパッケージにはコントロール・プログラム、ユティリティーの 入っている CD が付いていますが、常に最新のバージョンをインストールして頂くためには下記のサイトより無償でダウンロードすることができます。この日本語マニュアルは日本語版ソフトウエアと同様にユーザーの便宜のために用意されています。疑わしい場合は英文を参照して確認して下さい。ソフトウエアは何時でも以下からダウンロードの上、インストールしプログラマーの機種を選択の上、デモ・モードでもデバイスの選択やその情報の取得等が可能です。

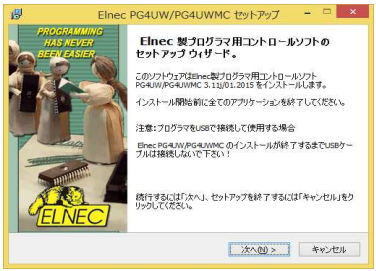
ソフトウエアの新しいバージョン

常にプログラマーの機能を最大にご利用頂くために最新のバージョンを
http://www.elnec.com/download/ から PG4UWarc-OnDemand.exe のソフトウエアをダウンロードされることをお薦めします。
PG4UWarc-OnDemand.exe は最新バージョンです。
*参考:PG4UWarc.exe(Regular Version) はレギュラー・バージョンと呼んでいますが、そのバージョンの最初のバージョンに過ぎません。

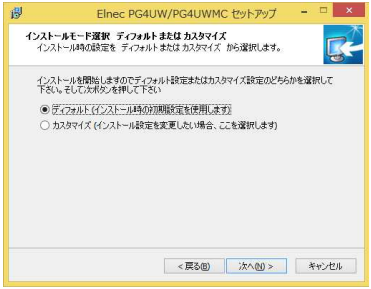
プログラマー・ソフトウエアのインストール

ダウンロード/保存した PG4UWarc-OnDemand.exe をダブルクリックします。

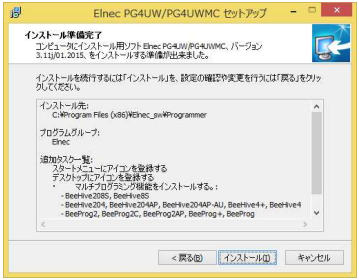
OK をクリック



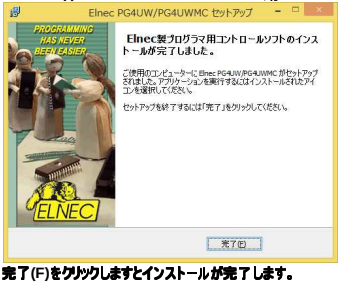
次へ(N)をクリック



次へ(N)をクリック



インストール(I)をクリックしますとインストールが自動的に行われます。



完了(F)をクリックしますとインストールが完了します。

ハードウエア(プログラマー))のセットアップ(インストール)

インストール中に望ましくない複雑さを避けるためにプログラマーを PC に接続する前にソフトウェア[PG4UW/MC]をインストールすることを推奨します。

警告 : プログラマの通信量が多いため各プログラマを USB 2.0 高速コントローラ (USB EHCI)に接続することを推奨します。新しい PC マザーボードの殆どには 2 つ以上の EHCI コントローラがチップセットに内蔵されています。
そうでない場合は、PCI(PCI-E) USB アドオンカード(ルネサス USB チップセットの使用を推奨)を使用できます。マザーボード・チップセットに内蔵されている EHCI を使用する場合は、マザーボードのマニュアル、又は、マザーボードメーカーの USB ポート・マッピングの技術サポートを参照して各プログラマを分離した EHCI に接続できるようにします。一般的にはプログラマを PC の USB ポート (USB HUB なし)に直接接続しマザーボードに搭載された直接(主に PC の背面にある)USB ポートに接続することをお勧めします。

プログラマが制御プログラムがインストールされる前に USB ポートに接続されると、Windows は新しいハードウェアを検出し、ドライバのインストール方法を選択するように自動的、又は、手動でユーザーに求めます。プログラマを正しく検出するには、制御プログラムのインストール CD をコンピュータの CD-ROM ドライブに挿入し、次の手順を実行する必要があります :

Step 1.
USB(LPT)ケーブルをプログラマの B タイプ USB(LPT)ポートに直接接続します。

Step 2.
USB(LPT)ケーブルを PC のタイプ A USB2.0(LPT)ポートに直接接続します (高速推奨)。

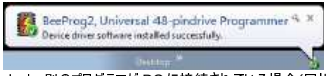
Step 3.
電源コードのコネクタをプログラマと電源プラグの適切なコネクタに接続します。

Step 4.
プログラマをオンにします。この時点で、全ての`動作結果`LED が連続的に点灯した後 LED が消灯します。

LPT 接続されたプログラマの場合は、すぐにプログラマと作業を開始することができます。
USB 接続されたプログラマの場合は、次の手順に進みます。

Windows 7/8/10
Step 5.
タスクバーの通知領域(主に右下)に通知バブルが表示されます :

プログラマのためのドライバが正常にインストールされた後に表示されます。



ノート: 別のプログラマが PC に接続されている場合(同じ USB ポートに接続されている可能性があります)、
“Installing device driver software”が再度起動します。
同じプログラマーが他の USB ポートに接続されている場合、追加のドライバーのインストールの必要はありません。

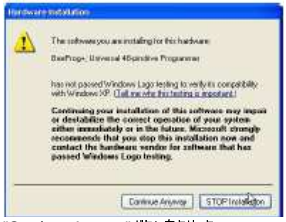
Windows XP, Service Pack 2と Windows Vista:
Step 5.
ウインドウズは“Found new hardware wizard”で開始されます。

"No, not this time"を選択、そして、"Next"ボタンをクリックして下さい。

全ての OS:

"Install the software automatically"を選択、そして、"Next"ボタンをクリック

Step 6.



"Continue Anyway"ボタンをクリック

Windows Vista:

"Install this driver software anyway"をクリック

Step 7.

セットアップを終了するために"Finish"ボタンをクリック

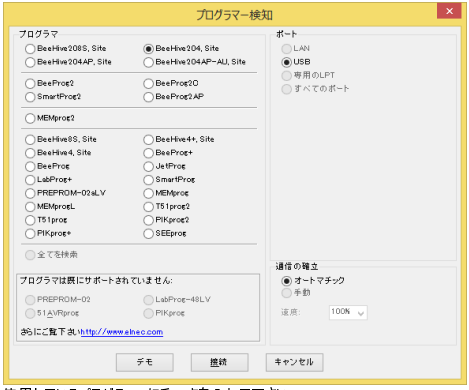
Step 8.
"Found new hardware wizard"が各プログラマーに対して 1 回起動されます (BeeHive204 の場合は 4 回)。
Step 5 でハードウェアの設定を続けます。

注意：PC 上の別の USB ポートをプログラマの次の接続に使用すると、"Found new hardware wizard"が再度起動し、新しい USB ドライバがインストールされます。

PG4UW ソフトウエア

PG4UW コントロール・プログラムの実行

PG4UW アイコンをダブル・クリックして下さい.



使用しているプログラマーにチェックを入れて下さい。
プログラマーを接続せずにアダプター等を知りたい場合等はデモをクリックして
デバイスを選択して検索して下さい。実際に使用する場合は接続をクリックして
下さい。

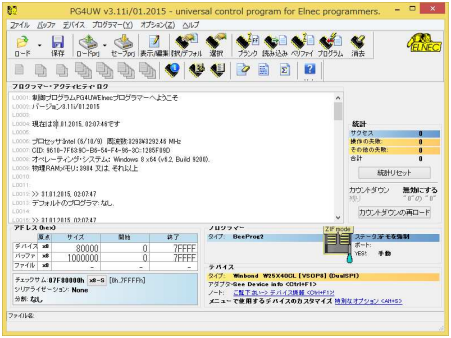
コントロール・プログラム(PG4UW)は自動的にすべてのポートをスキャンし、そして、
接続されているプログラマーをサーチします。

ノート: *PG4UW プログラムが開始されますと、標準のユーザー・メニューが表示されてユーザーからの指示を待ちます.*

もし、コントロール・プログラムがプログラマーと通信できないと、スクリーンにエラー・
コードと考えられる理由(プログラマーの接続が外れている、間違った接続をしてい
る、電源アダプターの不良、間違ったプリンター・ポート)を含むエラー・メッセージが現れます. 再点検の上、問題を取り除いて、そして、いずれかのキーを押してください。
もし、エラーがまだ有る場合は、プログラムはデモ・モードで再開されますので、プログラマーへのアクセスは出来ません. もし、エラーの原因が見つからないときはトラブ

ル・シューティングの指示に従ってください。コントロール・プログラムはデバイスのプログラムの前にプログラマーとの通信をチェックします。

ユーザー画面の説明



Toolbars[ツールバー]

メニューの下によく使用されるボタン・ショートカットを持ったツールバーがあります。ツールバーはメニューのオプション->表示で変更することが出来ます。

Log window[ログ・ウィンドウ]

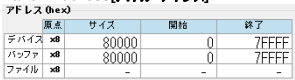
ログ・ウィンドウはPG4UWでの殆どの操作の流れについての情報を含みます。

操作:

- PG4UWの開始
- プログラマー・サーチ
- ファイル／プロジェクト・ロード/セーブ
- デバイスの選択
- デバイス操作(デバイス・リード, ブランク・チェック, プログラミング等)
- その他の各種情報

これらの情報は問題が有った時のためにカット&ペースでテキスト・ファイルにコピーすることも出来ますが、通常はヘルプ->プロブレム・レポート作成をクリックしますと"PG4UW_LOG_windows_content_xxxx.zip"としてデスクトップ上に作成しサポートのためにメールに添付して送ることが出来ます。

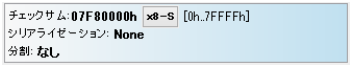
Panel Address[パネル・アドレス]



パネル・アドレスは現在選択されているデバイスの実際のアドレス範囲，ロードされたファイルとバッファの開始と終了アドレス設定についての情報を含んでいます。
ある種 のデバイスではメニューの**デバイス->デバイス・オプション->操作オプション**によってデフォルト・デバイスとバッファ・アドレス範囲を変更できます。
パネル・アドレスには Split、Serialization、及び、Buffer Checksum の現在の状態に関する高度な情報も含まれています。それぞれのオプションの詳細については、次をご覧ください：

- Split[スプリット] – メニュー デバイス/デバイス・オプション/操作オプション
- Serialization[シリアライゼーション] – メニュー デバイス/デバイス・オプション/シリアライゼーション
- Checksum[チェックサム] – メニュー バッファ/メインウィンドウに表示されるセクション・チェックサムのチェックサム

チェックサムには以下の様な機能も利用出来る様になっています。



例えば、x8-S をクリックしますと下 記のダイアログが表示され便利に使用することが出来ます。

Panel Programmer[パネル・プログラマ]

パネル・プログラマは現在選択されたプログラマについての情報を含みます。

- プログラマ・タイプ
- コンピューターに接続されているプログラマのポート
- 次の何れかのプログラマの状態
- Ready - プログラマーが接続され、正常に見つかり作業準備が出来ている

· Not found - プログラマーが見つかりません
· Demo - ユーザーがダイアログ Find programmer(プログラマの検索)でオプション(ボタン)でデモを選択したとき
· YES! mode - 一部のタイプのプログラマは次のいずれかの方法で次のデバイス動作を開始する特別なモードを使用することができます。
· 制御プログラムのダイアログ Repeat により手動
· ボタン START!により手動でプログラマに直接配置されます。
· 自動 - プログラマは自動的にデバイスの取り外しと新しいデバイスの挿入を検出します。
更に詳しくは Programmer / Automatic YES!をご覧ください。

Panel Device[パネル・デバイス]
現在選択されているデバイスに付いての情報を含みます。
· デバイス名(タイプ)とマニュファクチャラー
· 現在選択されているプログラマで使用する必要のあるデバイス・アダプタ(又は、モジュール)
· 詳細なデバイス情報ダイアログの参照。メニュー デバイス/デバイス情報でも使用できます。
· 高度なデバイス・オプションの参照 - これは一部のデバイスでのみ利用可能です。

Panel Statistics[パネル・スタティスティクス]
現在選択されているデバイスに付いての統計情報を含みます。
· 成功、失敗とデバイス操作合計の数
· カウント-ダウン・ステータスが残りのデバイスの表示
統計とカウント-ダウン・オプションはメニューコマンド Device / Device options / Statistics[デバイス/デバイス・オプション/統計]、又は、パネル Statistics[統計]をマウスの右ボタンでクリックし、ポップアップ・メニューから項目 Statistics[統計]を選択することで使用できます。

Panel File[パネル・ファイル]
パネルは PG4UW メイン・ウィンドウの下部に配置されます。パネルには現在ロードされているファイル、又は、プロジェクトの名前、サイズ、及び、日付が表示されます。

ホット・キーのリスト

<F1>	ヘルプ	ヘルプを呼ぶ
<F2>	セーブ	ファイルの保存
<F3>	ロード	ファイルをバッファにロード
<F4>	エディット	バッファのビュー/編集
<F5>	選択/デフォルト	最後に選ばれた 10 のデバイス・リストからターゲット・デバイスを選択
<Alt+F5>	選択/手動	デバイス/ベンダー名をタイプすることでターゲット・デバイスを選択
<F6>	ブランク	ブランク・チェック
<F7>	リード	デバイスの内容をバッファに読み込み

<F8> ベリファイ ターゲット・デバイスとバッファの内容を比較
<F9> プログラム ターゲット・デバイスをプログラム
<Alt+Q> 保存せずに終了 プログラムを終了
<Alt+X> 保存して終了 設定を保存してプログラムを終了
<Ctrl+F1> デバイス情報 現在のデバイスの追加情報を表示
<Ctrl+F2>イレース・バッファ 与えられた値でバッファをフィル
<Ctrl+Shift+F2> フィル・ランダム・データ ランダム値でバッファをフィル

File[ファイル]

このサブメニューはソース・ファイル(**binary**, **MOTOROLA**, **MOS Technology**, **Intel (extended) HEX**, **Tektronix, ASCII space**, **JEDEC** 及び **POF** のフォーマット)の各種操作として、設定とビュー・ディレクトリー、ドライブ変更、ファイルのロードとセーブの為のバッファ・メモリの開始アドレスと終了アドレスの変更に使用します。

File / Load[ファイル/ロード]

ファイル形式を解析した後、指定されたファイルからバッファにデータをロードします。ご使用に合った形式(**binary**, **MOTOROLA**, **MOS Technology**, **Tektronix**, **Intel (extended) HEX**, **ASCII space**, **JEDEC** 及び **POF**)を選んでください。コントロール・プログラムは、最後の有効なマスク情報をファイル・リストに順次記録して行きます. **options / Save options** コマンドで、コンフィグ・ファイルにマスク情報をセーブ出来ます.

<F3>によっていつでもどのメニューからでもこのメニューを呼び出す事が出来ます。

ファイルフォーマットの説明:

ASCII HEX フォーマット

各バイトデータは 2 つの 16 進数で表され、他の全てのデータ・バイトから空白スペースによって区切られています。
データ: バイトのためのアドレスは$Annnn, キャラクターのシーケンスを使ってセットされます。nnnn は 4 つの 16 進数アドレスです。後ろにカンマが必要です。
各データバイトがアドレスを持っているが明白でない場合、明示的なアドレスがデータストリームに含まれていない限りデータ・バイトは連続してアドレス指定されます。最初のデータバイトの前にアドレス部分が設定されていない場合は、ファイルは 0 から開始します。ファイル STX(Control-B) 文字 (0x02) で始まり ETX(Control-C)文字(0x03)で終わります。
ノート: チェックサム部分は$Sとカンマ文字間の 4 つの 16 進数文字列で構成されます。チェックサムはファイルの最後の部分になります。

ASCII HEX ファイルの例: データ"Hello, World"をアドレスの 0x1000 にロードしています:
^B $A1000,
48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 0A ^C
$S0452,

ASCII SPACE フォーマット
ASCII HEX とよく似た非常に単純な HEX フォーマットで開始(STX)と終了(ETX)文字列の無い HEX フォーマットです。各データバイトは 2 つの 16 進数文字として表現され、他のすべてのデータバイトの空白で区切られています。アドレス・フィールドはデータバイトから空白で区切られています。アドレスは 4-8 の 16 進数文字のシーケンスを使用して設定されます。

ASCII SPACE ファイル例：データ"Hello, World"をアドレスの 0x1000 にロードしています：
0001000 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 0A

Straight HEX フォーマット
ASCII HEXと同様の非常に単純な HEX ファイル・フォーマットでアドレスとチェックサムとスタート(STX)とエンド(ETX)を持たないフォーマットです。各バイトデータは 2 つの 16 進数で表され全ての他のデータ・バイトからはスペースによって区切られています。

Straight HEX ファイルの例：データ"Hello, World"を含む：
48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 0A

Samsung HEX フォーマット
Samsung HEX フォーマットは Intel HEX フォーマットを少し修正したもので、ファイルの形式、及び 構成等は Intel HEX フォーマットと略同等ですのでソフトウエアでは Intel HEX ファイル形式として認識され示されます。

特殊な x16 フォーマットのノート：
Intel HEXx16 は TMS320F デバイスのための 16bit データ・ワードを持った Intel Hex フォーマットです。
Motorola HEXx16 は TMS320F デバイスのための 16bit データ・ワードを持った Motorola ファイル・フォーマットです。

チェック・ボックスをチェックして下さい。**Automatic file format recognition** は自動的にプログラムがファイル形式を検出します。もしプログラムがサポートしている形式の中からファイル形式を検出出来ない場合は、バイナリー・フォーマットであることが考えられます。

Automatic file format recognition のチェックボックスにチェックが入っていない場合は、ユーザー側が **Selected file format** のパネル上から、手動によって利用出来るファイル形式のリストから適したフォーマットを選択します。バイナリーを選択した場合は、バッファの開始アドレスを指定することが出来ます。バッファの開始アドレスはファイルから読まれたデータがバッファ・メモリに書き込まれる時のバッファ・メモリの開始アドレスです。

注意：プログラムは ASCII Hex フォーマットを自動的に認識しません。バイナリーとして認識されますので、オプション automatic file format recognition を無効にして ASCII Hex フォーマットのダウンロードを行って下さい。

追加操作
チェック・ボックス **Erase buffer before loading** にチェックをいれますと入力されたイレース値を使って全てのバッファ・データをイレースします。
バッファ消去はファイルを読み込む前に直ちに実行されます。これはバイナルーと全ての HEX ファイル形式のための機能です。
このワンショット設定を使用は **Hex file options** のメニューの **Options/ General options** で **Erase buffer before loading option** を無効にします。

もし、チェック・ボックスに **Swap bytes** がチェックにされていますと、ユーザーはファイル読み込み中に 16 ビット・ワード(又は、2-バイト・ワード)内でスワッピング・バイト機能を有効にすることが出来ます。この機能はモトローラのバイト形式のファイル(ビッグ・エンディアン)でロードする時に便利です。標準のロード・ファイルではリトル・エンディアンのバイト形式を使用します。

ノート: *ビッグ-エンディアンとリトル-エンディアンとはコンピューターのメモリーに書き込まれるバイトの、シーケンス順を現す方法です。ビッグ-エンディアンはビッグ・エンド (Most significant 最上位の値) が最初にストアされます。(最下位の書き込みアドレス)。リトル-エンディアンはリトル・エンド(least significant 最下位の値) が最初にストアされます。例えば、ビッグ-エンディアンのコンピューターでは、ヘキサ・デシマス値 4F52 は2バイトが要求され、ワードの書き込みアドレス 1000H には 4F52H としてストアされます。4FH はバイト・アドレス 1000H に、そして、52H はバイト・アドレス 1001H となります。リトル-エンディアン・システムでは、それは 524FH (バイト・アドレス 1000H が 52H でバイト・アドレス 1001H が 4FH)としてストアされます。*
4F52H がメモリーにストアされる内容を次に示します。:

アドレス	*ビッグ・エンディアン システム*	*リトル・エンディアン システム*
1000H	*4FH*	*52H*
1001H	*52H*	*4FH*

Add blank spare area[ブランク・スペア領域の追加] - (for NAND Flash devices) チェックにしますとファイルのロード中にバッファ(選択したデバイスによる)内の関連する位置に空白のスペア領域データを追加します。

Buffer offset for loading[ローディングのためのバッファ・オフセット]
ローディングのためのバッファ・オフセットはファイルからバッファにデータをロードするためにワン-ショット・オフセット設定が出来ます。
標準設定はいつも"None"になります。ユーザーが使用する時にオフセット値を設定して下さい。設定はバッファリングするために保存するためにロードされたデータのオフセットをオプションで指定するために使用されます。
Load File ダイアログ・ウィンドウが開かれていますオフセットはデフォルトでは常に設定なしです。これはバッファに読み出しデータを格納するために使用されるオフセットはないことを意味します。

利用出来るオフセット・オプション:

None[なし] ファイルからバッファへのローディングにオフセットは適応されません。
Positive offset[ポジティブ・オフセット] オフセット値のポジティブ・オフセットは現在のアドレスにオフセットを加算してバッファにデータをストアします。このオフセットは全てのフォーマットで使 用でき、現在のバッファ構成が x8 の場合は x8 形式で、現在のバッファ構成が x16 の場合は x16 形式で使用されます。。
Negative offset[ネガティブ・オフセット] モードは 2 つのオプションを持っています:
Negative offset[ネガティブ・オフセット]と **Automatic negative offset**[自動ネガティブ・オフセット] -手動、又は、自動の 2 つの方法でセット:
手動セットはオプション Negative offset を使用し、希望するオフセット値とその編集ボックスに置きます。自動オフセット検出はオプションの Automatic negative offset を使用します。この値はデータをバッファに保存するため現在のアドレスから減算されます。

ネガティブ・オフセット値(手動定義、又は、自動検出)はデータをバッファに保存するため現在のアドレスから減算されます。ネガティブ・オフセットは x8 フォーマットを使った全ての HEX ファイルにのみ適応されます。ネガティブ・オフセット設定はバイナリー・ファイルと他の HEX で無いファイルでは無視されます。

ノート:
- 負のオフセットの値は実アドレスから減算されているので減算結果が負の数の場合は読み込めません。従って、正しい値を設定するように注意して下さい!
- 特殊な場合にのみネガティブ・オフセットの自動設定をお勧めします。
  このオプションはファイル内のある種の間違ったデータを扱うことが出来るヒューリスティック解析が含まれています。断片化されたアドレス範囲が含まれていたり、そして、選択されたデバイスのサイズを超えた様な問題のあるファイルに対して使用されます - ある種のブロックは無視することが出来ます。
- Automatic negative offset オプションは確実に指定されたブロックを持った HEX ファイルを必要とするような特殊なある種のデバイスでは利用出来ません。例として Microchip PICmicro デバイス。それらの特殊なデバイスは手動オフセット設定(None, Positive offset, Negative offset)のみを利用して下さい。

ネガティブ・オフセットの使用例:
Motorola S - フォーマットによるデータがふくまれるファイル
データ・ブロックがアドレス FFFF0H で開始するファイル
それは 3 バイトのアドレス・アレイの長さを持った S2 フォーマットです。
全てのデータの読み取りで Negative offset オプションを設定し、ネガティブ・オフセットの値を FFFF0H に設定することができます。
これは現在の実際のアドレスからオフセットが差し引かれデータがバッファ・アドレス 0 から書き込まれることを意味します。(即ち、ファイルの読込みを実行するとアドレス FFFF0H の前のデータは読み飛ばされ FFFF0H 以降のデータがバッファメモリの 0 番地から順にストアされます)

ファイル・フォーマットとエラー・コードのリスト
サポートされたフォーマットの幾つかでファイルのダウンロード中にエラーが起こり得ます。
エラーは LOG ウィンドウに次の様に書かれます "Warning: error #xxy in line rrr", xx はファイル・フォーマット・コード, y はエラー・コードと rrr は 10 進数での行番号。
File format codes:
#00y - binary
#10y - ASCII Space
#20y - Tektronix
#30y - Extended Tektronix
#40y - Motorola
#50y - MOS Technology
#60y - Intel HEX

Load file error codes:
#xx1 - bad first character - header
#xx2 - bad character in current line
#xx3 - bad CRC
#xx4 - bad read address
#xx5 - bad length of current line
#xx6 - too big negative offset
#xx7 - address is out of buffer range
#xx8 - bad type of selected file format
#xx9 - the file wasn't loaded all

File / Save[ファイル/保存]
作成や修正されたバッファ・メモリのデータや、デバイスから読み込まれたデータの保存です。希望するフォーマット(**binary**, **MOTOROLA**, **MOS Technology**, **Tektronix**, **Intel (extended) HEX**, **ASCII space**, **JEDEC** 及び **POF**)を選択することが出来ます.

もし、チェック・ボックスに **Swap bytes** がチェックにされていますと、ユーザーはファイル書き込み中に 16 ビット・ワード(又は、2-バイト・ワード)内でスワッピング・バイト機能を有効にすることが出 来ます。この機能はビッグ・エンディアンのファイルをモトローラのバイト形式 でロードする時に便 利です。標 準のセーブ・ファイル 操作ではリトル・エンディアンのバイト形式を使 用します。

<F2> のリザーブ・キーによって、このメニューをいつでも呼び出 すことが出 来ます。

File/Load project[ファイル/ロード・プロジェクト]
このオプションはデバイスの保存されたコンフィギュレーション・バッファ・データとユーザー・インターフェース・コンフィギュレーションを持ったプロジェクト・ファイルをローディングするために使 用されます。
標 準ダイアログ Load project は追加のウィンドウを含みます - Project description - ダイアログ説明] - ダイアログの下に置かれ ています。

標準ダイアログ Load project[ロード・プロジェクト] は追加のウィンドウを含みます - Project description[プロジェクト説明] - ダイアログの下に置かれています。このウィンドウはダイアログ Load project で現在選択されているプロジェクト・ファイルの情報を表示ためのものです。プロジェクト情報は下記の情報を含みます:
・プロジェクト内で最初に選択されたデバイスのメーカーと名前
・プロジェクト作成日時
・ユーザーが書いたプロジェクトの説明
ノート: シリアライゼーションを持ったプロジェクトはオン

シリアライゼーションは以下の手順でプロジェクト・ファイルから読み込まれます:
1. プロジェクトに記述したシリアライゼーション設定が受け付けられます。
2. 追加のシリアライゼーション・ファイルを検索が実行さます。ファイルが検出された場合、それが読み込まれ、そして、追加ファイルからのシリアライゼーション設定が受け付けられます。追加のシリアライゼーション・ファイルは常に特定のプロジェクト・ファイルに関連付けられます。追加のシリアライゼーション・ファイルの設定が受け入れられる時、プロジェクトのシリアル化の設定は無視されます。

追加のシリアライゼーション・ファイル名はファイルの名前を投影する拡張機能".sn"を追加することによって派生したプロジェクト・ファイル名になります。
追加のシリアライゼーション・ファイルは常に制御プログラムのディレクトリへのディレクトリ"serialization\"に置かれます。

例:
プロジェクト・ファイル名 :my_work.prj
コントロール・プログラムのディレクトリ: c:\Program Files\Programmer\

追加のシリアライゼーション・ファイルは:
c:\Program Files\Programmer\serialization\my_work.prj.sn
追加のシリアライゼーション・ファイルはデバイスのプログラム成功後に作成され更新されます。

シリアライズをオンにして追加のシリアライゼーション・ファイルを作成するための唯一の要件はシリアライゼーションをオンにしたロード・プロジェクトです。

ファイルが存在し、現在保存されたプロジェクトに関連する場合、コマンド File/Save project[ファイル/プロジェクトをセーブ]は追加のシリアライゼーション・ファイルを削除します。

Enter job identification dialog[入力ジョブ・アイデンティフィケーション・ダイアログ]
このダイアログはプロテクトされたプロジェクト・ファイルをローディングされて時に表示されます。

2 つの編集可能フィールドを持っています:
プロテクトされたプロジェクト・ファイルをリードした時、Job ID コードを入力するダイアログが表示されます。

・オペレーターID - このパラメータはプログラマーのオペレーターを認識するため使用されます。
オペレーターID は 3 文字以上でなければいけません。
プロテクトされたプロジェクトの Job Reportを作成する時にパラメータが必要ですので、ユーザーはオペレーターIDを入力しなければいけません。
・Job ID 入力 - 現在行っている作業の Job ID 認証を入力します。

ノート：ダイアログ Enter job identification はパスワードのダイアログではありません。オペレーター認証の値と Job ID は情報の目的に単に Job Report を含んでいます。プロテクトされた/又は、暗号化されたプロジェクト・パスワードとは関係ありません。

File/Save project[ファイル/プロジェクトをセーブ]
このオプションは保存されたデバイス構成の設定やバッファ・データを含むプロジェクト・ファイルを保存するために使用されます。プロジェクト・ファイルに保存されたデータはメニュー・コマンドの File/Load project でいつでも使用出来ます。

ファイル・リストから実際に選択されたプロジェクトの説明
ダイアログ Save project 内に現在選択されたプロジェクト・ファイルに付いての情報が表示されます。このボックスは情報のためですので変更は出来ません。

セーブされるプロジェクトの説明
上半分は現在選択されているデバイス, プログラム・モード, 日時等の実際のプログラム構成についての情報を表示します。内容等の変更は出来ません。
下半分はユーザー編集可能であり、通常はプロジェクト作成者やメモで構成されたプロジェクトの説明(任意のテキスト)が含まれます。

チェックボックス Encrypt project file (with password)は暗号アルゴリズムを使った特殊なフォーマットでプロジェクトをセーブするために使用されます。
これはパスワード無しでソフトウエアにプロジェクト・ファイルをロードするのを防ぎます。キーでボタンをクリックした後、password ダイアログが表示されますので、保存されるプロジェクトのための暗号化パスワードを指定するのに使用されます。

チェックボックス Set Protected mode of software after loading of this project file は Protect モードと呼ばれる特別なモードでプロジェクトを保存するために使用されます。キーでボタンをクリックした後、password ダイアログがが表示され、セーブされるプロジェクトの Protected mode password を指定、そして、オペレータのミスを防ぐための他のセキュリティ・オプション(他のプロジェクトのロード、デバイス操作の制限を無効にする)アクティブな Protected mode でセーブされたプロジェクトは Protected mode projectsと呼ばれる特殊なプロジェクトです。Protected mode プロジェクトに付いての更に詳しい情報は Options /Protected mode をご覧下さい。Protected mode がアクティブな時、ソフトウエアはプログラマー・アクティブ・ログの右上角の label Protected mode によってこれを表示します。

推奨：Encrypt project file (with password)と Set Protected mode of

software after loading of this project file のためのパスワードは同じものは使用しないで下さい。

チェックボックス Require project file checksum before first programming がアクティブな時, ロード・プロジェクトの後に最初のデバイス・プログラミングの開始前にソフトウエアはユーザーに正しいプロジェクト・ファイルのためのユニークな ID の入力を聞いてきます。この機能は正しいプロジェクト・ファイルが最新にロードされたかを追加チェックするために推奨されます。また、このチェックボックスはアクティブ Protected モードで使用することをお薦めします。プロジェクト・ファイルの固有の ID の要求がアクティブな場合、ソフトウエアは制御プログラムのメイン・ウィンドウの一番下のステータス行にプロジェクト・ファイル名の隣にラベル(ID)によって表示されます。

ノート:オプション最初のプログラミングの前の Require project file unique ID は以前の最初のプログラミングの前の Require project file checksum の替わりです。ジェネリック・チェックサムよりユニーク ID の利点は、固有の ID がメイン・デバイス・バッファのデータから計算されるだけでなくデバイスと使用可能なデバイス設定で使用されるセコンダリー・バッファ・データからも計算されることです。
プロジェクト・ファイルのチェックサムの要求がアクティブな場合、ソフトウェアが制御プログラムのメイン・ウィンドウの一番下のステータス行にプロジェクト・ファイル名の隣にこのラベル(CSum)を表示します。このオプションは Save project ダイアログボックスで使用できなくなりましたが、チェックサム要求が設定を持った古いプロジェクト・ファイルの読み込み後にアクティベートすることができます。

File/Reload file[ファイル/ファイルを再ロード]
最近使用したファイルを再ロードするためにこのオプションを選んで下さい。

ファイルを使用した時、Reload ファイル・リストに追加されます。ファイルは使用した時間の順番にリストされます。最後に使用したファイルは以前に使用したファイルの前にリストされます。

ファイルの再ロード:
1. ファイル・メニューから Reload ファイルを選択
2. 最後に使用したファイルのリストが表示されます。再ロードしたいファイルをクリック
ノート: ファイルを再ローディングする時、そのファイルは最後にロード/セーブされたファイルで使用されたファイル形式が使用されます。

File/Reload project[ファイル/プロジェクトの再ロード]
最近使用したプロジェクトを再ロードするためにこのオプションを選んで下さい。

プロジェクトを使用した時、Reload project リストに追加されます。プロジェクトは使用した時間の順番にリストされます。最後に使用したファイルは以前に使用したファイルの前にリストされます。

プロジェクトの再ロード:

1. ファイル・メニューから Reload プロジェクトを選択
2. 最後に使用したプロジェクトのリストが表示されます。再ロードしたいプロジェクトをクリック

File/Project options[ファイル/プロジェクト・オプション]
このオプションは実際にロードされたプロジェクトの表示/編集プロジェクト・オプションのために使用されます。プロジェクト・オpションは次のプロジェクト・データに含まれているプロジェクトのベーシックな説明を意味します:

・ デバイス名とマニュファクチャラー
・ プロジェクト作成日付
・ ユーザー定義プロジェクト説明(任意テキスト), 例えば、更に詳細なプロジェクト説明のための作者と他のテキスト・データ

ユーザーはユーザー定義プロジェクト説明のみ直接編集することが出来ます。デバイス名, マニュファクチャラー,プロジェクト・デートとプログラム・バージョンはプログラムにより自動的に生成されます。

File / Load encryption table[ファイル/暗号テーブルのロード]
このコマンドはディスクからバイナリー・ファイルでのデータをロードします。そして、それらをメモリーの一部にセーブ、暗号(セキュリティー)テーブルのためにリザーブされます。

File / Save encryption table[ファイル/暗号テーブルの保存]
このコマンドは暗号テーブルが含まれたメモリーの部分の内容を、バイナリー・データとしてディスクのファイルに書込みます。

File / Exit without save[ファイル/保存せずに終了]
設定を保存せずにプログラムを終了

File / Exit and save[ファイル/終了と保存]
INI ファイルに設定を保存してプログラムを終了

Buffer[バッファ]

このメニューはバッファ操作、ブロック操作、ストリングでのバッファの部分のフィル、イレース、チェックサムと編集とその他(検索とストリングス再配置、印刷...)の項目でもビューに使用します。

Buffer / View/Edit[バッファ/ビュー/エディット]

このコマンドはバッファのデータを見たり(ビュー・モード) 又は、編集(エディット・モード) するのに使用します(ビューは DUMP モードのみ) 。オブジェクトの編集をするための選択は矢印キーを使用してください。編集したデータはカラーで表示されます。
<F4> ホット・キーでも使用出来ます。

View/Edit Buffer [ビュー／バッファの編集]
このダイアログはバッファのデータを View(ビュー・モード) 又は、edit(編集モード)に使用されます。選択されたチップのために配置されたデータの領域外のバッファのデータはグレイ・バックグラウンドで示されます。

次のコマンドがバッファ・データの編集のために利用できます。全てのコマンドが全てのシチュエーションで利用できるわけではありません。選択されたデバイスとデバイスのために使用されるバッファに依存します。

F1	ヘルプの表示
F2	フィルのための開始と終了ブロックと要求されるヘキサ又は、ASCII ストリングをセットして下さい。
Ctrl+F2	指定したブランク値でバッファを消去
Ctrl+Shift+F2	ランダム・データでバッファをフィル
F3	ブロックのコピーは新しいアドレス上の現在のバッファのデータの指定されたブロックをコピーするのに使用されます。ターゲット・アドレスはソース・ブロック・アドレスの外側である必要はありません。
F4	ブロックの移動は新しいアドレス上の現在のバッファのデータのブロックを移動するのに使用します。ターゲット・アドレスはソース・ブロック・アドレスの外側である必要はありません。ソース・アドレス・ブロック(又は、一部分)はブランク・キャラクターによりフィルされます。
F5	スワップ・バイト・コマンドは現在のバッファ・ブロックのバイト・ペアのハイとローの順番をスワップします。 このブロックは偶数アドレスで開始しなければいけません。そして、バイトの偶数でなければいけません。 もし、この条件が満たされないと、プログラムのアドレス自身を修正します。(開始アドレスは低い偶数アドレスに移動されるか、又は、終了アドレスが高い奇数アドレスに移動されます。)
F6	プリント・バッファ
F7	ストリング検索(最大長 16 ASCII キャラクター)

F8 ストリングを検索し置き換え (最大 16 ASCII キャラクター)
F9 現在のアドレスを変更
F10 ビュー/編集モードを変更
F11 バッファ・データ・ビューのモードを 8 ビットと 16 ビットの間で切り替えます。
View/Edit mode buffer indicator [ビュー/編集モード・バッファ・イン ディケータ] の右のボタンをマウスでクリックすることでも行えます。このボタンは実際のデータ・ビュー・モード(8 ビット又は、16 ビット)も表示します。
F12 チェックサム・ダイアログはバッファの選択されたブロックのチェックサムをカウントします。
Arrow keys カーソル移動
Home/End 開始/終了へジャンプ
PgUp/PgDn 前/次ページへジャンプ
Ctrl+PgUp/PgDn 現在のページの開始/終了へジャンプ
Ctrl+Home/End 現在のデバイスの開始/終了へジャンプ
Shift+Home/End 現在のバッファの開始/終了へジャンプ
Backspace カーソルを1つ左へバック

ノート: *キャラクター 20H - FFH (ASCII モード)と番号 0..9, A..F (HEX モード) は即座に編集エリアの内容を変更します。*
警告: ワード・デバイスへの ASCII キャラクターの編集は出来ません。

Print buffer[プリンター・バッファ]
このコマンドはバッファの選択された部分をプリンター又は、ファイルに書きます。プログラムは外部テキスト・エディターを使用します。デフォルトでは Notepad.exe に設定されています。

Print buffer[プリント・バッファ]
このコマンドは選択されたバッファの部分をプリンター、又は、ファイルにを書くことが出来ます。プログラムは表示されている選択されたバッファのブロックを外部テキスト・エディターでプリント、又は、ファイルにセーブすることが出来ます。デフォルトでシンプルなテキスト・エディターnotepad.exe が設定されています。

ダイアログに次のオプションがあります。:
Block start[ブロック開始]
バッファ内の選択されたブロックの開始アドレスを定義
Block end[ブロック終了]
バッファ内の選択されたブロックの終了アドレスを定義

外部エディター
バッファの選択されたブロックのためのテキスト・ビューワーとして使用される外部プログラムのパスと名前を定義します。デフォルトでは **wordpad.exe** に設定されています。ユーザーはどのテキスト・エディターも定義することが出来ます。ユーザー定義テキスト・エディターでユーザーは印刷又は、バッファの選択されたブロックをフ

ァイルに保存することが出来ます。外部エディターのパスと名前は自動的にディスクにセーブされます。

Find [テキスト検索]ダイアログ・ボックス
テキスト入力ボックスに検索のためのストリングスを入力し検索します。検索を始めるために**<Find>** を選ぶか、又は、中止する場合は **<Cancel>** を選んで下さい。

Direction[検索方向] ボックスは検索する方向を指定します。 現在のカーソル位置から開始(エディット・モード).
Forward (現在の位置、又は、バッファの最初からバッファの最後へ) がデフォルトです。
Backward は始めに向かって検索します。ビュー・モードでは全バッファを検索します。

Origin[オリジン]は検索を開始する場所を指定します。

Find & Replace text[テキストの置換]ダイアログ・ボックス
Text to find[検索のためのテキスト**]**ストリング・入力ボックスに検索のためのストリングスを入力し、そして、**Replace with[置換]** 入力ボックスに置換のためのストリングスを入力します。

Options[オプション] ボックスで置換のプロンプトを選択することができます。

Origin[オリジン] は検索をどこから開始するか指定します。

Direction[検索方向] ボックスは検索する方向を指定します。 現在のカーソル位置から開始(エディット・モード).
Forward (現在の位置、又は、バッファの最初からバッファの最後へ) がデフォルトです。
Backward は始めに向かって検索します。ビュー・モードでは全バッファを検索します。

ダイアログ・ウィンドウを閉じるには **<Esc>** を押すか、又は、**Cancel[キャンセル]** ボタンをクリックします。

Replace[置換] ボタンを押しますと、ダイアログ・ボックスが閉じられ、そして、クエスチョン・ウィンドウが表示されます。このウィンドウは下記の選択を含んでいます。:
Yes 置換と次を検索
No 置換せずに次を検索
Replace All すべてを置換
Abort search このコマンドについて

View/Edit buffer for PLD[ビュー/PLD のためのバッファ編集]
Ctrl+F2 指定したブランク値でバッファを消去
Ctrl+Shift+F2 ランダム・データでバッファをフィル

F9	アドレスへ...
F10	ビュー/編集のモードを変更
F11	バッファのデータ・ビューのモードで 1 ビットと 8 ビット・ビューの間を切替えます。View/Edit mode buffer indicator [ビュー/編集モード・バッファ・インディケータ] の右のボタンをマウスでクリックすることでも行えます。このボタンは実際のデータ・ビュー・モード(1 ビット又は、8 ビット)も表示します。
Arrow keys	カーソル移動
Home/End	現在行の開始/終了へジャンプ
PgUp/PgDn	前/次ページへジャンプ
Ctrl+PgUp/PgDn	現在のページの開始/終了へジャンプ
Ctrl+Home/End	エディット・エリアの開始/終了へジャンプ
Backspace	カーソルを1つ左へバック

ノート: *0 と 1 のキャラクターがエディット・エリアの内容を即座に変更*

Buffer/Fill block[バッファ / ブロックのフィル]
このコマンドを選択することで、要求したヘキサ(又は、ASCII)ストリングによりバッファの選択されたブロックをフィルします。フィルのためのブロックの開始と終了とヘキサ(又は、ASCII)ストリングを設定して下さい。

選択オプション “Allow address history logging”は最近確認された値のセーブをアクティベートします。これらは各デバイスは別個にセーブされます : カウントは最後の 15 個に制限されています。
ノート : アドレス履歴値は全てのバッファ・データ操作ダイアログで共通です。

デフォルト・アドレス範囲は選択されたデバイスのバッファ範囲に従ってセットされます。選択オプション “Maintain last inserted values”は次回このダイアログを開いた時に前回確認された値がデフォルトとして再ロードされます。

Buffer/Copy block[バッファ /ブロックのコピー]
このコマンドは新しいアドレス上の現在のバッファのデータの指定されたブロックをコピーするのに使用されます。ターゲット・アドレスはソース・ブロック・アドレスの外側である必要はありません。

Buffer/Move block[バッファ /ブロックの移動]
このコマンドは新しいアドレス上の現在のバッファのデータのブロックを移動するのに使用します。ターゲット・アドレスはソース・ブロック・アドレスの外側である必要はありません。ソース・アドレスのブロック(又は、一部分)は一般的にブランク・キャラクターによりフィルされます。

選択オプション “Allow address history logging”は最近確認された値のセーブをアクティベートします。これらは各デバイスで別個にセーブされ カウントは最後の 15 個に制限されています。
ノート : アドレス履歴値は全てのバッファ・データ操作ダイアログで共通です。

デフォルト・アドレス範囲は選択されたデバイスのバッファ範囲に従ってセットされます。選択オプション “Maintain last inserted values”は次回このダイアログを開いた時に前回確認された値がデフォルトとして再ロードされます。

Buffer/ Swap block[バッファ/スワップ・ブロック]
このコマンドは現在のバッファ・ブロックのバイト・ペアのハイとローの順番をスワップします。このブロックは偶数アドレスで開始しなければいけません。そして、バイトの偶数を持たなければいけません。もし、この条件が満たされないと、プログラムをアドレス自身を修正します。(開始アドレスは低い偶数アドレスに移動されるか、又は、終了アドレスが高い奇数アドレスに移動されます。)

次のスワップ・モードが利用出来、ユーザーは選択することが出来ます:
1.Swap 2-bytes inside 16-bit words 16-bit ワード内をバイト・ペアでスワップ
2.Swap 4-bytes inside 32-bit words 32-bit ワード内をバイト 4 でスワップ
3.Swap nibbles inside bytes バイト内をニブルでスワップ
4.Mirror bits inside bytes バイト内のビットを逆にします。

バッファ内のスワップ操作の例:
開始アドレス 0 から終了アドレス N までのスワップ・バイト操作は次のテーブルによりバッファ内のデータを修正

Address	Original Data	Swap 2-bytes inside 16-bit words	Swap 4-bytes inside 32-bit words	Swap nibbles Insides Bytes	Mirror bit inside bytes
0000h	b0	b1	b3	b0n	b0m
0001h	b1	b0	b2	b1n	b1m
0002h	b2	b3	b1	b2n	b2m
0003h	b3	b2	b0	b3n	b3m
0004h	b4	b5	b7	b4n	b4m
0005h	b5	b4	b6	b5n	b5m
0006h	b6	b7	b5	b6n	b6m
0007h	b7	b6	b4	b7n	b7m

b0, b1, b2..はアドレス 0, 1, 2..からのオリジナル・バッファ・バイト値
b0n, b1n, b2n..は次のルールによるニブル・スワップ・オリジナル・バイト b0, b1, b2

Original Byte bits bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0
Nibbled-swapped Byte Bits bit3 bit2 bit1 bit0 bit7 bit6 bit5 bit4

Original Byte bits bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0
Mirrored Byte Bits bit0 bit1 bit2 bit3 bit4 bit5 bit6 bit7

オプション"Allow address history logging" を選択しますと最近確認された値のセーブをアクティベートします。これらは個々のデバイスのために別々にセーブされ:カウントは最後の 15 項目に制限されています。

ノート : アドレス履歴値は全てのバッファ・データ操作ダイアログで共通です。
デフォルトのアドレス範囲は選択されたデバイスのバッファ範囲に従ってセットされます。オプション"Maintain last inserted values"を選択しますと次回にこのダイアログを開いた時に以前に確認された値がデフォルトとして再ロードされます。

Buffe/Erase[バッファ/イレース]
このコマンドを選択しますと、バッファの内容をブランクでフィルします。

オプション"Allow address history logging" を選択しますと最近確認された値のセーブをアクティベートします。これらは個々のデバイスのために別々にセーブされ : カウントは最後の 15 項目に制限されています。
ノート : アドレス履歴値は全てのバッファ・データ操作ダイアログで共通です。

デフォルトのアドレス範囲は選択されたデバイスのバッファ範囲に従ってセットされます。オプション"Maintain last inserted values"を選択しますと次回にこのダイアログを開いた時に以前に確認された値がデフォルトとして再ロードされます。

<Ctrl+F2>はどのメニューにいても、このメニューを呼び出すことが出来ます。

Buffer/ Fill random data[バッファ]
もし、このコマンドが選択されますと、バッファの内容がランダム・データでフィルされます。

オプション**"Allow address history logging"** を選択しますと最近確認された値のセーブをアクティベートします。これらは個々のデバイスのために別々にセーブされ : カウントは最後の 15 項目に制限されています。
ノート : アドレス履歴値は全てのバッファ・データ操作ダイアログで共通です。

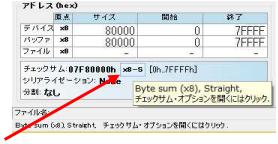
デフォルトのアドレス範囲は選択されたデバイスのバッファ範囲に従ってセットされます。オプション**"Maintain last inserted values"**を選択しますと次回にこのダイアログを開いた時に以前に確認された値がデフォルトとして再ロードされます。

<Shift+Ctrl+F2> キーでいつでも、どのメニューにいても、このメニューを呼び出すことが出来ます。

Buffer /Duplicate buffer contents[バッファ/バッファ内容のコピー]
このコマンドはコピー先の EPROM の範囲にソースの EPROM の範囲のバッファの内容をコピーします。これは、例えば、27C512 EPROM 位置へ 27C256 データを使用するのに適切です。
ノート : 手順は常にバッファ開始アドレス 00000h を使用

Buffer / Checksum[バッファ/チェックサム]

PG4UWのバッファにストアされたデータのチェックサムはバッファのデータが正しいかを照合するのに便利です。PG4UW はチェックサムに関する次の機能を含んでいます:



ここをクリックしますと下のダイアログが現れます。

“チェックサム・オプション(Z)をクリックしますと”チェックサム・ダイログが次ページの様に現れます。

- タブ **Checksum calculator[チェックサム・カイキュレータ]**, これはバッファ内の各種のデータ・ブロックの各種チェックサムを計算し、そして、表示出来るオン-デマンドの **checksum calculator[チェックサム・カイキュレータ]** です。(*1)

- タブ**Main checksum options[メイン・チェックサム・オプション]** はテーブル **Address**とPG4UWの**Log windows**でPG4UWのメイン・ウィンドウに表示されるメイン・チェックサム値を持った**Automatic checksum calculator**［自動チェックサムの計算]のためのオプションが含まれています。(*2)

Checksum calculator[チェックサム・カリキュレータ] はオン-デマンド・チェックサム・カリキュレータを含んでいます。(*1)

- フィールド **From address[アドレスから]**と**To address[アドレス]**がメイン・チェックサム計算のためのアドレス範囲の入力に使用されます。アドレスはチェック・ボックス**Enabled[有効]** にチェックが入っている時のみ使用することが出来ます。アドレスは常にバイト・アドレスで定義されます。
- グループ **Exclude buffer block(s) from checksum calculation[チェックサム計算からバッファ・ブロックを除外する]** は、例えば、**serialization[シリアライゼーション]**には便利です。シリアライゼーションは通常はバッファの指定されたアドレスのデータを修正します。従って、シリアライゼーション・エンジンによってデバイスのプログラミングの前にデータの或るアドレスが変更された時はバッファのチェックサムのチェックに問題があります。シリアライゼーションのために使用するバッファ(データ・ブロック)の部分をチェックサム計算が除外した場合はバッファ・データのチェックサムはシリアライゼーションによって変更されません。1つ以上の除外されるブロックを指定することが出来ます。
  - 計算された**チェックサム・タイプ**の値を表示するフィールド: 後述のタイプの説明をご覧下さい。

- **STRAIGHT**は 追加の調整無しのチェックサム計算
- **NEGATED**はチェックサムの反転 SUM + NEG. = FFFFH.
- **SUPPLEMENT**はチェックサムの補数 SUM + SUPPL.=0 (+ carry).
- **Insert checksum options[チェックサムの挿入オプション]** ボックス – このボックスは**Calculate & insert[計算と挿入]** 操作のための次のオプションを含みます:
- **Insert checksum[チェックサムの挿入]** Calculate & insert[計算と挿入]が実行されたときバッファに書込まれるチェックサムの種類
- **Insert at address[アドレスの挿入]** Calculate & insert[計算と挿入]が実行された時きに選択されたチェックサムの結果を書込むバッファのアドレス。アドレスは <From address> から <To address>の範囲内で指定することが出来ません。アドレスはバイト・アドレスとして定義されます。
- **Size[サイズ]** 選択されたチェックサム結果が書込まれるバッファのサイズ。チェックサムのサイズはByte (8-bit)又は、Word(16-bit)、又は、DWORD(32-bit)です。
  もし、選択されたチェックサム・サイズより小さい場合は、チェックサム値の口ー・バイトのみがバッファに書込まれます。
  ノート: *もし、ワード・サイズが選択されますと、チェックサム値の口ー・バイトがInsert address[アドレス挿入]ボックスで指定されたアドレスが書込まれ、そして、ハイ・バイトが1つずつインクレメントされたアドレスに書込まれます。DWORDに対しても同様です。*
- **Calculate button[計算ボタン]** - **Calculate** ボタンをクリックしますと、バッファ内の選択されたブロックのチェックサムを計算します。バッファへの書込みは行われません。
- **Calculate & insert button[計算と挿入]** - Calculate & insert ボタンをクリックしますと、バッファ内の選択されたブロックのチェックサムを計算され、そして、選択されたチェックサムがInsert address[アドレス挿入]で指定されたアドレスでバッファに書込まれます。この機能はByte, Word, CRC-CCITTとCRC-XMODEMチェックサムで利用出来ます。
- **Close button[クローズ・ボタン]** – ダイアログChecksumを閉じます。

(*1) これらの値はプロジェクトに保存されません。それぞれの新しいデバイス選択でデフォルトに初期化されます。

タブ Main checksum options[メイン・チェックサム・オプション]は自動チェックサム計算のモードをセットすることが出来ます。(*2)

• Custom address range for main checksum[メイン・チェックサムのためのメイン・チェクサム]

• Enabled[有効にする] – ユーザー定義アドレスがバッファのデータのチェックサムの計算に使用されます、他方、もし、**Disabled[無効]**の場合はバッファのデータのチェックサムの計算にグローバル・バッファ開始とバッファ終了アドレスが使用されます。

• フィールド**From address[アドレスから]** と**To address[アドレスまで]** はメイン・チェックサム計算のアドレス範囲の入力に使用されます。アドレスはチェックボックス**Enabled[有効にする]**にチェックが入っている時のみ使用されます。

• 選択グループ **Checksum type[チェックサム・タイプ]**は メイン・チェックサムに使用する希望するタイプの選択使用します。詳しは下記のチェックサム・タイプをご覧下さい。

• フィールド **Checksum[チェックサム]** は最後に計算されたチェックサムの実際の値を含みます。
• グループ**buffer block(s) exclude from checksum calculation** – はチェックサム計算のタブと同じです。
• ボタン**Apply[適用]**は**Main checksum options[メイン・チェックサム・オプション]**からのチェックサム設定を確認するために使用します。ノート：一度ボタンが押されますと、前回のチェックサム設定は失われます。
• ボタン**Close[閉じる]** はチェックサム・ダイアログを閉じるために使用されます。もし、設定で変更を加えた場合、**Apply[適用]** を押すまで変更は反映されません。
(*2) それらの値はコフィギュレーション・ファイルとプロジェクト・ファイルにストアされます。プロジェクト・ファイルからの設定が優先されます。

チェックサム・タイプ

Byte sum (x8)
バッファのデータは現在のバッファのビュー・モード(x8/x16/x1)構成に関係なくバイトごとに加算されます。32ビットを超えるキャリービットは無視されます。このチェックサム・モードでは文字列(x8)をメイン・プログラム・ウィンドウのチェックサム値の後に表示されます。

Word sum Little Endian (x16)
バッファのデータは現在のバッファのビューモードの構成に関係なくワード単位で加算されます。32ビットを超える任意のキャリー・ビットは無視されます。このチェックサム・モードはメイン・プログラム・ウィンドウのチェックサム値の後に表示され文字列(x16LE)よって示されます。リトル・エンディアンはバッファのチェックサムがリトル・エンディアン・モードでバッファから読み出されワードから計算されます。

Word sum Big Endian (x16)
バッファのデータは現在のバッファのビューモードの構成に関係なくワード単位で加算されます。32ビットを超える任意のキャリービットは無視されます。このチェックサム・モードはメイン・プログラム・ウィンドウのチェックサム値の後に表示され文字列(x16BE)よって示されます。ビッグエンディアンはバッファのチェックサムがビッグ・エンディアンモードでバッファから読み出されワードから計算されます。

CRC-CCITT
多項式 x^16+x^12+x^5+1 (0x1021)を使ってバッファ・データbyteをWordで計算，初期値 0, XOR out 0, reflexions in/out はoff

CRC-XMODEM
多項式 x^16 + x^15 + x^2 +1 (0x8005)を使ってバッファ・データbyteをWordで計算，初期値 0

CRC-16
多項式 x^16+x^15+x^2+1 (0x8005)を持った標準CRC-16 アルゴリズムを使ってバッファ・データbyteをWordで計算，初期値 0, そして、XOR out 0

CRC-32
多項式0x04C11DB7を持った標準CRC-32 アルゴリズムを使ってバッファ・データbyteをWordで計算，初期値 0xFFFFFFFF, そして、XOR out 0xFFFFFFFF

MD5
MD5 hash は32桁の16進数のシーケンスで表示されます。(128 bits)

SHA-1
"Secure Hash Standard" は40桁の16進数のシーケンスで表示されます。(160 bits)

Checksum forms

Straight – 追加の調整無しのチェックサム
Negated – チェックサムを反転 SUM + NEG. = FFFFH.
Supplement はチェックサムの補数 SUM + SUPPL. = 0 (+ carry).
デバイス依存チェックサム – いくつかのデバイスが適応されます。
例えば、STMicroelectronics's STM8ファミリー。**メイン・チェックサム**のためのチェックサム・モードはメイン・プログラムのラベル・チェックサム上でクリックすることでポップ-アップ・メニュー(又は、メニュー・ショートカット)でセットすることが出来ます。

Shift+Ctrl+1 - Byte sum (x8),
Shift+Ctrl+2 - Word sum Little Endian (x16) **Shift+Ctrl+3** - Word sum Big Endian (x16) etc...

Wordは16-bit word. **DWORD**は32-bit word.

Device[デバイス]

この機能は選択されたプログラマブル・デバイスの操作に使用します。- デバイス選択，デバイスからのデータの読み出し，デバイスのブランク・チェック，プログラム，ベリファイとイレース

Device / Select from default devices[デバイス/デフォルト・デバイスから選択]
このウィンドウはデフォルト・デバイスのリストからデバイスのタイプを選択することが出来ます。これはデバイス・オプションで最後に選択されたデバイスにストアされる周期バッファです。このリストは **File / Exit and save[ファイル/終了とセーブ]**コマンドによりディスクに保存されます。.

現在のデバイスの追加情報を表示したい場合は**<Ctrl+F1>** キーを使います。このコマンドはデバイスのサイズ、構成、プログラミング・アルゴリズムとこのデバイスがサポートされてるプログラマー(追加モジュールを含む)のリストを提供します。パッケージ情報と他の一般的な情報もご覧頂けます。

デフォルト・デバイスのリストから現在のデバイスを削除するためには **<Del>** キーを使用します。このリストを空白にすることはできません。最後のデバイスはバッファに残っておりますので、**<Del>**キーは受け付けられません。

Device / Select device ...[デバイス/選択されたデバイス….]
このウィンドウは現在のプログラマーによりサポートされている全てのデバイスのタイプを選択することが出来ます。デバイスを**名前**、**タイプ**又は、**マニファクチャラー**により選択することが可能です。

ノート 1: ソフトウェアでプログラマブル・デバイスの名前はチップの上部に表示されていたり、データシート区間番号に記載されている全てのキャラクターが含まれているわけではありません。名前はデバイスの識別に必要な全てキャラクターが含まれていますが、プログラミングに影響がない例えば、温度コード、スピードコード、梱包タイプコードは含まれていません。そのようなコード文字が名前の最後にある場合は省略され、途中にある場合は'x'に置き換えられています。

例えば
- *デバイス Am27C512-150, Am27C512-200と27C512-250はソフトウエアではAm27C512と表示されます。*
- *S29GL064N11TF1010 デバイスはソフトウエアでは S29GL064NxxTxx01 と表示されます。*

ノート 2: もし、あるデバイスで 2 つ表示されていて、2 番目にサフィックス x16 とある場合、それはプログラミング・アルゴリズムがより早いワード・モードを提供していることを意味します。

Selected device[選択されたデバイス] は自動的にデフォルト・デバイスのバッファにセーブされます。このバッファは **Device / Select from default devices[デバイス/デフォルト・デバイスからの選択]** コマンドからアクセスすることが

出来ます。検索 マスク・フィールドではデバイス名、マニュファクチャラー及び/又は、プログラミング・アダプタ名で全体のデバイス・リストのフィルターリングのためにマスクを入力することができます。スペースはフィルター項目の区切り文字として "OR" 機能を持っています。スペースを含め正確なフィルタ文字列を入力したい場合はクオテーション・マーク " を使用します。

もし、現在のデバイスについての追加情報を表示したい場合は、<Ctrl+F1> キーを使ってください。このコマンドはデバイスのサイズ、構成、プログラミング・アルゴリズムとこのデバイスがサポートされてるプログラマー(追加モジュールを含む)のリストを提供します。パッケージ情報と他の一般的な情報もご覧頂けます。

Save currently displayed list to file ボタンを押すことで現在表示されているデバイス・リストをテキスト・ファイルに保存することができます。

Select device ... / All[デバイス選択…/全部]
このウィンドウは現在のプログラマーでサポートされている全てのデバイスからターゲット・デバイスのタイプを選択することが出来ます。サポートされているデバイスはリスト・ボックスに表示されます。

デバイスはマニファクチャラー名とデバイス番号をリストの行でダブル・クリックするか、又は、サーチ・ボックス(セパレート・キャラクターとして**<Space>** を使って、そして、**<Enter>** を押すか、又は、**OK** ボタンをクリックして下さい。)で入力することで選択することが出来ます。

現在選択されているデバイスを反映せずにデバイス選択をキャンセルするときは、いつでも、**<Esc>** キーを押すか、又は、**Cancel** ボタンをクリックして下さい。

Selected device[選択されたデバイス] は自動的にバッファにデフォルトのデバイス(最大10デバイス)がセーブされます。このバッファは **Device / Select from default devices[デバイス/デフォルト・デバイスからの選択]** コマンドからアクセスすることが出来ます。

もし、現在のデバイスについての追加情報を表示した場合は、**<Ctrl+F1>** キーを使ってください。このコマンドはデバイスのサイズ、構成、プログラミング・アルゴリズムとこのデバイスがサポートされてるプログラマー(追加モジュールを含む)のリストを提供します。パッケージ情報と他の一般的な情報もご覧頂けます。

Select device / Only selected type[デバイス選択/選択されたタイプのみ]
このウィンドウはデバイスのターゲット・タイプを選択することが出来ます。最初に、デバイス・タイプ(EPROM の様に)を選択しなければいけません。そして、マウス又は、カーソル・キーを使いながら、次にデバイスのサブ・タイプ(64Kx8 (27512)の様に)を選択して下さい。そうしますと、マニファクチャラーのリストとデバイスが表示されます。

デバイスはマニファクチャラー名とデバイス番号をリストの行でダブル・クリックするか、又は、サーチ・ボックス(セパレート・キャラクターとして**<Space>** を使って、そして、

<Enter> を押すか、又は、**OK** ボタンをクリックして下さい。)で入力することで選択することが出来ます。

現在選択されているデバイスを反映せずにデバイス選択をキャンセルするときは、いつでも、**<Esc>** キーを押すか、又は、**Cancel** ボタンをクリックして下さい。

Selected device[選択されたデバイス] は自動的にバッファにデフォルトのデバイス(最大10デバイス)がセーブされます。このバッファは **Device / Select from default devices[デバイス/デフォルト・デバイスからの選択]** コマンドからアクセスすることが出来ます。

もし、現在のデバイスについての追加情報を表示した場合は、**<Ctrl+F1>** キーを使ってください。このコマンドはデバイスのサイズ、構成、プログラミング・アルゴリズムとこのデバイスがサポートされてるプログラマー(追加モジュールを含む)のリストを提供します。パッケージ情報と他の一般的な情報もご覧頂けます。

Select device... /Only selected manufacturer[デバイス選択.../選択マニュファクチャラのみ]
このウィンドウはマニュファクチャラー別によるデバイス・タイプを選択できます。最初にマウス、又は、カーソル・キーを使用してマニュファクチャラー・ボックスで希望するマニュファクチャラーを選択します。選択されたマニュファクチャラーのデバイスのリストが表示されます。

デバイスは希望のマニファクチャラー名とデバイス番号をリストの行でダブル・クリックするか、又は、デバイス番号をサーチ・ボックス(セパレート・キャラクターとしてキー **<Space>** を使用) で入力する、そして、**<Enter>** を押すか、又は、**OK** ボタンをクリックすることで選択することが出来ます。

現在選択されているデバイスを反映せずにデバイス選択をキャンセルするには、いつでも、**<Esc>** キーを押すか、又は、**Cancel** ボタンをクリックして下さい。

選択されたデバイスは自動的にデフォルト・デバイスのバッファにセーブされます。このバッファは**Device/Select from default devices[デバイス/デフォルト・デバイスからの選択]** コマンドからアクセスすることが出来ます。

もし、現在のデバイスについての追加情報を表示した場合は、ボタン**Device info**、又は、<Ctrl+F1> キーを使ってください。このコマンドはデバイスのサイズ、構成、プログラミング・アルゴリズムとこのデバイスがサポートされてるプログラマー(追加モジュールを含む)のリストを提供します。パッケージ情報と他の一般的な情報もご覧頂けます。

Device /Select EPROM /Flash by ID[デバイス/ID による EPROM 選択]
このコマンドはデバイス IDを読むことでアクティブ・デバイスとして EPROMを自動選択するのに使用します。プログラマーはチップに焼き付けられているマニファクチャラーとデバイスの IDを読むことで EPROMを自動的に認識します。これは、この機能をサポートしている EPROM のみに適応されます。もし、デバイスがチップ

IDとマニファクチャーIDをサポートしていないときは UNKOWN 又は、NOT SUPPORTED DEVICE であることを告げるメッセージを表示します。

他に一致したチップ IC とマニュファクチャーID が検知されますと、これらのデバイスのリストが表示されます。リストから、その番号(又は、マニファクチャー名)を選ぶことで、このリストから対応デバイスを選択することが出来ます。そして、**<Enter>** を押すか、又は、**OK** ボタンをクリックして下さい。
現在選択されているデバイスを反映せずにデバイス選択をキャンセルするときは、いつでも、**<Esc>** キーを押すか、又は、**Cancel** ボタンをクリックして下さい。

警告: 制御プログラムは 28 ピンと 32 ピンの EPROM と Flash のみをサポートします。どのプログラマもピン番号を自動的に決定します。他のプログラマはこの番号を手動で入力する必要があります。

プログラマはソケットの相応のピンに高電圧を印加します。これはシステムがデバイス ID を読み取るために必要です。 EPROM、又は、Flash 以外のデバイスをソケットに挿入しないでください。プログラマが高電圧を印加すると破損することがあります。

このコマンドを次のように適用することはお勧めしません:
1) 2764 と 27128 の EPROM タイプ。そのほとんどは ID がサポートしていないためです。
2) 非標準のピン配置を有する Flash メモリ (e.g. Firmware Hub Flash)
3) A9 ピンで Vid 電圧を受け付けない Flash メモリ
4) 低電圧 EPROと Flash メモリ

Device / Device options [デバイス/デバイス・オプション]
このメニューのすべての設定はプログラミング・プロセス、シリアライゼーションと関連ファイルのコントロールに使用されます。

Device / Device options / Operation options [デバイス/デバイス・オプション/操作オプション]
このコマンドのすべての設定はプログラミング・プロセスのコントロールに使用されます。これはターゲット・デバイスとプログラマーのタイプに関連した項目を含むフレキシブルな環境です。項目はターゲット・デバイスに対しては有効ですが、現在のプログラマーではサポートされておりませんので、ディスエーブルになっています。これらの設定は **File / Exit and save[ファイル/終了と保存]** コマンドにより関連デバイスと共にディスクにセーブされます。

通常使われる項目のリスト:

アドレス・グループ
Device start address デバイス開始アドレス (デフォルト 0)
Device end address デバイス終了アドレス (デフォルト・デバイス・サイズ-1)
Buffer start address バッファ開始アドレス (デフォルト 0)

スプリット
このオプションはプログラミングやデバイスを読み取る際にバッファの特殊なモードを設定することが出来ます。16-bit、又は、32-bit のアプリケーションを 8bit のデータ・メモリ・デバイスに書き込みに使用する際に分割オプションを使用すると特に有用です。

次の表はバッファからデバイスとデバイスからバッファのデータ転送を説明しています。

Split type	Device Buffer	Address assignment
None	Device [ADDR]	Buffer [ADDR]
Even	Device [ADDR]	Buffer [2*ADDR]
Odd	Device [ADDR]	Buffer [1+ (2*ADDR)]
1./4	Device [ADDR]	Buffer [4*ADDR]
2./4	Device [ADDR]	Buffer [1+(4*ADDR)]
3./4	Device [ADDR]	Buffer [2+(4*ADDR)]
4./4	Device [ADDR]	Buffer [3+(4*ADDR)]

実際のアドレスは次のようになります: (全てのアドレスは16進数です)

Split type	Device addresses	Buffer addresses
None	00 01 02 03 04 05	00 01 02 03 04 05
Even	00 01 02 03 04 05	00 02 04 06 08 0A
Odd	00 01 02 03 04 05	01 03 05 07 09 0B
1./4	00 01 02 03 04 05	00 04 08 0C 10 14
2./4	00 01 02 03 04 05	01 05 09 0D 11 15
3./4	00 01 02 03 04 05	02 06 0A 0E 12 16
4./4	00 01 02 03 04 05	03 07 0B 0F 13 17

用語説明:
デバイス・アドレス ADDR へのアクセスは Device[ADDR]と書かれています。
バッファ・アドレス ADDR へのアクセスは Buffer[ADDR]と書かれています。
ADDR 値は 0 からデバイス・サイズ(バイト)にすることが出来ます。
全てのアドレスはバイト指向アドレスです。

インサーション・テスト・グループ:
Insertion test 装着テスト (デフォルト ENABLE)
有効の場合、プログラマは ZIF ソケットとの接続が正しいかチップの全てのピンをチェックします。プログラマはチップの誤装着と逆挿しの接触不良を認識します。

Device ID チェック・エラーは操作を終了 (デフォルト ENABLE)
プログラマーは選択された各アクションの前に ID チェックを提供します。
これはデバイスの製造業者によって定義された ID コードをデバイスから読み出した ID コードとを比較します。ID エラーの場合、制御プログラムが次のように動作します:
- 項目を ENABLE[有効]にセットしている場合、選択された動作は終了します。
- 項目を DISABLE[無効]にセットしている場合は、選択した動作が続行されます。制御プログラムは単に ID エラーに関する警告メッセージを LOG ウィンドウに書き込みます。

有効にした場合、プログラマーはプログラムされたチップの電子 ID をチェックします。

ノート 1: ある種の古いチップは電子的な ID 機能を持っていません。
ノート 2: ある特殊な場合、チップ内のコピー防止機能が設定されている場合、ある種の特殊な場合はマイクロコントローラは制御プログラムのデバイス ID チェック設定を"Enable[有効]"に設定されていても ID 機能は利用出来ません。

コマンド実行グループ:
プログラミング前にブランクチェック(デフォルト DISABLE)
プログラミング前にイレース (デフォルト DISABLE)
読み込み後のベリファイ (デフォルト ENABLE)
プログラミング後のベリファイ (ONCE, TWICE)
ベリファイ・オプション (nominal VCC 5%,
nominal VCC 10%,
VCCmin VCCmax)

ターゲット・システム電源供給パラメータ
ある種のタイプのデバイスでは ISP モードを利用出来ます。次のセッティングを含んでいます:
Enable target system power supply[ターゲット・システムの電源供給を有効にする] - プログラマーからターゲット・システムへの電源供給を可能にします。プログラムされたデバイスとのアクションの前にターゲット・システムのための供給電源がスイッチオンされ、そして、アクションが完了した後にスイッチオフされます。もし、操作が有効にされた後に定義されたレベルでの ISP の信号を保持する場合、プル-アップ/プル-ダウン抵抗が無効にされた後、プログラマーは電源供給のスイッチをオフにします。

Voltage[電圧] - ターゲット・システムへの供給電源。供給電圧は 2V～6V です。
ノート:ターゲット・システムに供給する電圧値はターゲット・システムへの電流に依存します。ターゲット・システムへの適切な電源供給に達するためには、適切な電圧と最大電流値が定義される必要があります。最大電流値はターゲット・システムの実際の消費電流と同じで可能な限り正確でなければいけません。

Max. current[最大電流] - 電源供給されたターゲット・システムの最大電流消費。電流消費範囲は 0～300mA

Voltage rise time[電圧立ち上がり時間] - ターゲット・システム電源供給電圧の立ち上がり時間のスキュー・レートを決定します。(供給電圧をスイッチオン)

Target supply settle time[ターゲット供給セトル時間] - ターゲット・システムで供給した電圧が設定値で安定し、ターゲット・システムでプログラムされるデバイスでの動作が準備される時間を決定。

Voltage fall time[電圧立ち下がり時間] - ターゲット・システム電源供給電圧の立ち下がり時間のスキュー・レートを決定します。(供給電圧をスイッチオフ)

Power down time[パワー・ダウン時間] - ターゲット・システム内でターゲット・システムの電源供給がスイッチオフされた後、残留供給電圧を維持する時間(例え

ば充電されたコンデンサから)決定。この時間の経過後ターゲット・システムは供給電源なしでプログラマーから安全に切り離すことが出来ます。

ターゲット・システムのパラメータ
これはある種のデバイスのタイプの ISP モードで利用出来ます。次のセッティングを含みます:
Oscillator frequency (in Hz)[オシレータ周波数(in Hz)] - デバイス(ターゲット・システム)のオシレータの周波数。コントロール・プログラムはそれによってプログラミング速度をセットしますので、正しい値をセットする必要があります。

Supply voltage (in mV) 供給電圧(in mV) - ターゲット・システム側の供給電源。制御プログラムは全てのアクションの前にターゲット・システムで入力された供給電圧をチェック。又は、セット(プログラマの種類によって異なります)します。

Disable test supply voltage[供給電圧のテストを無効にする] - デバイスでの動作の前に Supply voltage edit[供給電圧エディット]ボックスでセットしたプログラムされたデバイスの供給電圧の測定とチェックを無効にします。

Delay after reset active[リセット・アクティブ後の遅延] - このパラメータはデバイスで動作を開始するためにリセット信号アクティブ後の遅延を決定します。この遅延はデバイスのリセット回路で使用されるデバイスの値に依存し、そして、
次の値から選択することが出来ます:10ms, 50ms, 100ms, 500ms 又は、1s.

Inactive level of ISP signals ISP[信号の待機レベル] - このパラメータはターゲット・デバイスへのアクセス終了後 ISP 信号のレベルを決定します。ISP コネクターの信号はプル-アップ(信号が 22k 抵抗を経由して供給電圧に接続されている)、又は、プル-ダウン(信号が 22k 抵抗を経由してグランドに接続されている)にセットすることが出来ます。

Keep ISP signals at defined level after operation[操作後定義されたレベルで ISP 信号を保持] - ターゲット・デバイスにアクセス終了後に ISP 信号のセットレベルを保持します。制御プログラムは警告ウィンドウを表示することでアクティベートされたプル-アップ/プル-ダウン抵抗を示します。ユーザーがこのウィンドウ制御プログラムを閉じますと抵抗を無効化します。

プログラミング・パラメータ
これはある種のデバイス・タイプに利用出来ます。プログラムされるデバイスのブロック、又は、領域、チップのプロテクトのセッティングを含みます。

イレース・パラメータ
これはある種のデバイス・タイプに利用出来ます。選択されたデバイスのイレース・モードのセッティングを含みます。

Device / Device options / Serialization[デバイス/デバイス・オプション/シリアライゼーション]

シリアライゼーションはプログラムの特殊なモードです。シリアライゼーション・モードがアクティブな時、各デバイスにプログラミングする前に指定した値が自動的にバッファの前もって定義されたアドレスに挿入されます。そして、次から次へとデバイスをプログラムする時、自動的にシリアル番号の値が.変更してデバイスのプログラミングの前にバッファに挿入されます。従って、各々のデバイスが特有のシリアル番号を持つことが出来ます.

シリアライゼーションには３つのタイプがあります。:
•インクレメンタル[増加]・モード
•ファイルからのモード
•カスタム・ジェネレーター・モード

ダイアログ **Serialization[シリアライゼーション]** はシリアライゼーションがオンにされた場合にプロジェクト・ファイルで使用される関連したシリアライゼーション位置ファイルのための設定も含まれています。さらに詳しくは" シリアライゼーションとプロジェクト"をご覧下さい。

シリアライゼーションのベーシック・ルール:

•シリアライゼーションは最近選択されたデバイスにのみ関連付けられます。 新しいデバイスを選択すると、シリアライゼーション設定がリセットされます(シリアライゼーションは無効に設定されます)。
•最近のデバイスのシリアライゼーション設定はデバイスのプロジェクト・ファイル、又は、アプリケーションが閉じられたときのコンフィギュレーション・ファイルと他の設定と共に保存されます。
•シリアライゼーション・エンジンは各デバイス・プログラミングが開始される前に新しい(次の)シリアル番号を要求します(ノート１を参照)。
•使用されたシリアル番号はシリアル番号の値の後に(*)で示されます。シリアル番号を使用すると、次のデバイス・プログラミングは次のシリアル番号を使用します(ノート１を参照)。

ノート 1): オプションの Serial number usage if programming action fails[プログラミング動作が失敗した場合のシリアル番号の使用]により以前のデバイス・プログラミングの結果が失敗した場合、プログラミング前に新しいシリアル番号要求を呼び出すことを抑制できます。:
•Reuse generated serial number for next programmed device[次のプログラムされるデバイスのために生成されたシリアル番号を再使用]オプションが選択されている場合、以前のデバイス操作結果が失敗した場合に新しい(次の)シリアル番号の要求が抑制されます。つまり、使用されたシリアル番号が再び使用され、正常なデバイス・プログラミングが完了するまで同じシリアル番号が使用されます。
•Throw away (use the serial number only once, regardless result of the programming) Throw away[捨てる](プログラミングの結果に関係なくシリアル番号を 1 回だけ使用する)が選択された場合、以前のプログラミング操作の

結果に関係なく、各デバイス操作の前に新しいシリアル番号の要求が実行されます。

シリアライゼーションはある種のデバイスのタイプに対しては PG4UW コントロール・プログラムのメイン・バッファ、又は、使用可能な拡張バッファを操作することが出来ます。例えば、データ EEPROM メモリーを搭載したマイクロチップ PIC16FXXX 等、一部のタイプのデバイスで使用可能な拡張バッファで動作します。どのバッファをシリアライゼーション・ルーチンにより使用するかはダイアログ Serialization[シリアライゼーション]で選択可能です。Buffer[バッファ]設定ボックスが表示されていない場合、現在選択のデバイスのシリアライゼーション・モードは拡張バッファをサポートしていません。

Device / Device options / Serialization / Incremental mode & SQTP[デバイス/デバイス・オプション/シリアライゼーション/インクレメント・モード & SQTP]

Incremental mode & SQTP[インクレメンタル・モードと SQTP]は各プログラム・デバイスに個別のシリアル番号を割り当てることが出来ます。各デバイスのプログラム操作に対してユーザーにより入力された開始番号が指定されたステップで増加され、そして、各デバイスのプログラミングに先立ち、選択されたフォーマットで指定されたバッファ・アドレスにロードされます。

インクレメンタル・モードのためにユーザーが修正することが出来るオプションには以下の項目があります:

S / N size[S /N サイズ]
S/N サイズ・オプションはバッファに書込まれるシリアル値のバイトの数を定義します。S/N サイズでは Bin(バイナリー) シリアライゼーション・モードの値は 1-8 が有効で、そして、ASCII シリアライゼーション・モードでは 1-16 の値が有効値です。

Addres[アドレス]
アドレス・オプションはシリアル値が書込まれるバッファ・アドレスを指定します。アドレス範囲はデバイスの開始と終了のアドレスの範囲内でなければいけません。アドレスはシリアル値の最後(最上位、又は、最下位)バイトがデバイスの開始と終了のアドレス範囲の中に指定されなければいけませんので、正しく指定されなければいけません。

Start value[スタート値]
スタート値オプションはシリアライゼーションが開始されるイニシャル値を指定します。一般的にシリアライゼーションの最大値は 32bit long word で$1FFFFFFF です。実際のシリアル値が最大値を超えた場合は、シリアル番号の 3 つの最上位ビットがゼロにセットされます。このアクションの後、数値は常に 0 .. $ 1FFFFFFF の間隔内にあります(これはオーバーフロー処理の基本スタイルです)。

Step[ステップ]
ステップ・オプションはシリアル値のインプリメンテーションの増加ステップを指定します。

S/N mode[S/N モード]
S/N モード・オプションはバッファに書込まなければいけないシリアル値の形式を定義します。2つのオプションが利用できます:
• ASCII
• Bin
ASCII - シリアル番号が ASCII 文字列としてバッファに書込まれることを意味します. 例えば、番号$0528CD は ASCII モードで 30h 35h 32h 38h 43h 44h ('0' '5' '2' '8' 'C' 'D') としてバッファに書込まれます, 即ち、6 バイトです。
Bin - シリアル番号が直接バッファに書込まれることを意味します。もし、シリアル番号が1バイト長以上の場合, 2 つの可能なバイト・オーダーの1つに書くことができます。バイト・オーダーは Save to buffer[バッファにセーブ]項目で変更することが出来ます。

Style[スタイル]
スタイル・オプションはシリアル番号ベースを定義します。2 つのオプションがあります:
• Decimal[デシマル - 10 進数]
• Hexadecimal[ヘキサデシマル - 16 進数]
Decimal[デシマル]番号は'0' から '9'のキャラクターを使って入力と表示がされます。Hexadecimal[ヘキサ・デシマル]番号は'A' から'F'のキャラクターを使います。特別なケースは Binary Dec[バイナリー・デシマル]で、これは BCD 番号スタイルを意味します。BCD はデシマル番号がヘキサ・デシマル番号にストアされることを意味します、即ち、各ニブルが 0 から 9 までの値を持たなければいけません。
A から F の値は BCD 番号のニブルとしては使用出来ません。シリアル開始値とステップの数字を入れるまえに"Style"[スタイル]オプションでベースを選択して下さい。

Save to buffer[バッファにセーブ]
Save to buffer[バッファにセーブ]オプションはバッファに書込むためのシリアル値のバイト・オーダーを指定します。このオプションは Bin S / N モード(ASCII モードには役立ちません)に対して使用されます。2 つのオプションが利用出来ます。:
• LSByte first (インテルのプロセッサーで使用されています。) はシリアル番号のリースト・シグニフィカント・バイトをバッファの最下位アドレスに置きます。
• MSByte first (モトローラのプロセッサーで使用されています。) はモースト・シグニフィカント・バイトをバッファの最下位アドレスに置きます。

Split serial number[スプリット・シリアル番号]
オプションはシリアル番号を個々のバイトにスプリットし、そして、バッファの各N番目のアドレスにバイトを配置することが出来ます。この機能はデバイスのシリアル番号をRETLWまたはNOP命令のグループとしてプログラムメモリの一部とすることが出来る時、マイクロチップ社のPICデバイスのためのSQTPシリアライゼーション・モードのために特に有用です。詳細については以下のサンプルのサンプル2を参照して下さい。

次のスプリット・オプションが利用可能:
•チェック・ボックス **"Split serial number"** –スプリット機能のターンオン/オフ

•**Split gap** – スプリット・シリアル番号のフラグメント間に置くバイト数を指定
•**S/N fragment size** – シリアル番号はこのオプションにより指定されたサイズでフラグメントにスプリットされます。

サンプル 1:
アドレス7FFFAHでAT29C040デバイスにシリアル番号を書く、シリアル番号のサイズは4バイト。開始値は16000000H。インクリメンタル・ステップは1。シリアル番号の形式はバイナリーそして、最下位バイトはデバイスのシリアル番号の下位アドレスに配置されます。

上記に記載のシリアライゼーションを作成するにはシリアライゼーション・ダイアログで次の設定をする必要があります:

モード: インクリメンタル・モード
S/N size: 4 bytes
S/N mode:: Bin Style: Hex
Save to buffer: LS Byte
first Address: 7FFFCH
Start value: 16000000H

Step: 1
以下の値がデバイスに書き込まれます:
1番目のデバイス
アドレス　データ
007FFF0　xx xx xx xx xx xx xx xx xx xx xx xx 00 00 00 16
2番目のデバイス
アドレス　データ
007FFF0　xx xx xx xx xx xx xx xx xx xx xx xx 01 00 00 16
3番目のデバイス
アドレス　データ
007FFF0　xx xx xx xx xx xx xx xx xx xx xx xx 02 00 00 16 etc.
"xx" はデバイスにプログラムされるユーザー・データ
シリアル番号はデバイスのアドレス7FFFCH から7FFFFHに書き込まれます。
シリアル番号のサイズは4バイトです。

サンプル 2:
次のサンプルはシリアル番号がマイクロチップPIC16F628デバイスに対してRETLW命令にスプリットされる時のSQTPシリアライゼーション・モードの使用方法を示します。

ノート: シリアル・クイック・ターン・プログラミング(SQTP)はマイクロチップ社のPICマイクロコントローラのシリアル・プログラミングのために、マイクロチップ社に指定された標準方式です。 マイクロチップPICデバイスを使用すると各マイクロコントローラに固有のシリアル番号をプログラムすることができます。この数はエントリコード、パスワード、又は、ID番号として使用することが出来ます。

シリアライゼーションはリテラル・データとして、シリアル番号のバイトで、RETLW(リターンリテラルW)命令を連続使用して行われます。シリアライズするには、インクリメンタル・モードのシリアライゼーション、又は、From File modeシリアライゼーションを使用することが出来ます。

インクリメンタル・シリアライゼーションはシリアル番号を分割するSplit機能を提供

します。シリアル番号分割機能は、偶数又は奇数バイトに分割された増加分の使用を可能にし、そして、シリアル番号の各バイトの間にはRETLW命令コードが挿入されます。

"From file"シリアライゼーションは独自のシリアル番号ファイルを使用しています。このファイルは色々なシリアル番号で構成することが出来ます。この番号はSQTPに適した形式を持つことが出来ます。たとえばRETLW b1 RETLW b2等です。ノート:PG4UWのシリアル・ファイル形式はマイクロチップ社のMPLABによって生成されるSQTPシリアル・ファイルとは互換性がありません。

サンプル 2a:
Microchip PIC16F628デバイスに対してシリアライゼーションの分割を使用すると、RETLW命令で分割します。

PIC16F628は14ビット幅命令ワードを持っています。RETLW命令は14ビット・オペコードを持っています:

説明		MSB	14-Bit word	LSB
RETLW	Wリテラルで返す	11	01xx kkkk	kkkk

xは00に置き換えることが出来、そして、kはデータ・ビット、即ち、シリアル番号のバイトです。

RETLW命令のオペコードは、KKがデータ・バイト(シリアル番号のバイト)であり、ヘキサで 34KKH です。

例えば、シリアル番号1234ABCDHをデバイスPICに4つのRETLW命令の1部として書きたいとします。シリアル番号の最上位バイトがMSB(最上位バイト)です。デバイスのプログラム・メモリーのアドレス40Hにシリアル番号を書きたいとします。シリアル番号分割はこのような状況では非常に便利です。シリアル番号の分割なしでシリアライゼーションは次のバッファとデバイスに番号を書きます:

```
アドレス   データ
0000080    CD AB 34 12 xx xx xx xx xx xx xx xx xx xx xx xx
```

ノート: アドレス80Hはバッファがバイト構成を持っており、PICはワード構成を有していますので、プログラム・メモリーのアドレス40Hと同等であるためです。バッファがワード構成 x16を持っている場合、アドレス40Hと番号1234ABCDHは次のようにバッファに配置されます:

```
アドレス   データ
0000040    ABCD 1234 xxxx xxxx xxxx xxxx xxxx xxxx
```

RETLW命令を使いたいと仮定しますとバッファは:

```
アドレス   データ
0000040    34CD 34AB 3434 3412 xxxx xxxx xxxx xxxx
```

これは次のステップで行うことが出来ます:

A) メイン・バッファのアドレス 40H に 4 つの RETLW 命令を書く(これは手動でバッファを編集するか、又は、適切な内容のファイルをロードすることによって行うことができます)。各 RETLW 命令の下位 8 ビットは重要ではありません、それは

各 RETLW 命令の下位 8 ビットには、シリアライゼーションの正しいシリアル番号のバイトが書き込まれる為です。

デバイスのプログラムを開始する前のバッファの内容は例えば以下の様に見えます:

```
アドレス   データ
0000040    3400 3400 3400 3400 xxxx xxxx xxxx xxxx
```
各RETLW命令8ビットはゼロです。それらは如何なる値も持つことが出来ます。

B) 次のような シリアライゼーション・オプションをセット:

S/N size:	4 Bytes
Address:	40H
Start value:	1234ABCDH
Step:	1
S/N mode:	BIN
Style:	HEX
Save to buffer:	LS Byte first
Split serial number:	checked
Split gap:	1 byte(s)
S/N fragment size:	1 byte(s)

上述のスプリットの設定は、2番目のバイト毎に、バイトによるシリアル番号を分けてバッファします。正しいシリアル番号は、デバイスのプログラミング操作が開始される前に、しっかりと設定されます。

最初のデバイスがプログラミングされる時のシリアル番号のバッファ内容は:

```
アドレス       データ
0000040    34CD 34AB 3434 3412 xxxx xxxx xxxx xxxx
```

2番目のデバイスは:

```
アドレス       データ
0000040    34CE 34AB 3434 3412 xxxx xxxx xxxx xxxx
```

次のデバイスは同じフォーマットのシリアル番号を持ち、各デバイスに対して1でインクリメントされます。

サンプル 2b
Microchip PIC24FJ256 デバイスのためのNOP 命令を持った シリアライゼーション・スプリットの使用。

デバイス PIC24FJ256は24ビット幅の命令ワードを持っています。NOP命令はコード00xxxxhを持っています。Microchip MPLAB®で指定されているSQTPシリアライゼーションと同じ方法のシリアライゼーションを使用するとします:

次のステップでこれを行うことが出来ます:

A) PG4UWのメイン・バッファのアドレス800hにNOP命令(00xxxxh)を書き込みます。これは編集バッファを手動、又は、正しい内容を持ったファイルをロードすることによって行うことが出来ます。PG4UW バッファ内のアドレス800hは

PIC24Fxxxプログラム・メモリーのアドレス200hと同等です。詳細についてはPG4UWのPIC24FJ256デバイス用のデバイス情報を見て下さい。
例えば、デバイスのプログラム開始前において、アドレス800hのNOPでのバッファ内容は次の様になります:

```
アドレス          データ
0000800       00 00 00 00 00 00 00 00 xx xx xx xx xx xx xx xx
```

xx – はバイト値を意味します。

B) 次のようなシリアライゼーション・オプションをセット:

```
S/N size:                3 bytes
Address:                 800h
Start value:             123456h
Step:                    1
S/N mode:                BIN
Style:                   HEX
Save to buffer:          LS byte first
Split serial number:     checked
Split gap:               2 byte(s)
S/N fragment size:       2 byte(s)
```

上述のスプリットの設定はフラグメント間の2バイトのギャップで16ビット(2バイト)サイズのフラグメントにシリアル番号のスプリットをバッファします。正しいシリアル番号は、デバイスのプログラミング操作が開始される前に、しっかりと設定されます。

最初のデバイスがプログラミングされる時のシリアル番号のバッファ内容は:

```
アドレス          データ
0000800       56 34 00 00 12 00 00 00 xx xx xx xx xx xx xx xx
```

2番目のデバイスは:

```
アドレス          データ
0000800       57 34 00 00 12 00 00 00 xx xx xx xx xx xx xx xx
```

次のデバイスは同じフォーマットのシリアル番号を持ち、各デバイスに対して1でインクリメントされます。

サンプル 3:
次のサンプルでは代わりにシリアル番号スプリット・ギャップが2と3に設定されているサンプル2aと同じシリアライゼーション・オプションを使用しています。.

スプリット・ギャップが3バイトにセットさている時、バッファ内容は次の様に見えます:

バイト・バッファ構成:

```
アドレス     データ
0000080    CD xx xx xx AB xx xx xx 34 xx xx xx 12
```

ワード16 バッファ構成:

```
アドレス     データ
0000040    xxCD xxxx xxAB xxxx xx34 xxxx xx12 xxxx
```

ノート: シリアライゼーション・オプションの効果がわからない時は、バッファに書き込まれる実際のシリアル番号をテストすることが可能です。テストは次のステップで行うことが出来ます。
1.ダイアログ"シリアライゼーション"で希望するシリアライゼーションを選択し、OKボタンで確認します。
2.デバイス操作オプションでインサーション・テストとDevice IDチェックを無効にします。
3.プログラマーのZIFソケットにデバイスが装着されていないことを確認して下さい。
4.デバイス・プログラム操作を実行(ある種のデバイスではプログラミングの開始前にプログラミング・オプションを選択する必要があります)
5.プログラミング操作が終了後(殆どの場合、デバイスが装着されていませんのでエラーとなります)、どこにシリアル番号が置くかはアドレスのメイン・バッファ(View/Editバッファ)で見て下さい。

ノート: シリアライゼーションのためのアドレスは常に制御プログラムが現在のデバイスに使用している実際のデバイスとバッファの構成に対して割り当てられます。もし、バッファ構成がバイトorg (x8)であれば、シリアライゼーション・アドレスはバイト・アドレス。もし、バッファ構成がバイトより広い、例えば、16ビット・ワード(x16)より広いならシリアライゼーション・アドレスはワード・アドレスになります。

Device / Device options / Serialization / Classic From file mode
[デバイス/デバイス・オプション/シリアライゼーション/クラシック・フロム・ファイル・モード]
Classic From-file mode[クラシック・フロム・ファイル・モード]を使用する場合、シリアライゼーション・ファイルにはシリアル値が直接含まれています。シリアライゼーション・データはシリアライゼーション・ファイルからファイルに指定されたアドレスのバッファに直接読み込まれます。クラシック・フロム・ファイル・モードはPG4UWコントロール・プログラムのメイン・ウインドウとj情報ウインドウに**"From File"**シリアライゼーションとしてパネル"Serialization"で表示されます。
2つのユーザー・オプションがあります:
Start label[スタート・ラベル]
開始ラベルは入力ファイルの開始ラベルを定義します。 ファイルからのシリアル値は定義された開始ラベルから読み取りを開始します。

File name[ファイル名]
Classis from file[クラシック・フロム・ファイル]のためのシリアライゼーションの入力ファイルは正しい形式でなければいけません。.

File format[ファイル・フォーマット]
クラシック From-file シリアライゼーション入力ファイルはテキスト形式です。このファイルはバッファ・アドレスとバッファに書き込むデータを定義するバイトのアドレスと配列が含まれます。入力ファイルにはテキスト形式の形式があり、その構造は次のとおりです:

```
[label1]   addr byte0 byte1 .. byten ...
[labeln]   addr byte0 byte1 .. bytem , addr byte0 byte1 ... bytek
          \_____________________/   \_____________________/
```

| |
ベーシック・パート オプショナル・パート

; Comment[コメント]

意味は:

ベーシック・パート
基本部分はバッファ・アドレスとバッファに書き込むバイトの配列を定義します。基本部分は常にラベルの行の後に定義する必要があります。

オプショナル・パート
オプショナル部分は2番目のバイトの配列とバッファに書き込むバッファ・アドレスを定義します。オプショナル部分の一部はデータの基本部分の後に定義することができます。

label*1*, label*n* – ラベル
ラベルは入力ファイルの各行の識別子です。これらはファイルの各行のアドレス指定に使用されます。ラベルはユニークでなければいけません。ファイルの行をアドレス指定するとは、ユーザーが入力する必要な開始ラベルはシリアル値の読み込みを開始する入力ファイルでの行を定義します。

addr -
Addrはアドレスに続くデータを書き込むバッファ・アドレスを定義します。

byte*0*..byte*n*, byte*0*..byte*m*, byte*0*..byte*k* -
バイト配列byte0..byten、byte0..bytemとbyte0..bytekはバッファに書き込むために割り当てられるデータを定義しています。アドレスに続く1つのデータ・フィールドの最大バイト数は64バイトです。データ・バイトはアドレスaddrからaddr+nまでのバッファに書き込まれます。
特定のバイトをバッファに書き込むプロセスは次のとおりです:

byte*0* to addr
byte*1* to addr + *1*
byte2 to addr + 2
....
byte*n* to addr + *n*

Optional part[オプショナル部分]は最初のデータ部分から文字“,“(カンマ)で区切られ、その構造は最初のデータ部分と同じです。即ち、アドレスとそれに続くデータバイトの配列です。

特別使用の文字:
[ ] - ラベルは角括弧の中に定義する必要があります。
',' – データのベーシック・パート[基本部分]とオプショナル部分を区切る文字
";" - セミコロン文字はコメントの先頭を意味します。";"から行末までの全ての文字は無視されます。コメントは個々の行または定義行の最後に置くことができます。
ノート:
• *ラベル名は '['と']' を除く全ての文字を含めることができます。ラベル名は大文字と小文字を区別しないように分析されます。即ち、文字 'a' は 'A' と同じで 'b' は 'B' と同じです。*

- *入力ファイルの全てのアドレスとバイト番号の値は16進数です。*
- *許容されるアドレス値のサイズは1 ～4バイトです。*
- *1行のデータ配列の許容サイズは1から64バイトの範囲です。1行に2つのデータ配列がある場合、それらのサイズの合計は最大80バイト迄です。*
- *正しいアドレスをセットするように注意してください。アドレスはデバイスの開始アドレスとデバイス終了アドレスの範囲内で定義する必要があります。アドレスが範囲外の場合、警告ウィンドウが表示されシリアライゼーションは無効にセットされます。*
- *シリアライゼーションのためのアドレスは制御プログラムが現在のデバイスに使用している実際のデバイス構成とバッファ構成に常に割り当てられます。バッファ構成がバイト構成の場合(x8)、シリアライゼーション・アドレスはバイト・アドレスになります。バッファ構成がバイトよりも広い場合、例えば、16ビットワード(x16)の場合、シリアライゼーション・アドレスはワード・アドレスになります。*

Classic From file[クラシック・フロム・ファイル]シリアライゼーションの典型的な入力ファイルの例]:

[nav*1*] A7890 78 9 56 02 AB CD ; comment*1*
[nav*2*] A7890 02 02 04 06 08 0A
[nav*3*] A7890 08 09 0A 0B A0 C0 ; comment*2*
[nav*4*] A7890 68 87 50 02 0B 8D
[nav*5*] A7890 A8 88 59 02 AB 7D

;次の行には2番目の定義も含まれます
[nav*6*] A7890 18 29 36 42 5B 6D , FFFF6 44 11 22 33 99 88 77 66 55 16

; これは最後の行です - ファイルの最後

この例のファイルでは labels „nav1“, „nav2“, ...“nav6“の6つのシリアル値が定義されています。各値はアドレス$A7890のバッファに書き込まれます。 全ての値のサイズは6バイトです。 „nav6“ラベルの行はまたアドレス$FFFF6にバッファリングされサイズが10バイトである第2の定義 値を持っています。即ち、この値の最後のバイトはアドレス$FFFFFに書き込まれます。

ノート: シリアライゼーションのアドレスは制御プログラムが現在のデバイスに使用している実際のデバイス構成とバッファ構 成に常に割り当てられます。バッファ構成がバイト構 成の場合、(x8)、シリアライゼーション・アドレスはバイト・アドレスになります。バッファ構 成がバイトよりも広い場合、例えば、16ビットワード(x16)の場合、シリアライゼーション・アドレスはワード・アドレスになります。

Device / Device options / Serialization / Playlist From file mode [デバイス/デバイス・オプション/シリアライゼーション/ファイルからのモード]

Playlist From-file mode[プレイリスト・ファイルからのモード]を使用するとシリアライゼーション・ファイルは含まれるシリアル値を直 接は持っていません。ファイルはシリアライゼーション・データが含まれ ている外部ファイルの名 前のリストが含まれています。シリアライゼーション・データはこれらの外部データ・ファイルから読み出され、各ファイルは1つの シリアライゼーション・ステップ(1つのデバイスがプログラムされる)を意味します。Playlist From-file mode[プレイリスト・ファイ

ルからのモード]はPG4UW制御プログラムのメイン・ウィンドウと情報ウィンドウに"From-file-pl[プレイリスト・ファイルからのモード]"シリアライゼーションとして"Serialization[シリアライゼーション]"パネルに表示されます。

ファイル・フォーマット
From-file[ファイルから]シリアライゼーション・プレイリスト・ファイルはシリアライゼーション・データを持ったファイル名のリストを含みます。そのファイル・フォーマットはクラシック・シリアライゼーション・ファイル・フォーマットに似ています。ファイル・フォーマットの違いはプレイリスト・ファイルにおいては次の通りです:

1. playlistファイルはファイルの最初に空白行でない特別なヘッダを持つ必要があります。そのヘッダ行のフォーマットはテキスト形式です。
   FILETYPE=PG4UW SERIALIZATION PLAYLIST FILE
2. 各シリアル・データ・パッチは別の行で次のフォーマットで表わされます。
[label x] datafilename

labelx - ラベルを現わします。
ラベルは入力ファイルの各行が空白でない事を示すための識別子です。これらはファイルの各行をアドレス指定するために使用されます。ラベルはファイル内でユニークである必要があります。ファイルの行のアドレス指定はユーザーにより入力された必要な開始ラベルがシリアル値の読み込みを開始する入力ファイルの行を定義することを意味します。

datafilename[データ・ファイル名] - シリアライゼーション・データを含むデータ・ファイルの名前を定義します。シリアライゼーションが新しいシリアル値を必要とする場合、データ・ファイルは標準のPG4UW "Load file[ロード・ファイル]"の手順で、PG4UWのバッファへロードされます。ファイル形式はバイナリー又は、ヘキサ・ファイル(Intel Hex等)に対応しています。自動認識システムは適切なファイル形式を認識し、そして、正しいファイル形式のファイルのロードを行います。データ・ファイル名はペアレント(playlist)のシリアライゼーション・ファイルと関連しています。

playlist シリアライゼーション・ファイルのサンプル:
;---- 次のファイル・ヘッダーが必要です。----------
FILETYPE=PG4UW SERIALIZATION PLAYLIST FILE

;----- シリアライゼーション・データ・ファイルの参照
[nav1] file1.dat
[nav2] file2.dat
[nav3] file3.dat
...
[label n] filex.dat
;-------- end of file ----------

シリアライゼーション・タイプ From-file playlistのより詳細で完全に機能するサンプル例については、次のようにPG4UWインストレーション・ディレクトリのExamples\subdirectoryにあるサンプル・ファイルを参照してください:
<PG4UW_inst_dir>\Examples\Serialization\fromfile_playlist_example\
一般的なパスは以下の様になります:
C:\Program Files

(x86)\Elnec_sw\Programmer\Examples\Serialization\
fromfile_playlist_example\

次のステップでシリアライゼーションをテストすることが出来ます:
1. PG4UWを実行
2. ELNECプログラマが接続されて正しくPG4UWで認識されている必要があります。
3. 希望するデバイスを選択、イレース可能なメモリ・デバイスをお薦めします。(OTPメモリではありません)
4. Device | Device Options | Serializationメニューからダイアログを選択
5. パネルFrom-file modeオプションでFrom-file modeをセットしサンプルのシリアライゼーション・ファイル fromfile_playlist.serを選択して下さい。
6. 新しいシリアライゼーションの設定を受け付けるためにOKボタンをクリックします。
7. デバイス操作で "Program[プログラム]"を実行して下さい。

PG4UWのメイン・ウィンドウでシリアライゼーションがラベルを表示し、またデバイスのプログラミング中とプログラミングのリピートを情報プログレス・ウィンドウで見ることが出来ます。

使用されたファイルで追加の操作
このグループ・ボックスには操作の3つのタイプが含まれています。ユーザーは"Playlist From-file mode"で使用されたシリアライゼーション・データ・ファイルの操作の1つを選択することが出来ます。次の操作が利用可能:
- **option Do nothing**
  プログラムは使用されたシリアライゼーション・データ・ファイルでいずれの操作も行いません。
- **option Move used file to specified directory**
  プログラムは使用されたシリアライゼーション・データ・ファイルをユーザー指定の使用されたシリアライゼーション・ファイルのディレクトリーに移動します。
- **option Delete used file**
  プログラムは使用されたシリアライゼーション・データ・ファイルを削除します。

ディレクトリー
このオプションは"playlist From-file"シリアライゼーション・モードでオプション"Move used file to specified directory[指定されたディレクトリーに使用するファイルを移動]"が選択されますと利用出来ます。ユーザーがどのシリアライゼーション・データ・ファイルに移動するかのターゲット・ディレクトリーを指定することが出来ます。

次のエラー表示がPlaylist From-fileシリアライゼーションで使用されます:
- s/n error #3 シリアライゼーション・データ・ファイルは存在しません。
- s/n error #34 使用されたシリアライゼーション・データ・ファイルを削除出来ません(シリアライゼーション・ファイルは書き込みプロテクト・ディスクに置かれているかも知れません)
- s/n error #35 使用されたシリアライゼーション・データ・ファイルを使用されたシリアライゼーション・ファイルのターゲット・ディレクトリーへ移動出来ません(シリアライゼーション・ファイルは書き込みプロテクト・ディスクに置かれているか、又は、ターゲット・ディレクトリーが存在しないかも知れません)

Device / Device options / Serialization / Custom generator mode[デバイス/ デバイス・オプション/ シリアライゼーション/ カスタム・ジェネレーター・モード]

ユーザーが自分でシリアライゼーション・システムを全て持つ場合は、カスタム・ジェネレータ・シリアライゼーション・モードが最もフレキシブルなシリアライゼーション・モードを提供します。

シリアライゼーションのCustom generator mode[カスタム・ジェネレータ]モードが選択された時、PG4UW、又は、PG4UWMCで各デバイスがプログラムされる前にユーザーが作成したプログラムによって"on-the-fly[オンザフライ]"でシリアル番号が生成されます。カスタム・ジェネレーター・モードのシリアライゼーションはユーザーが望むユニークなシリアル番号のシーケンスを生成することが出来ます。シリアル番号はリニア・シーケンス、又は、完全な非リニア・シーケンスとしてインクリメントすることが出来ます。ユーザー作成シリアル番号ジェネレーター・プログラムの詳細は以下の**Custom generator program**セクションで説明します。

サンプル:

利用出来るサンプル .exe と C/C++ ソース・ファイルが有ります。ファイルは次の様にPG4UWインストレーション・ディレクトリーExamples\ subdirectoryに有ります:

<PG4UW_inst_dir>\Examples\Serialization\custom generator_example\

一般的なパスはこのように見えます:

C:\Program files\Elnec_sw\Programmer\Examples\Serialization \customgenerator_example\

PG4UWコントロール・ソフトウエアの**Custom generator serialization[カスタム・ジェネレーター・シリアライゼーション]** のための次のオプションがあります: ダイアログ "Serialization" の**Mode**パネル・オプションでCustom generator modeを選択。次のオプションが表示されます:

Serialization Data File[シリアライゼーション・データ・ファイル]

現在のシリアル番号が含まれるデータ・ファイルのパスと名前を指定します。デバイスをプログラムする時、PG4UWソフトウェアはユーザーが作成したデータ・ファイルを更新するシリアル番号ジェネレーターを呼び出します。データ・ファイルの推奨される拡張子は.datです。 弊社のカスタマーの多くがBP Microsystems社のプログラマーも使用しているため、ユーザーは普通同じシリアライゼーション・ソフトウェアを使用することを希望するためです。

従って、シリアライゼーション・データ・ファイルはBPマイクロ社のソフトウェアで利用できる"Complex serialization"の.datファイルと互換性が有ります。

ノート: データ・ファイルは全て定期的にデバイスのプログラミング中にシリアライゼーションで上書きされます。希望する .datファイルの正しい名前を確実に入力して下さい。例えば: "c:\serial_files\serial.dat"です。

Serialization generator[シリアライゼーション・ジェネレーター]
シリアライゼーション・データ・ファイルが生成される実行ファイルのためのパスと名前を指定します。.

最初のシリアル番号
このオプションはカスタム・ジェネレーター・シリアライゼーション・プログラムに渡される最初のシリアル番号を指定する必要があります。番号は入力されますと16進形式で表示されます。

最後のシリアル番号
このオプションは許可されたシリアル番号の最大値を指定します。値がゼロでない場合に、シリアライゼーション・ジェネレーター・プログラムへ渡されます。ジェネレーターは最後のシリアル番号の値を、テストし、そして、現在のシリアル番号が最後のシリアル番号より大きい場合には、そのシリアライゼーション.datファイルに適切なエラー内容を持ったシリアル .datファイルを生成します。最後のシリアル番号の値がゼロの場合、その値はジェネレーター・プログラムに渡されません。

チェック・ボックス **Call generator with -RESULT parameter after device operation completed [デバイスの操作が完了した後に-RESULTパラメータと共に、ジェネレータをコール]**

この新しいオプションは特別な目的を持っています。特殊なパラメータ -RESULTでカスタム・ジェネレータを呼び出す要件がある場合は、チェックボックスにチェックを入れておく必要があります。それ以外の場合は、チェックを外して、オフにしておく必要があります。(デフォルトの状態はオフです) チェックした場合、各デバイス操作が完了した後、カスタム・ジェネレーターはデバイス操作の結果がOK、又は、エラーに関係なくPG4UW制御プログラムによって呼び出されます。ジェネレーターのパラメータはPG4UWシリアライゼーション・エンジンによって作成されます。
2つのパラメータが使用されます:

-RESULT[*n*]=TRUE | FALSE

n はマルチプログラミングが使用されている場合のオプショナルなプログラマー・サイトの番号。TRUEはデバイス操作がOKで終了したことを意味します。FALSEはデバイス操作がエラーで終了したことを意味します。

-N<serial number>

シリアライゼーション・ジェネレーターの通常の呼び出しと同様で、現在のシリアル番号を指定。

カスタム・ジェネレータ・プログラム
カスタム・ジェネレーター・プログラム、又は、シリアライゼーション・ジェネレータはシリアル番号のユニークなシーケンスを発生し、そのシリアル・データをシリアライゼーション.datファイルに書き込むプログラムです。このプログラムはユーザー側で作成します。シリアライゼーション・プログラムのパスと名前は、カスタム・ジェネレーター・モード・オプションのシリアライゼーション・オプションのダイアログで指定する必要があります。

プログラムは新しいシリアル・データが生成される度にPG4UWから呼び出されます。これは通常各デバイスのプログラミング操作の前に行われます。PG4UWコントロール・プログラムはシリアライゼーション・プログラムにコマンド・ライン・パラメータを渡し、そして、シリアライゼーション・プログラムはPG4UWコントロール・プログラムによって読み込まれるシリアライゼーション.datファイルを生成します。

以下のコマンド・ライン・パラメータが使用されます:

-N<serial number> 現在のシリアル番号を指定
-E<serial number> 最終(又は、最後)のシリアル番号を指定。
パラメータは、PG4UWソフトウエアのダイアログ "シリアライゼーション"において、最後のシリアル番号の値がゼロで無い時のみ渡されます。シリアライゼーション・プログラムは、もし現在のシリアル番号が最後のシリアル番号よりも大きい場合、シリアライゼーション.datファイルにエラーレコードT06を返します。詳細については"シリアライゼーション.datファイル形式"のセクションを見て下さい。

シリアライゼーション .dat ファイル・フォーマット
シリアライゼーション・ジェネレータによって生成されたシリアライゼーション.datファイルは、次のテキスト形式でなければいけません。シリアライゼーション.datファイルはレコードとシリアル・データ・セクションで構成されています。

レコードは以下に説明する様にTxxプリフィックスの1つで始まる行です。"xx"の値はレコードタイプのコードを表します。レコードはPG4UWソフトウェアにシリアライゼーションの状態(現在と最後のシリアル番号、シリアライゼーション・データとデータフォーマット、エラー等)を知らせるために使用されます。 必要なレコードはレコードT01, T02, T03とT04です。その他のレコードはオプションです。

T01:<serial number> コマンド・ライン・パラメーター -N<serial number> によってジェネレーターに渡す現在のシリアル番号が含まれています。
T02:<serial number> PG4UWが次のシリアライゼーションで使用する次のシリアル番号値を含んでいます。この値はシリアライゼーション・ジェネレーターで生成し、PG4UWに現在のシリアル番号に続くシリアル番号を知らせます。
T03:<data format code> シリアライゼーション・・データ形式を指定。
次のフォーマットがサポートされています:
T03:50 又は、T03:55 ASCIIスペース・データ形式
T03:99 - Intel Hexデータ形式
T04: シリアライゼーション・データが次の行からファイルの最後に続くことを示します。シリアライゼーション・データは例えばIntel Hex, ASCII Space等々の標準のASCIIデータ・ファイル形式の1つで保存されます。データに使用するフォーマットはレコードT03で指定する必要があります。

サンプル: 典型的なシリアライゼーション・データ・ファイル:

T01:000005

T02:001006
T03:99
T04:
:0300000000096B89
:03000300000005F5
:02000C005A0197
:01003F004F71
:00000001FF

ファイルは以下の情報で構成されています:
line T01 - 現在のシリアル番号 000005h
line T02 - 最終(最後)のシリアル番号 001006h
line T03 - 行 T04の後のシリアライゼーション・データ・フォーマットはIntel Hexです。
line T04 -デバイス・プログラミングの前にPG4UWのバッファにロードされるシリアライゼーション・データ、データはインテルHEXフォーマットで表現されます。

オプショナル・レコードは:

T05:<message> ワーニング、又は、エラー・メッセージ。このレコードはシリアライゼーションが中止されたために起こり、そして、PG4UWソフトウエアでワーニング、又は、エラー・メッセージが表示されます。

T06: 現在のシリアル番号が制限より大きい。このレコードはシリアライゼーションを停止し、PG4UWソフトウェアにより警告、又は、エラー・メッセージが表示される原因となります。シリアライゼーションをオフにする理由は現在のシリアル番号が許可された最大値の最終シリアル番号より大きいためです。このレコードはEコマンド・ライン・パラメータが指定されている場合に使用することが出来、それはシリアライゼーション・ダイアログでシリアル値の指定がゼロでないことを意味します。

T11:<message> 余り重要でないワーニング、又は、メッセージ。 シリアライゼーションは停止されません。

カスタム・ジェネレーター・シリアライゼーションを含んだデバイス・プログラミングのフローチャート

カスタム・ジェネレータのシリアライゼーションが使用される場合、各デバイスのプログラミングが開始される前に、シリアライゼーション・エンジンがシリアル.datファイルを生成するために実行可能なシリアライゼーションを呼び出します。PG4UWのシリアライゼーション・エンジンはシリアライゼーション・ジェネレーターを呼び出すために適切なコマンド・ライン・パラメータを管理します。.datファイルからのデータは直ちに内部プログラマー・バッファに読み出され、そして、プログラミング・デバイス用のデータとして使用されます。また、次のシリアル番号情報(レコード T02)はPG4UWに記憶されています。

デバイス・プログラミングの典型的なフローチャートは次の通りです:

1. プログラミング・バッチを開始
2. デバイス装着テスト
3. シリアライゼーションのシーケンスは4つのステップからなります:

•シリアライゼーション .datファイルを発生させるために適切なコマンド・ライン・

パラメーターでシリアライゼーション・ジェネレーターを呼び出す
•利用可能なシリアライゼーション.datファイルを待つ
•シリアライゼーション .datファイルのデータをプログラマー・バッファへ読み込む
(データはプログラミング・デバイスのために使用)
•データを読み込んだ後シリアライゼーション.datファイルを削除
4. デバイス・プログラミング
5. デバイス・ベリフィケーション
6. 操作結果のチェック.
これは全てPG4UWコントロール・プログラムによって管理されます。シリアライゼーション・ジェネレーターの操作結果はどの操作とも関係がありません。コントロール・プログラムは要求されたコマンド・ライン・パラメータでシリアライゼーション・ジェネレーターを呼び出します。

OK - PG4UWは次のシリアル番号の要求をします。次のシリアル番号はステップ3で .datファイルから読み込まれています。
シリアライゼーション・ジェネレーターの呼び出しにより、コマンド・ラインで指定された次のシリアル番号を持ちます。

ERROR - PG4UWは新しいシリアル番号の要求をしません。最新のシリアル番号は次のデバイスで使用されます。
次のシリアライゼーション・ジェネレーターの呼び出しはコマンド・ラインで指定された最新のシリアル番号を持ちます。

7. 次のデバイスへのプログラミングを繰り返しますか?
Yes ステップ2へ行く
No ステップ8を継続
8. プログラミング・バッチの終了

ノート:
エラー・プログラミングの場合、最新のシリアル番号が使用されますが、ジェネレーターはステップ3で呼び出されます。とにかくもし同じ番号が以前にプログラムされたデバイス用として用いた場合であっても、呼び出されます。もし、シリアライゼーション .datファイルのエラーが検出された場合、プログラムPG4UWはシリアライゼーション・エラーを報告し、即プログラミングのバッチ処理を中止します。

Device / Device options / Statistics[デバイス/デバイス・オプション/スタティスティクス(統計)]
スタティクスは選択されたタイプのデバイスで処理されるデバイス操作の実際のカウントについての情報を提供します。もし、1つのデバイスが1つの操作に対応している場合、すなわち、プログラミング、デバイス操作の数がプログラムされるデバイスと同じ場合です。

スタティクスの次の機能は**Count down[カウント・ダウン]**です。カウント・ダウンはデバイス操作の数、そして、デバイス操作がおこなうべきデバイスの数をチェックします。それぞれの成功したデバイス操作の後にカウント・ダウンのカウンターは反対に減少します。カウント・ダウンはユーザーが定義したデバイスの開始番号を持っています。カウント・ダウン値がゼロに達しますと、指定したデバイスの数が完了し、そして、カウント・ダウンの完了についてのユーザー・メッセージが表示されます。

Statistics[スタティスティクス] ダイアログは下記のオプションを含んでいます。:

チェック・ボックス - **Program[プログラム]**, **Verify[ベリファイ]**, **Blank[ブランク]**, **Erase[イレース]** と **Read[リード]** はスタティクス値がインクレメントされた後でオプションを定義します。

如何なる選択され実行されたデバイス操作も**Total** カウンターをインクリメントし、そして、デバイス操作の結果(成功または失敗)に応じて**Success[成功]**又は、**Failure[失敗]**になります。
部分操作の組み合わせも1つの操作としてカウントされます。例えば、Readの後のVerifyを含むRead操作は1つの操作です。Eraseと/又は、Verify操作を含むプログラム操作も1つの操作としてカウントされます。

チェック・ボックス- **Count down[カウント・ダウン]** はカウント・ダウンの有効、又は、無効を設定します。カウント・ダウンに続くエディット・ボックスはカウント・ダウンが開始されるカウンターの最初の番号を定義します。

Statistics[スタティクス] ダイアログは Statistics パネルで右マウス・ボタンを押して、そして、表示されている項目 Statics はクリックすることで開くことができます。実際のスタティクス値は Statistics[スタティクス]パネルのコントロール・プログラムのメイン・ウィンドウに表示されます。
スタティクス・ダイアログは 7 つの値が含まれます – **Success[成功]**, **Operational Failure[操作失敗]**, **Adapter test failure[アダプター・テスト失敗]**, **ID check failure[ID チェック失敗]** と他の **Failure[失敗]**(prog. SW, HW)と **Total[合計]**

値の意味は:
Success 成功して完了した操作の数
Operational failure デバイス・エラーで失敗した操作の数
Adapter test failure アダプターによる失敗した操作の数
Insertion test failure アダプターの誤った位置により失敗した操作の数
ID check failure デバイスからの ID コードの読み出しで失敗した操作の数
Other failure(prog. SW, HW) ハードウエア・エラー、又は、制御ソフトウエアのエラーにより失敗した操作の数
Total 全操作数
実際の Statistics[統計]値はメイン・ウィンドウの **Statistics**[統計]パネルに表示されます。

Statistics[スタティクス] パネルは 4 つの統計値を含みます - **Success[成功]**, **Operational Failure[操作失敗]**, **Other Failure[他の失敗]Total[合計]**

値の意味は:
Success 成功して完了した操作の数
Operational failure デバイス・エラーで失敗した操作の数
Other failure デバイス・エラー以外の理由で失敗した操作の数
Total 全操作数
Count down カウント・ダウン(有効、又は、無効)の情報
Remains デバイス操作の残り数の情報

ノート: 新しいデバイス・タイプが選択されたとき、すべてのスタティクス値はゼロにセットされ、そして、**Count down[カウント・ダウン]** は **Disabled[ディスエーブル ド]** にセットされています。**Statistics** パネルの **Reset[リセット]** ボタンはスタティクス値をリセットします。**Statistics** パネルの **Reload Count down[カウント・ダウンの再ロード]** ボタンは**カウント・ダウン**に初期値を再ロードします。

PG4UW ソフトウェアを使用する場合は、PG4UW を閉じるときに統計情報がログ・ウィンドウに保存されます。
PG4UWMC ソフトウェアのマルチ・プログラミングの場合、統計情報はジョブ・サマリ・レポートに保存されます。

Device / Device options / Associated file[デバイス/デバイス・オプション/アソシエーテイド・ファイル(関連ファイル)]
このコマンドはターゲット・デバイスの関連ファイルを設定するために使用されます。これはデフォルト・デバイス選択リスト又は、コントロール・プログラムをスタートした後にバッファに自動的にロードすることが出来るファイルです。

ユーザーはファイル名ボックスで関連ファイル名を編集することが出来ます。パス名をフルに付けて下さい。コントロール・プログラムはディスクのこのファイルの存在をチェックします。また、このファイルの自動ロードをイネーブル又は、ディスエーブルも変更出来ます。

File / Exit and save[ファイル/終了と保存] コマンドで両方、すなわち、関連ファイルと自動ロードのイネーブルをディスクにセーブ出来ます。

Device/Device options/Special options[デバイス/デバイス・オプション/スペシャル・オプション]
使用する全ての用語の説明についてはプログラムしたいチップのドキュメントをお読みください。このメニュー項目の名前が"View/Edit ..."で始まる場合、デバイスの読み込みコマンドはチップ構成の内容を読み込みこのメニュー・コマンドで表示および編集できます。

Device / Blank check[デバイス/ブランク・チェック]
このコマンドは、もし、可能な場合は、全てのデバイス又は、そのパーツのブランク・チェックを行ないます。コントロール・プログラムは INFO[情報]ウィンドウと LOG に警告のメッセージを書くことにより、このアクションの結果を報告します。

メニュー・コマンド Device / Device options / Operation options [デバイス/デバイス・オプション/操作オプション] で利用できる操作オプションをカスタマイズすることが出来ます。

Device / Read[デバイス/リード]
このコマンドはバッファに全てのデバイス又は、その一部分を読み込むことが出来ます。リードはチップのコンフィギュレーション(もし、有って、読み取り可能な場合)の内容も読み取ります。スペシャル・デバイス・コンフィギュレーション領域はメニュ

－View/Edit bufferとメニュ－Device / Device options / Special options[デバイス/デバイス・オプション/特別オプション](Alt+S)で利用できるダイアログで見たり編集することが出来ます。

コントロール・プログラムはINFO[情報]ウィンドウとLOGにメッセージを書くことによりこのアクションの終了を報告します。

読み込み処理が完了したらバッファ同期処理が開始されます。読み取ったデータはプログラマの内蔵 SSD ディスクから PC に転送されます。データ転送の進捗状況は別のウィンドウに表示されます。<Esc>キーを押すか、又は、ウィンドウの終了ボタンを押すと転送をキャンセルできます。

メニュー・コマンド Device / Device options / Operation options [デバイス/デバイス・オプション/操作オプション] で標準としてその他のワーキング・エリアを設定することが出来ます。このメニュー・コマンドで、オプション **Verify data after reading[読み出し後にデータをベリファイ]** を設定することは、デバイスの読み出しにより高い信頼性を持たせることを意味します。

Device / Verify[デバイス/ベリファイ]
このコマンドはバッファにあるデータと全てのデバイス又は、その一部分のプログラムされたデータを比較照合します。コントロール・プログラムはINFO[情報]ウィンドウとLogウィンドウにエラー・メッセージを書くことにより、このアクションの結果を報告します。

メニュー・コマンド Device / Device options / Operation options [デバイス/デバイス・オプション/操作オプション] で利用できる操作オプションをカスタマイズすることが出来ます。
タブ Error は PG4UW メニュー・コマンド Device / Device options / Operation options [デバイス/デバイス・オプション/操作オプション] で標準としてその他のワーキング・エリアを設定することが出来ます。

ノート:
- *ベリファイ操作はソフトウエアでデータを持ってチップ全体の内容を比較しますので、従って、不完全にプログラムされたチップの場合 - プログラミングの後のベリフィケーションではエラーは無くても、ソロ(単体)ベリファイ操作ではパスしないかも知れません。*
- *ベリファイ操作はデータのアクティブなリード・プロテクションを持ったプロテクトされたデバイスの場合もエラーを報告することがあります。*
- *一般に、"デバイス全体"はデバイス/デバイス・オプション/操作オプションのダイアログで設定されたデバイス開始アドレスとデバイス終了アドレスの間のデバイス範囲を意味します。全てのデバイスがStart-Endデバイス・アドレスをカスタマイズできるわけではありません。一部のデバイス(例えば、NAND FLASH)はカスタマイズ可能なセクタまたはページ数/範囲があり、"デバイス全体"はこれらのオプションで指定されたデバイスの範囲を意味します。*

Device / Program[デバイス/プログラム]
このコマンドはバッファにあるデータを全てのデバイス又は、その一部分にプログラムすることが出来ます。コントロール・プログラムは INFO[情報]ウィンドウと LOG にエラー・メッセージを書くことにより、このアクションの結果を報告します。

メニュー・コマンド Device / Device options / Operation options[デバイス/デバイス・オプション/操作オプション] でプログラムされる領域をカスタマイズすることができ、そして、その他の操作オプションを設定することが出来ます。

Device / Erase[デバイス/イレース]
このコマンドはすべてのプログラマブル・デバイスを消去することが出来ます。プログラムはエラーなしで終了、又は、エラーで終了したかを INFO[情報]ウィンドウと Log ウィンドウにエラー・メッセージを書くことによりこのアクションの結果を報告します。

メニュー・コマンド Device / Device options / Operation options [デバイス/デバイス・オプション/操作オプション] で利用できる操作オプションをカスタマイズすることが出来ます。

イレース後、もし、デバイス(チップ)がイレース・ベリファイ・コマンドをサポートしていない場合、ブランク・チェック操作がイレース操作のベリファイ成功を代行します。

Device / Test[デバイス/テスト]
このコマンドはサポート・デバイスのリストから選択されたプログラマー(このテストをサポートしている)上のデバイス(すなわち、スタティクス RAM)のテストを実行します。

- sRAM は 3 つのベーシック・ステップでおこなわれます:

- データドライバ[デバイス出力ピン]機能のテスト

ドライバ・テスト ... D0..D7 のテストは CE\, OE\と WE\の信号反応を示します:
- 最初のサイクルでアドレス 0x0(CE/=L WE/=L OE/=H) にデータ 0x55 を書き込み、同じアドレス(CE/=L WE/=H OE/=L)から読みだされたデータと比較します。データは有効でなければいけません。
- そして他の組み合わせ制御ピン(CE/=L WE/=H OE/=H), (CE/=H WE/=H OE/=L), ..., がセットされ、データが無効であることもチェックします - データバスドライバは非アクティブです。

- sRAM テスト, ベーシック・パーツ

プログラマはランダムなデータを sRAM デバイスに書き込んだ後、その内容を確認します。

- RAM テスト, アドバンスト(オプショナル)

"Walking one"(1 を書き込む) と"Walking zero"(0 を書き込む) テストを行うことが出来ます。
http://www.google.com/search?q=memory+test+walking+one

http://www.google.com/search?q=memory+test+walking+zero

ノート:
- ビットのリークを検出する目的で書き込み動作とプログラムされたデータのベリファイ後との間の遅延を(デバイスが供給されている状態で)選択することが可能です。
- プログラマは信号ピン上の電流が大きすぎるか、又は、アナログ・エラーを検出することはできません。
- 全てのテストは低周波数(テストされるデバイスの最高速度と比較される)で行われるため、このようなテストの使用は制限されています。

結論:
- デバイス・プログラマは sRAM の健全性に関する基本的な回答のみを提供できます。
- sRAM をより深くテストする必要がある場合は、特化された sRAM テスタを使用してください。

Device / IC Test[デバイス/テスト]
このコマンドは IC に対するテスト・セクションをアクティベートします。主に標準のロジック IC。IC はグループ/ライブラリーへのテクノロジーのタイプによってソーティングされています。

最初に適切なライブラリ、希望のデバイス、テスト・ベクトルの実行モード(LOOP、SINGLE STEP)を選択します。制御シーケンスとテスト結果はプログラマのアクティビティ・ログに表示されます。必要な場合には、テスト・ベクトルをユーザーが直接定義することが可能です。テスト・ベクトルの作成の構文と方法の詳細な説明はプログラム・インストール・フォルダにある example_e.lib ファイルに記述されています。

ノート:
IC のテストはある程度(かなり低い)速度でテスト・ベクトルを使用して行われます。テスト・ベクトルによるテストではチップの全ての欠陥を検出することはできません。言い換えれば、IC テストが "FAIL"と報告した場合、デバイスに欠陥があります。しかし、"PASS" が報告された場合は、チップがテストに合格したことを意味しますが、テストされた IC の他の主にダイナミックなパラメータをチェックするテストに合格しない可能性があります。
プログラマの立ち上がり/立ち下がりエッジはチップのプログラミングに合わせて調整されるためチップに欠陥がない(例えばカウンタなど)にもかかわらず一部のチップのテストが失敗することがあります。

Device / Jam/VME/SVF/STAPL/mDOC ... Player
Jam STAPL は AlteraR 社により開発され、そして、プログラマブル・ロジック・デバイス(PLD)製造業社, プログラミング装置メーカーとテスト装置製造業者のコンソーシアムによりサポートされています。

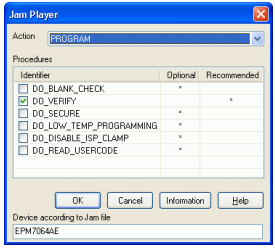
Jam™ 標準テストとプログラミング言語(STAPL), JEDEC standard JESD-71 は ISP(イン-システム・プログラミング)の目的のための標準ファイル・フォーメット です。Jam STAPL はフリー・ライセンスのオープン・スタンダードです。それは IEEE 1149.1 Joint Test Action Group (JTAG)インターフェースを使用した プログラミング・デバイスと電気回路システムのテストのプログラミング、又は、コンフィ ギュレーションをサポートしています。デバイスはプログラム、又は、ベリファイ出来ま すが、Jam STAPL はデバイスのリードのような他の機能は一般的には許可され ていません。

Jam STAPL プログラミング・ソリューションは 2 つのコンポーネントから構成されています: Jam Composerと Jam Player.
Jam Composer はプログラムは、一般的にデバイスにデザインをプログラムするのに必要なユーザー・データとプログラミング・アルゴリズムを含む Jam file (.jam)を生成し、プログラマブル・ロジック・ベンダーにより書かれています。
Jam Player は Jam ファイルを読み取り、プログラミングや JTAG チェーンでデバイスのテストのためのベクターを適用するプログラムです。さらに詳しい情報はウェブサイト: http://www.altera.com でアプリケーション・ノートを見て下さい:
"AN 425: Altera デバイスをプログラムするために Jam Playerを使用",
"AN 100: イン-システム・プログラマビリティ・ガイドライン",
"AN 122: 組み込みプロセッサー経由でISP & ICRのためにJam STAPLを使用" と詳細のための関連アプリケーション・ノート:.

ソフトウエア・ツール:
Altera: MAX+plus II, Quartus II, SVF2Jam utility (シリアル・ベクター・ファイルをJam fileに変換), LAT2Jam utility (ispLSI3256A JEDECファイルをJamファイルに変換);
Xilinx: Xilinx ISE Webpack 又は、Foundation software(ユティリティ SVF2Jamで使用するためにSTAPLファイル、又は、SVFファイルを生成);;
Actel: Actel Libero® Integrated Design Environment (IDE) (STAPLEファイルと/又は、PDBファイルを生成), Actel FlashPro (PDBファイルをSTAPLEファイルに変換)..

JAM player dialog[JAM playerダイアログ]

Jam Player version 1 (ActionとVariables controlsをご覧ください)



Jam Player version 2 (Actionと Procedures controls をご覧ください)

Action[動作]
実行したい動作を選択して下さい。
version 2 の Jam file は動作で構成されています。動作は実行させる手順の呼び出しを含んでいます。
version 1 の Jam file はステートメント'action'と'procedure'を知りません, 従って、Action の選択はアクセス可能ではありません。プログラム・フローはプリフィックス DO_something を持ったブーリアン変数に応じた命令を実行するために開始します。もし、プリフィックス DO_something を持った新しいブーリアン変数が必要な場合はご連絡下さい。

Procedures[手順]
プログラム・フローは各手順からのステートメントを実行します。手順はオプションと推奨をご使用下さい。推奨手順は暗黙的にマークされています。ニーズに応じて有効、又は、無効手続きにすることができます。Jam Playerはマークされた手順のみを実行します。その他の手順は無視されます。手順の数はJamファイルに依存して異なります。

Variables[変数]
version 1のJam fileはステートメント'action'と'procedure'を知りません。プログラム・フローはプリフィックス DO_somethingを持ったブーリアン変数に応じた命令を実行するために開始します。Jam Playerはアルゴリズム内の全てのマークされたDO_somethingを実行します。変数(手順)は一定であり、それはJamファイルに依存しません。もし、プリフィックス DO_somethingを持った新しいブーリアン変数が必要な場合はご連絡下さい。

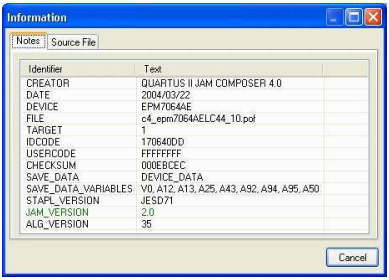
OK
マークされた適切な手順で選択されたアクションを受付けます。

Information
Jamファイルについての情報を表示。ダイアログでノートとソース・ファイルをプレビューすることが出来ます。

Device according to Jam file

Fァイルは特定のデバイスのために作られています。デバイス名は Jam ファイルの NOTE identifier DEVICE に含まれています。デバイス名はダイアログ Select device[デバイス選択]で選択したデバイスの名前と同じでなければいけません。デバイスが異なる場合、ソフトウェアがJam Playerの開始中の警告メッセージによってこの状況を示します。

JAM file information dialog[JAM file情報ダイアログ]



ノート: ステートメントはJamファイルに付いての情報をストアするために使用されます。 NOTEフィールドにストアされたこの情報は関連した如何なるタイプのドキュメントと特定のJamプログラムに関連した属性を含んでいます。

ソース・ファイルは Jam 言語でのプログラムを含んでいます。Jam プログラムはステートメントのシーケンスで構成されています。Jam のステートメントはオプション、命令と引数であるラベルを含みセミコロン(;)で終了します。引数はリテラル定数、変数、又は、目的のデータタイプ(即ち、ブーリアン、又は、整数)での結果となる式になります。各ステートメントは通常 Jam プログラムが 1 行を占有しますが、これは必須ではありません。改行は終了コメントを除いて Jam 言語の構文に重要ではありません。アポストロフィ分(')はインタプリタで無視されることを除いてはコメントを表わすために使用することが出来ます。言語は行の長さ、ステートメントの長さ、又は、プログラム・サイズのための制限を指定していません。より詳しい情報はウェブサイト上で見つけることが出来ます:http://www.altera.com

拡張子 .jbcの付いたJamファイルはエディターで表示出来ないJam STAPL Byteコード形式です。

XILINX デバイスのためのJED ファイルからJam STAPL ファイルへの変換

· フリー・ダウンロード Xilinx Integrated Software Environment(ISE) 6.3i ソフトウエアのインストール: WebPACK_63_fcfull_i.exe + 6_3_02i_pc.exe (315MB 又は、概ね)
· Xilinx ISE 6/Accessories/iMPACT を起動

· ダイアログ “Operation Mod Selection: What do you want to do first?” 選択:“Prepare Configuration Files,
· ダイアログ “Prepare Configuration Files: I want create a:” 選択: “Boundary-Scan File”,
· ダイアログ “Prepare Boundary-Scan File: I want create a:” 選択: “STAPL File”,
· ダイアログ “Create a New STAPL File”拡張子 .stapl にした Jam ファイル名を入力,
· ダイアログ “Add Device”拡張子.jed の JED ファイルを選択,
· JTAG チェーンで作成されたデバイスを選択 例えば: XC2C32A そして、シーケンス操作(即ち: Erase, Blank, Program, Verify; right mouse button),
· メニューの選択 項目で“Output/Stapl file/Stop writing to Stapl file”を選択
· PG4UW を実行,デバイスを選択、即ち: Xilinx XC2x32A [QFG32](Jam), Jam ファイルをロード(Files of type: select STAPL File)
· “Device operation options Alt+O”を選択、“Jam configuration”ボタンを押します。警告 メニューからのデバイス・選択"Select Devices“とJamファイルは恐らく違います! 続けますか?” はいを選択 (Xilinxソフトウエアは行: NOTE "DEVICE" "XC2x32A";をJamファイルに含んでいません). ダイアログ“Jam player”でアクションと手順を選択、ダイアログと終了、ツール・バーから“Play Jam”をクリック、そして、Logウィンドウを読みます。

STAPL ファイルを使用したACTEL デバイスのプログラミングについての情報

PG4UW プログラムでの Actel 社製 flash FPGA のプログラムは Actel Jam player を使用して実行されます。全ての一般的な操作アイコン(program, erase, verify...)ボタンは STAPL に置き換えられます。

Actel デバイスの操作(program, erase, verify...)は以下のいくつかのステップを含んでいます。

***.stp (STAPL ファイル)をロード**

メイン・ツールバー上の“Load”アイコンをクリックして適切なSTAPLEファイル (Actel design software LIBRO IDE)をロード。このSTAPLEファイルにはユーザー・データとデバイスに設計をプログラムするために必要なプログラミング・アルゴリズムを含んでいます。

動作を選択

STAPL ファイルのロード完了後、デバイス操作オプション(Alt+O ショートキー)/STAPL configuration... .(STAPL configuration ...)で意図する動作の操作を選択して下さい。デバイスをプログラムするにはアクション・リスト内の PROGRAM を選択して下さい。プログラミング・ファイルに関する全ての動作のリストに関する説明は<http://www.actel.com>の ACTEL FlashPro User`s Guide を参照して下さい。

アクションを実行

選択したアクションを実行させるためにSTAPLボタンをクリックして下さい。操作(即ち、プログラミング)が完了しますとLogウィンドウに“Exit Code = 0... Success”と表示されます。

ACTEL デバイスのためにPDB ファイルをJAM STAPLEに変換

Actel PDB ファイルは Actel プログラマー、即ち、FlashPro プログラマーのみでサポートされている専用ファイル・フォーマットです。PG4UW コントロール・プログラムは Actel デバイスを Jam STAPL ファイルでのみプログラムすることが出来ます。従って、PDB と STAPL 間のファイル変換が必要です。

ACTELデバイスのためにPDBファイルをJam STAPLファイルに変換:
· ソフトウエア・ツール FlashPro をインストール(Actel Libero tool suite のコンポーネント、又は、スタンド・アローン
・バージョンとして<http://www.actel.com>からダウンロード)
· FlashPro を起動
· New Project[新規プロジェクト]ボタンをクルック、又は、ファイル・メニューから New Project[新規プロジェクト]を選択、そして、Project nameフィールドにプロジェクト名を入力して下さい。希望するプログラミング・モードを選択 – single device[シング・デバイス]、又は、chain[チェーン]を選択しOKをクリックして下さいK
· コンフィギュレーション・メニューからLoad Programming File[プログラミング・ファイルのロード]を選択し、そして、変換するための一致する*.pdbファイルを選択して下さい。
· ファイル・メニューから、Export/Export Single Device STAPL File... を選択し、ファイル名を入力し、そして、指定したディレクトリーに STAPL ファイルをエクスポートするために Save[セーブ]ボタンをクリック
· PDBファイルからSTAPLへの変換が完了し、作成した*.stpファイルはActelデバイスのプログラミングために使用することが出来ます。

Actelについてよくある質問

Q: 既にプログラムされた Actel デバイスをどのように ID チェック/ベリファイを行うのですか?
A: これを行うための幾つかのオプションがあります。各オプション(action)は既にプログラムされた Actel デバイスをロードされた STAPL ファイルで互いに比較しベリファイする方法です。次に STP ファイルでの適切なアクションを記載します:
DEVICE_INFO: デバイスをリードし、ログ・ウィンドウに表示されているデバイスにプログラムされるプログラミング環境のチェックサムをご覧下さい。この値は手動で STAPL ファイル(情報ウィンドウでも見ることが出来ます)のヘッダーの値と比較することが出来ます。注意: プログラムされたデバイスのチェックサムの値は実際の(壊れているかも知れません)デバイス・データの内容からカウントされませんが、この値はプログラミング中にスペシャル・メモリ・ローカライゼーションに保存され、そして、それだけ読み取っています ! VERIFY_DEVICE_INFO: 前のオプションと類似していますが、違いはプログラムされたデバイスのチェックサムと STAPLE ファイルのチェックサムを自動で比較します。比較の結果はメッセージ・ウィンドウに success 又は、error の何れかで表示されます。VERIFY: STAPLE ファイルの内容とプログラムされたデバイス内容のデータの比較に関して、最も安全ですが、最も遅い(オプション 1と 2 の約 1 秒に比べデバイスの要領に依存しますが数 10 秒)オプションです。選択されたファミリー機能の比較(FPGA Array,

targeted FlashROM pages, security setting...)は bit ごとに実行されます。そして、もし、データのミスマッチが起こりエラー・メッセージがログ・ウィンドウに書かれますとベリフィケーション・プロセスは早期に終了することが出来ます。

Q: PG4UW で２つの異なった STAPLE ファイルを１度のプログラム操作で Actel デバイスにプログラムは可能ですか?
A: はい。可能です。PG4UW コントロール・プログラムは上記の様な状況に対して内蔵のマルチ-プロジェクト機能を持っています。例としてデータ・コンテンツ(最初の STAPLE ファイル)と一緒にセキュリティ暗号化キー(2 番目の STAPL ファイル)をプログラムすることが出来ます。

IspVM Virtual Machine

IspVM Virtual Machineはバウンダリー・スキャン・テストのためのIEEE 1149.1 Standardと互換性のあるプログラミング・デバイスのための仮想マシーンです。IspVM EMBEDDEDツールはバウンダリー・スキャン・プログラミングとテストのための工業標準シリアル・ベクター・フォーマット(SVF)言語と Lattice's IspVM Virtual Machine™のパワーが兼ね 備わっています。

ispVM システム･ソフトウェアは IEEE 1149.1 standard 規格と SVF 又は、IEEE 1532 フォーマットをサポートした ispJTAG と非ラティス JTAG ファイルの両方をサポートしている VME ファイルを生成します。VME ファイルは IspVM システム・ウィンドウからチェーン情報を得ることが出来る hex コード・ファイルです。

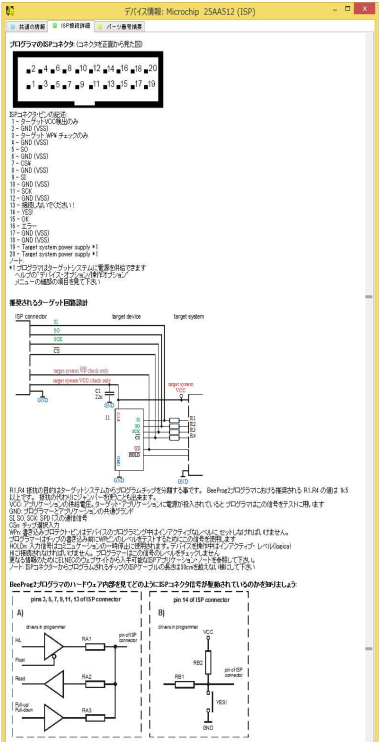
さらに詳しい情報はウェブサイト: http://www.latticesemi.comをご覧下さい。

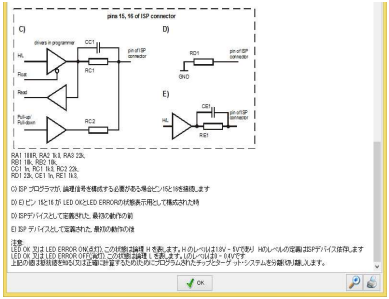
ソフトウエア・ツール:

Lattice: ispLEVER, IspVM System ISP Programming Software, PAC-Designer Software, svf2vme utility (シリアル・ベクター・ファイルをVME fileに変換) をご覧ください。

Device / Device info[デバイス/デバイス情報]

このコマンドは現在選択されているデバイス、デバイスのサイズ、構成織、プログラミング・アルゴリズムとデバイスをサポートするプログラマ(ソケット・モジュールを含む)の追加情報を提供します。ここでは現在のデバイスに関するパッケージ情報やその他の一般的な情報も確認できます。





<Ctrl+F1> キーでいつでも、どのメニューにいても、このメニューを呼び出すことが出来ます。

Programmer[プログラマー]

Programmer / Find programmer[プログラマ/プログラマ検出]
この項目は新しいタイプのプログラマと通信パラメータを選択します。このコマンドには次の項目が含まれています。:
Programmer*[プログラマ]* - 検出のための新しいタイプのプログラマを設定します。**Search all[すべて検索]**を選択すると制御プログラムはサポートされているすべてのプログラマを検出します。
Establish communication*[通信を行う]* - 新しいプログラマのために**手動**、又は、**自動**により通信を確立することができます。
Speed - 手動通信が選択された場合、どのPCがプログラマにデータを送信するか速度をセット。速度は最大速度からのパーセントで表されます。

Port - 要求されたプログラマのためにスキャンされるポートを選択。All port が選択されている場合、制御プログラムは標準のアドレスで使用可能な全てのポートをスキャンします。

キー**<Enter>**、又は、**OK**ボタンを押すとセットされたパラメータでプログラマのスキャンが開始されます。制御プログラムの開始時と同じアクティビティがあります。このコマンドは新しく選択されたプログラマがこのデバイスをサポートしている場合、現在のデバイスのないデフォルト・デバイスのリストをクリアします。

このセッティングはコマンド Options / Save options[オプション/セーブ・オプション]によりディスクにセーブされます。

Programmer / Refind programmer[プログラマ/プログラマ再検出]
このメニュー・コマンドは現在選択されているプログラマを再検出(通信を再確立する)ために使用されます。

他のタイプのプログラマを選択するには、プログラマの通信パラメータと新たに選択されたプログラマとの通信を確立するためにメニュー**Programmer/ Find programmer[プログラマ/プログラマ]**を使用します。

Programmer / Handler[プログラマ/ハンドラー]
ダイアログ **Handler[ハンドラー]**でハンドラーのタイプとハンドラーの通信パラメータを設定できます。ハンドラーは制御プログラム内のデバイス操作を特別に制御するための外部デバイスです。None Handlerが選択されている場合、これは制御プログラムのデフォルト状態を意味します、即ち、ハンドラーとの協調によりデバイス動作が自動的に制御される場合、デバイス動作はユーザーによって直接制御されます。
ダイアログ **Handler[ハンドラー]** は次の項目を含みます:
Selected Handler - 希望するハンドラー・タイプを選択
Search at port - 要求されるハンドラーをスキャンするCOMポートを選択

キー**<Enter>**、又は、**OK**ボタンを押すと設定されたパラメータによってハンドラーのスキャンが開始されます。選択されたHandler typeが**None**の場合、Handlerスキャンは処理されません。現在のハンドラの設定は、コマンドに Options/Save options[オプション/セーブ・オプション]、又は、制御プログラムが閉じられたときにコンフィギュレーション・ファイルに保存されます。ハンドラーは販売しておりません。

Programmer/Credit box info[プログラマ/クレジット・ボックス情報]
メニュー項目'Credit box info'[クレジットボックス情報]はPCに添付されているクレジットボックスに関する全ての必要な情報(タイプ、シリアル番号、有効化に関する情報、利用可能なクレジットに関する情報)を提供します。複数の'クレジットボックス'がPCに接続されている場合、ソフトウェアは利用可能な全てのクレジットに関する情報も表示します。

クレジット・ボックスの利用可能性と添付されているクレジット・ボックス内のクレジットの量に関する情報もメイン・ウィンドウに表示され、'統計情報'、及び、'カウントダウン'セクションの下にある'クレジットボックス'ボタンに表示されます。

このボタンは'Paid ISP support'[有料 ISP サポート]に属するデバイスが選択され、少なくとも1つのクレジット・ボックスが存在する場合に動的に表示されます。

'Credit box'[クレジットボックス]ボタンの下部にある棒グラフは添付されたクレジット・ボックスのクレジットのステータスを示します:
緑色バーグラフ - 85%のクレジットが利用可能
黄色バーグラフ - 50%のクレジットが利用可能
赤色バーグラフ - 10%のクレジットが利用可能
バーグラフなし - 0%は利用可能なクレジットがゼロ

利用可能なクレジットに関する情報は定期的に更新されます(開かれたクレジットボックス情報ウィンドウ中も)。

デバイスでの作業は'Paid ISP support'[有償 ISP サポート]カテゴリに属します。

お客様の要望やニーズに応じて、プログラマブル・デバイスはますます複雑になります。また、プログラマブル・デバイスの範囲はますます広くなっています。 結果として、私たちはデバイス・プログラマの製造元として、より多くの開発リソースを費やさなければなりません。新しい複雑なプログラマブル・デバイスのサポートの実装は過去と比較して難しく、実装する必要のあるデバイスの数も非常に多いためです。

オフライン・プログラミング(ZIF ソケット、プログラミング・アダプタを使用)の場合、新しいデバイス・サポートの実装に費やされる開発コストは、プログラミングアダプタの販売によってもカバーされます。しかし、ISP モードでプログラミングする場合、顧客はプログラマーのみを購入し、新しいバージョンのソフトウェアに含まれるアフター・プログラマー・サポート費用は何の影響も受けません。
従って、'Paid ISP support'[有償 ISP サポート]が必要となって来ております。

上記の様に ISP モードでプログラムされたデバイスについても、ソフトウェア・アップデートのダウンロードを無料に保つため、複雑なデバイスの ISP によるプログラミングには非常に僅かな料金が掛かるシステムを取っております。ISP サポートの実装には非常に時間が掛かります。

この'Paid ISP support'システムは簡単です。このようなデバイスを使用するにはデバイス・プログラマが接続されている PC にクレジット・ボックスを接続する必要があります。クレジット・ボックスは、技術的には USB ポートの小さなドングルで、モデルに応じて 25 から 500,000 のクレジットが含まれています。このような各デバイス・プログラミングのマイクロペイメントはクレジット・ボックス内のクレジット額の減少によって行われます。
クレジットの実際の手数料は非常に低く、クレジット・ボックスの CB-25k バージョン(=25,000 クレジット)に対して約 0.01 US ドルで開始し、クレジット・ボックスの CB-500k バージョンのクレジットでは約 0.003 US ドルまで急激に下がります(=500,000 クレジット)。クレジット・ボックスの有効性は regular バージョンのソフトウェアの 10 バージョンに限られます。つまり、クレジット・ボックスが最初の使用時にアクティブ化されると、それはその時のバージョンと次の 9 つの regular バージョンで使用することができます。
例 : 3.01,3.01 バージョンのソフトウェアで起動する場合、3.01,3.02 ... 3.10 バージョンまでのソフトウェア(全ての OnDemand バージョンを含む)で使用できます。

クレジット・ボックスの状態は、PG4UW、及び、PG4UWMC ソフトウェアのメイン・ウィンドウに表示されます。

詳細については、当社 Web サイトのアプリケーションノート - 'Paid ISP support'カテゴリにあるデバイスのクレジット・ボックスの使用方法 - をご覧ください。

Programmer / Automatic YES! [プログラマ Automatic YES!]
このコマンドは**Automatic YES!** モードの設定に使われます。このモードではプログラムされたデバイスを取り除いて新しいデバイスをZIF ソケットに装着しますと最後の操作が自動的にリピートされます。プログラムが自動的に新しいデバイスの装着を検知し、最後に行った操作をキー又は、ボタンを押すことなく実行します。ZIFへのデバイスの装着は画面に表示されます。リピート操作の実行はZIF

から/への装着/取り外しを待っている間に<Esc> キーを押すことでキャンセルされます。

デバイスで操作が実行された後、プログラマ上のOK又はERRORのステータスLEDが操作結果により点灯します。そして、BUSY LEDが点滅します。
プログラムがデバイスが取り除かれたことを検知しますと、ステータスLEDはオフになり、新しいデバイスが最後の操作を繰り返すためにプログラムが用意できていることを示すためにBUSY LEDが点滅します。プログラムがプログラマのZIFソケットにある新しいデバイスの1つ又は、それ以上のピンを示した後、BUSY LEDは連続して点灯します。ここでプログラムは新しいデバイスの残りのピンが挿入されるための要求された時間を待ちます。もし、要求された時間(デバイス挿入完了時間)が過ぎたり、デバイスが正しく挿入されない場合、プログラムはこのステータスをERROR LEDで示します。新しいデバイスが正しく挿入された後、プログラムはBUSY以外のステータスLEDをオフにします。そして、新しいデバイスで操作を開始します。このモードは**Automatic YES!** モードにより有効、又は、無効にすることが出来ます。もし、新しいプログラマが**Options/Find programmer[オプション・プログラマ検出]**で選択されますと、このモードは無効になります。

Response time[応答時間]はZIFソケットへのチップ装着と選択されたデバイス操作の開始の間隔となります。もし、ZIFソケットでのチップの長いポジショニングが必要な場合は**elongated** response time[延長した応答時間]を選択して下さい。
Programming adapter used[使用されるプログラミング・アダプタ]は現在選択されたデバイスで使用されるアダプタ名を示します。
Pins of ZIF excluded from sensing[感知から除外されたプログラマのピン]はAutomatic YES!によるテストで無視されたピンのリストです。ピンの無視の殆どの理由はそれらのピンへのコンデンサの接続になります。

ボタン**Setting Automatic YES! parameters[Automatic YES!パラメータのセッティング]** は完全に接続されたピン(コンデンサーがあるピン)を検出できるウィザードを実行し、これらのピンをセンシングから除外されたピンのリストに設定します。デバイスの選択の後, 除外されたピンのリストには選択されたデバイス・アダプタに対してデフォルトの除外されたピンを含んでいます。もし、ユーザーによりユニバーサル・プログラマーと/又は、デバイス・アダプタに他のバイパス・コンデンサが追加する場合は、デフォルト・パラメータを無視し優先するためとコンデンサーのあるその他のピンを検出するために**Automatic YES! parameters wizard**[パラメータ・ウィザード]を実行する必要があります。

Device removal hold off time[デバイス・リムーバル保持時間] はZIFソケットからデバイスを取り除いた時とソフトウエアが新しいデバイスの装着をソケットでチェックを開始する時の間の時間間隔です。この時間は秒間隔で1～120 (デフォルト値は2 秒)でなければいけません。
Device insertion complete time[デバイス装着完了時間] はプログラムが不正に挿入されたデバイスを検出しない様にするために最初のピン(複数)が検知された後に全てのピンが適切に挿入されなければならない時間をセットすることが可能です。この時間は秒間隔で1～120 (デフォルト値は2 秒)でなければい

けません。
Suspend on error[エラーで停止]はAutomatic YES!機能でエラーが起こった時に一時停止して操作の結果を見るか、又は、停止せずに続けるかを定義します。
このオプションは**Device/Select device[デバイス/デバイス選択]**で新しいデバイスが選択された後はデフォルトにセットされます。
このセッティングはコマンド**Options/Save options[オプション/オプション保存]**によりディスクに保存し、選択したデバイスをFile/Save project[ファイル/プロジェクトをセーブ]…でプロジェクト・ファイルにセーブすることが出来ます。

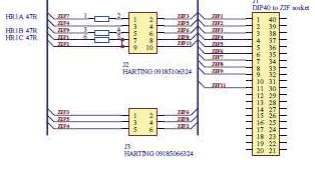
ノート: 供給電源をバイパスするためにコンデンサ等、一部のpassive部品、又は、active部品を備えたソケット・アダプターを使用している時、Automatic YES!機能はそれらのピンをコンデンサ・リストのピンにセットする必要があります。それはSetting Automatic YES! parameters[Automatic YES!パラメータのセッティング] ウィザードで行われます。これはAutomatic YES! 機能が正しく動作するために必要です。さもないとAutomatic YES! 機能はピンが未だ接続されていると考え、ユーザーが新しいデバイスを挿入して新しいプログラミングを開始できません。

Programmer / Selftest[プログラマ セルフテスト]
コマンドはプログラマの標準的な付属品に含まれる 48pin Diagnostic POD[診断 POD]を使用して、現在のプログラマのセルフテストを実行します。セルフテストを実行する頻度はメンテナンスのセクションをご覧下さい。

Programmer / Selftest ISP connector[プログラマ ISP セルフテスト・コネクター]
コマンドはISPコネクター用のDiagnostic POD[診断POD]を使用して現在のプログラマーのISPコネクターのセルフテストを実行します。

ISPコネクタの診断ポッドはプログラマの6と10ピンISPコネクタのテストに使用する必要があります。SmartProg2のみISPコネクタの診断ポッド(注文番号: 70-0208)はオプションです。

ISPコネクターのためのDiagnostic PODの回路図 (お急ぎの場合):



6ピンISPコネクターのテストのシーケンス:

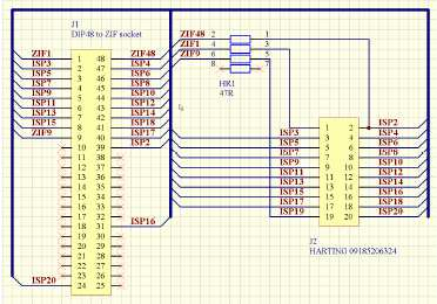
1. ISPコネクタの診断ポッドをプログラマのZIFソケットに挿入します。ISPコネクタの診断ポッドは40ピン・デバイスとして挿入する必要があります。
2. ISPケーブルでプログラマのISPコネクタにISPコネクタのための診断ポッドの6ピン・コネクタを相互接続します。ピンが正しく相互接続(即ち 1-1, 2-2, 6-6)されていることを確認してください。
3. PG4UW(Programmer/Selftest ISP connector…[プログラマ/ISPコネクタ・セルフテスト])でISPコネクタのセルフテストを実行して下さい。

10ピンISPコネクターのテストのシーケンス:
1. ISPコネクタの診断ポッドをプログラマのZIFソケットに挿入します。ISPコネクタの診断ポッドは40ピン・デバイスとして挿入する必要があります。
2. ISPケーブルでプログラマのISPコネクタにISPコネクタのための診断ポッドの6ピン・コネクタを相互接続します。ピンが正しく相互接続(即ち 1-1, 2-2, 10-10)されていることを確認してください。
3. PG4UW(Programmer/Selftest ISP connector…[プログラマ/ISPコネクタ・セルフテスト])でISPコネクタのセルフテストを実行して下さい。

ISPコネクター#2用のDiagnostic POD[診断POD]はプログラマーの20ピンISPコネクターをテストするために必要です。BeeHive208S, BeeHive204, BeeProg2, BeeProg2Cには**Diagnostic POD[診断POD]**は標準で付属しています。

ISPコネクター#2のためのDiagnostic PODの回路図 (お急ぎの場合):

20ピンISPコネクターのテストのシーケンス:



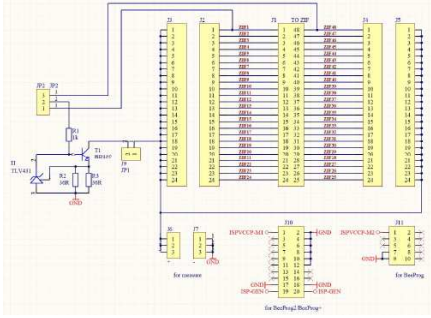
1. ISPコネクタ#2の診断ポッドをプログラマのZIFソケットに挿入します。ISPコネクタ#2の診断ポッドは48ピンデバイスとして挿入する必要があります。
2. プログラマーに標準付属の20ピンISPケーブルをプログラマーのISPコネクターに接続し、フラット・ケーブルの他方を48ピンZIFソケット上のISPコネクター#2のた

めのDiagnostic PODの20ピンの角形コネクターに接続します。ピンが相互に正しく接続されていることを確認して下さい(即ち、1-1, 2-2, ..., 20-20).
3. PG4UW (Programmer/Selftest ISP connector[プログラマー/セルフテストISPコネクター…)でISPコネクターのセフルテストを実行して下さい。
このテストを6か月毎に実行されることをお薦めします。

Programmer/Calibration test[プログラマ/カリブレーション・テスト]
コマンドはBeeHive204、BeeProg2、及び、BeeProg2Cのオプションのアクセサリである48ピン・キャリブレーション・テスト・ポッド(Type I)を使用してプログラマの校正値のテストを実行します。
注文番号: 70-0438.
ZIFソケットの各ピンのTTLドライバ、VCCP、VPP1とVPP2電圧のテストされた電圧レベルがあります。キャリブレーションテストの結果はファイルに保存したり、印刷したりすることができます(次の使用のために)。

- プログラマのキャリブレーション・テストポッドとDIL48 ZIFソケット間の適切で安定した接触を保証するためにZIFソケットのポッドとピンのリードをクリーニングすることを強く推奨します。
- aキャリブレーション・テストポッドをZIFソケットに挿入した後、プログラマのZIFソケット内のポッドをわずかに動かし、左右に移動してZIFソケットからポッドのリードを抜かないようにします。

48 Pins Calibration test pod, Type Iの回路図



48 Pins Calibration test pod, Type Iテスト用シーケンス
1. J9にAメーターを接続
2. J7にGNDを接続
3. J6に+5Vを接続
4. JP2-2に+5Vを接続した後、70mAの電流を流すAメーターを起動します。

5. 測定電流が56mAから84mAの場合、48 Pins Calibration test pod, Type IIはOKです。
その他の接続は校正試験手順中に試験されます。

Options[オプション]

このオプション・メニューはユーザーが各種デフォルト設定を見て、そして、変更するためのコマンドを含んでいます。

Options / General options[オプション/一般オプション]
一般オプション・ダイアログはユーザーがプログラムの下記のオプションをコントロールし各種オプションを PG4UW コンフィギュレーション・ファイルをコマンド **Options/Save options[オプション/オプション保存]**によりセーブすることが出来ます.

File options [ファイル・オプション]
ファイル・オプション・ページは現在のファイルとローディング、自動再ロードの前にイレース・バッファとロードされたファイルのファイル・フォーマットの認識方法のためのオプションをセットすることが出来ます。

Erase buffer before loading options [ローディング・オプション前にバッファをイレース]はデータ・ファイルのローディングの前にイレース・バッファ(希望する値で)自動的にセットします。

グループ内の現在のファイルが別のプロセスによって変更された場合は、実際にロード(現在の)ファイルの再ロードのモードを設定することができます。 3 つの選択があります:
· 再ローディング・ファイル前のプロンプト
· 自動的に再ロード
· 現在ファイルの変更スキャンニングを無視
ファイルの修正がテストされた時の 3 つのシチュエーションが有ります:
· 他のアプリケーションからコントロール・プログラムへの切り替え
· デバイス操作 Vefiry、又は、Program の選択
· 最後のデバイス操作のリピートがダイアログ"Repeat?"で選択された時

Load file format ロード・ファイル・フォーマット]はファイルのロードのためのファイル・フォーマット認識のモードをセッティング。自動ファイル形式が選択された時、プログラムで利用可能なサポートされたフォーマットの各々に対してローディング・ファイルとテスト・ファイルのプログラム・フォーマットを解析します。
ファイル・フォーマットがサポートされている形式のいずれかと一致する場合はファイルが検出された形式でバッファリングするために読み込まれます。
マニュアル・ファイル形式はユーザーがサポートされているファイル形式のリストから明示的に望んだファイル形式を選択することができます。
ファイル形式がユーザが選択した形式と一致しない場合、ファイルは不完全、又は不正にロードされます。

チェックボックス Show "Load recent project"ダイアログはプログラムの開始でアプリケーション PG4UW 起動時に表示されるダイアログを設定します。ダイアログ Load recent project は最近のプロジェクト(プロジェクト履歴)のリスト含んでいます。ユーザーは直ちにリストから選択し、そして、プロジェクト・ファイルをロード、又は、プロジェクト・ファイルをロードせずにダイアログを閉じることが出来ます。

File extensions[ファイル拡張子]
ファイル拡張子ページはファイル・マスクをセットすることが出来ます。
ロードするファイル形式の拡張子とプロジェクト・ファイルの拡張子を指定します。

File format mask[ファイル・フォーマットのマスク]は全てのファイル・フォーマットに対する File/Save[ファイル/保存]と File/Load[ファイル/ロード] ファイル・ウィンドウでのファイル・リストのためのフィルターとしてファイル名のマスクを設定するのに使用します。マスクは少なくともワイルドカード(*, ?)を1つを含んで、そして、構文に正しく適応していなければいけません。

ノート: 各ファイル・フォーマットに対して複数のマスクを指定することが出来ます。セミコロンは拡張子のための区切り文字として使用されます。
例: Motorola: *.MOT;*.S19
2 つのファイル・マスクを定義 *.MOT と *.S19 を定義 - Motorola ファイル形式。

プロジェクト・ファイルのデフォルトの拡張子は File/Load project[ファイル/プロジェクトのロード]と File/Save project[ファイル/プロジェクトのセーブ]ダイアログのデフォルトの拡張子として使用されるプロジェクト・ファイル拡張を設定するために使用されます。

Buffer[バッファ]
このオプション設定により、デバイスを選択した時にバッファメモリ内容を自動で指定 データにクリアすることが出来ます。
消去時データ
- 自動設定 を選択した時は使用するデバイスの BLANK 状態にクリアします。
- ユーザーが指定する を選択した時はその後ろにある - 消去データを指定する 部分に入力したデータでクリアします。

指定したバッファファイルのパスを有効にする
このオプションは特定のバッファエリアをファイル データでクリアする時に指定します。この機能は動作が遅いためあまりお勧めはしません。

Erase buffer before selecting of new device[新しいデバイスの選択前にバッファをイレース]動作を選択することが出来ます。
これは特定のアドレスのデータの正確な型を必要とする特別なデバイスのいくつかの種 類に便利に使用することができ、そして、データはこのデバイスのためにバッファにロードされたデータのファイルの一部ではありません。

バッファは選択したデバイス、又は、カスタム定義の値を持つ"blank"値はデフォルトで消去(フィル)することが出来ます。これはグループボックス Erase value and

Custom erase value edit field[イレース値とカスタム・イレース値編 集フィールド]で制御することができます。

ノート : バッファの消去を行うために 多くの時間を消費しますで大規模なデバイス(8MB 以上メガバイト)の場 合この機能を使 用することはお薦めしません。
設定は PG4UW コンフィギュレーション・ファイルにセーブされます。プロジェクト・ファイルにはセーブされません。

Language[言語]
メニュー、ボタン、ダイアログ、情報とメッセージの様なユーザー・インターフェースのために他の言 語を選択することが出 来ます。

Sound[サウンド]
パネル Sound settings[サウンド設定]ページではプログラムのサウンド・モードを選択 できます。プログラムはいくつかのアクティビティの後にサウンドを生 成します。
デバイス上の活動(プログラミング、ベリファイ、リード等)。
警告、又は、エラー・メッセージが表示されたときにも、サウンドが生 成されます。
ユーザーは Windows システムサウンド(必要なサウンドカードがインストールされている)、PC スピーカー、又は、サウンドなしのサウンドを選択できるようになりました。パネル次のアクションのサウンドを許 可するには、以下のオプションがあります :

- チェックボックス Successful operation[操作成功]

オンにすると、デバイスの操作が正常に完了した後にサウンドが生 成されます。
チェックを外すと、正常にデバイスを操作した後でサウンドは生 成されません。

- チェックボックス In case of error[エラーの場合]

オンにすると、デバイス操 作がエラー終 了した後にサウンドが生 成されます。
チェックを外 すと、デバイスの操 作がエラーで終 了した後でサウンドが生 成されません。
パネル Programmer internal speaker sound settings[プログラマの内 部スピーカーのサウンド設 定]では、内 蔵スピーカーを内 蔵しているプログラマーのためにサウンドオプションを設定することができます。
デバイスの動作結果を示すために各デバイスの動作後に内蔵のプログラマスのピーカからピープ音が鳴ります :
良い結 果と悪い結 果。

Errors[エラー]
このオプションはデバイス・ベリファイ・エラーをファイルにセーブする設 定を行います。
ベリファイ・エラーした時、45 個まで Log ウィンドウに書かれます。もし、ユーザーがベリファイ・エラー(データの差 異)とファイルにセーブしたい場合、セクション Save device verify errors to file[デバイス・ベリファイ・エラーをファイルにセーブ]のオプションで 2 つの方法の何れ かでセットすることが出来ます : 同じファイルに対する全てのベリファイ動作からの累 積エラー、又は、最 後のベリファイ動作からのみファイルにエラーをセーブ。
ベリファイ・エラーは指定された名 前で Error file name edit box[エラー・ファイル名編 集ボックス]によりファイルにセーブされます。

次のエラー・レポート・ファイル・オプションが利用出来ます:
・オプション No[いいえ](デフォルト)はベリファイ・エラーをファイルにセーブするのを無効にします。
・オプション New save[新規セーブ]はベリファイ・エラーLog ウィンドウに表示されると同時に最後のベリファイ動作からのみファイルに書かれます。
新しいベリファイ動作書き込み前にファイルが削除されて新しいものが作成されます。
・オプション Append[追加]は全てのベリファイ動作からのベリファイ・エラーが同じファイルに累積されます。もし、ファイルが存在しない場合は、新しいファイルが作成されます。ボックス Error report file size limit[エラー・レポート・ファイル・サイズ制限]はファイルにセーブされるベリファイ・エラーの最大数をセッティング出来ます。

次のオプションが含まれています:
・チェックボックス Stop verification after max. number of errors reached[最大エラー数に達した後、ベリファイを停止]
チェックにされていますと、ベリファイ動作はファイルにエラーの最大数に達した場合停止します。チェックにされていないと、全てのベリファイ・エラーがファイルにセーブされます。
・編集ボックス Max. number of errors specifies number of verify errors[ベリファイ・エラーの数を最大エラー数で指定]はエラー・ファイルに 1 つのベリファイ操作を書くことが出来ます。

Log file[ログファイル]
このオプションは Log window[ログ・ウィンドウ]の使用と関連しています。このウィンドウに関するすべてのリポートは Log file[ログ・ファイル]にも書込まれます。ログ・ファイル名はデフォルトで"Report.rep"です。
コントロール・プログラムが Log file name edit box[Log ファイル名編集ボックス]で指定された名前で指定されたディレクトリーにこのファイルを作成します。
次のログ・ファイル操作が利用出来ます:
· No デフォルトは Log ウィンドウの内容を Log ファイルにコピーしません。即ち、全レポートは Log ウィンドウのみに表示されます。
· Rewrite 古いログ・ファイルがある場合は削除され、コントロール・プログラム開始毎に 新しいファイルが作成されます。
· Append 既存のログ・ファイルに全リポートを追加します。
ファイルがない場合は新しいファイルを作成します。

チェックボックス Add date information to Log file name[日付の情報をログファイル名に追加]は Log file name edit box[Log ファイル名編集ボックス]でユーザーが指定した日付情報をセットすることが出来ます。チェックボックスにチェックが入れられている時、プログラムは自動的に次のルールでユーザー指定の Log ファイル名に現在の日付を追加します:

ユーザーが指定したログ・ファイル名がフォーマット持っている場合:
<user_log_file_name>.<log_file_extension>

日付が追加された名前は
<user_log_file_name><-yyyy-mmm-dd>.<log_file_extension>

日付を表す新しい部分は yyyy - year, mmm - month and dd - day で構成されます。

サンプル: ユーザーが指定した Log ファイル名 : c:\logs\myfile.log
追加された日付の最後のログ·ファイル名はこのようになります(2006 年 11 月 7 日の場合):
c:\logs\myfile-2006-nov-07.log

日付情報の前にプリフィックス無しでログ・ファイル名を付けたい場合、次の様にログ・ファイル名をとして指定することができます。
.<log_file_extension> - .(dot)はファイル名の最初

サンプル: ユーザー指定ログ・ファイル名: c:\logs\.log
追加された日付の最後のログ·ファイル名はこのようになります(2006 年 11 月 7 日の場合):
c:\logs\2006-nov-07.log

アドバンスド・オプション Log ファイル・サイズ制限の利用について:
- オプション Use Log file text truncating when file size limit is reached[ファイルサイズの上限に達したときにログ・テキスト切り捨てのファイルを使用] - チェックが入っている時、ログ・ファイルのサイズ制限がオン。 これはログ・ファイルのサイズが指定された値に達した時にログファイルに含まれているテキストの一部が切り捨てられることを意味します。オプションにチェックが入っていない時、ログ・ファイルのサイズは無制限で、PC のフリー・ディスク容量のみにより制限されます。
- オプション Maximum Log file size[最大ログ・ファイル・サイズ]は kB 単位でログ・ファイルの最大サイズを指定します。
- オプション Amount of truncated text[切り捨てられたテキストの量]は最大ログ・ファイル・サイズに達した後に切り捨てられるログ・ファイル・テキストの%を指定することが出来ます。大きい値はより多くのテキストがログ・ファイルから切り捨てられる(削除)されることを意味します。ログ・ファイル設定は Options/Save options[オプション/オプションをセーブ]によってディスクにセーブすることが出来ます。

Job Report[ジョブ・レポート]
ジョブ・レポートは最近のデバイスで行った操作の概要説明を表します。Job はプロジェクト・ファイルに関連付けられロード・プロジェクトで開始された操作からる新しいプロジェクトのローディング、又は、プログラム PG4UW のクローズ迄の情報です。ジョブ・レポートは次の情報を含みます:
- プロジェクト名
- プロジェクト日付
- プロテクト・モード・ステータス
- PG4UW ソフトウエア・バージョン

· プログラマー・タイプとシリアル番号
· ジョブの実行の開始時間(ロード・プロジェクト操作がおこなれた時間)
· ジョブが実行された終了時間(ジョブ・レポートの作成の時間)
· デバイス名
· デバイス・タイプ
· チェックサム
· デバイス操作オプション
· シリアライゼーション情報
· 統計情報

ジョブ・レポートは次の場合に作成されます:
· ユーザー・コマンドのロード・プロジェジェクトが選択された時
· 閉じる、又は、切断されるプログラマーのサイトが選択された時
· PG4UW を終了した時
· デバイス・カウント・ダウンが 0 になった時(スタータス完了)
· ユーザーによって、メニュー"File | Job Report"が使用された時

ジョブレポートは最近ロードされたプロジェクト・ファイルのために、合計の統計値が 0 より大きい時のみ生成されます。これは少なくとも 1 つのデバイス操作(program, verify,...)が行われなければならないことを意味します。

次のオプションがジョブ・レポートのために利用できます:

チェックボックス Enable Job Report function[ジョブ・レポート機能を有効にする] - チェックされた時、ジョブ・レポート機能がアクティブ(有効)にされます。そうでない場合はジョブレポート機能は無効にされています。

チェックボックス Automatically save Job Report file[ジョブ・レポート・ファイルを自動的にセーブする] - チェックされた時、ジョブレポートは編集フィールド・ジョブレポート・ディレクトリーで指定したディレクトリーに自動的に保存され、そして、次のファイル名で作成されます:

job_report_<ordnum>_<prjname>.jrp

<ordnum> はファイルの 10 進数の順序です。もし、同じ名前の既存のレポート・ファイルが存在する場合、新しいレポート・ファイルの順序は既存のファイルにインクリメントされます。
<prjname> は最近使用したプロジェクトのプロジェクト・ファイル名で、プロジェクト・ファイル名の拡張子はありません。

Example 1:
プロジェクト・ファイル c:\myproject.eprj を使用し、ジョブ・レポートのためのディレクトリーが d:\job_reports\ にセットされている場合。
ジョブ・レポート・ディレクトリーにレポート・ファイルが無い場合、最後のジョブ・レポートのファイル名は: d:\job_reports\job_report_000_myproject.jrp

Example 2:

Example 1 からの条件を使用し、しかし、1 つのレポート・ファイルが既にあると仮定します。
このファイルの名前は d:\job_reports\job_report_000_myproject.jrp

最終的に新しいレポートのジョブ・レポート・ファイル名は:
d:\job_reports\job_report_001_myproject.jrp
ノート: ファイル名に含まれている番号の順番が 1 つインクリメントされています。

自動的にジョブ・レポート・ファイルをセーブするセッティングに設定されている時、ジョブ・レポートを生成する時に Job Report ダイアログは表示されません。
新しく生成されたジョブ・レポートはダイアログやメッセージ無しでセーブされます(ファイルのセーブ中にエラーが起こらない場合)。

もし、チェックボックスが Automatically save Job Report file[自動的にジョブ・レポート・ファイルをセーブ]のチェックが外されていますと、PG4UW は Job Report ダイアログは必要なら毎回表示されます。
Job Report ダイアログでユーザーはジョブ・レポートで行う操作を選択することができます。もし、何も選択しない場合(ボタンを閉じる)、ジョブ・レポートは PG4UW ログ・ウィンドウにのみ書かれます。

Automatic YES![オートマチック YES!]
ユーザーはプログラマーとソフトウエアがアクティブ Automatic YES!モードでプログラムされたデバイスの取り除きと新しいデバイスの装着を待つ時の状態の表示のデフォルト・セッティング(PG4UW ソフトウェアのプリセットとして)を無視することが出来ます。

デフォルト・セッティング(PG4UW ソフトウェアのプリセットとして) - プログラマーはデバイスがプログラムされた時とプログラマーが新しいデバイスに交換されるのを待っている時にその状態を表示します。を点滅させます。
マルチ-ソケット・プログラマーではこの表示はされません(quiet mode)。シングル-ソケット・プログラマーは LED ビジー点滅によりこの状態を表示します。By LED Busy blinking セッティングをご覧下さい。

LED 表示無し(Quite mode) - プログラマーは ZIF ソケットの数に関係なく(マルチ-プログラマー、シングル・ソケット・プログラマー)共にデバイスがプログラムされて新しいデバイスの装着を待つ時に状態を表示しません。デバイス操作の後、その操作の結果によりステータス LED error 又は、OK の何れか1つのみが点灯します。この LED は ZIF ソケットからデバイスが取り除かれるのを検知しますと直ちにオフになります。

By LED Busy blinking[LED ビジー点滅] - プログラマーは ZIF ソケットの数に関係なく(マルチ-プログラマー、シングル・ソケット・プログラマー)共にデバイスがプログラムされて新しいデバイスの装着を待つ時に LED ビジーで点滅することで状態を表示します。
デバイス操作の後、その操作の結果によりステータス LED error 又は、OK の何れか1つのみが点灯し、そして、LED ビジーが点滅します。もし、プログラムが

ZIF ソケットからデバイスが取り除かれるのを検知しますと LED はオフになりますが、新しいデバイスで操作が繰り返すためにプログラムが用意していることを示すために LED ビジーは点滅します。プログラムが ZIF ソケットで(新しい)デバイスの 1 つ以上のピンを検知しますと、LED ビジーは連続して点灯します。この時点からプログラムは新しいデバイスの残りのピンが装着されるために要求された時間待ちます。
もし、要求された時間(デバイス装着完了時間)がオーバーフローしたり、デバイスが正しく装着されなかった場合、プログラムはこの状態を示すために LED Error を点灯します。デバイスが正しく装着された時、ステータス LED はオフになり、デバイスでの新しい操作が開始されます。

Remote control[リモートコントロール]
PG4UW コントロール・プログラムのリモート・コントロールは他のアプリケーションによる PG4UW アプリケーションのいくつかの機能を制御することが出来ます。
これはマス-プロダクション・ハンドラー、又は、他のアプリケーションにデバイス・プログラマーを統合するために大変適した機能です。

PG4UW を制御するリモート・アプリケーションはサーバーとして動作します。プログラム PG4UW はクライアントとして動作します。PG4UW とリモート・コントロール・プログラムは TCP プロトコル経由で通信されます。
- これは PG4UW が 1 つの PC にインストールされ、そして、リモート・コントロール・アプリケーションが他の PC にインストールされ、そして、これらの PC がネットワーク経由で接続します。

リモート・コントロールのためのデフォルト TCP 通信設定は:

Port: telnet Address: 127.0.0.1 又は、ローカル・ホスト

アドレス設定は PG4UW(クライアント)のみに適用されます。ポート設定は PG4UW(クライアント)とサーバー・アプリケーションにも適用されます。

デフォルト設定は 1 台の PC(アドレス・ローカルホスト)上のリモートコントロールを使用を許可します。
PG4UW(クライアント)とリモート・コントロール・サーバーを同じ PC にインストールする必要があります。

ノート: もし、ファイアウォールがシステムにインストールされている場合、ファイアウォールはリモート・コントロール・サーバー、又は、クライアントがスタートする時に警告メッセージを表示することが出来ます。
ファイアウォールがリモート・サーバー、又は、クライアントのネットワーク・アクセスの強化、又は、拒否を尋ねる質問を警告表示された時、'Allow'オプションを選択して下さい、そうでないとリモート・コントロールが動作しません。勿論、指定されたアドレスとポートのみでリモート・サーバー/クライアント・アクセスを許可するためにより厳格な権限をファイアウォール・オプションで指定することが出来ます。

PG4UW のリモート・コントロールとデモ用リモート・コントロール・アプリケーションの詳細については PG4UW がインストールされているディレクトリ内の\RemoteCtrl サブ・ディレクトリにあるアプリケーション・ノート remotemanual.pdf を参照してください。

リモート・コントロールのマニュアルは PG4UW のインストール中に作成された Windows スタート/プログラム・メニューリンクからリモート・マニュアルにもあります。
*Remote control of PG4UW, user's manual
5. Using remote control with multiply programmers (multiprogramming)
Restrictions:を特に注意してご覧下さい。弊社でのサポートは出来ません。

Save options[オプションの保存]
プログラム終了時のオプション設定のセーブの選択が出来ます。3 つのオプションが利用出来ます。

Don't save	プログラム終了中にオプションをセーブしません、そして、保存オプションも聞いてきません。
Auto save	保存オプションを聞くことなくプログラム終了中にオプションをセーブ
Prompt for save	プログラム終了前に保存オプションを聞いてきます。ユーザーはオプションをセーブするか、又は、しないかを選択することが出来ます。

Other[その他]
プログラムの処理優先レベルをセットするために使用します。システム内のより要求の厳しいアプリケーションの実行がある場合は優先レベルの設定はプログラマーのパフォーマンス(デバイス・プログラミング時間)に影響を与えます。
アプリケーション優先レベルを Low にセッティングした場合書き込み時間が非常に遅くなりますので注意して下さい。

Tool buttons[ツール・ボタン], メイン・プログラム・ウィンドウでのツール・ボタン hint display[ヒントの表示]オプションを変更することが出来ます。
パネル Start-up directory[起動ディレクトリ]はプログラムの起動時にディレクトリを選択するモードを選択することが出来ます。
デフォルトの起動ディレクトリはプログラムが呼び出されるディレクトリを意味します。
プログラムは最後に終了されたディレクトリはプログラムの最後に終了した最後のカレント・ディレクトリを意味します。
このディレクトリはディレクトリ履歴リストから最初のディレクトリを前提としています。

Colors and LEDs[カラーとLED]
このオプションはプログラマのLEDの動作を異ならせることができます。ステータスLEDのカラースキーム、全LEDの輝度と制御プログラムの視覚的オプションを選択することができます。
プログラマの動作結果LEDのカラー:
- 標準カラー・スキーム (ERROR=red, BUSY=yellow)
- 前のカラー・スキーム (ERROR=yellow, BUSY=red)

ノート: 新しいタイプのプログラのみでこれらの設定を利用出来ます。もし、メニュ

ーにセッティングが無い場合、又は、メニューで編集を有効にならない場合、ご使用のプログラマーではLEDカラー・スキームのカスタマイズはサポートされていません。
ソフトウエアでの動作結果表示のカラー:
- 標準カラー・スキーム (ERROR=red, BUSY=yellow)
- プログラマのLEDに従う(ERROR=yellow, BUSY=red)

ノート: *これらのセッティングは古いタイプのプログラマでのみ利用出来ます。*
LEDの明るさはLEDの光強度のカスタマイズを可能にします。
ノート: *これらのセッティングは新しいタイプのプログラマでのみ利用出来ます。*
ラベル・スタイルではアプリケーションのメイン・ウィンドウで重要なラベル(プログラマ・タイプ、デバイス？タイプ、統計、チェックサム等)を強調表示できます。

Options /View[オプション/ビュー]
ツール・バーのようなプログラム環境で違った要素を表示又は、非表示にするためにはビュー・メニュー・コマンドを使います。

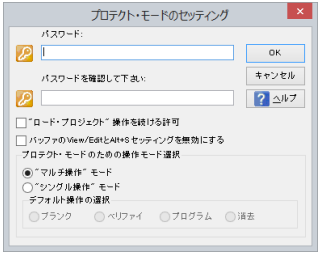
Options /View/Main Toolbar[オプション/ビュー/メイン・ツールバー]
メイン・ツール・バーの表示又は、非表示はこのコマンドを選びます。

Options /View/Additional Toolbar[オプション/ビュー/アディショナル・ツールバー]
アッディショナル・ツール・バーの表示又は、非表示はこのコマンドを選びます。

Options / View / Device options before device operation[オプション/ビュー/デバイス操作前のデバイス・オプション]
デバイス操作が確認される前にデバイス・オプションの表示を有効/無効にするにはこのコマンドを選択します。

Options / Protected mode[オプション/プロテクト・モード]
Protected mode[プロテクト・モード] はプログラムの特別なモードです。プログラムがProtectedモードの時は特定のプログラム操作とコマンドのバッファ、又は、デバイス設定の変更が無効にされます。Protectedモードは不用意にバッファ、又は、デバイス設定を変更することを防ぐために使用されます。Protectedモードは同じ種類のデバイスの大量のプログラミングに適しています。
Protectedモード機能はシングル・プログラミング制御ソフトウェアPG4UWとマルチ・プログラミング制御ソフトウエア**PG4UWMC**で独立して使用可能です。

PG4UWでのProtectedモード
プログラムをProtectedモードに切り替える2つの方法があります:
1. メニュー・コマンド Options/Protected mode[オプション/プロテクト・モード]を使用。このコマンドはパスワード・ダイアログを表示します。ユーザーはパスワードが正しいかを確認するために2度入力しなければいけません。パスワード確認後にプログラムはProtectedモードに切り替えられます。パスワードの入力はProtectedモードをスイッチ・オフするためにも使用されます。



2. Protectedモードで以前にセーブされたプロジェクトを読み込む。詳しくは **File/Save project**[ファイル・セーブ・プロジェクト]をご覧ください。

チェックボックス **Keep "Load project" operation allowed** はデフォルトではインアクティブにセットされています。- それはLoad project operation[ロード・プロジェクト操作]ボタンとメニューはProtectedモードがアクティブな時は無効にされます。もし、オプションが有効(チェックされている)になっている場合、Load project operation[ロード・プロジェクト操作]ボタンとメニューはProtectedモードで許可されます。

チェックボックス **Disable view/edit buffer**はデフォルトではインアクティブにセットされています。- それは View/Edit buffer[ビュー/バッファ編集] ボタンとメニューはProtectedモードがアクティブな時有効にされます。これはバッファの内容を見ることが出来ますが、編集は出来ません(Protectedモードがアクティブなため)。

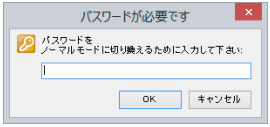
Protectedモードでバッファの内容を見れないようにしたい場合はこのオプションをアクティベートして下さい。この場合、オプション Encrypt project file (with password)[暗号アルゴリズムを使った特殊なフォーマットでプロジェクトをセーブ]もアクティベートされることをお薦めします。
詳しはFile/Save project[ファイル/プロジェクトをセーブ]をご覧下さい。

プロテクト・モードのための操作モード選択
オプション **"Multi operation[マルチ操作]" モード** - リード以外のデバイス操作(blank, verify, program, erase)の全てが利用できるプロテクト・モードの基本を現わしています。これは偶発的、又は、意図的なリード操作によってバッファ・データを変更してしまうことを防ぐ確実性を提供します。1つのプロジェクト(マルチ・プロジェクト)で全てのサポートされているデバイス操作を行いたいときに便利です。

オプション **"One operation[シングル操作]" モード** - 全ての使用可能な1つの操作のみが有効されているプロテクト・モードの拡張フォームを現わします。デバイス操作の間違ったタイプを実行することを防ぐより良い確実性を提供します。

マルチ-プロジェクト・ウィザードを使って"One operation[シングル操作]"モードでセーブされた複数のプロジェクトをビルドすることで制御SWのデバイス操作の標準でないフロー(例えば、Program + Verify + Verify + Verify)も一緒にすることが出来ます。

プログラムをProtectedモードからNormalモードに切り替えるためにはメニュー・コマンドOptions/Normal mode[オプション/通常モード]を使って下さい。"**Password required**" ダイアログが現れます。Protectedモードに切り替えるために使ったのと同じパスワードを入力しなければいけません。



Protectedモードをキャンセルする他の方法はプログラムを閉じて下さい。次にプログラムを起動すればNormal[通常]モードで開始します(唯一の例外はProtectedモードでセーブされたプロジェクトの名前でコマンド・ライン・パラメータによりプロジェクトがロードされている場合)。

Protectedモードがアクティブな時、ソフトウエアはプログラマー・アクティビティ・ログの右上にラベル Protected[プロテクト]モードが表示されます。

他のオプション **Require project file unique ID before first programming**[最初のプログラム開始前にプロジェクト・ファイルのユニークなIDが必要]に付いての情報はコントロール・プログラムの下のボタン・ステータス行にプロジェクト・ファイル名の隣にラベル(ID)により表示されています。
File/Save project[ファイル/プロジェクトのセーブ]をご覧ください。

Protected mode in PG4UWMCでのProtectedモード
PG4UWMCのAdministrator Mode と Operator Mode

プログラムPG4UWMCはデフォルトでAdministrator Modeにセットされています。これはユーザーのためにブロックしている操作が適用されないことを意味します。しかし生産で、いくつかのメニュー・コマンドをブロックするのに適しており、確実にするために、ユーザーは重要なプログラムの設定や構成を変更しません。オペレーター・モードはこの目的のために使用されます。

OperatorとAdministrator ModeはHelpのOptions/Switch to Operator Mode (PG4UWMC)を参照して下さい。

プログラムPG4UWMCはプログラムPG4UWに非常によく似たProtectedモードを持っています。違いはProtectedモードはメニュー・コマンドによりアクティベートすることが出来きますがプロジェクト・ファイルによりアクティベートすることが出来ません。
もう1つの違いは、PG4UWMCのProtectedモード設定はPG4UWMCが閉じられている間にPG4UWMCのコフィギュレーション .iniファイルにセーブされます。次のPG4UWMC開始の間 .iniファイルから取得した最新のProtectedモードの設定が使用されます。

PG4UWMCで使用できる1つのメニュー・コマンドがあります -
Options/Protected mode[オプション/Protectedモード]
メニューOptions/Protected mode[オプション/Protectedモード]選択後、パスワード・ダイアログが現れます。ユーザーはパスワードが正しいかを確認するために2度入力しなければいけません。パスワード確認後にプログラムはProtectedモードに切り替えられます。

Protectedモード設定はPG4UWMCが閉じられている間にPG4UWMCのコフィギュレーション .iniファイルにセーブされます。次のPG4UWMC開始の間 .iniファイルから取得した最新のProtectedモードの設定が使用されます。

チェックボックス **Keep "Load project" operation allowed** はデフォルトではインアクティブにセットされています。- それはLoad project operation[ロード•プロジェクト操作]ボタンとメニューはProtectedモードがアクティブな時は無効にされます。もし、オプションが有効(チェックされている)になっている場合、Load project operation[ロード・プロジェクト操作]ボタンとメニューはProtectedモードで許可されます。

プログラムをProtectedモードからNormalモードに切り替えるためにはメニュー・コマンド**Options/Normal mode**[オプション/通常モード]を使って下さい。
"**Password required**" ダイアログが現れます。Protectedモードに切り替えるために使ったのと同じパスワードを入力しなければいけません。

Protectedモードがアクティブな時、ラベル "Protected mode"がPG4UWMCメイン・ウィンドウのLog windowsのトップの近くに見えます。
*ノート: 時にProtectedモードがアクティブ状態からインアクティブ状態(Normal mode)に切り替えられる時、ある種のコマンド(例えば、"Load project")は無効のまま保持されます。これはボタン **Stop ALL** をクリックすることで解決することが出来ます。*

Multi-projects[マルチ-プロジェクト]

Multi-projectは**sub-projects**と**multi-project**の作成中にセーブされた情報をベースにした如何なるデバイスでの操作のシーケンスを実行するための特別な機能です。
実際にマルチ-プロジェクトを使用する:
• マルチ-チップ・デバイスのプログラム

• デバイス操作のシーケンスの実行(即ち、Program + Verify + Verify + Verify)
更に詳しくは操作モードをご覧下さい。

マルチ-プロジェクトに関する基本事項:
• **Multi-project file[マルチ-プロジェクト・ファイル]**は全ての**Multi-project[マルチ-プロジェクト]**情報を含む特別なファイルです。マルチ-プロジェクト・ファイルには1個以上のプロジェクト・ファイルを含むことが出来ます。マルチ-プロジェクト(ファイル)に含まれるプロジェクトはsub-projectsと呼ばれます。
• **Sub-project[サブ-プロジェクト]**はマルチ-プロジェクト・ファイルのビルド中にマルチ-プロジェクト・ファイルに入れられたクラシック[従来の]・プロジェクト・ファイルです。
• **Project file[プロジェクト・ファイル]** – バッファ・データ、デバイス操作オプション、スペシャル・オプションといくつかの安全機能が組み合わさった特別なタイプのファイルです。デバイスをどのように扱うかを完全に定義します。1度セーブされると、いつでも再ロード出来、そして、操作を繰り返し確実に行えます。
• **Multi-chip device[マルチ-チップ・デバイス]** は2つ以上の独立したチップ(同じ、又は、各種タイプ)を持ったシングル・パッケージのデバイス
• **Sub-device[サブ-デバイス]** - マルチチップ・デバイスの独立したパート。Sub-deviceはPG4UWデバイス・リストから選択することが出来ます。1度選択されますと、デバイス毎のアルゴリズムでアクセスします。パーシャル・チップ単独で定義、テストとプロジェクト・ファイルにセーブすることが出来ます。
• **Master device[マスター・デバイス]** - マルチチップ・デバイス・ユニットはsub-devicesの構成です。マスター・デバイスもPG4UWのデバイス・リストから選択可能です。1度選択しますと、マルチ-プロジェクト・ウィザードを使用して個々のプロジェクト・ファイルからマルチ-プロジェクト・ファイルをビルドし、そして、セーブ/ロード/それを実行。シングル-チップ・デバイスからビルトされたマルチ-プロジェクトの場合はマスター・デバイスは定義出来ません。
• **Device operation[デバイス操作]** - 各操作はメニューから選択して直接実行することが出来、ツール・バー・ボタンでクリックするか、又は、リモート・コマンド(Blank, Read, Verify, Program, Erase)で呼び出すことも出来ます。これらの操作の幾つか(特にProgramとErase)は組み込まれたsub-operationを含み、メニュー/デバイス/デバイス・オプションで編集出来ます。
• **Multi-project Wizard[マルチ-プロジェクト・ウィザード]** - マルチ-プロジェクト・ファイルをビルドするためのアシスタント機能です。ウィザードはユーザーがマルチ-プロジェクトに含むべきプロジェクトを選択し、そして、1つのマルチ-プロジェクト・ファイルにそれらをセーブします。選択されたプロジェクト・ファイルを1つのマルチ-プロジェクト・ファイルにセーブするプロセスはマルチ-プロジェクト・ファイル・ビルディングと呼ばれます。ウィザードはマルチ-プロジェクトに含まれたプロジェクト(sub-devices)に従ってデバイス操作を開始することも出来ます。更に詳しくは後述のマルチ-プロジェクト・ウィザードをご覧下さい。

Multi-project Wizard[マルチ-プロジェクト・ウィザード]
マルチ-プロジェクト・デバイス操作にはマスター・デバイスのsub-devices(chips)に関連した部分的なsub-projectsが含まれているマルチ-プロジェクト・ファイルが必要です。マルチ-プロジェクト・ファイルはマルチ-プロジェクト・ウィザードで作成することが出来ます。ウィザードは次の主機能を持っています:

- sub-projectsを選択、そして、最終的なマルチ-プロジェクトをビルドします。
- 既存のマルチ-プロジェクト・ファイルをロード *1
- 最新のマルチ-プロジェクトのデバイス操作を開始

***ノート* 1:** *既存のマルチ-プロジェクト・ファイルはメニューFile|Load projectを使ってPG4UWのメイン・メニューから、又は、multi-prjコマンドをロードすることでマルチ-プロジェクト・ウィザードからロードすることが出来ます。*

Multi-project Wizard[マルチ-プロジェクト・ウィザード]は次の制御を含んでいます:

- ボタン**Load multi-prj** は既存のマルチ-プロジェクト・ファイルをロードするために使用されます。
- ボタン**Build Multi-project** はテーブル Sub-projectsにリストされたプロジェクトを使って新しいマルチープロジェクトのビルドのために使用されます。
- **Table 1: Sub-projects** は最新のマルチープロジェクトに含まれるプロジェクトのリストを含んでいます。
- ボタン**Add project** はTable 1にあるプロジェクト・ファイルのリストに新しいプロジェクト・ファイルを追加するのに使用されます。
- ボタン**Remove project** はTable 1にあるプロジェクト・ファイルのリストから選択されたプロジェクト・ファイルを削除するために使用されます。
- ボタン**Move up a Move down**はTable 1にある選択されたプロジェクトを1つ上、又は、下に移動するために使用されます。プロジェクトは最も上の行(#1)を最初としてここで指定されたシーケンス順に処理されます。
- ボタン**Help** はこのhelpです。
- デバイス操作**Blank, Verify, Program, Erase, 又は、Run**のボタンはテーブルSub-projectsにリストされている全てのチップ(sub-devices)の選択された操作を実行するために使用されます。
  - **"Multi operation" mode**で, 全ての利用出来る操作を1度に実行(各sub-device上で同じ操作)することが出来ます。
  - **"One operation" mode**で, マルチ-プロジェクトが構成されているプロジェクトによっては1つの操作だけを実行(各sub-device上で同じ操作)が実行出来ます。また、各sub-projectがその1つの操作を実行することが出来ます。

ノート:

- *マルチ・プログラミング・モードではシリアライゼーションはサポートされていません。(シングル・プログラミングのみシリアライゼーションをサポート)*
- *カウント-ダウン機能もサポートされていません。*

マルチ-チップ・デバイス(シングル-チップ・デバイスも同様)をプログラムするためにマルチ-プロジェクトを使用する時は次の2つの基本動作を行う必要があります。:

- マルチ-プロジェクト(又は、マルチ-プロジェクト・ファイル)の作成(ビルド)
- デバイス操作の実行のためにマルチ-プロジェクトを使用

マルチ-プロジェクト(又は、マルチ-プロジェクト・ファイル)の作成(ビルド)

マルチ-プロジェクト・ファイルの作成時は以下のステップを推奨します:

- “classic”プロジェクトを作成, マルチチップ・デバイスの各sub-deviceのための1つのプロジェクト。ジェネリック・デバイスのためのプロジェクトと同じ方法でプロジェクトを作成:
- マルチチップ・デバイスに搭載されている希望するチップに従いsub-deviceを選択 *1
- デバイスのパラメータをセット、設定し、そして、希望のデバイスのデータをPG4UWのLoad Fileコマンドでバッファにロード
- 確認のためにデバイス上でデバイス操作を実行することでデバイスのテストを行って下さい。
- 全てOKの場合、Save projectコマンドでプロジェクト・ファイルを作成することが出来ます。
- そのマルチ-プロジェクトを使用すべきマスター・マルチチップ・デバイスを選択します。マルチチップ・デバイスを選択後、マルチ-プロジェクト・ウィザードは自動的に開かれます。
- マルチ-プロジェクト・ウィザードのAdd projectボタンにより必要なプロジェクトを追加します。各プロジェクトはマルチチップ・デバイスの対応した1つのsub-deviceを表します。
- sub-project選択の完了後、最終的なマルチープロジェクト・ファイル作成のためにボタンBuild Multi-projectを使用します。プログラムは新しいマルチ-プロジェクト・ファイルの名前を聞いて来ますので名前を付けて下さい。最終的なマルチープロジェクト・ファイルはTable 1: Sub-projectsにリストされている全てのsub-projectsを含んでいます。

ノート: 全ての従来(classic)使用していたプロジェクト・ファイルからもマルチ-プロジェクトの作成が可能です。マスター・デバイスとの関連付けは必須ではありません。ユーザーへの配慮のため1つのマルチ-プロジェクト内に正しいサブ・デバイス(sub-project)を結合する方法をサポートしているだけです。この機能はJTAGチェーンでISP プログラミングを使用して異って定義されたプロジェクトを使用する時には特に便利です。
マルチ-プロジェクト・ウィザードは次の動作の1つにより開くことが出来ます:

- *PG4UWのSelect Deviceダイアログからマスター・マルチチップ・デバイスを選択*
- *作成したマルチ-プロジェクト・ファイルをローディング*
- *PG4UWのメニューOptions | Multi-project Wizardから直接 マルチ-プロジェクト・ウィザード・ダイアログを開く*

**1 PG4UW デバイス・リストでのマスター・デバイスとサブ・デバイス・パーツの名前*
Master-device: マルチチップのオリジナル部品名[パッケージ・タイプ]
Sub-devices: マルチチップのオリジナル部品名[パッケージ・タイプ] (part1)
マルチチップのオリジナル部品名[パッケージ・タイプ] (part2) .
……..
マルチチップのオリジナル部品名[パッケージ・タイプ] (part n)
サンプル:
Master-device: TV0057A002CAGD [FBGA107]
Sub-devices #1 TV0057A002CAGD [FBGA107] (NAND)
#2 TV0057A002CAGD [FBGA107] (NOR)

デバイス操作実行のためのマルチ-プロジェクトの使用

既存マルチ-プロジェクト・ファイルの典型的な使用方法は次の順序です。.

PG4UWでのシングル・プログラミング:

- PG4UWメイン・ウィンドウのメニュー・コマンド**File/Load project**[ファイル/ロード・プロジェクト]によって作成したマルチープロジェクトをロードするか、又は、マルチ-プロジェクト・ウィザードで**Load multi-prj**ボタンを使用。マルチ-プロジェクトのローディング完了後、マルチ-プロジェクト・ウィザードは自動的に開きます。
- ウィザードで利用出来るデバイス操作ボタン(Blank, Verify, Program, Erase)の1つを使って希望するデバイス操作を実行、最も使用されるのはプログラム・デバイス操作です。選択されたデバイス操作はマルチ-プロジェクトで定義された各sub-deviceのためのsub-projectのロードとその結果としてのsub-deviceのプログラミングのシーケンスとして実行されます。そして、これはマルチ-プロジェクトの主目的です - マルチチップ・デバイスの各チップのためのデバイス操作のシーケンスを自動で実行。このコンセプトのサイド・イフェクトは、そのデバイスの進捗インジケーターは各sub-device操作の開始時に0にリセットされますので、マルチチップ操作の実行中にプログレス・バーが0数回に"ジャンプ"されているように見える動作になります。
- 全てのsub-devicesのプログラミングが完了(又は、エラー)後、標準の"Repeat"ダイアログが表示されます。プログラマーのソケットからプログラムされたデバイスを取り除き、そして、新しいデバイスを挿着することが出来ます。"Repeat"ダイアログ上でYesボタンを押すか、又は、プログラマーのYES!ボタンを押しますとマルチチップ・デバイスのプログラミング・シーケンスが再開されます。
- Automatic YES!機能がオンされている場合、デバイス操作終了後にRepeatダイアログは表示されずAutomatic YES!ウィンドウが表示されます。このウィンドウにはプログラマー・ソケット状態とプログラムされたデバイスの取り除きと新しいデバイスの装着が表示されます。新しいデバイスの装着後、マルチチップ・デバイス操作シーケンスが自動的に開始します。Automatic YES!の詳細は **Programmer / Automatic YES!** を参照して下さい。

ノート:

- *マルチ・プログラミング・モードではシリアライゼーションはサポートされていません。(シングル・プログラミングのみシリアライゼーションをサポート)*
- *カウント-ダウン機能もサポートされていません。*

PG4UWMCによるマルチ・プログラミング、又は、スタンドアローン・プログラマ

- Load projectメニューでマルチ-プロジェクトをロード
- 利用出来るデバイス操作ボタン(Blank, Verify, Program, Erase)の1つを使って希望するデバイス操作を実行、最も使用されるのはプログラム・デバイス操作です。選択されたデバイス操作はマルチ-プロジェクトで定義された各sub-deviceのためのsub-projectのロードとその結果としてのsub-deviceのプログラミングのシーケンスとして実行されます。そして、これはマルチ-プロジェクトの主目的です - マルチチップ・デバイスの各チップのためのデバイス操作のシーケンスを自動で実行。このコンセプトのサイド・イフェクトは、そのデバイスの進捗インジケーターは各sub-

device操作の開始時に0にリセットされますので、マルチチップ操作の実行中にプログレス・バーが0数回に“ジャンプ”されているように見える動作になります。
• 全てのsub-devicesのプログラミングが完了(又は、エラー)後、PG4UWMCにデバイス操作の結果情報が表示されます。プログラマーのソケットからプログラムされたデバイスを取り除き、そして、新しいデバイスを挿着することが出来ます。サイトに対する操作ボタンを押すか、又は、プログラマー・サイトのYES!ボタンを押しますとマルチチップ・デバイスのプログラミング・シーケンスが再開されます。
• Automatic YES!機能がオンされている場合、プログラマー・ソケット状態とプログラムされたデバイスの取り除きと新しいデバイスを装着後デバイス操作シーケンスが自動的に開始します。Automatic YES!の詳細はProgrammer / Automatic YES! を参照して下さい。

ノート:
• *マルチ・プログラミング・モードではシリアライゼーションはサポートされていません。(シングル・プログラミングのみシリアライゼーションをサポート)*
• *カウントダウン機能もサポートされていません。*

Options /Save options [オプション/セーブ・オプション]
このコマンドは、オート・セーブがターン・オフになっていても、保存に対する現在サポートされている全ての設定をセーブします。

Help[ヘルプ内は英文]
メニュー・ヘルプにはサポートされているデバイスやプログラマ、プログラム・バージョンに関する情報を表示するためのコマンドが含まれています。

<F1>キーを押すとヘルプにアクセスします。メニュー項目を選択して<F1>を押すと、状況依存ヘルプにアクセスします。PG4UWがプログラマで操作を実行している間は<F1>は応答しません。
次のヘルプ項目が強調表示されます。
• 現在のヘルプで参照されるキーを表す単語[英文]
• 他のすべての重要な単語[英文]
•現在の相互参照。この相互参照をクリックすると詳細情報が表示されます。

個々のメニュー・コマンドの詳細については統合オンラインヘルプを参照してください。

ノート: *このマニュアルに記載されている情報はリリース時点での正確さを期していますが全ての製品を継続的に改善しています。*
https://www.elnec.com/en/download/#manuals のマニュアルを参照して下さい。

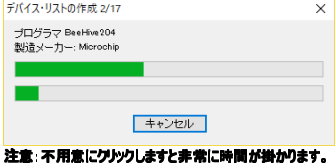
ヘルプシステムは制御プログラムと共に継続的に更新されるため、このマニュアルに記載されていない情報が含まれている可能性があります。

Help / Supported devices[ヘルプ サポートされているデバイス]
このコマンドはサポートされ ている全てのプログラマの少なくとも1つのタイプによってサポートされている全てのデバイスのリストを表示します。これは特に、少なくとも1つのタイプのプログラマによってサポートされているデバイスを探したい場合に便利です。デバイスの名 前の前に"g_"というプリフィックス(Prefix "g_")が付いていればデバイスはマルチ・ソケット・プログラマでサポートされています。
.

Help / Supported programmers[ヘルプ サポートされたプログラマ]
このコマンドはこのプログラムがサポートされているプログラマに関する情報を表示します。

Help / Device list (current programmer)[ヘルプ デバイス・リスト(現在使用のプログラマ]
このコマンドは現在使 用中のプログラマの全てのデバイス・リストを作成し、それを????? **DEV.txt**テキスト・ファイルと????? **DEV.htm** HTMLファイルとして制御プログラムが実行され ているディレクトリに保存します。マーク?????は現在のプログラマの略称に置き換えてデバイス・リストが生 成されます。

Help / Device list[ヘルプ/デバイス・リスト(すべてのプログラマ]
※非常に時間が掛かりますのでお薦め致しません。

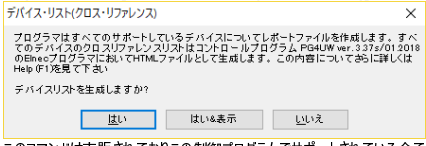


注意:**不用意にクリックしますと非常に時間が掛かります。**
このコマンドは全 てのプログラマのデバイス・リストを作成し、それを?????**DEV.TXT**テキスト・ファイルと????? **DEV.HTM** HTMLファイルが制御プログラムが実行され ているディレクトリに保存します。
例 : Creating device list ...
2 の 17 操作 ...
BeeHive204プログラマのためのデバイス・リストを作成 中 :
"C:\ユーザー\任意の管理者名]\AppData\Roaming\Elnec\Pg4uw\beeh204dev.txt"
"C:\ユーザー\任意の管理者名]\AppData\Roaming\Elnec\Pg4uw\beeh204dev.htm" ...
デバイス・リストのHTMLファイルをディレクトリに書き出し中 :
"C:\Users\Owner\AppData\Roaming\Elnec\Pg4uw\dev_html\beeh204dev\" ...
Duplicate devices found!: デバイスメーカー名 デバイス名 [パッケージ]完了
経過時間 : 0:05:23

キャラクター?????はプログラマの略称に置き換えればデバイス・リストが生成されます。

ノート: このコマンドの実行後、制御プログラムは現在のデバイスに関するすべての情報を失います。
DEVICE[デバイス]・メニューのいずれかの選択方法で希望のデバイスを再選択し直して下さい。

Help / Device list[ヘルプ デバイス・リスト (クロス・リファレンス)]

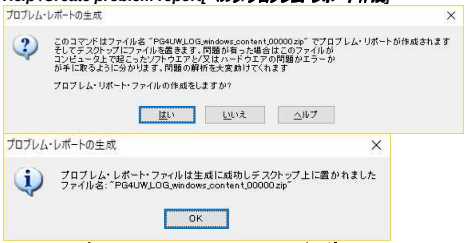


このコマンドは市販されておりこの制御プログラムでサポートされている全てのプログラマがサポートする全てのデバイスの相互参 照[クロス・リファレンス]リストを作成します。結果のリストはHTML形式であり以下のファイルで構 成されています:

- 1つのメインHTMLファイル**TOP_DEV.htm**にサポートされているデバイス・メーカーがリストアップされます
- 部分的なHTMLファイルには各デバイス・メーカーのサポートされているデバイスのリストが含まれています。

メインHTMLファイルはこのプログラマ向けの制御プログラムが置かれているディレクトリーに置かれます。部分HTMLファイルはプログラマのための制御プログラムが置かれ ているディレクトリに置かれ たサブディレクトリDEV_HTMLに置かれます。
*OSにより場所は異なります。.

Help /Create problem report[ヘルプ プロブレム・レポート作成]



プロブレム・レポート作成コマンドは特定の診断情報を**ログ・ウィンドウ**に書き込むために使用され 、結果としてログウィンドウの内容を
含む全ての情報を1つの
"**PG4UW_LOG_windows_content_xxxxxx.zip**"[LOGファイルと呼びま

す]ファイルにし、デスクトップ上に置かれます。ログ・ウィンドウの内容はPG4UWの画面をコピーすることでクリップボードから任意のテキスト・エディタに配置できます。プロブレム・レポートは制御プログラムやプログラマにエラーが発生し、エラーの種類が自分で解決できず、プログラマーの製造元に連絡しなければならない場合に便利です。この場合、ユーザーは必ず問題について製造元にメッセージを送るときにはLOGを送り下さい。
LOGはElnecがエラーの理由をローカライズして早期に解決するのに役立ちます。

About[プログラムについて]
メニューからInfoコマンドを選択すると著作権とバージョン情報を示すウィンドウが表示されます。

PG4UWMC マルチ制御プログラム

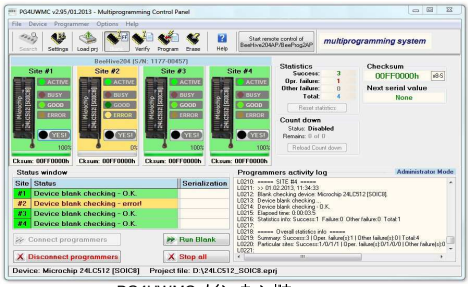
プログラム**PG4UWMC**は複数のプログラマ、又は、USBポートに接続されたプログラマが同じコンピュータにマルチ・プログラム可能なプログラマで完全に独立した同時デバイス・マルチ-プログラミングに使用されます。*1つのPC(及び、PG4UWMC)に標準モードで最大8つのプログラミング・サイトを接続するか、ネットワークモードで最大32のプログラミング・サイトを接続できます。

PG4UWMCは大量生産オペレーションの容易な監視を簡単に行うことに焦点を当てています。PG4UWMCのオペレーター・フレンドリーなユーザー・インターフェイスは多くの強力な機能と使い易さを組み合わせて重要でない詳細にオペレーターに負担をかけることなく全ての重要な動作と操作結果の概要を提供します。**PG4UWMC**はマルチ-プログラミング・システムを制御するためにプロジェクト・ファイルを使用します。プロジェクト・ファイルはユーザー・データ、チップのプログラミング設定情報、チップ構成データ、オート?プログラミング?コマンド?シーケンス等を含んでいます。プロジェクト・ファイルが正常に作成され、そして、技術者により確認された上で全ての必要な情報が簡単な操作で行えますので、従って、オペレーターのエラーが最小化されます。オプショナルのプロテクト・モードプロジェクト・ファイルの不要な変更を回避するためにプロジェクト・ファイルを設定することが出来ます。各チップはシリアル番号、設定とカリブレーション情報などの別のデータを用いてプログラムすることが出来ます。

プログラム **PG4UWMC**は次のメイン・ウィンドウで構成されています:

- メイン・ウィンドウ
- 設定ダイアログウィンドウ
- "Search for Programmers"ダイアログ・ウィンドウ

PG4UWMC のメイン・パートの基本説明



PG4UWMC メイン・ウィンドウ

PG4UWMCのメイン・ウィンドウは次のパーツで構成されます:

メニューとツール・ボタン
ツール・ボタン・セッティング
ボタンはPG4UWMCセッティング・ダイアログを開くために使用されます。セッティング・ダイアログは以下の通りです。

パネル Site #1, Site #2,... パネルは以下について表示します:
- •選択されたプログラマーのサイト
- •プログラマー・サイトの動作状況
- •現在のデバイス操作状態と/又は、結果

各パネルはデバイス操作を開始するために使用されるボタン**Run** 又は、ボタン**YES!** を含んでいます。

ボックス Statistics
Statistics[統計]はプログラムされたデバイスと成功と失敗したデバイスの数について通知します。
ノート: *'Reset statistics' ボタンはサイト上で操作が実行されるまで無効にされます。操作を中止したい場合はボタン'Stop All'を押して下さい。*

Checksum
チェックサムは現在のプロジェクト・ファイルからロードされたデータのチェックサムを表示

パネル Status window
パネル Status windowは各サイトの現在の状態を知らせます。状態は:

Blank	サイトは非アクティブ
Ready	サイトはアクティブで動作用意が出来ています。プログラマは接続されていますがデバイス操作は実行されていません。
その他の情報	デバイス操作が実行されている、結果、プログラマー接続状態等々

Log window ステータス・ウィンドウの右側
Log windowはProgrammers activity logと表示され、プログラマーの接続/非接続、デバイス操作結果やその他の情報が含まれています。

ボタン Connect programmers
このボタンは選択されたプログラマー/プログラマーのサイトの全てを接続するために使用されます。通常はPG4UWMC開始後の最初のステップとして使用されます。

ボタン Disconnect programmers
このボタンは接続されている全てのプログラマー・サイトを切断し、プログラマーのサイトの制御プログラムは閉じられます。ボタンは接続されているプログラマーで実行されているデバイス操作が行われていない場合にのみ適用されます。

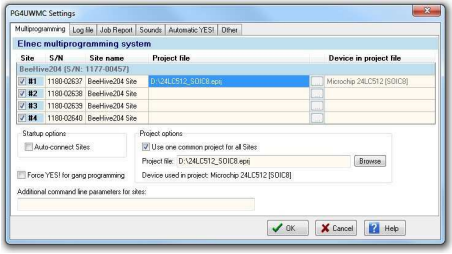
ボタン Run <operation>
ボタンは全ての接続されているプログラマーで同時にデバイス操作を開始するために使用されます。<operation> の値は次のタイプの1つです: Program, Verify, Blank check, Erase.

ボタン Stop ALL
ボタンは全ての接続されているプログラマー・サイトで現在行われているデバイス操作を中止するために使用されます。

ボタン Help
ヘルプを表示するために使用します。

ボタン Start remote control of BeeHive204AP/BeeProg2AP
このボタンは自動化プログラマーBeeHive204AP/BeeProg2APでのみ利用可能で、PG4UWMCの設定 ダイアログ/マルチプログラミング/プロジェクトのオプションで**Use Site #1 project for all Sites**[全てのサイトのためにサイト＃1のプロジェクトを使用]をチェックにされている場合。これはPG4UWMCインターフェースのリモート・コントロールを開始するために使用されます。
ボタンのキャプションでプログラマーは最近接続に使用したプログラマーの名前に置き換えられます。オプションはサイト上でロードされたプロジェクトに応じてモジュールの検出をアクティベートします。

PG4UWMC Settings ダイアログ



PG4UWMC Settings ダイアログは次のオプションをセット、又は、表示するために使用されます:
• プログラマー・サイトのためのセッティング情報を含んでいます: サイト番号, サイトのシリアル番号, サイトに関連したプロジェクト・ファイル
•チェック・ボックス Use one common project for all Sites[全てのサイトで1つの共通プロジェクトを使用]
•チェック・ボックス Auto-connect sites settings[自動接続サイト・セッティング]
•チェック・ボックス Force gang multiprogramming mode[強制ギャング・マルチプログラミング・モード]
•パネル Log file settings[Logファイル・セッティング]
•パネル Job Report settings[Jobレポート・セッティング]
•パネル Automatic YES! Settings[Automatic YES!セッティング]
•パネル Other[その他]

パネル Multiprogramming
PG4UWMCの最初のコントロール・パネルは次を含んでいます:
•指定されたサイト番号で個々のプログラマー・サイトを使ってサイト番号 #1,#2,#3,#4を有効/無効にするチェック・ボックスを含みます。
•プログラマー・サイトのシリアル番号に付いての情報

•Project fileは各PG4UWの実行にロードする個々のプロジェクトのセッティングのための編集行Project: #1, Project: #2, ...Project: #4を含みます。プロジェクト・ファイル名は手動で入力出来ますが、又は、ダイアログ**Select project file** によって各プロジェクト編集行の右側におかれたボタン"..."上でクリックすることで各サイトのために開くことが出来ます。もし、プロジェクト名編集行がブランクの場合、自動プロジェクト・ロードは行われません。

チェックボックス Use one common project for all Sites [全てのサイトで1つの共通プロジェクトを使用]がサイト番号とプロジェクト・テーブルに置かれます。

同じデータで同じデバイス・タイプをプログラムする必要がある時、チェックボックスはチェックにされていなければいけません。
•もし、チェック・ボックスがチェックにされている場合、サイト#1のためのプロジェクト・ファイルが全ての他のプログラマー・サイトのために使用されます。このモードでは全てのサイトはプロジェクト・データの同じシェアされたバッファを使用し同じデバイス・タイプをプログラムします。
•もし、チェック・ボックスがチェックにされていない場合、各サイトはコラム・プロジェクト・ファイルのサイトのテーブル内の名前によって定義されたそのプロジェクト・ファイルを使用します。このモードでは各サイトは各サイトで同時に異なるタイプのデバイスに異なるデータをプログラムすることができ、プロジェクト・データの独自のバッファを使用します。

Auto-connect sitesはPG4UWMC開始後に覚えているサイトがある場合は接続され用意された状態になります。

Force YES for gang multiprogramming mode[ギャング・マルチプログラミング・モードのための強制YES]

各プログラミング・サイトが独立して動作し、他のプログラミング・サイトが実行されている間にオペレーターがプログラムされたデバイスを再ロードできる時、マルチプログラマーでのマルチプログラミング操作の標準モードは**concurrent multiprogramming mode[同時マルチプログラミング・モード]** です。**gang multiprogramming mode[ギャング・マルチプログラミング・モード]**ではあらかじめ定義された操作を任意のYES!ボタンを押すことで**同時に全てのプログラミング・サイト上で操作を開始**します。
ノート:
- *全てのアクティブ(現存し有効な)サイトで動作*
- *何れかのサイトがビジーな間は開始はブロックされます。*
- *このモードではAutomatic YES!は無効にされます。*

パネル Log file セッティングはLogファイル・レポートのモードを設定するために使用します。

Log file は PG4UWMC 制御プログラムの操作フローに関する情報を含むテキスト・ファイルです。これはロードされるプロジェクト・ファイル、デバイス操作のタイプとデバイス操作結果についての情報を意味します。マルチ・プログラミング・システムは幾つかのログ・ファイルを生成します。プログラム PG4UWMC の 1 つのメイン・ログ・ファイルと実行中の各プログラマ・サイトのログ・ファイルです。各サイトは独自のログ・ファイルを 1 つ持っています。サイトのログ・ファイルの名前は編集ボックスのログ・ファイルで指定したログ・ファイルの名前と同じプリフィックスを持っています。ファイル名のプリフィックスは_#<Snum>形式のサイト番号が続きます。

例えば:
Logファイル名はユーザーにより指定されます: "report.log". するとLog ファイルの名前は: - PG4UWMC メイン・ログ・ファイル名 - "report.log"
- Site's #1 Log file name - "report_#1.log"
- Site's #2 Log file name - "report_#2.log"
- Site's #3 Log file name - "report_#3.log"
等々...

次のオプションをLogファイル作成のために設定することが出来ます。
- オプション**Append Log file**はLogファイルの使用法をセットにします。Logファイルは PG4UWMCの最初のリスタート後に作成されます。PG4UWMCの全ての他の開始では、既存のLogファイルはプリザーブされ、そして、新しいデータは既存Logファイルに追加されます。
- オプション**Rewrite Log file**はLogファイルの使用法をセットにします。Logファイルは PG4UWMCの最初のリスタート後に作成されます。PG4UWMCの全ての他の開始では、既存のLogファイルは書き換えら、そして、新しいLogファイルが作成されます。以前のLogファイルのデータは削除されます。

チェックボックス **Add date information to Log file name**[日付の情報をログファイル名に追加]は Log file name [Log ファイル名] 編集ボックスでユーザーが指定した日付情報をセットすることが出来ます。チェックボックスにチェックが入れられている時、プログラムは自動的に次のルールでユーザー指定の Log ファイル名に現在の日付を追加します:
ユーザーが指定したログ・ファイル名がフォーマット持っている場合:

<user_log_file_name>.<log_file_extension>

日付が追加された名前は:
<user_log_file_name><-yyyy-mmm-dd>.<log_file_extension>

日付を表す新しい部分はyyyy - year, mmm - month and dd - day で構成されます。

例えば: *ユーザー指定ログ・ファイル名: c:\logs\myfile.log*

追加された日付の最後のログ·ファイル名はこのようになります(2006年11月7日の場合):
c:\logs\myfile-2006-nov-07.log

日付情報の前にプリフィックス無しでログ・ファイル名を付けたい場合、次の様にログ・ファイル名をとして指定することができます:
.<log_file_extension> - dotはファイル名の最初

例えば: *ユーザー指定ログ·ファイル名: c:\logs\.log*

追加された日付の最後のログ·ファイル名はこのようになります(2006年11月7日の場合):
c:\logs\2006-nov-07.log

アドバンスド・オプション Logファイル・サイズ制限の利用について:
•オプション Use Log file text truncating when file size limit is reached - チェックが入っている時、ログ・ファイルのサイズ制限がオン。 これはログ・ファイルのサイズが指定された値に達した時にログファイルに含まれているテキストの一部が切り捨てられることを意味します。オプションにチェックが入っていない時、ログ・ファイルのサイズは無制限で、PC のフリー・ディスク容量のみにより制限されます。
•オプション Maximum Log file size[最大ログ・ファイル・サイズ]は kB 単位でログ・ファイルの最大サイズを指定します。
•オプション Amount of truncated text[切り捨てられたテキストの量]は最大ログ・ファイル・サイズに達した後に切り捨てられるログ・ファイル・テキストの%を指定することが出来ます。大きい値はより多くのテキストがログ・ファイルから切り捨てられる(削除)されることを意味します。
ノート: *'+'で開始される行はログファイルに表示されますが、画面のログには表示されません。*

共通情報:
Index of Programmer Site[プログラマー・サイトのインデックス]は明確に実行中の各プログラマー・サイトを定義する1から8の整数番号です。
Serial number of Programmer Site[プログラマー・サイトのシリアル番号]は明確に使用されているプログラマー、又は、プログラマー・サイトを定義しています。シリアル番号とプログラマー(サイト)を見つけるまでUSBバス上に接続された全てのプログラマーを検索します。PG4UWMCはプログラマー(サイト)が見つからない場合は"Not found"の表示と共にDEMOモードに設定されます。

1つのコンピューターで8つの同じタイプのプログラマーをサイトとして同時に実行することが出来ます。

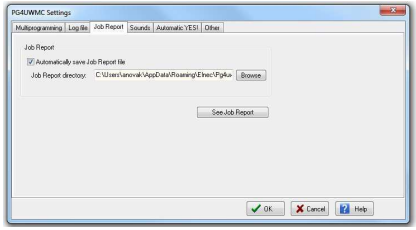
Job Report設定はJobレポート使用のモードをセットするために使用されます。ジョブ・レポートは最近のデバイスで行った操作の概要説明を表します。Job はプロジェクト・ファイルに関連付けられロード・プロジェクトで開始された操作からる新しいプロジェクトのローディング、又は、プログラム PG4UW のクローズ迄の情報です。

Job Report は次の情報を含んでいます:
- プロジェクト名
- プロジェクト日付
- プロテクト・モードの状態
- PG4UWMCソフトウエア・バージョン
- プログラマーのタイプとシリアル番号
- Job実行の開始時間(ロード・プロジェクト操作が行われた時間)
- Jobの実行終了時間(Jobレポートが作成された時間)
- デバイス名
- デバイス・タイプ
- チェックサム
- デバイス操作オプション
- シリアライゼーション情報
- 統計情報

Job Reportは次の場合に生成されます:
- コマンドLoad project[プロジェクトのロード]が選択された場合
- プログラマー・サイトの閉じる、又は、切断が選択された場合
- PG4UWMCを閉じた場合
- デバイス・カウント・ダウン・カウンターが0に達した場合(ステータスの完了)
- ユーザーにより手動でメニュー"File | Job Report[ファイル|ジョブ・レポート]"が使用された場合

ジョブ・レポートは最近ロードされたプロジェクト・ファイルのために、合計の統計値が 0 より大きい時のみ生成されます。これは少なくとも 1 つのデバイス操作(program, verify,...)が行われなければならないことを意味します。

Job Reportダイアログの設定はタブJob ReportのダイアログPG4UWMC Settings (メニューOptions / Settings)です。
次のオプションがJob Reportで利用出来ます:



チェックボックス Automatically save Job Report file[ジョブ・レポート・ファイルを自動的にセーブする] - チェックされた時、ジョブレポートは編集フィールド・ジョブレポート・ディレクトリーで指定したディレクトリーに自動的に保存され、そして、次のファイル名で作成されます:

job_report_<ordnum>_<prjname>.jrp

<ordnum> はファイルの 10 進数の順序です。もし、同じ名前の既存のレポート・ファイルが存在する場合、新しいレポート・ファイルの順序は既存のファイルにインクリメントされます。
<prjname> は最近使用したプロジェクトのプロジェクト・ファイル名で、プロジェクト・ファイル名の拡張子はありません。

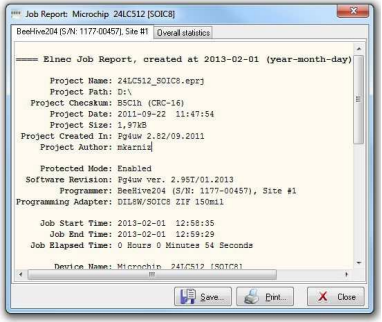
例 1: プロジェクト・ファイル c:\myproject.eprj を使用し、ジョブ・レポートのためのディレクトリーが d:\job_reports\ にセットされている場合。
ジョブ・レポート・ディレクトリーにレポート・ファイルが無い場合、最後のジョブ・レポートのファイル名は:
d:\job_reports\job_report_000_myproject.jrp

例 2: Example 1 からの条件を使用し、しかし、1 つのレポート・ファイルが既にあると仮定します。
このファイルの名前は d:\job_reports\job_report_000_myproject.jrp
最終的に新しいレポートのジョブ・レポート・ファイル名は:
d:\job_reports\job_report_001_myproject.jrp

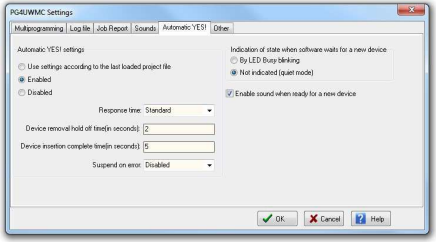
ノート : ファイル名に含まれている番号の順番が 1 つインクリメントされています。

Automatically save Job Report file[自動的にジョブ・レポート・ファイルをセーブ]セッティングに設定されている時、ジョブ・レポートを生成する時に Job Report ダイアログは表示されません。新しく生成されたジョブ・レポートはダイアログやメッセージ無しでセーブされます(ファイルのセーブ中にエラーが起こらない場合)。

もし、チェックボックスが Automatically save Job Report file[自動的にジョブ・レポート・ファイルをセーブ]のチェックが外されていますと、PG4UW は Job Report ダイアログは必要なら毎回表示されます。
Job Reportダイアログでユーザーはジョブ・レポートで行う操作を選択することができます。もし、何も選択しない場合(ボタンを閉じる)、ジョブ・レポートは PG4UWログ・ウィンドウにのみ書かれます。:



Automatic YES! セッティング



このモードではプログラムされたデバイスを取り除いて新しいデバイスをZIF ソケットに装着しますと最後の操作が自動的にリピートされます。プログラムが自動的に新しいデバイスの装着を検知し、最後に行った操作をキー又は、ボタンを押すことなく実行します。ZIFへのデバイスの装着は画面に表示されます。リピート操作の実行はZIFから/への装着/取り外しを待っている間に<Esc> キーを押すことでキャンセルされます。

この機能はある種のタイプのプログラマーでは利用できません。

Use settings according to the last loaded project file[最後にロードされたプロジェクトによるセッティングを使用] – Automatic YES!オプションはプロジェクト・ファイルのセッティングによって設定されます。Automatic YES!の設定の項目の1つは使用されるプログラミング・アダプターに依存する'Pins of programmer's ZIF excluded from sensing'です。これは別のプログラマーが同じデバイスで異なるプログラミング・アダプターを使用することが可能であるため、この設定はこの場合無視されログ・ウィンドウには次のメッセージを見つけることが出来ます:
"None connected pins setting was not accepted due to different programming adapter. Please use automatic YES wizard again."[プログラミング・アダプターが異なるため接続されていないピンの設定が受付られません。Automatic YESウィザードを再度使用して下さい]。もし、これがマスター・プログラミング・サイト(オプション'Use Site #1 project for all Sites'でPG4UWMCを実行している場合)、又は、前述のメッセージがログに書かれたプログラミング・サイトで起こった場合はProgrammer/Automatic YES![プログラマー/Automatic YES!]オプションでボタン'Setting Automatic YES! parameters'をクリックして下さい。'Indication of state when software waits for a new device'と'Enable sound when ready for a new device'のセッティングはプロジェクト・ファイルにストアされません。

Enabled - Automatic YES! 機能はPG4UWMCにによりパラメータ・セットで全ての接続されているプログラミング・サイトを有効にされます。

Disabled - Automatic YES! 機能は全ての接続されているプログラミング・サイトが無効にされます。もし、デバイスのプログラムで次の操作を開始するためにボタンYES!を使用する必要がある場合はこのセッティングを使用して下さい。

Response time - ZIFソケットへのチップ装着と選択されたデバイス操作の開始の間隔となります。もし、ZIFソケットでのチップの長いポジショニングが必要な場合は**elongated** response time[延長した応答時間]を選択して下さい。

Device removal hold off time - ZIFソケットからデバイスを取り除いた時とソフトウエアが新しいデバイスの装着をソケットでチェックを開始する時の間の時間間隔です。この時間は秒間隔で1～120 (デフォルト値は2 秒)でなければいけません。

Device insertion complete time - プログラムが不正に挿入されたデバイスを検出しない様にするために最初のピン(複数)が検知された後に全てのピンが適切に挿入されなければならない時間をセットすることが可能です。この時間は秒間隔で1～120 (デフォルト値は2 秒)でなければいけません。

Suspend on error - Automatic YES!機能でエラーが起こった時に一時停止して操作の結果を見るか、又は、停止せずに続けるかを定義します。

ソフトウエアが新しいデバイスを待つ時の状態表示:

Not indicated (quiet mode) - プログラマーはZIFソケットの数に関係なく(マルチ-プログラマー、シングル・ソケット・プログラマー)共にデバイスがプログラムされて新しいデバイスの装着を待つ時に状態を表示しません。デバイス操作の後、その操作の結果によりステータスLED error又は、OKの何れか1つのみが点灯します。このLEDはZIFソケットからデバイスが取り除かれるのを検知しますと直ちにオフになります。

By LED Busy blinking[LEDビジー点滅] - プログラマーはZIFソケットの数に関係なく(マルチ-プログラマー、シングル・ソケット・プログラマー)共にデバイスがプログラムされて新しいデバイスの装着を待つ時にLEDビジーで点滅することで状態を表示します。デバイス操作の後、その操作の結果によりステータス LED error 又は、OKの何れか1つのみが点灯し、そして、LEDビジーが点滅します。もし、プログラムがZIFソケットからデバイスが取り除かれるのを検知しますとLEDはオフになりますが、新しいデバイスで操作が繰り返すためにプログラムが用意していることを示すためにLEDビジーは点滅します。プログラムがZIFソケットで(新しい)デバイスの1つ以上のピンを検知しますと、LEDビジーは連続して点灯します。この時点からプログラムは新しいデバイスの残りのピンが装着されるために要求された時間待ちます。もし、要求された時間(デバイス装着完了時間)がオーバーフローしたり、デバイスが正しく装着されなかった場合、プログラムはこの状態を示すためにLED Errorを点灯します。デバイスが正しく装着された時、ステータスLEDはオフになり、デバイスでの新しい操作が開始されます。

Enable sound when ready for a new device[新しいデバイスの用意がされた時にサウンドを有効にする] -
チェックされている時、もし、SWが完全な空のZIFソケットを検出し、そして、ZIFソケットに新しいデバイスを受け入れる準備ができている場合には音が発せられます。

前のいずれかのオプションを選択してOKボタンで確認された場合、PG4UWMCは接続されている全てのプログラミング・サイトに選択した設定を送信します。また、もし、マスター・プログラミング・サイトでAutomatic YES!パラメータをセットしていますと、それらのセッティングが全ての接続されたスレーブ・プログラミング・サイトとPG4UWMCに送信されます。

Automatic YES! 機能についての詳しくはProgrammer / Automatic YES![プログラマー/オートマチックYES!].をご覧ください。

Other
プログラマーの動作結果のLEDの色:
•標準カラー・スキーム (ERROR=red, BUSY=yellow)
•旧タイプ・カラー・スキーム (ERROR=yellow, BUSY=red)

ノート: *これらのセッティングはある種のプログラマーのためのみにこれらの設定を利用出来ますもし、メニューにセッティングが無い場合、又は、メニューで編集を有効にならない場合、ご使用のプログラマーでは LED カラー・スキームのカスタマイズはサポートされていません。*

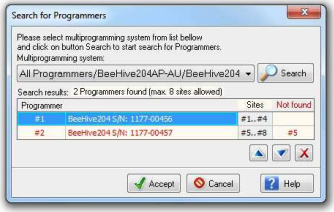
Timer refresh rate[タイマー・リフレッシュ・レート] PG4UWMCプログラムが実行しているプログラマー・サイトからステータス情報を要求する頻度を定義します。ステータス情報には現在のデバイス操作タイプ、進行、結果等を意味します。現在のステータス情報がPG4UWMCのメイン・ウィンドウに表示されます。デフォルトのタイマーのリフレッシュ・レート値は200msです。もし、PG4UWMCの操作パネルに表示されるステータス情報の表示をより早くリフレッシュしたい場合、短いリフレッシュ間隔を選択して下さい。より速いリフレッシュを使用している時に、もし、システムのパフォーマンスのスローダウンに気付いた場合、リフレッシュ頻度が少な

くするためにより高いリフレッシュ値を選択します。Pentium 4コンピューターではタイマーのリフレッシュ・レートに依存するパフォーマンスの低下は殆どありませんが、遅いコンピューターでは時々長い(より少ない)タイマー間隔を選択することは有用です。

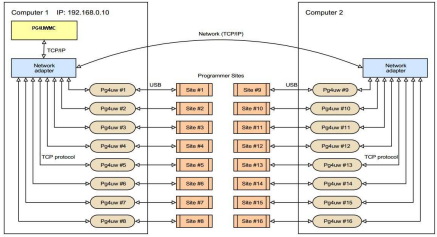
PG4UWMC "Search for Programmers[プログラマーのサーチ]"

ローカル・コンピューター上をサーチ
プログラマー検索のこのモードはデフォルトでPG4UWMCのインストール後にアクティブです。もし、ネットワーク経由で別のコンピュータに接続されたプログラマーで操作したい場合はネットワーク・モードを試してみて下さい。

下のスクリーン・ショットで赤色のプログラマーは存在することが期待されているいくつかのサイトがあることを示していますが見つけることが出来ない状態を示しています。これらのサイトは"Not found"列にリストされています。そうでない場合は列が非表示になります。



ネットワーク上の定義されたプログラマー・グループでサーチ[Search in defined Programmers group on network.]
PG4UWMC はネットワークモードに切り替えるとネットワークコンピュータ上のPG4UW を検索、開始、制御、及び、監視することができます。PG4UWMC とPG4UW 間の通信は各コンピュータ上で動作する PG4UWMC Network Agent を介して実現されます。ネットワーク上のすべての PG4UW、PG4UWMC ネットワークエージェントと PG4UWMC の制御は同じバージョンでなければなりません。この機能は自動プログラマにのみ使用でき、主にハンドラ・マシンで使用することを意図しています。

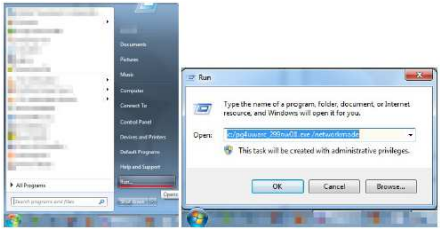


2 台のコンピュータ上で実行される遠隔制御されたマルチプログラミングシステムの典型的な構成

インストレーション

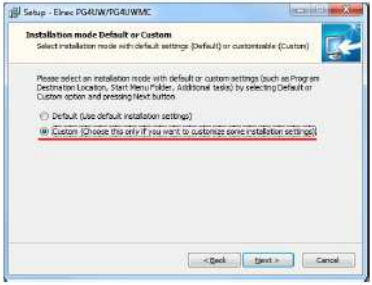
インストール中、ネットワークモード機能はデフォルトではインストールされません。コマンド-ライン・パラメータ/networkmode を使用してインストール手順を実行することでアクティベートする必要があります(例：Start/Run/C:\pg4uwarc.exe/networkmode)。

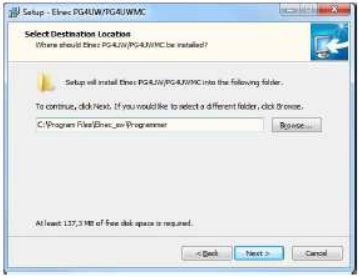
いくつかの初期画面の後、PG4UWMC Network Agent のインストールと Programmer Group の選択を含めるオプションが表示されます。このインストールされたコンピュータが属する Programmers グループの名前を定義してください。PG4UWMC Network Agent は Windows で始まるように設定されます。



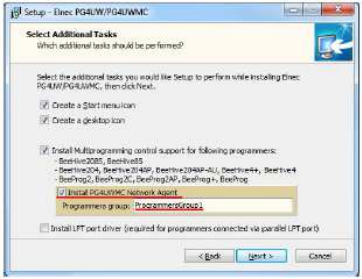
コマンド-ライン・パラメータ /networkmode によるインストール手順


Setup - Elnec PG4UW/PG4UWMC
PROGRAMMING HAS NEVER BEEN EASIER
ELNEC
Welcome to Elnec programmers software setup wizard
Setup will install PG4UW/PG4UWMC universal control program for Elnec programmers, version 2.99rw(x)/08.2022.
It is recommended that you close all other applications before continuing. In case of unusual behaviour during the installation please check your antivirus/firewall configuration especially custom rules for files copying.
Note for programmers with USB connection:
Please, disconnect the programmer's USB cable until installation of the Elnec PG4UW/PG4UWMC software has been completed.
Click Next to continue, or Cancel to exit Setup.
Next >
Cancel


Setup - Elnec PG4UW/PG4UWMC
Installation mode Default or Custom
Select installation mode with default settings (Default) or customizable (Custom)
Please select an installation mode with default or custom settings (such as Program Destination Location, Start Menu Folder, Additional tasks) by selecting Default or Custom option and pressing Next button
Default (Use default installation settings)
Custom (Choose this only if you want to customize some installation settings)
< Back
Next >
Cancel


Setup - Elnec PG4UW/PG4UWMC
Select Destination Location
Where should Elnec PG4UW/PG4UWMC be installed?
Setup will install Elnec PG4UW/PG4UWMC into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\Elnec_sw\Programmer
Browse...
At least 137,3 MB of free disk space is required.
< Back
Next >
Cancel

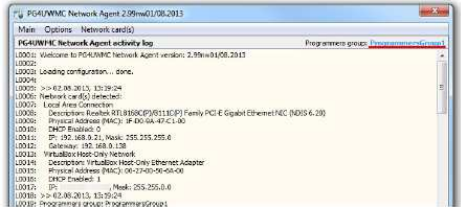
インストール手順 - カスタマイズ



PG4UWMC Network Agent の Installationと選択された Programmers group 名がチェックされたインストール手順

この方法で Programmers group で動作すると考えられるネットワーク上の各コンピュータに PG4UW をインストールする必要があります。

Programmers group 内の各コンピュータは PG4UWMC Network Agent をバック・グラウンドで実行している必要があります。PG4UWMC Network Agent がインストール後に実行されていない場合は、Start menu / All Programs...から実行してください。



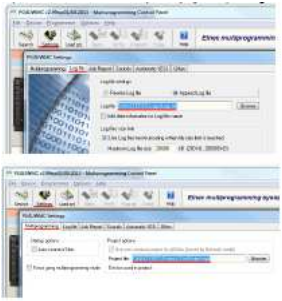
PG4UWMC Network AgentのInstallationと選択されたProgrammers group名がチェックされたインストール手順

各コンピュータでインストールが完了したら、PG4UWMCの初期設定に進むことができます。

コンフィギュレーション
プログラミング・プロセス全体を制御するコンピュータでPG4UWMCを実行します。
Menu /OptionsでNetwork modeをチェックします。

PG4UWMC Network AgentのInstallationと選択されたProgrammers group名がチェックされたインストール手順

ネットワーク上にいますで、ネットワーク・パスをプロジェクト・ファイルとログ・ファイルに設定する必要があります。



ネットワークからPG4UWMCプロジェクト読み込みの設定、ネットワーク・パスへのログの保存

ここで定義されたProgrammers groupのネットワーク上で最初に検索を実行することができます。
- プログラマーを探す
- 見つかったものを評価する
- レジェンドをチェックする(何をすべきかを知るためにはヘルプを見て下さい)
- 制限を満たすために問題を解決する

- 望むようにプログラマーを有効、無効、移動、削除
- 変更を適用するかキャンセル

ネットワーク上のProgrammers group をサーチ

この時点から、PG4UWMCを使用する作業はいつものように行う必要があります。

トラブル・シューティング
検索プログラマーが期待どおりに終了しない場合は、以下を確認してください:
· Programmers group内の各コンピュータはPG4UWMC Network Agentと同じProgrammers groupを実行する必要があります。
· ファイアウォール設定がネットワーク通信を妨げている可能性があります。ファイアウォール・ルールをチェックするか、又は、一時的にファイアウォールを無効にして下さい(自己責任で注意して行って下さい)。

Command line parameters[コマンド・ライン・パラメータ]
プログラム PG4UWMCは以下のコマンド・ライン・パラメータをサポートしています:

/prj:<file_name>
プロジェクト・ファイルをロードします。パラメータ<file_name>は完全、又は、相対プロジェクト・ファイルのパスと名前を意味します。プリフィックス /prj...を付けずにプロジェクト・ファイル名を入力することで、コマンドラインからプロジェクトをロードすることもできます。

サンプル:
pg4uwmc.exe c:\projects\myproject.eprj
Makes load project file "c:\projects\myproject.eprj".
/autoconnectsites
コマンドはPG4UWMCが最近閉じられた時に使用された全のサイトに対してPG4UWMCの起動時にPG4UWMCにプログラマー・サイト(各サイトの制御プログラムPG4UWを開始)を強制的に接続させます。また、PG4UWMCの

"Settings"ダイアログで利用出来る"Auto-connect Sites"の同等のオプションもあります。.

PG4UWMC によりサポートされているプログラマ

現在サポートされているプログラマーのリストはメニューのヘルプ/サポートされているプログラマーによってPG4UWMCに表示されます。一般的に、PG4UWMCのサポートされているプログラマはUSBインタフェースを備えた48ピン汎用プログラマです。また、全てのUSB接続マルチ・プログラミング・システムがサポートされています。PG4UWMCは1から8のプログラマー・サイトを扱うことが出来ます。1つのプログラマー・サイトは1つのZIFソケットモジュールを意味します。

トラブルシューティング

シリアル・ナンバー

追加プログラマの使用を成功させるためにはパネルのシリアル番号で使用される各プログラマの正しいシリアル番号を指定する必要があります。シリアル番号のフィールドが空白の場合、プログラマー・サイトに対してアプリケーションPG4UWは起動しません。PG4UWMCアプリケーションが"Search for programmers"ダイアログで接続されているプログラマを検索している時、プログラマーのシリアル番号が自動的に検出されます。ユーザーはシリアル番号を自分で指定する必要はありません(また指定することも出来ません)。

プログラマーをサーチ中の通信エラー

何らかの通信エラーが発生した場合は、全てのPG4UWアプリケーションとPG4UWMCを終了し、そして、PG4UWMCを起動し"Connect programmers"ボタンをクリックして各サイトのPG4UWアプリケーションを起動しプログラマを接続してください。

全てのプログラマーが正しく接続されているが動作が不安定

プログラマとの通信がデバイス操作(例えば、デバイス・プログラミング)中に無作為に失われた場合、他のプログラム、特に大量のシステム・リソースを消費するプログラム(マルチメディア、CAD、グラフィック・アプリケーション等)を終了してください。

ノート: また、コンピュータの背面にあるコンピュータのUSBポートを使用し、マザーボードに直接接続することをお勧めします。ケーブルを介して間接的にコンピュータのマザーボードに接続されたコンピュータのUSBポートは高速USB 2.0転送モードを使用する際には信頼できない可能性があります。この推奨はプログラマだけでなく他のデバイスに対しても有効です。

Common notes – 共通ノート

メンテナンス

プログラマの高い信頼性を長期間維持するためにここでの説明と注意事項に従うことをお勧めします。

プログラマのメンテナンスは使用者の姿勢とその使用量に依存します。何れにしましても、以下の推奨事項が一般的に受け入れらて実行されるべきです:

- プログラマを粉塵の多い場所で使用したり保管しないで下さい。
- 湿度はZIFソケットの塵や埃の沈降を促進しますので乾燥した場所で使用して下さい。
- 使用後は付属のダストカバー付きのプログラマのZIFソケットをカバーして下さい。
- プログラマは直射日光や熱源の近くに置かないようにして下さい。ファンが故障した場合は重要なダメージに繋がります。.

集中的な日常使用 (プログラミング・センタ、生産現場)

日々のメンテナンス

プログラマとソケット・コンバーターのZIFソケットの状態と装着状態を確認して下さい。清潔で乾燥した圧縮空気でZIFソケットやアダプタやモジュールからゴミや埃を取り除きます。閉じた状態と開いた状態の両方でZIFソケットを清掃します。

週毎のメンテナンス

全てのプログラマ、又は、プログラミング・モジュールに対してセルフテストを実行します。

四半期毎のメンテナンス

柔らかい布の上でイソプロピル・アルコール(*注意)、又は、工業用アルコールでプログラマの表面を静かに清掃します。LCDは柔らかい布で拭いて水だけで湿らせます。*イソプロピル・アルコールはLCDの表面を損傷する可能性があります。プログラマがこの機能をサポートしている場合は較正テストを実行します。

日々の使用 (開発研修、オフィス)

作業が終了したらカバー付きのプログラマのZIFソケットをカバーします。また、アダプタ(又は、モジュールのZIFソケットを埃や汚れから保護することをお勧めします。

週毎のメンテナンス

プログラマとソケット・コンバーターのZIFソケットの状態と装着状態を確認して下さい。清潔で乾燥した圧縮空気でZIFソケットからゴミや埃を取り除きます。閉じた状態と開いた状態の両方でZIFソケットを清掃します。

四半期毎のメンテナンス

プログラマ、又は、プログラミング・モジュールに対してセルフテストを実行します。

半年毎のメンテナンス

柔らかい布の上でイソプロピル・アルコール(*注意)、又は、工業用アルコールでプログラマの表面を静かに清掃します。LCDは柔らかい布で拭いて水だけで湿らせます。*イソプロピル・アルコールはLCDの表面を損傷する可能性があります。プログラマがこの機能をサポートしている場合は較正テストを実行します。

時折の使用

日々のメンテナンス

作業が終了したらカバー付きのプログラマのZIFソケットをカバーします。また、アダプタ(又は、モジュールのZIFソケットを埃や汚れから保護することをお勧めします。

四半期毎のメンテナンス

プログラマとソケット・コンバーター(アダプタ)やモジュールのZIFソケットの状態と装着状態を確認して下さい。清潔で乾燥した圧縮空気でZIFソケットからゴミや埃を取り除きます。閉じた状態と開いた状態の両方でZIFソケットを清掃します。

半年毎のメンテナンス

プログラマ、又は、プログラミング・モジュールに対してセルフテストを実行します。

年間でのメンテナンス

柔らかい布の上でイソプロピル・アルコール(*注意)、又は、工業用アルコールでプログラマの表面を静かに清掃します。LCDは柔らかい布で拭いて水だけで湿らせます。*イソプロピル・アルコールはLCDの表面を損傷する可能性があります。プログラマがこの機能をサポートしている場合は較正テストを実行します。

警告:

プログラマのZIFソケットとソケット・コンバータ(アダプタ)は消耗品とみなされます。プログラマのZIFソケットのライフサイクルは約25,000機械サイクルです。幾つかの特別なBGA ZIFソケットのライフ・サイクルが約500,000の機械的サイクルであっても、ソケット・コンバータ(アダプタ)のZIFソケットのライフサイクルは一般的に5,000～10,000までの機械サイクルです。プログラムされたデバイス、環境、及び、ZIFソケットのメンテナンスは、ZIFソケットの実際の電気的寿命に直接影響を与えます(ZIFソケットがまだプログラミングの失敗を引き起こさないことを意味します)。ZIFソケットの接触部が指で汚れたりするとプログラミングが失敗する可能性があるためZIFソケットの接触部から指を離して下さい。プログラミングの失敗回数が増えていることに気づいた場合はZIFソケット、又は、アダプタを交換して見て下さい。

摩耗や汚れがありプログラマとの作業中に大量の故障を引き起こすZIFソケットには保証は適用されません。

Software[ソフトウエア]

PG4UWはELNECの全てのプログラマにとって共通の制御プログラムです。PG4UWarc-OnDemand.exeをダウンロードしてご使用下さい。

一部の特別なデバイス(Philips CoolRunnerファミリ等)では標準のPG4UW SW配信CDには存在しない外部DATファイルが必要です。これらのデバイスをプログラムする必要がある場合はwww.elnec.comのダウンロードからダウンロードして下さい。

コマンド・ライン・パラメータ

特別なユーティリティpg4uwcmd.exeを使用してPG4UWのコマンドライン・パラメータ制御を行うことを推奨します。下位互換性のためにpg4uw.exeで直接コ

マンドライン・パラメータを使用することも可能ですが、より多くのコマンドライン・コマンドをサポートするpg4uwcmd.exeを使用しますと、ExitCode(又は、ErrorLevel)値を返す機能も備えていてコマンドライン・パラメータの実行の成功、又は、エラーの結果を示します。PG4UWのコマンド・ライン・コントロールの使用についての詳しい情報はPG4UWがインストールされたディレクトリの "remotectrl"フォルダのremotemanual.pdf[英文]の6. Remote command line control of PG4UWをご覧下さい。

pg4uw.exeで直接使用できるコマンドライン・パラメータ

/Prj:<file_name> プログラムの起動時、又は、プログラムがすでに実行中であってもプロジェクトのロードを強制的に実行します。<file_name>は完全、又は、相対プロジェクト・ファイルのパスと名前を意味します。

/Loadfile:<file_name> プログラムの起動時にファイルのロードを強制するか、プログラムであっても<file_name>はロードする必要があるファイルへの絶対パス、又は、相対パスを意味しファイル・フォーマットは自動的に検出されます。

/Saveproject:<file_name> このコマンドは現在選択されているデバイス・タイプ、バッファ内容、及び設定をプロジェクト・ファイルに保存するために使用されます。コマンド /Saveproject:.はユーザーが選択したコマンドと同等です。PG4UW制御プログラムでプロジェクトを保存します。

ファイル名はWindowsの規則を満たさなければならないことに注意して下さい。つまり、ファイル名にスペースが含まれている場合、コマンドライン・パラメータのファイル名は引用符で囲む必要があります。

.

サンプル:

/prj:c:\myfile.eprj 名前がc:\myfile.eprjのプロジェクト・ファイルをロード

/loadfile:"c:\filename with spaces.bin"
バッファに"c:\filename with spaces.bin"ファイルをロードします。

/Program[:switch] プログラムの起動時に自動的に"Program device[プログラム・デバイス]"操作の開始を強制するか、又は、プログラムがすでに実行中であっても次のオプション・スイッチの1つを使用することができます:

switch 'noquest' 問題なくデバイス・プログラミングを開始します。

switch 'noanyquest' 問題なくデバイス・プログラミングを開始し、デバイス上の操作が完了した後、プログラムは"Repeat"操作ダイアログを表示せずメイン・プログラム・ウィンドウに直接行きます。

サンプル:

/Program

/Program:noquest
/Program:noanyquest

/Close　　このパラメータは/Programパラメータのみと共に意味を持ち、デバイス・プログラミングが正常に終了したら自動的にプログラムを閉じるようにします。

/Close: always　　このパラメータは/Programパラメータのみと共に意味を持ちデバイスの操作が成功したかどうかに関わらずデバイスのプログラミングが終了した後に自動的にプログラムを終了させます。

/Eprom_Flash_Autoselect[:xx]
プログラムが起動している時、又は、プログラムがすでに実行中であっても自動的にEPROM、又は、FLASHをIDで強制的に選択します。xxはZIFのデバイスのピン数を表します(有効な28ピン、又は、32ピンのみ)。これは挿入テスト機能のない古いプログラマにのみ必要です。他のプログラマではこの値は無視されます。

エグゼクティブ・コマンドライン・パラメータを使用するための基本的なルール:

1. コマンドライン・パラメータは大文字と小文字が区別されません。
2. コマンドライン・パラメータはプログラムの最初の起動時、又は、プログラムがすでに実行中の時に使用できます。
3. プログラムがすでに実行されている場合、プログラムがビジーでない(プログラムで操作が現在実行されていない)場合にのみ、いずれかのコマンドライン操作が処理されます。プログラムは基本的な状態でなければいけません、即ち、メイン・プログラム・ウィンドウがフォーカスされ、モーダル・ダイアログが表示されず、メニュー・コマンドがオープンされていないか、又は、実行されていない状態でなければいけません。
4. 複数のパラメータを同時に使用する時コマンドライン・パラメータの処理順序は次のようにしっかりと定義されます:
1. プロジェクトのロード (/Prj:...)
2. ファイルのロード (/Load file:...)
3. IDによるEPROM/Flash選択
4. プログラム・デバイス (/Program[:switch])
5. コントロール・プログラムを閉じる (/Close only together with parameter /Program)

利用可能なコマンドライン・パラメータ:

/Axxx　　アドレスxxxのみのLPTポートにあるプログラマをチェック
サンプル: /A3bc

/SPP　　PC <-> プログラマの単方向モードでの通信を強制

デモ・モードでプログラムPG4UWを起動するために使用できるコマンドライン・パラメータ

デモモードは使用可能なプログラマ・デバイスがない場合に便利です。デモモードはプログラマの検索ダイアログでデモ・ボタンをクリックするかコマンドラインパラメータ/demoで使用できます。パラメータの推奨使用方法は次の通りです:

pg4uw.exe /demo /<programmer name>
<programmer name>はPG4UW制御プログラムで使用されている希望のプログラマ名に置き換える必要があります。

PG4UW のリモート・コマンドライン制御

PG4UWはコマンドライン(コマンドライン・パラメータ)からコマンドのセットを受け入れることができます。リモート・コントロールはこれらのコマンドライン・パラメータでも実現できますが、より効率的な方法は特別なツールpg4uwcmd.exeを使用することです。これは多くの利点があります。主な利点はpg4uwcmdのサイズでpg4uwcmdを呼び出すとPG4UWを直接呼び出すよりもはるかに高速な応答が得られます。

プログラムpg4uwcmd.exeを以下に使用出来ます:
1. 指定されたコマンドライン・パラメータでPG4UWアプリケーションを起動する
2. 既に実行中のPG4UWにコマンドライン・パラメータを強制的に適用する

pg4uwcmd.exeの非常に良い機能はPG4UWのコマンドライン・パラメータの操作結果に応じたそのリターンコードです。

pg4uwcmd.exeの戻り値
PG4UWで処理されたコマンドライン・パラメータが成功した場合、pg4uwcmd.exeのExitCode(又は、ErrorLevel)はゼロです。それ以外の場合ExitCode値は1以上です。プログラムpg4uwcmd.exeの戻り値はバッチ・ファイルでテスト出来ます。

pg4uwcmd.exeで使用できるエグゼクティブ・コマンドライン・パラメータは次の通りです。

/Prj:<file_name> プロジェクト・ファイルを読み込みます。パラメータ<file_name>は完全、又は、相対プロジェクト・ファイルのパスと名前を意味します。

/Loadfile:<file_name> ファイルを読み込みます。パラメータ<file_name>はロードする必要があるファイルへの絶対パス、又は、相対パスを示します。ファイル形式は自動的に検出されます。

/Program[:switch] プログラムの起動時に自動的に"プログラム・デバイス"の動作を強制的に開始します。また、次のオプション・スイッチのいずれかを使用できます:

switch 'noquest' 問題なくデバイス・プログラミングを開始

switch 'noanyquest' 問題なくデバイス・プログラミングを開始し、デバイス上の操作が完了した後、プログラムは"Repeat"操

作ダイアログを表示せずメイン・プログラム・ウィンドウに直接入ります。

サンプル:
/Program
/Program:noquest
/Program:noanyquest

/Close このパラメータは /Programパラメータのみと共に意味を持ち、デバイスのプログラミングが終了した後(操作が成功したか否かにかかわらず)プログラムPG4UWを自動的に閉じます。

/Saveproject:<file_name>
このコマンドは現在選択されているデバイス・タイプ、バッファの内容とコンフィギュレーションをプロジェクト・ファイルに保存する ために使用されます。Command / Saveproject ...はPG4UW制御プログラムでユーザーが選択したコマンドSave projectと同等です。

/Eprom_Flash_Autoselect[:xx]
プログラマのZIFソケットに現在挿入されているチップから電子IDを読み取ることによりEPROM、又は、FLASHのタイプを自動選択します。オプションのパラメータxxはZIF(有効ピン数は28pin、又は、32pinのみ)のデバイスのピン数を示し、そして、挿入テスト機能のない古いプログラマにのみ必要です。他のプログラマにとっては無視されるためxxパラメーターは省略することが出来ます。

サンプル:
/Eprom_Flash_Autoselect
/Eprom_Flash_Autoselect:32

/writebuffer:ADDR1:B11,B12,B13,B14,...,B1N[::ADDR2:B21,B22,B23,B24,...,B2M]...
コマンド /writebufferはPG4UWメイン・アドレスの指定されたアドレスにByteのブロックを書き込むために使用されます。Write bufferコマンドは必要な1ブロックのデータとオプショナルな他のブロック(複数可)のデータ([...]でマークされた)を持っています。コマンドにスペースやタブを使用しないでください。バッファ・アドレスは常にバイト・アドレスとして定義されます。つまり、バッファ構成x16の場合、バッファ内のアドレスAAAAx16はコマンド /writebufferで2*AAAAx(8)として指定する必要があります。

サンプル 1:
/writebuffer:7FF800:12,AB,C5,D4,7E,80
アドレス7FF800Hのバッファに6バイトの12H ABH C5H D4H 7EH 80Hを書き込みます。
アドレッシングは次の様に見えます:
最下位アドレスの最初のバイト
バッファ・アドレス・データ

サンプル 1:
/writebuffer:7FF800:12,AB,C5,D4,7E,80

アドレス7FF800Hのバッファに6バイトの12H ABH C5H D4H 7EH 80Hを書き込みます。
アドレッシングは次のようになります:
最下位アドレスの最初のバイト
バッファ・アドレス データ
7FF800H 12H
7FF801H ABH
7FF802H C5H
7FF803H D4H
7FF804H 7EH
7FF805H 80H

サンプル 2:
/writebuffer:7FF800:12,AB,C5,D4,7E,80::FF0000:AB,CD,EF,43,21
_____/ _____/
データの最初のブロック データの2番目のブロック
データ2つのブロックをバッファに書き込みます。
データの最初のブロック - 6バイト12H ABH C5H D4H 7EH 80Hはサンプル1と同じ方法でアドレス7FF800Hのバッファに書き込まれます。

データの第2ブロック - 5バイト ABH CDH EFH 43H 21Hはアドレス FF0000Hのバッファに書き込まれます。
アドレッシングは次のようになります:
最下位アドレスの最初のバイト
バッファ・アドレス データ
FF0000H ABH
FF0001H CDH
FF0002H EFH
FF0003H 43H
FF0004H 21H

/writebufferex:INDEX:ADDR1:B11,B12,B13,B14,...,B1N[::ADDR2:B21,B22,B23,B24,...,B2M]..

コマンド /writebufferexは指定されたアドレスにバイトのブロックをPG4UWのメイン・バッファへの書き込みに使用されます。このコマンドはもう1つのパラメータ - INDEXを除いて、コマンド /writebufferと非常によく似ています。INDEXパラメータはデータが送信されるバッファの順序を指定します。メイン・バッファのインデックスは '1'です。最初のニ次バッファはインデックス'2'、等を持っています。セコンダリー・バッファはいくつかの種類のデバイスのみ(例 : Microchip PIC16F628)のみ利用可能なことに注意して下さい。パラメータ buffindexによってインデックスされるバッファの種類はアプリケーションPG4UWのダイアログView/Editバッファ内のバッファの順序に依存します。例えば、デバイスMicrochip PIC16F628にはラベル "Data EEPROM"付きの追加バッファがあります。このバッファはbuffindex = 2が指定されている場合、この関数によってデータ書き込みのためにアクセスできます。

サンプル 1:
/writebufferex:1:7FF800:12,AB,C5,D4,7E,80
そのコマンドは以下のコマンドと同等です。
/writebuffer:1:7FF800:12,AB,C5,D4,7E,80
コマンド /writebuffer.についてのセクションで説明

サンプル 2:
/writebufferex:2:2F:12,AB,C5,D4,7E,80
このコマンドはアドレス2FHでのインデックス"2"を持つ2次バッファに6バイト 12H ABH C5H D4H 7EH 80Hを書き込みます。
アドレッシングは次のようになります:
最下位アドレスの最初のバイト
バッファ・アドレス　データ
00002FH　12H
000030H　ABH
000031H　C5H
000032H　D4H
000033H　7EH
000034H　80H

エグゼクティブ・コマンドライン・パラメータを使用するための基本的なルール:
1. プログラム pg4uwcmd.exeはプログラムpg4uw.exeと同じディレクトリになければなりません。
2. pg4uwcmd.exeが呼び出された時にpg4uw.exeが実行されていない場合、自動的に起動します。
3. コマンドライン・パラメータでは大文字と小文字は区別されません。
4. コマンドライン・パラメータはプログラムの最初の起動時、又は、プログラムがすでに実行中のときに使用できます。
5. プログラムがすでに実行中の場合、プログラムがビジーでない時(プログラムで現在実行中の操作がない)にのみコマンド行操作のいずれかが処理されます。プログラムはベーシックな状態でなければいけません。即ち、メイン・プログラム・ウィンドウがフォーカスされモーダルダイアログが表示されずメニュー・コマンドが開かれたり実行されたりすることがない状態。
6.より多くのパラメータを一緒に使用する場合のコマンドライン・パラメータの処理順序は次のようにしっかりと定義されています。
　step1 ロード・ファイル(/ Loadfile:...)
　step2 プロジェクトをロード(/ Prj:...)
　step3 EPROM/FLASH自動選択
　step4 プログラム・デバイス(/Program[:switch])
　step5 制御プログラムの終了(パラメータ /Programと一緒にのみ /Close)

サンプル 1:
pg4uwcmd.exe /program:noanyquest /loadfile:c:\empfile.hex
以下の操作が実行されます:
1. pg4uw.exeを起動します(まだ実行していない場合)
2. ファイル c:\empfile.hexをロード

3. プログラムデバイスの動作を開始
4. pg4uwcmd.exeはまだ実行されており、定期的にpg4uw.exeの状態を確認しています。
5. デバイス・プログラミングが完了すると、pg4uwcmd.exeが閉じられ、ロードファイルとpg4uw.exeのデバイス・プログラミング結果に応じてExitCodeが返されます。全ての操作が成功すると、pg4uwcmd.exeは0を返し、それ以外の場合は値1、又は、それ以上を返します。

サンプル 2:
pg4uwcmd.exe /program:noanyquest /prj:c:\emproject.eprj
操作はサンプル1と同じです; Load file操作はLoad project file: \emproject.eprjコマンドに置き換えられます。

サンプル 3:
pg4uwcmd.exeを使ってバッチ・ファイルでpg4uwcmd.exeの戻りコードをテストします。

```
rem ------- beginning of batch -------------------------
@echo off
rem Call application with wished parameters
pg4uwcmd.exe /program:noanyquest /prj:c:\emproject.eprj
rem Detect result of command line execution
rem Variable ErrorLevel is tested, value 1 or greater means the error occurred
if ErrorLevel 1 goto FAILURE
echo Command line operation was successful
goto BATCHEND
:FAILURE
echo Command line operation error(s)
:BATCHEND
echo.
echo This is end of batch file (or continue)
pause
rem ------- end of batch -------------------------
```

サンプル 4:
PG4UW制御プログラムが動作していてユーザが選択したデバイスを持っていると仮定します。データをPG4UWデバイス・バッファにロードし、選択したデバイスの設定とバッファの内容をプロジェクト・ファイルに保存する必要があります。デバイスに必要なデータはファイル c:\15001-25001\file_10.binに保存されています。プロジェクト・ファイルはc:\projects\project_10.eprjに保存されます。
希望の動作を実現するには次のコマンドライン・パラメータを指定する必要があります:

```
pg4uwcmd.exe /loadfile:c:\15001-25001\file_10.bin
/saveproject:c:\projects\project_10.eprj
```

PG4UWがコマンドを受信すると、次の手順を実行します:
1. データ・ファイル c:\15001-25001\file_10.bin をロード
2. 現在選択されているデバイス設定を保存し、データをプロジェクト・ファイルにバッファリング

c:\projects\project_10.eprj
実行された操作の結果がOKの場合、PG4UWcmdアプリケーションは ExitCode(又は、ErrorLevel)値0を返します。
エラーがある場合(ファイルをロード出来ない、又は、プロジェクト・ファイルに保存できない場合)、PG4UWcmdアプリケーションはExitCode値同等か、又は、1以上のExitCode値を返します。

ノート : 上記のコマンドを使用する場合、PG4UWがデバイスの操作、例えば、デバイス・プログラミング等を実行していないことを確認する必要があります。
PG4UWがビジー状態の場合、コマンドを拒否し、エラー・ステータスを返します(ExitCode同等、又は、値1以上)。

ハードウエア[LPT ポート]

多種多様なパラレル・ポートタイプのためプログラマがPCと"協調する"ことができない場合があります。この問題はPCとプログラマの間の通信が不完全であるか、又は、信頼性の低い通信によって示される場合があります。この現象が発生した場合は、プログラマを他のPCやその他のパラレル・ポートに接続してみて下さい。解決策が見つからない場合は、その状況を文書化して下さい、即ち、PCコンフィグレーションの正確な情報を下さい。具体的には問題のPCのメーカーと機種を連絡して下さい。PCの種類、メーカー、スピード、操作システム、常駐プログラム等の情報を下さい。パラレルポートI/Oの製造元とタイプ。この目的のためにはデバイスの問題報告フォームを使用してください。

警告:
Class A ITE notice
このマニュアルに記載されているデバイスはクラスAの製品です。家庭環境ではこの製品は電波干渉を引き起こす可能性があり、その場合、ユーザーは適切な処置を講ずる必要があります。

BeeHive204、BeeProg2、及び、BeeProg2Cは内部電源を備えているためこれらの特別な注意事項に従ってください。
- *サーキットブレーカ(過電流保護)は電気設備を構築するための一環でなければいけません。*
- *電源コードのプラグをコンセントから引き抜いてプログラマを電源から切り離します。コンセントはプログラマの近くに配置し容易にアクセス可能でなければいけません。*

ISP (イン-システム・プログラミング)

定義

イン-システム・プログラミングではエンドシステム内部に配置されたデバイスのプログラミングと再プログラミングが可能です。シンプルなインターフェースを使用してISPプログラマはデバイスとシリアル通信しチップ上の不揮発性メモリを再プログラミングします。イン-システム・プログラミングはシステムからのチップの物理的除去を省け

ます。これによりラボでの開発中、及び、フィールド内のソフトウェア、又は、パラメータを更新する際に時間とコストを節約できます。

ターゲット・デバイスはシステム内にあるデバイス(マイクロコントローラ、PLD等)です。

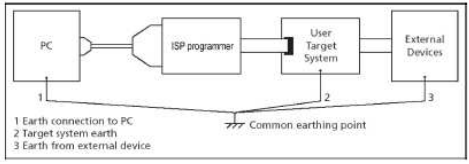
ターゲット・システムは物理的なプリント回路基板(PCB)であり、イン-システムでプログラムされるデバイスを含みます。

ISPプログラマはプログラマでありイン-システム・プログラミング能力を有する(例えば、BeeProg2、BeeProg2C、SmartProg2 ...等)です。

イン-システム・プログラミングの一般規則

PC、ISPプログラマ、ターゲット・デバイス、又は、ターゲット・システムの損傷を避けるために下記のルールを守ることをお薦めします:

- ターゲット・システム、ISPプログラマ、PCの共通アースポイントを確保
- ラップトップ、又は、一般的なアースポイントに接続されていない他のPCの場合:ラップトップから一般的なアースポイントにハードワイヤード接続します(例えば、VGAコネクタを使用).
- ターゲット・システムに接続されているデバイスは全て共通接地点に接続する必要があります。



Elnec ISP プログラマをターゲット・システムに接続する方向:

イン-システム・プログラミング中にISPプログラマとターゲット・システムの2つの電気装置を接続します。非適合な接続はこれらのデバイスを損傷する可能性があります。

ノート: *指示に従わないでシステム内プログラミング中にプログラマに損害を与えた場合、それは無条件操作によるプログラマの損傷であり保証対象外です。.*

1. ISPプログラマとターゲット・デバイスの両方のデバイスをオフにします。
2. 全てのデバイスに同じGND電位を割り当てます。例えば、全てのデバイスのGNDをワイヤで接続して下さい。
3. ISPプログラマにISPケーブルの1本のコネクタを差し込み、プログラマをオンにしてプログラムを制御します。
4. 制御プログラムでターゲット・デバイスと操作オプションを選択します。
5. ターゲット・デバイス上でアクションを開始します(読み込み、プログラム)。
6. f制御プログラムの指示の後、他のISPケーブル・コネクタをターゲット・システムに接続し、それをオンにします。
7. 制御プログラムの指示の後、他のISPケーブル・コネクタをターゲット・システムから切り離し電源を切ります。
8. ターゲット・デバイスで別のアクションが必要な場合はステップ5に進みます。.

ISP でプログラムされたデバイスを使用したターゲット・システムの設計に関する推奨事項

ターゲット・システムはイン-システム・プログラミングに使用する全ての信号がISPコネクタを介してISPプログラマに直接接続できるように設計する必要があります。ターゲット・システムがこれらの信号を他の機能に使用する場合は、これらの信号を分離する必要があります。ターゲット・システムはイン-システム・プログラミング中にこれらの信号に影響してはいけません。

イン-システム・プログラマブル・デバイスの場合、メーカーはアプリケーション・ノートを発行します。これらのアプリケーション・ノートを考慮したElnecプログラマの設計は、適切なインシステム・プログラミングを可能にします。条件はこれらのアプリケーション・ノートを厳密に尊重しています。ElnecがISPプログラマで使用するアプリケーション・ノートはwww.elnec.comのセクション **Supportt / Application Notes**.

推奨される回線に従うための注意をお読みください。

• D1 ダイオードの目的はISP プログラマによって提供されるより高い電圧からターゲット回路を保護することです。

• ターゲットボードの電源が上記の5Vと異なる場合は、この電源電圧に応じてツェナー・ダイオード(D1) 電圧を選択してください。

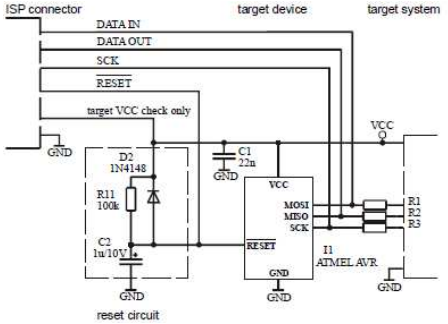
• ターゲット・デバイスとターゲット・システムを分離するために抵抗R1, R2, (R3) を使用することを推奨します。ISP プログラミングに必要なピンがターゲット・システムの入力である場合、抵抗による分離で十分であり抵抗でもローパス・フィルタが可能です。ピンが出力の場合、抵抗を使用するとプログラミング時間が節約されます。 勿論、分離抵抗器R1, R2, (R3)は必要に応じてスイッチ、又は、ジャンパで置き換えることが出来ます。この場合、ターゲット・デバイスのISP プログラミング中にスイッチ(ジャンパー)がオープンしていなければいけません。しかし、スイッチ(ジャンパー)を使用すると次の操作時間がプログラミング手順に追加されます。

アプリケーションノートの例

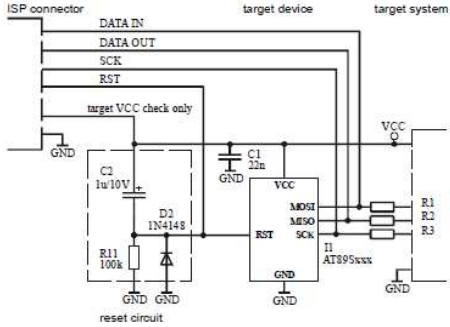
マイクロコントローラAtmel AVR。及び、AT89Sxxxシリーズ

このインタフェースはAtmelアプリケーション・ノートAVR910：イン-システム・プログラミングに対応しています。このアプリケーション・ノートではターゲット・システムの推奨ISPインターフェイス・コネクタ・レイアウトについて説明します。(トップ・ビュー).

ATMEL AVR対応のエルネック推奨回路:



AT89Sxxx対応のエルネック推奨回路:



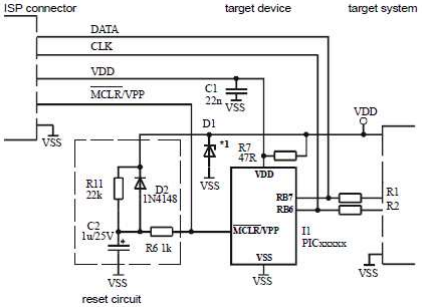
PICmicro® マイクロコントローラ

このインタフェースはマイクロチップのアプリケーション・ノートTB013, TB017, TB016: PIC16CXXX OTP(PIC12C5XX OTP)(PIC16F8Xフラッシュ)MCUを使用したICSP™ の実装方法に対応しています。
このアプリケーション・ノートではイン-システム・プログラミング・デバイスとISPプログラマを使用するターゲット・システムの要件について説明します。以下の信号はPICmicro® マイクロ・コントローラのイン システム・プログラミングに使用されます。

- MCLR\ / VPP リセット / プログラミング・モードへの切り替え
- RB6 (GP1) クロック
- RB7 (GP0) データ入力 / 出力
- VDD 電圧供給
- GND グラウンド

PICmicro® デバイスがプログラムされている場合; ピン MCLR\ / VPPは約12Vに駆動されます。
従って、プログラマが提供するこの電圧からターゲット・システムを分離する必要があります。RB6、及び、RB7信号はイン-システム・プログラミング用にPICmicro®によって使用されます。従って、ターゲット・システムはイン・システム・プログラミング中にこれらの信号に影響を与えてはなりません。限界値検証はプログラミング後に使用されます。プログラマは最小、及び、最大電源の両方でプログラム・メモリの内容を確認する必要があるためプログラミング中はPICmicro® のVDDピンをターゲット・システムの残りから絶縁する必要があります。

PICmicro対応のエルネック推奨回路:



ノート: *外部リセット回路が必要なのはVDDパワーアップ勾配が低すぎる場合のみです。*

その他

Elnecプログラマのための制御プログラムの定期的な実行のためには接続されたプログラマがこのプログラマ専用に予約されたプリンタ・ポートが必要です。それ以外の場合は、他のプログラムでそのプリンタ・ポートを同時に使用しないで下さい。PG4UW SWはLPTポート(フルIEEE 1284サポート)の全モードを処理できるためElnecプログラマの接続にLPTポートを設定する必要はありません。BUSY LEDが点灯している間、情報ウィンドウを移動しないで下さい - 監視回路を起動して通信中のPCプログラマ・エラーの場合はプログラマを安全な状態に切り替えることができます。

LPT ポート・ドライバ
パラレルLPTポートを介して接続されたプログラマの場合、制御プログラムには正しくインストールされたLPTポート・ドライバが必要です。LPTポート・ドライバのインストールとアンインストールはプログラムのインストールによって自動的に行われます。通常、ドライバに問題はありませんが、ドライバを正しく初期化できないことがあります。特にWindows NT/2000/XPオペレーティング・システムでLPT1ポートが存在しない場合。LPTポート・ドライバが初期化されていない場合、制御プログラムはシステム内のLPTポートを検出出来ません。
PCMCIA、又は、Express CardのLPTポートを使用する場合は、オペレーティング・システムを起動する前にカードをコンピュータにインストールする必要があります。
LPTドライバはオペレーティング・システムでポートLPT1が存在する必要があります。LPT1がシステムに存在するパラレル・ポートを確認して下さい。

短い説明: オペレーティング・システムに存在するLPTポートを表示する方法:
1. "スタート"メニューをクリック
2. "マイコンピュータ"の項目を右マウス・クリックしメニューの"プロパティ"を選択します。
3. "システム・プロパティ"ダイアログで"ハードウェア"ページを選択し"デバイス・マネージャー"ボタンをクリックします
4. "デバイス・マネージャー"ダイアログで "Ports (Com & LPT) "をダブルクリックすると現在のLPTポートとCOMポートの一覧が表示されます。
少なくとも1つの現在のLPTポートが表示されている必要があります。
1つ、又は、複数のLPTポートが存在しLPT1以外の番号がある場合は、LPTポートの1つをLPT1ポートに変更する必要があります。
以下の手順に従います(ステップ1-4から続きます)
5. ポートのプロパティを表示するために選択されたLPTポートをダブル・クリック
6. "LPTポート・プロパティ"ダイアログで"ポートの設定"ページを選択します。
7. "LPTポート番号"設定でLPTポート番号をLPT1に変更
8. OKボタンをクリックします。
9. オペレーティング・システムを再起動します(システムが再起動を要求しない場合でも、LPTポート・ドライバを正しく初期化するためにシステムを再起動する必要があります)
それで全部です。当社のソフトウェアはLPT接続されたプログラマで適切に動作するはずです。
USB接続のプログラマを使用する場合はLPTポート・ドライバは不要です。

トラブルシューティングと保証

トラブルシューティング

Elnec社はユーザーが弊社製品を快適に使用されることを望んでいますが、それにも拘わらず、問題が発生する可能性があります。
そのような場合は、以下の手順に従って下さい。

- プログラマ、又は、その制御プログラムPG4UWを正しく操作していないかも知れません。.
- 同梱のマニュアルをすべて慎重にお読み下さい。恐らく、直ぐに必要な答えを見つけることが出来ると思います。
- 上手く動作しない場合はプログラマとPG4UWを別のコンピュータにインストールしてみて下さい。システムが他のコンピュータで正常に動作している場合、最初の1台のPCに問題がある可能性がある場合もあります。両方のコンピュータの違いを比較します。
- 社内のより知識の豊富な担当者に聞いて下さい。
- 既にプログラマをインストールした経験を持った人に尋ねて下さい。
- 問題が解消されない場合は、先ず、作業中のLOGをセーブして下さい。その上でプログラマを購入した販売店に連絡するかElnecに直接メールでお問い合わせ下さい。ほとんどの問題は電子メールで解決出来ております。連絡したい場合:
- **E-mail** - インターネットのウェブ・サイト Support -> Problem Reportからフォームを使用してフォームの最後に記載されている指示に従って下さい。*重要: I如何なる場合もLOGファイル(メニューのヘルプからドロップダウンで"プロブレム・レポート作成"をクリックすることでデスクトップ上に作成されます)

プログラマのモデル名、ソフトウェアのバージョン、及び、ターゲット・デバイスに関連すると考えられる全ての情報が必要ですので、上記のLOGと共にお送り下さい。
Eメールで地域の販売店に送付するか、又は、**elnec ドットコムで(迷惑メールにならないようにし)**に送付してください。

- プログラマが不良品であると診断された場合は、お住まいの国のElnecの代理店、又は、Elnecにご相談ください。

パッケージに次のアイテムを慎重に入れてください:Elnec社に送る場合:

- 問題の製品名
- "**DEVICE PROBLEM REPORT**" フォームに入力
- 購入日証明のコピー

全項目なしではプログラマの修理を
受け付けることは出来ません。

ノート: *"DEVICE PROBLEM REPORT" フォームはインターネット (www.elnec.com)* でSupport -> Problem Reportからフォームを使用出来ます。

サポートされていないターゲット・デバイスがある場合

プログラマ用の制御プログラムでサポートされていないターゲット・デバイスを使用する必要がある場合は、次の手順に従って下さい:

- インターネットサイトの最新バージョンのコントロール・プログラムのデバイス・リストを見て下さい(ダウンロード、プログラマに対応するファイルを参照)。新しいターゲ

ット・デバイスが既に日々の最新OnDemandバージョンに含まれている可能性があります。

· 記載のない場合は、インターネットのElnecのウェブサイトのSupport-> AlgOR(New device support)フォームに入力して下さい。

ターゲット・デバイスの詳細なデータシートとサンプルが必要な場合があります。

保証期間

製造元、Elnec s.r.o. Presov, Slovakiaは購入日から3年間(BeeHive204、BeeProg2、BeeProg2C、及び、SmartProg2)のプログラマ、及び、その部品、材料、及び、ワークマンシップの全ての無故障動作を保証します。この保証はプログラマ本体のDIL ZIFソケットで25,000サイクル、又は、他のZIFソケットで10,000サイクルに制限されています)。製品が欠陥品であると診断された場合、Elnec s.r.o. が欠陥部品を無償で修理または交換します。交換、及び/又は、プログラマ全体に使用される部品は、元の保証期間内として保証されます。保証期間内の修理の場合、顧客は購入日を証明しなければいけません。この保証期間はElnec社から直接プログラマを購入するお客様に有効です。Elnecの代理店の保証条件は対象となる国の法律、又は、ディストリビューターの保証ポリシーによって異なる場合があります。

摩耗や機械的損傷を受けた製品には保証は適用されません。同様にElnecの許可を受けていない人物によって開封、改造された製品、又は、誤用、乱用されたり誤ったインストールによって故障した場合には保証は適用されません。不当な修理の場合、材料、サービス時間、貨物の交換費用に応じて請求されます。エルネック、又は、その代理店は不良品を修理、又は、交換するかどうかを決定し保証が適用されるかどうかを判断します。

製造元:

✉: **Elnec s. r. o.**, Jana Bottu 5, SK - 08001 Presov, Slovakia
☎: +42151/77 34 328, 77 31 007, fax 77 32 797
www.elnec.com, e-mail (nospam version): elnec at elnec dot com

Elnecは安定した信頼性の高いハードウェアとソフトウェアの開発に最善を尽くしています。Elnecはハードウェア及び、ソフトウェアに"バグ"、エラー、又は、欠陥がないことを保証するものではありません。エルネックの責任は購入者が支払った契約の正味価値に常に制限されます。

Elnecは以下の責任を負いません:

· 製品の不適切な使用、又は、操作ミスに起因する損失.

· ユーザー、又は、*第三者が製品を変更、又は、改造しようとすることによる損害*.

· ハードウェアのエラー、又は、ソフトウェアの"バグ"によって引き起こされるそれ以上の損害、又は、結果的損害

*バグにはElnecによる意図しない結果を生じるヒューマン・エラー(人為的ミス)、又は、予想出来なかった間違いも含まれます。

例えば: *利益の損失、第三者からのクライアントに対する請求、記録されたデータやファイルの破損、又は、使用不可能による損失 等.*