

## 日本語ユーザーズ・マニュアル

### BeeHive304

超高速ユニバーサル 4x 64ピン・ドライブ生産マルチプログラマ  
大容量メモリ・プログラミングに焦点を当てた設計



### BeeProg3

超高速ユニバーサル 64ピン・ドライブ・プログラマ  
大容量メモリ・プログラミングに焦点を当てた設計



この文書は ELNEC s.r.o., Presov, Slovakia に著作権があります。全著作権所有。本書、又は、その一部は ELNEC s.r.o. の書面による事前の許可なくいかなる形またはいかなる形でも複製、複製、翻訳することはできません。

制御プログラムは ELNEC s.r.o., Presov, Slovakia の著作権です。制御プログラム、又は、その一部はあらゆる目的のためにも、いかなる形式でも、いかなる媒体上でも、分析、解体、又は、変更することはできません。

このマニュアルに記載されている情報はリリース時点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。www.elnec.com のマニュアルを参照してください。

ELNEC s.r.o. とその代理店はこのマニュアルの誤読、誤用には一切責任を負いません。

E ELNEC s.r.o. は本書に記載されている製品を予告なく変更または改善する権利を留保します。このマニュアルには企業、ソフトウェア製品などの名称が含まれており、それぞれの所有者の商標である可能性があります。ELNEC s.r.o. これらの商標を尊重します。

COPYRIGHT © 1991 – 2019

Elnec s.r.o. Presov, Slovakia

1<sup>st</sup> March 2019

ZLI-0330

## 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ムの使用方法について説明しています。ユーザーにはPCやソフトウェ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 プログラム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](http://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。

## Table of contents

このマニュアルの使い方.....	3
<b>イントロダクション.....</b>	<b>6</b>
製品構成.....	7
必要なPCスペック.....	8
ソフトウェア.....	9
フリー・アディショナル・サービス.....	9
<b>クイック・スタート.....</b>	<b>10</b>
<b>詳細な説明.....</b>	<b>13</b>
<b>BeeHive304.....</b>	<b>14</b>
イントロダクション.....	15
BeeHive304構成要素.....	19
BeeHive304をPCに接続.....	21
プログラムされるデバイスの操作.....	21
セルフテストとカリブレーション・チェック.....	21
テクニカル・スペシフィケーション.....	23
<b>BeeProg3.....</b>	<b>29</b>
イントロダクション.....	30
BeeProg3構成要素.....	33
BeeProg3をPCに接続.....	34
プログラムされるデバイスの操作.....	34
BeeProg3によるマルチ・プログラミング.....	35
セルフテストとカリブレーション・チェック.....	35
テクニカル・スペシフィケーション.....	36
<b>セットアップ.....</b>	<b>43</b>
ソフトウェア・セットアップ.....	44
ハードウェア・セットアップ.....	49
<b>PG4UWソフトウェア.....</b>	<b>58</b>
PG4UWプログラマ・ソフトウェア.....	59
File[ファイル].....	62
Buffer[バッファ].....	69
Device[デバイス].....	77
Programmer[プログラマ].....	106
Options[オプション].....	110
Help[ヘルプ].....	122
<b>PG4UWMCソフトウェア.....</b>	<b>125</b>
<b>共通ノート.....</b>	<b>143</b>
メンテナンス.....	144
ソフトウェア.....	145
ハードウェア.....	147
その他.....	147
<b>トラブルシューティングと保証.....</b>	<b>148</b>
トラブルシューティング.....	149
サポートされていないターゲット・デバイスがある場合.....	150
保証期間.....	150

## 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・ソフトウェ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。全ての記録はチェックサムで保護されています。

---

## イントロダクション

---

**BeeHive304**はデスクトップ・プログラマですが、自動プログラマや自動テスト装置(ATE)のコアとしても使用できます。大容量メモリの大量生産プログラミング用に設計された超高速ユニバーサル4x64-pindrive並行マルチ・プログラミング・システムです。チップはほぼ理論上の最大プログラミング速度でプログラムされます。

**BeeProg3**はデスクトップ・プログラマですが、自動プログラマや自動テスト装置(ATE)のコアとしても使用できます。パワフルな64ピン・ドライバーを備えた超スピード・ユニバーサル・プログラマであり、少量生産プログラミング用に設計されています。

保護回路、オリジナルのブランド・コンポーネント、慎重な製造を含む先進的な設計によりBeeHive304とBeeProg3とプログラマの部品と労力に3年間の保証を提供します。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販売者の保証条件は対象となる国の法律、又は、Elnecの販売者の保証ポリシーによって異なる場合があります。

## 製品構成

	programmer	external power supply	internal power supply	power cord	USB cable	AP3 diagnostic POD	programming module fixing screw	screw with washers for ground connection	M4x70 DIN912 screw with washer	tie and tie mount for fixing cables	CD with software and user manual	sticker register your programmer	sticker Elnec programmer inside	leaflet Notes about ESD	antistatic set	vacuum handling tool kit	Transport case
BeeHive304	•	•	•	•	•	1x	4x	•	6x	3x	•	•	2x	•	•	•	•
BeeProg3	•	•	•	•	•	•	•				•	•		•			•

使用する前に、パッケージに次の全ての部分が含まれていることを慎重に確認して下さい。

それぞれのパーツリストとの間に矛盾がある場合、又は、これらのアイテムが破損している場合は直ちに販売店にご連絡下さい。

## 必要な PC スペック

上記は S/W バージョン 3.38(2/2018) 以降のご使用を前提。最新のバージョンは <https://www.elnec.com/sw/pg4uwarc3-ondemand.exe> をご覧下さい。

Windows 10 の自動アップデート後の予期しないリスタートが起こる場合は、Windows 7 を推奨します。  
※現在の Windows 10 では自動アップデートを設定することで回避することが出来ます。

### 最低必要 PC 要件

	OS - Windows	CPU	RAM [MB]	free disk space [MB]	USB 2.0 high speed	LAN
2x BeeHive304	XP	C2Quad	2000	400	•	1Gb
BeeHive304	XP	C2D	2000	400	•	1Gb
BeeProg3	XP	C2D	1000	200	•	100Mb

1024 x 768 は最低モニタ解像度です。

### 推奨される PC 要件

	OS - Windows	CPU	RAM [MB]	free disk space [MB]	USB 2.0 high speed	2x USB 2.0 high speed controllers	LAN
2x BeHive304	Win 7	Core i5 <sup>*1</sup>	4000 <sup>*2</sup>	2000 <sup>*2</sup>		•	1Gb
BeeHive304	Win 7	Core i3 <sup>*1</sup>	2000 <sup>*2</sup>	2000 <sup>*2</sup>	•		1Gb
BeeProg3	Win 7	Core i3 <sup>*1</sup>	2000 <sup>*2</sup>	1000 <sup>*2</sup>	•		1Gb

1024 x 768 以上のより高い解像度のモニタを推奨します。

\*1 - 又は、それ以上 \*2 - 又は、それ以上

もし、2つのプログラムを1台のPCで使用頂くためには、各プログラムを別々のUSB2.0 High speed controller(USB EHCI)に接続してご使用されることを強くお勧めします。更に詳しくは”ハードウェア・セットアップ”の章をご覧ください。

**警告:** BeeHive304、又は、BeeProg3プログラムをLANケーブル経由でPCに直接接続することは出来ません。BeeHive304、又は、BeeProg3はスイッチ、又は、ルーターのみを使用してLANケーブル経由で接続できます。

ディスクの空き容量の要件は使用されているICデバイスのサイズと接続されているプログラミング・サイトの数によっても異なります。大きなデバイスの場合、ディスクに必要な空き容量は約1000MB + 2xデバイス・サイズ x そのPCIに接続されたプログラミング・サイトの数になります。

PCハードウェア/ソフトウェア構成がPG4UW/PG4UWMCの現在のソフトウェア・バージョンと現在の状況に十分適しているかどうかを簡単にチェックします:

- Windowsタスクマネージャを実行する(Ctrl+Alt+Del)
- [パフォーマンス]タブを参照して下さい。
- 最大値を表示する必要があります。プログラミング・システムのフル生産時のCPU使用率の80%



## ソフトウェア

Elneceは全てのデバイス・プログラマに共通のソフトウェアを提供しています。正規バージョンとOnDemandバージョンのソフトウェアを利用できます。正規バージョンは通常3~4週間ごとにリリースされ、OnDemandのバージョンは頻繁に毎日程新しいデバイスのサポートやバグ修正のためにカスタマの要求に基づいてリリースされます。Elneceは全てのプログラマに無料のソフトウェア・バージョンをダウンロードできます。

### **最新バージョンの制御プログラムを使用することが重要なのはなぜですか？**

- 半導体メーカーは製品設計、及び、製造における柔軟性、品質、スピードの必要性をサポートするために新しいテクノロジーで製造された新しいパッケージ・タイプの新しいデバイスを継続的に導入しています。ペースを維持し最新の状態に保つため通常は1年以内に7000個以上の新しいデバイスを制御プログラムに実装しています。
- 更に、典型的なプログラマブル・デバイスはその技術的特性、及び、プロセス歩留まりを維持、又は、向上させるために寿命期間中にいくつかの変更を受けます。これらの変更はアップグレードする必要があるプログラミング・アルゴリズムに影響することが良くよくあります(プログラミング・アルゴリズムはプログラマに特定のターゲット・デバイスにデータをプログラムする方法を指示する命令セットです)。プログラミング・プロセスで最新のアルゴリズムを使用することは高品質の結果を得るための鍵です。多くの場合、古いアルゴリズムではまだデバイスがプログラムされますが、最適なアルゴリズムで可能なレベルのデータ保持は提供されません。最新のアルゴリズムを使用しないとプログラミングの歩留まりが低下しプログラミング時間が長くなることがあります。また、プログラムされたデバイスの長期信頼性に影響を及ぼすこともあります。

当社のコミットメントはこの新しいデバイス用に作成された最新、及び又は、最適なプログラミング・アルゴリズムを使用しているかどうかを確認できるように、これらの新規、又は、変更されたデバイスのサポートをリリース前、又は、出来るだけ早く実施することです。

フリー・ソフトウェア・アップデートは弊社の  
<https://www.elnec.com/en/download/>  
から利用出来ます

### **フリー・アディショナル・サービス:**

- フリー・テクニカル・サポート(e-mail)
- ウェブ・サイト経由フリー・ライフタイム・ソフトウェア・アップデート

またお客様のサポート・プログラムでは次の新しいサービスを提供しています:

- **Keep-Current** はElneceがプログラマ向けの最新のバージョンの制御プログラムと更新されたデバイス情報を提供するサービスです。Keep-Currentサービスは最新のソフトウェアとドキュメントに最小限のコストでいつでもアクセスできる、手間のかからない保証です。詳細については[www.elnec.com](http://www.elnec.com)を参照して下さい。
- **AlgOR** ((Algorithm On Request)サービスではElneceソフトウェアから現在のデバイス・リストで未だ使用できないプログラミング・デバイスのサポートを受けることができます。詳細については[www.elnec.com](http://www.elnec.com)を参照して下さい。

---

クイック・スタート

---

## プログラマ・ハードウェアのインストール

- 付属のケーブルを使用してプログラムの 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。

プリンター・ポートと電源の接続の順序かどちらが先でも問題はありません。

**※重要:USB の場合はソフトウェ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・ソフトウェアのインストール

CD が付属しておりますが、ご購入時はご使用時には

<http://www.eltec.com/downloads.php> から最新のソフトウェアをダウンロードされることをお勧めします。

## コントロール・プログラムの実行



Eltec PG4UW をダブルクリックでコントロール・プログラムを実行するために PG4UW.EXE を立ち上げて下さい。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 プログラムをスキャンします。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、チェックサムと他の項目(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], ヴェリファイ[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**”はプログラマで使用する機能のため使用されます。


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

---

*詳細な説明*

---

---

## *BeeHive304*

---



## イントロダクション

**BeeHive304**はEl nec**同時[非同期]ユニバーサル・マルチプログラマ**・シリーズの次世代メンバーであり、大容量メモリのための**非常に高速で信頼性の高い**マルチ・プログラマへの強い要求に応えるために作られています。

技術的な完成度とハードウェアのスピードを重視して設計されたこのプログラマは、**高品質のデスクトップ・プログラミング**だけでなく、最高の品質と全体的な歩留まりを保証する**自動プログラミング・システム**やATEマシンにも最適です。

**BeeHive304** は**BeeProg3**プログラミング・コアハードウェアをベースとする4つの独立した独立したユニバーサル・プログラミング・サイトで構成されています。従って、プログラミング・サイトは**非同期(同時マルチ・プログラミング・プログラミング・モード)**に実行できます。各プログラミング・サイトはチップがソケットに正しく挿入されたことが検出された時点で、他のプログラミング・サイトのステータスに依存せずにプログラミングを開始します。その結果、4つ目のサイトでプログラムされたチップを置き換えている間でも3つのプログラミング・サイトが機能します。

**BeeHive304** マルチ・プログラマは各プログラミング・サイトが独立して動作するため**BeeProg3**プログラマと同じ数のチップをサポートしプログラミング速度の明らかな低下なしにサポートします。また必要に応じて、各プログラミング・サイトは異なるチップをプログラムすることが出来ます。

**BeeProg3**プログラミング・コアは最先端の**FPGA**、強力な**ARM**プロセッサと内部SSDに基づいているため、**BeeHive304**は理論上可能な速度でデバイスをプログラムする準備ができています。達成された超高速プログラミング速度 - 22.5 MB/秒以上 - はこれまでサポートされていた実際のデバイスよりも実際には高くなります。これは非常に短いプログラミング時間に反映されます。例えば、**2 GB eMMC NAND**フラッシュはプログラムされたメモリがその速度を許容する場合、100秒未満で実行できます。

テストでは**BeeHive304**は現在この価格帯の全ての競合他社より高速であり、多くのチップでは最も速いことが示されています。

**BeeHive304**は現在のあらゆる種類のシリコン・テクノロジーと将来の**プログラマブル・デバイス**をサポートします。古いデバイスも部分的にサポートしています。新規のデバイスのサポートでは殆どの場合(必要に応じて)シンプルなプログラミング・モジュールでソフトウェアの更新のみが必要となるため**所有コストを最小限に抑えることが出来ます**。

ハードウェアのモジュラー構成 - プログラミング・サイトが独立して機能 - はプログラミング・サイトの一部が動作不能になったときに連続的な操作を可能にします。また迅速かつ簡単にサービスを提供します。

プログラミング・モジュールのソケット内のデバイスの適切な配置を検出する検出回路は**チップの挿入の直後にプログラミングを開始**することを可能にします。オペレーターは完了したチップを取り除き、新しいチップを挿入するだけです。従って、オペレータ・トレーニングは最小限に抑えられます。

**BeeHive304**は**USB(2.0)**ハイスピードポート、又は、**LAN**ポート(スイッチ、又は、ルータ経由)を介してMS Windows OSを実行しているIBM PC互換のパーソナル・コンピュータとインターフェースします。

ESD保護制御を容易に実装するための静電気防止用リストストラップ接続用バナナジャック、及び、アース線用ワッシャ付きネジ。

非常に競争力のある価格と信頼性の高いプログラミングのための優れたハードウェア設計により、おそらくこのクラスの最高の"価格に見合った価値のある"プログラマです。

64ピンの豊富な機能を備えたBeeProg3プログラミング・コアの精密でパワフルなピン・ドライバは、ノイズ、グラウンド・バウンス、オーバーシュートを排除することで、デバイス的高速、高精度、クリーンな波形信号を提供し、プログラミング歩留まりを最大限に高めています。これにより、プログラマブル・デバイス - (E)EPROM, Flash, MRAM, PCM, ... に使用される事実上すべての不揮発性テクノロジーを単一のデバイス・プログラマで確実にサポートすることができます。

**FPGAベースの完全再構成可能なTTLピンドライバ**はデバイスの各ピンにH/L/pull\_up/pull\_down、及び、読み出し機能を提供します。デュアルH/Lドライバは追加ロジックなしでプログラムされたデバイスのコア信号とI/O信号の両方に2つの異なるHレベルを提供することが出来ます。プログラマのピン・ドライバは0.8Vまで動作するためプログラマは最新、及び、将来の高度な超低電圧デバイスのフルレンジをプログラムする準備が来ています。

FPGAベースのステートマシン、高速プロセッサ、及び、SSDを使用することによって達成される**非常に高速なプログラミング**によりプログラマの内部で時間に拘らず重要なルーチン、及び、データ転送が実行されます。

プログラマは各デバイスをプログラムする前にプログラマとプログラムされるデバイスとの間の適切な信号経路のチェックに基づいてデバイス挿入テストを実行します。プログラミング構成に依ってプログラムされるデバイスとプログラミング・モジュールのZIFソケットとの間の接触ミス、接触不良、プログラミング・モジュールとプログラマとの間の接触不良、又は、非接触を識別、ZIFソケット、又は、プログラミング・モジュール内のデバイスの位置(移動、回転、逆方向)が間違っていることを識別します。これらの機能は、**過電流保護とシグネチャバイト・チェック**によってサポートされオペレータ・エラーによるチップの損傷を防止します。

セルフテスト機能によりソフトウェアの診断部分を実行してプログラマの状態を完全にチェックすることができます。

**内蔵保護回路**は環境やオペレータの失敗によるプログラマ、及び/又は、プログラムされたデバイスの損傷を排除します。BeeProg3プログラミングコアの全入力 - プログラミング・モジュール・インタフェースのピン(ピン・ドライバ信号、及び、サポート信号)、PC、及び、電源入力への接続は最大15kVのESDに対して保護されています。

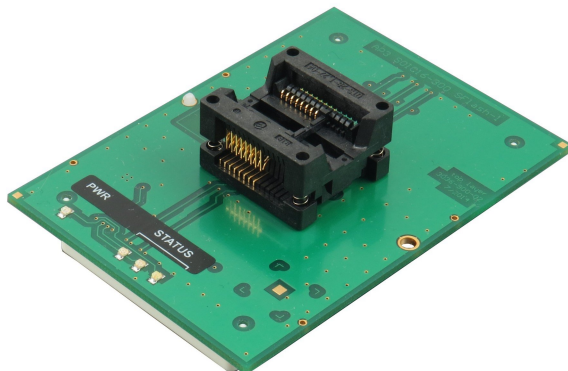
超高速メモリの適切で信頼性の高いプログラミングのためにBeeHive304はプログラムされるデバイスのニーズに従って特定のデバイス・ファミリ用に**最適に設計された特殊なモジュール**を使用しますが可能な限りICパッケージ・タイプ専用の汎用プログラミング・モジュールが使用されます。プログラミング・モジュールはBeeProg3プログラミング・コアに基づいた全てのモジュールと同じです。

プログラミング・モジュールの構造はプログラマの上部で**完全な安定性を確保するように設計**されておりメカニカル・アームによるチップの挿入/交換を行うのに十分に頑丈であり、モジュール交換後もZIFソケットの同一位置を保つことができます。

### プログラミング・モジュールは

PDIP, PLCC, JLCC, SOIC, SDIP, SOP, PSOP, SSOP, TSOP, TSOPII, TSSOP, QFP, PQFP, TQFP, VQFP, QFN(MLF), SON, BGA, EBGA, FBGA, VFBGA, UBGA, FTBGA, LAP, CSP, SCSP, LQFP, MQFP, HVQFN, QLP, QIP や他のパッケージを含みます。





*ノート: ZIFソケットとデバイス・リファレンス・ピンの向きは完璧な機能と人間工学に基づいて設定されているためAP3プログラミング・モジュールで異なる場合があります。*

**BeeHive304**プログラマはデスクトップ・プログラマとして設計されていますが、自動プログラミング・システムでの使用に適した特別な構造を持っています。ハンドラのアームの動きのオーバーヘッドを最小限に抑えるためにこの機能と高品質のハードウェアのプログラマの寸法を技術的な最小限に抑えています。この構造は機械的に安定しており動作中の振動に対して耐性があります。BeeHive304のケースは、プログラマ本体の上、又は、下から自動プログラマ作業場所に固定する準備ができています。

**BeeHive304**は自動化されたプログラマ(例えば、古いプログラマの代わりに)やより多くの方法を使っていくつかのハンドラに実装することができます:

- 1台、又は、2台のBeeHive304プログラマが全てのジョブを実行するのに十分であれば、マシン制御とプログラマ・ソフトウェアの両方に対してプログラミング・システム全体で1つの制御PCを使用すれば十分です。BeeHive304はUSB、又は、LANネットワークを使用してこのコンピュータに接続できます。
- 自動化されたプログラマに2台以上のBeeHive304プログラマを実装する必要がある場合は、さらに多くのコンピュータを使用する必要があります。全システムはBeeHive304プログラマと制御PCを掛け合わせることで1つの自動プログラミング・システムで最大64のプログラミング・サイトに拡張可能です。最大構成は16x BeeHive304プログラマから構成されています。システム内の1台の制御コンピュータがマスター・ユニットとして機能します。ここではマルチ・プログラミングの制御ソフトウェア、シリアルライズ・エンジン、及び、ホスト・システムへのインターフェイスも実行します。ネットワーク全体での制御PCのインターフェイスは標準LANネットワークを介して行われます。

**自動化されたプログラマやハンドラのソフトウェアへのBeeHive304の実装は、PG4UWMC制御ソフトウェアの簡単なリモート制御を使用することによるものです。**

**BeeHive304** プログラマはWindows XPからWindows 10 64ビットまでの全てのバージョンのMS Windowsで動作する快適で使いやすい制御プログラムによって駆動されます。

• **PG4UWMC** はプロダクション・モードの制御ソフトウェアです。

ソフトウェアのこの部分は大量生産オペレーションの簡単なモニターリングに焦点を当てています。オペレーター・フレンドリーなコントロール・ソフトウェアは多くの強力な機能と使いやすさを兼ね備えています。グラフィック・ユーザー・インターフェースは重要でない詳細に対してオペレーターの負担なし全ての重要な活動結果の概要を提供します。

BeeHive304マルチ・プログラミング・システムを制御する**プロジェクト・ファイル**が使用されています。プロジェクト・ファイルにはユーザー・データ、チップ・プログラミング設定情報、チップ・コンフィギュレーション・データ、自動プログラミング・コマンド・シーケンスなどが含まれます。従って、プロジェクト・ファイルは通常、設計技術者によって作成され認証されてオペレーターに与えられるためオペレータ・エラーは最小限に抑えられます。オプションの保護モードはプロジェクト・ファイルの望ましくない変更を避けるためにプロジェクト・ファイルに設定することができます。各チップはリッチ・フィーチャー・シリアルライゼーション・システムを使用してシリアル番号、構成、及び、較正情報等の異なるデータでプログラムすることができます。

• **PG4UW** はエンジニアリング・モード、又は、プログラミング・サイト・ドライバ用の制御ソフトウェアです。

ソフトウェアのこの部分はプロダクションモード制御ソフトウェアでの使用のためのプロジェクト・ファイルの迅速かつ簡単な準備に焦点を当てています。

各プログラミング・サイトはプルダウン・メニュー、ホットキーとオンライン・ヘルプを備えた快適で使いやすい制御プログラムによって駆動されます。これは他の全てのElneqシングル・サイトプログラマに使用され長年の実績を証明されたソフトウェアです。

デバイスの選択はそのクラス、製造元、又は、単にベンダ名及び/又は、部品番号の一部を入力することによって実行されます。いくつかの**テスト機能**(挿入テスト、シグネチャバイト・チェック)といくつかの**特殊機能**(自動インクリメント、生産モード - ソケットへのチップ挿入直後のプログラミングの開始)によって標準デバイス関連コマンド(読み取り、ブランク・チェック、プログラム、バリファイ、イレース)が強化されています。

全ての既知のデータ形式がサポートされています。自動ファイル形式の検出とファイル読み込み中の変換が実行されます。

また、ソフトウェアは - デバイス情報セクション - でプログラムされるデバイスに関する多くの情報を提供します。特に、**全ての利用可能なパッケージの図面**が提供されています。ソフトウェアはまたサポートされている**各チップのチップ・マーキング(チップのプリフィックスとサフィックスの意味)**の説明も提供しています。

豊富な機能を備えた**シリアルライゼーション機能**によりプログラムされた各デバイスに個々のシリアル番号を割り当てることができます。- 又は、単にシリアル番号をインクリメントするか、又は、その機能によりシリアル番号、又は、プログラムされたデバイス識別シグネチャをファイルから読み取り、プログラムされるデバイスにプログラムすることができます。

JEDEC標準JESD-71のJamファイルはJam Playerによって解釈されます。Jamファイルはそれぞれのプログラマブル・デバイスの製造元によって提供される設計ソフトウェアによって生成されます。チップはJTAG(IEEE 1149.1 Joint Test Action Group)インターフェースを介してプログラミングされます。

**VMEファイル**はVME Playerによって解釈されます。VMEファイルはSVFファイルの圧縮されたバイナ形式であり、高水準のIEEE 1149.1バス操作を含んでいます。SVFファイルはSVF Playerによって解釈されます。SVFファイル(シリアル・ベクトル形式)には高水準のIEEE 1149.1バス操作が含まれています。SVFファイルはそれぞれのプログラマブル・デバイスの製造元によって提供される設計ソフトウェアによって生成されます。チップはJTAGインタフェースを介してプログラムされます。VMEファイルはそれぞれのプログ

ラマブル・デバイスの製造元によって提供される設計ソフトウェアによって生成されます。

**BeeHive304**は真の意味でのユニバーサル・プログラマであるため殆どの新しいデバイスのサポートには**ソフトウェア・アップデートのみ**が必要であることを覚えておくことは重要です。当社の迅速なサービスにより数時間で新しいデバイスのサポートを追加することができます。詳細については**AlgOR**(Algorithm On Request)サービスとOnDemandソフトウェアを参照してください。

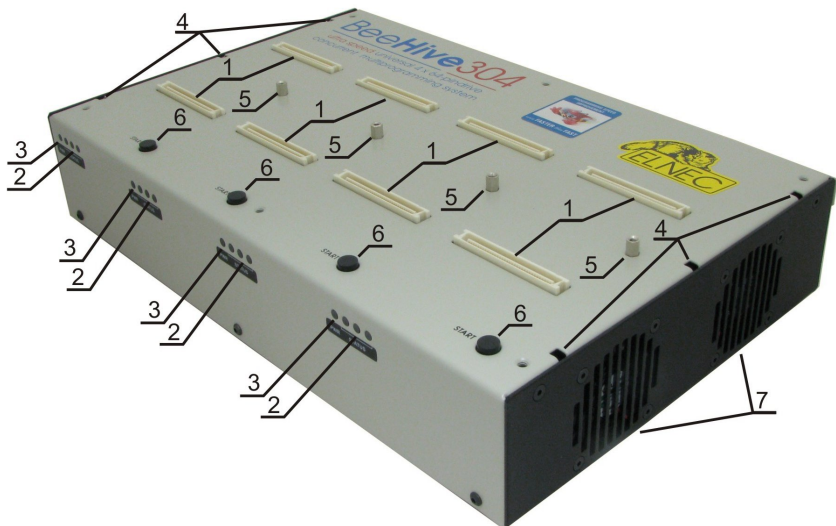
このサービスは殆どの場合無料です。但し、新規サポートのためのサンプル・デバイスの無償提供をお願いすることがあります、また、開発および製造コストが高すぎる場合は、お客様に費用を分担するよ依頼することがあります。

保護回路、オリジナル・ブランド・コンポーネント、慎重な製造と焼き付けを含むBeeHive304ユニバーサル・プログラマの先進的な設計により、プログラマの部品や製造技術に対する3年間の保証を提供しております。

保証は購入日から有効です。製品の登録は修理依頼を迅速化します。登録は購入日から60日以内に行う必要があります。

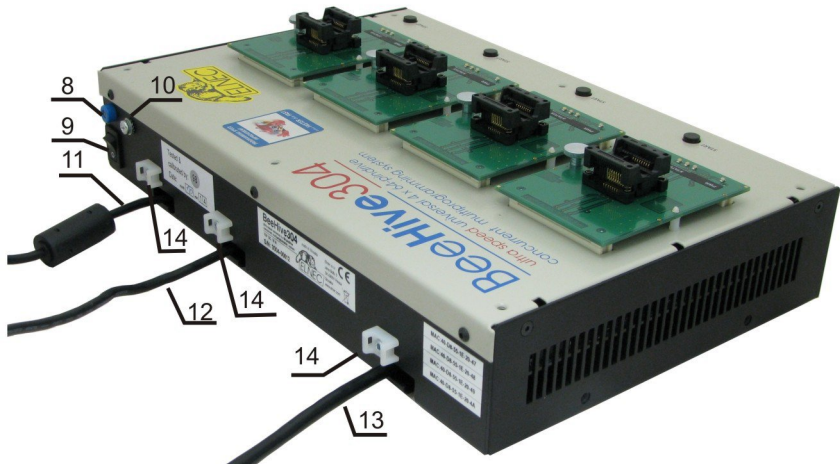
## BeeHive304 構成要素

- 1) プログラミング・モジュール・インターフェース(PMI)コネクタ
- 2) サイトの動作結果 LED
- 3) サイトのパワー/スリープ LED
- 4) BeeHive304をボトム、又は、トップ・プレートに固定するための6 x 4.2 mmの穴
- 5) プログラミング・モジュール固定ネジ
- 6) ボタン START
- 7) 温度制御クーリング・ファン



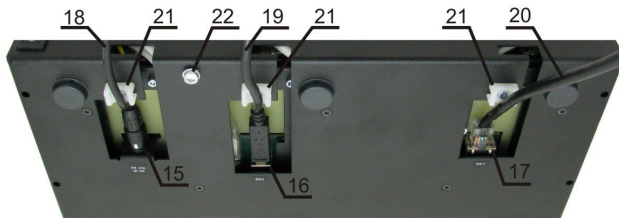
BeeHive304の正面と上面ビュー

- 8) ESDリスト・ストラップ接続のためのバナナ・ジャック
- 9) ON/OFF スイッチ
- 10) 後面のアース接続用ワッシャー付きネジ
- 11) 後面からプログラマに接続された15V DC電源ケーブル
- 12) 後面からプログラマに接続されたUSBケーブル
- 13) 後面からプログラマに接続されたLANケーブル
- 14) 電源、USB、及び、LANケーブルを固定するための背面側ケーブル・タイ・マウント



BeeHive304の背面ビュー

- 15) パワー・コネクタ, 15V 6A
- 16) type B PC用USBコネクタ ↔ BeeHive304通信ケーブル
- 17) ネットワーク用LANコネクタ ↔ BeeHive304通信ケーブル
- 18) 底面からプログラマに接続された15V DC電源ケーブル
- 19) 底面からプログラマに接続されたUSBケーブル
- 20) 底面からプログラマに接続されたLANケーブル
- 21) 固定側電源、USB、又は、LANケーブルの底面ケーブルタイマウント
- 22) 底側のアース接続用ワッシャー付きネジ



BeeHive304の底面ビュー

## BeeHive304 を PC に接続

### USBポートの使用

プログラマとPCの接続に関する推奨事項:

1. プログラマとPC、又は、他のグラウンド間を接地接続
2. USBケーブルでプログラマをPCと接続
3. プログラマに電源を接続

### LANポートの使用

プログラマとPCの接続に関する推奨事項:

1. プログラマとPC、又は、他のグラウンド間を接地接続
2. Cat5eイーサネット・ケーブルを使用してプログラマを最寄りのネットワーク・デバイス(スイッチ、ハブ、又は、ルータ)に接続
3. ネットワーク内のDHCPサーバーが設定されていることを確認
4. プログラマに電源を接続

## プログラムされるデバイスの操作

制御プログラムでDevice/Select device (Alt+F5)を選択します。デバイス名の一部、又は、全部を入力して目的のデバイスを選択します。右側の列には希望するデバイスでの作業に必要なAP3モジュールの名前が表示されます。

AP3プログラミング・モジュールをプログラミング・モジュール・インターフェイス(PMI)コネクタに挿入します。開いているZIFソケット(ZIFの上部を押して)に目的のデバイスを挿入し、ソケット(ZIFの上部を離して)を閉じてください。ZIFソケットのプログラムされたデバイスの正しい方向はZIFソケットの近くに示されています。

### ノート:

プログラマの電源を切る必要はなく、またソフトウェアが次の場合に実行されている可能性があります:

- プログラミング・モジュール・インターフェイス(PMI)コネクタへ/からのAP3プログラミング・モジュールの挿入/取り外し
- AP3プログラミング・モジュールZIFソケットへ/からターゲット・デバイスの挿入/取り外しを行いますが、ターゲット・デバイスの操作は無効です(LED BUSYライトが消灯)。

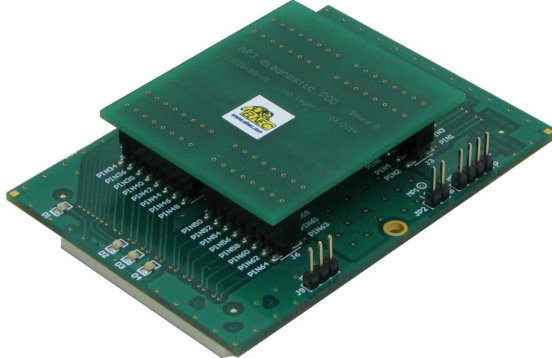
プログラマの電子保護はターゲット・デバイスとプログラマ自身を短期、又は、長期の停電から保護し一部はPCの障害からも保護します。しかし、誤ったユーザー選択したプログラミング・パラメータにターゲット・デバイスの完全性を与えることは不可能です。ターゲット・デバイスはプログラマへの物理的な接続を取り除くことによって制御プログラムの強制的な中断(PCのリセット、又は、スイッチオフ)によって破壊されることはありませんが、実際にプログラムされたセルの内容は不定義のままです。

## セルフテストとカリブレーション・チェック

プログラマが期待通りに動作しないと感じる場合は、標準パッケージに同梱されているAP3診断PODを使用してプログラマの“Selftest”を実行してください。

### プログラマ・サイトのセルフテスト

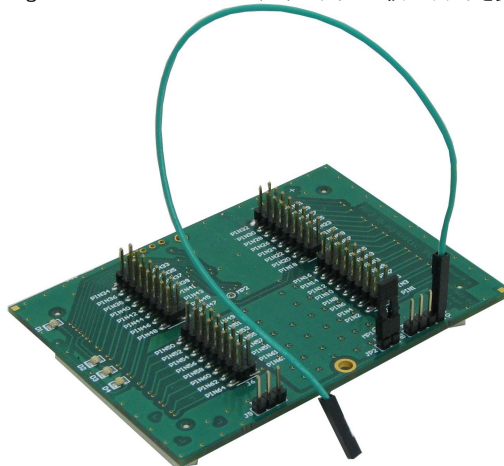
- ボードAをAP3診断PODボードBに接続
- プログラマ・サイトのプログラミング・モジュール・インタフェース(PMI)コネクタに**AP3診断POD**を挿入します。
- プログラマのセルフテストをPG4UW(メニュー Programmer/Selftest)で実行します。



Selftest plus[セルフテスト・プラス]のためのAP3 diagnostic POD

### カリブレーション・テスト

- ボードAを**AP3診断POD**のボードBから外します。
- プログラマ・サイトのプログラミング・モジュール・インタフェース(PMI)コネクタに**AP3診断POD**のボードAを挿入します。
- PG4UW(メニュー Programmer/Calibration test)でプログラマの校正テストを実行します。



カリブレーション・テストのためのAP3 diagnostic POD

## テクニカル・スペシフィケーション

### ハードウェア

#### ベース・ユニット, DACs

- USB 2.0ハイスピード互換ポート、最大480Mb/s転送レート
- 100Mbit LANポート
- オンボード・インテリジェンス: 強力なプロセッサ(ARM9 400MHz)とFPGAベースのステートマシン (基本クロック50MHzプラスPLL)
- 内部バッファ(128GB、大容量にアップグレード可能)として内蔵mSATA SSD (\*)
- VCC1, VCC2とVPP用の3つのD/Aコンバータ、制御可能な立ち上がりと立ち下り時間
- VCC1, VCC2 範囲 0.8V..7V/1A (10mVステップ)
- VPP 範囲 0V..25V/1A (25mVステップ)
- 温度制御されたファン
- セルフテスト機能
- 電源入力に対するサージとESDに対する保護、プログラミング・モジュール・インターフェース (PMI)(IEC1000-4-2: 15kV空気、8kV接点)のUSBとLANポートとピン
- ESD[静電気防止用]リスト・ストラップ接続用のバナナジャック
- グラウンドに接続するための2つのワッシャー付きネジ

#### (\*) ノート:

- バッファ内のデータはフラグメントに格納され、そして、また部分的に圧縮されるため、バッファは現在のバッファ・サイズより大きなサイズのデバイスについても標準データを保持できます。
- このアップグレードは機械的に複雑でElneqでのみ実行できます。
- s/n 3004-00041(を含む)までのプログラムは内部バッファとして32GBのSSDを提供されました。
- 128GB SSDは比例してより大きいサイズのSSDに対して約110GBのランダム・データ(32GB SSDの場合は28GB)をバッファとして保存できます。デバイス・サイズの全範囲でランダムなデータを含む可能性があるeMMC/NANDデバイスを作成/コピーする予定の場合は、プログラムに等しい容量のバッファを使用することをお勧めします。

#### ピン・ドライバ(プログラミング・モジュールのインターフェースで利用可能 - プログラミング・モジュール用のコネクタ)

- ピン・ドライバ: 64 ユニバーサル
- VCC1/VCC2とVPPは各ピンに接続することが出来ます。
- 各ピンの完璧なグラウンド
- 2つの独立したFPGAベースのTTLドライバは全てのピンドライバ・ピンでH, L, CLK, プルアップ, プルダウン, ロジック・レベル 0.75V - 5V(IOLとIOH電流20mA)を提供します。
- ロジック信号周波数: 最大125MHz (3.3V), 80MHz (5V)
- 0.8V~25Vのアナログ・ピンドライバ出力レベルを選択可能
- 電流制限、過電流シャットダウン、停電シャットダウン
- 導通テスト: 各ピンはプログラミング動作の前にテストされます。

## デバイス・サポート

### プログラマ、プログラミング・モジュールを使用:

- NAND FLASH: Samsung K9xxx, KFxxx, SK Hynix (ex Hynix) HY27xxx, H27xxx, Toshiba TC58xxx, TH58xxx, Micron MT29Fxxx, (ex Numonyx ex STM) NANDxxx, Spansion S30Mxxx, S34xxx, 3D-Plus 3DFNxxx, ATO Solution AFNDxxx, Fidelix FMNDxxx, Eon Silicon Sol. EN27xxx, ESMT F59xxx, LBA-NAND Toshiba THGVNxxx
- serial NAND FLASH: Micron MT29Fxxx, GigaDevice GD5Fxxx
- eMMC: Hynix H26Mxxxxxxxx, Kingston KE44B-xxx/xxx, Micron MTFCxxxxxx, Numonyx NANDxxxxxxxx, Phison PSM4A11-xx, Samsung KLMxxxxxxx, SanDisk SDINxxx-xx, Toshiba THGBMxxxxxxxxxxx
- Multi-chip devices: NAND+RAM, NOR+RAM, NOR+NOR+RAM, NAND+NOR+RAM
- Serial Flash: standard SPI, high performance Dual I/O SPI and Quad I/O SPI (25Bxxx, 25Dxxx, 25Exxx, 25Fxxx, 25Lxxx, 25Mxxx, 25Pxxx, 25Qxxx, 25Sxxx, 25Txxx, 25Uxxx, 25Vxxx, 25Wxxx, 25Xxxx, 26Vxxx, 45PExx, MX66Lxxx, S70FLxxx), DataFlash (AT45Dxxx, AT26Dxxx)
- parallel NOR Flash: 28Fxxx, 29Cxxx, 29Fxxx, 29GLxxx, 29BVxxx, 29LVxxx, 29Wxxx, 49Fxxx series, Samsung's K8Fxxxx, K8Cxxxx, K8Sxxxx, K8Pxxxx series, ...
- EPROM: NMOS/CMOS, 27xxx and 27Cxxx series
- EEPROM: NMOS/CMOS, 28xxx, 28Cxxx, 27EExxx series, 3D Plus 3DEExxxxxxxxx
- mDOC H3: SanDisk (ex M-Systems) SDED5xxx, SDED7xxx, MD2533xxx, MD2534xxx, Hynix HY23xxx
- FRAM: Ramtron
- MRAM: Everspin MRxxxxx8x, 3D Plus 3DMRxxxxxxxx
- NV RAM: Dallas DSxxx, SGS/Inmos MKxxx, SIMTEK STKxxx, XICOR 2xxx, ZMD U63x series
- Serial E(E)PROM: Serial E(E)PROM: 11LCxxx, 24Cxxx, 24Fxxx, 25Cxxx, 30TSExx, 34Cxxx, 34TSxx, 59Cxxx, 85xxx, 93Cxxx, NVM3060, MDAXxx series, full support for LV series, AT88SCxxx
- Serial FRAM: Cypress(Ramtron): FM24xxxxxx, FM25xxxxxx, Fujitsu: MB85RCxxxx, MB85RSxxxx, Lapis(OKI, Rohm): MR44xxxxxx, MR45xxxxxx
- Serial MRAM: Everspin MH20xxx, MH25xxx
- Configuration (EE)PROM: XCFxxx, XC17xxxx, XC18Vxxx, EPCxxx, EPCSxxx, AT17xxx, AT18Fxxx, 37LVxx
- Wire E(E)PROM: DS1xxx, DS2xxx
- PLD Altera: MAX 3000A, MAX 7000A, MAX 7000B, MAX 7000S, MAX7000AE, MAX II/G/Z, MAX V
- PLD Lattice: ispGAL22V10x, ispLSI1xxx, ispLSI1xxxEA, ispLSI2xxx, ispLSI2xxxA, ispLSI2xxxE, ispLSI2xxxV, ispLSI2xxxVE, ispLSI2xxxVL, LC4xxxB/C/V/ZC/ZE, M4-xx/xx, M4A3-xx/xx, M4A5-xx/xx, M4LV-xx/xx, ispCLOCK, Power Manager/II, ProcessorPM
- PLD: Xilinx: XC9500, XC9500XL, XC9500XV, CoolRunner XPLA3, CoolRunner-II
- SPLD/CPLD series: AMD, AMI, Atmel, Cypress, Gould, ICT, Lattice, National Semicond., Philips, STMicroelectronics, TI (TMS), Vantis, VLSI
- FPGA: FPGA: Microsemi(Actel): ProASIC3, IGLOO, Fusion, ProASICplus, SmartFusion
- FPGA: Lattice: MachXO, MachXO2, LatticeXP, LatticeXP2, ispXPGA
- FPGA: Xilinx: Spartan-3AN
- Clocks: TI(TMS), Cypress
- Special chips: Atmel Tire Pressure Monitoring ATA6285N, ATA6286N; PWM controllers: Zilker Labs, Analog Devices; Multi-Phase ICs: IR(Chil Semiconductor); Gamma buffers: AUO, Maxim, TI, ...
- Microcontrollers MCS51 series: 87Cxxx, 87LVxx, 89Cxxx, 89Sxxx, 89Fxxx, 89LVxxx, 89LSxxx, 89LPxxx, 89Exxx, 89Lxxx, all manufacturers, Philips LPC series



- Microcontrollers Atmel ARM. AT91SAM7Sxx, AT91SAM7Lxx, AT91SAM7Xxx, AT91SAM7XCxx, AT91SAM7SExx series;
- Microcontrollers Atmel ARM9: AT91SAM9xxx series;
- Microcontrollers ARM Cortex-M3: ATSAM3Axxx, ATSAM3Uxxx, ATSAM3Nxxx, ATSAM3Sxxx, ATSAM3D20, ATSAM3Xxxx series
- Microcontrollers ARM Cortex-M4: ATSAM4Sxxx series
- Microcontrollers Atmel AVR 8bit/16bit: AT90Sxxxx, AT90pwm, AT90can, AT90usb, ATtiny, ATmega, ATxmega series
- Microcontrollers Atmel AVR32: AT32UC3xxxx, ATUCxxxD3/D4/L3U/L4U series
- Microcontrollers TI (Chipcon): CC11xx, CC24xx, CC25xx, CC85xx series
- Microcontrollers Coreriver: Atom 1.0, MiDAS1.0, 1.1, 2.0, 2.1, 2.2, 3.0 series
- Microcontrollers Cypress: CY7Cxxxxx, CY8Cxxxxx
- Microcontrollers ELAN: EM78Pxxx
- Microcontrollers EPSON: S1C17 series
- Microcontrollers Explore Microelectronic: EPF01x, EPF02x series
- Microcontrollers Generalplus: GPM8Fxxx series
- Microcontrollers GreenPeak: GPxxx series
- Microcontrollers Infineon(Siemens): XC800, C500, XC166, C166 series
- Microcontrollers MDT 1xxx and 2xxx series
- Microcontrollers Megawin: MG87xxx, MPC82xxx series
- Microcontrollers Microchip PICmicro: PIC10xxx, PIC12xxx, PIC16xxx, PIC17Cxxx, PIC18xxx, PIC24xxx, dsPIC, PIC32xxx series
- Microcontrollers Motorola/Freescale: HC05, HC08, HC11, HC12, HCS08, RS08, S12, S12X, MC56F, MCF51, MCF52 series, Kinetis (K,L), Qorivva/5xxx Power Architecture
- Microcontrollers Myson MTV2xx, 3xx, 4xx, 5xx, CS89xx series
- Microcontrollers National: COP8xxx series
- Microcontrollers NEC: uPD70Fxxx, uPD78Fxxx series
- Microcontrollers Novatek: NT68xxx series
- Microcontrollers Nordic Semiconductor: nRF24LExxx, nRF24LUxxx, nRF315xx, nRF51xxx Flash and OTP series
- Microcontrollers Nuvoton ARM Cortex-Mx: NUC1xx, M05x, Mini51, Nano1xx series
- Microcontrollers Nuvoton (Winbond): N79xxx, W77xxx, W78xxx, W79xxx, W83xxx series
- Microcontrollers NXP (Philips) ARM Cortex-Mx: LPC11xx, LPC11Cxx, LPC11Dxx, LPC11Uxx, LPC12xx, LPC12Dxx, LPC13xx, LPC17xx, LPC11Axx, LPC11Exx, LPC11xxLV, LPC18xx, LPC43xx, LPC8xx, EM7xx, series
- Microcontrollers NXP (Philips) UOC series: UOCIII, UOC-TOP, UOC-Fighter (TDA1xxxx) series
- Microcontrollers NXP (Philips) ARM7: LPC2xxx, MPT6xx, PCD807xx, SAF7780xxx series
- Microcontrollers NXP (Philips) ARM9: LPC31xx series
- Microcontrollers Pasat: TinyModule DIL40, DIL50 series
- Microcontrollers Scenix (Uvicom): SXxxx series
- Microcontrollers Syntek: STK6xxx series
- Microcontrollers Renesas: R8C/Tiny series, RX series, uPD70Fxxx, uPD78Fxxx series, RL78 series, R32C series
- Microcontrollers SyncMOS: SM39xxx, SM59xxx, SM73xxx, SM79xxx, SM89xxx series
- Microcontrollers & Programmable System Memory STMicroelectronics: uPSD, PSD series
- Microcontrollers STM (ex SGS-Thomson): ST6xx, ST7xx, ST10xx, STR7xx, STR9xx, STM32F/L/W, STM8A/S/L series, SPC5 (Power Architecture)
- Microcontrollers Silicon Laboratories(Cygnal): C8051 series
- Microcontrollers Silicon Laboratories(Energy Micro): EFM32Gxx,

- EFM32GGxx, EFM32LGxx, EFM32TGxx, EFM32WGxx series
- Microcontrollers Silicon Laboratories: SiM3Cxxx, SiM3Lxxx, SiM3Uxxx series
  - Microcontrollers Texas Instruments: MSP430 series, MSC12xx series, TMS320F series, CC430 series,
  - Microcontrollers Texas Instruments (ex Luminary Micro): LM3Sxxx, LM3Sxxxx series, LM4Fxxxx series, TM4C series
  - Microcontrollers ZILOG: Z86/Z89xxx and Z8Fxxxx, Z8FMCxxxxx, Z16Fxxxx, ZGP323xxxxxx, ZLF645xxxxxx, ZLP12840xxxxx, ZLP323xxxxxxx series
  - Microcontrollers other: EM Microelectronic, Spansion(Fujitsu), Goal Semiconductor, Hitachi, Holtek, Novatek, Macronix, Princeton, Winbond, Samsung, Toshiba, Mitsubishi, Realtek, M-Square, ASP, Coreriver, Gencore, EXODUS Microelectronic, Topro, TinyARM, VersaChips, SunplusIT, M-Square, QIXIN, Signetic, Tekmos, Weltrend, Amic, Cyrod Technologies, Ember, Ramtron, Nordic Semiconductor, Samsung, ABOV Semiconductor...

#### ノート:

- サポートされているその時点でのデバイスについては  
<https://www.elnec.com/en/search/device-list/beeprog3/> の実際のデバイス・リストを参照してください。

#### パッケージ・サポート

- DIP, SDIP, PDIP, PLCC, JLCC, SOIC, SDIP, SOP, PSOP, SSOP, TSOP, TSOPII, TSSOP, QFP, PQFP, TQFP, VQFP, QFN(MLF), SON, BGA, EBGA, FBGA, VFBGA, UBGA, FTBGA, LAP, CSP, SCSP, LQFP, MQFP, HVQFN, QLP, QIP や他のパッケージを含みます。

#### プログラミング・スピード

##### ノート:

- プログラムの速度テストには常に乱数データ・パターンを使用することが重要です。競合他社の中には少量の非ブランク・データのみがプログラムされているか、又は、僅かなビット数(FE、EF等)のデータが使用されている"スカース・データ・パターン"を使用するものがあります。この不正行為のアプローチはプログラミング時間をかなり短縮することができます。プログラムの速度を比較したい場合は常にどのパターンを使用するかを尋ねておいて下さい。
- プログラミングのためのデータとプログラミング・アルゴリズムの主要部分がプログラムの内部に格納されているためプログラミング速度は実際にはPCの種類には余り依存しません。
- 2つのNAND Flashを含む下記の全てのデバイスが最大速度でプログラムされプログラミング時間は短くできません。
- 申し訳ありませんが、現時点では20+MB/sのBeeHive304プログラミング速度を利用できるデバイスは余り多くはありません。

Device	Size [bits]	Operation	Time
JS28F00AM29EWH(parallel NOR Flash)	4000080hx16(1Giga)	programming and verify	10 sec
MT29F1G08ABAEAWP(parallel NAND)	8400000hx8(1Giga)	programming and verify	19.5 sec
SDIN7DP2-8G (eMMC NAND FLASH)	1D2000000hx8(64Gig)	programming *1	342 sec
S25FL164K(serial Flash)	800300hx8(64Mega)	programming and verify	30.7 sec
AT89LP51RD2(microcontroller)	10000hx8	programming and verify	5.2 sec
PIC32MX360F512L(microcontroller)	80000hx8	programming and verify	8.9 sec

**条件:** Core2 Duo, 3.16 GHz, 1G RAM, USB 2.0 HS, Windows 7. Version of SW: 3.09, 10/2014.

- \*1 プログラミングの検証は、eMMCデバイスの内部コントローラによって行われます。デバイスはデータとCRCを加えたブロックを受信し、一致した場合、内部コントローラは適切なプログラミングを確認します。

## ソフトウェア

- **アルゴリズム:** マニ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:** ソフトウェ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。OnDemand[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・バージョン 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**
  - インサクション・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・ジェネレーター・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

- ファイル/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

- 外部電源供給ユニット: 操作電圧 100-240V AC, 90-264 VAC 最大, 47-63Hz. 出力電圧: 15V, 6A, 出力ケーブル長 1200mm (47.2 inch)
- 消費電力 最大. 90W アクティブ
- **BeeHive304の寸法** プログラム: 320,5 x 205 x 58,4 mm (12,6 x 8,1 x 2,3 inch). 寸法はプログラミング・モジュールを挿入しない状態での測定、プロジェクションを含みません。プログラミング・モジュールを挿入したBeeHive304プログラムの全高は、ZIFソケットの高さに依存し76~88mmの間で変化します。
- 重さ(プログラミング・モジュールを含まない): 3.6kg(7,9 lb)
- 操作温度: 5° C ÷ 40° C (41° F ÷ 104° F)
- 操作湿度: 20%.80%, 非結露

---

## BeeProg3

---



## イントロダクション

**BeeProg3**は大容量メモリのための**非常に高速で信頼性の高い**プログラマへの強い要求に応えるために設計された**Elneqユニバーサル・プログラマ**・シリーズの次世代のメンバーです。

**BeeProg3**はハードウェアの技術的完成度とスピードを重視して全体的な歩留まりを保証する**最高品質なデスクトップ・プログラミング**としてだけでなく**自動プログラミング・システム**やATEマシンのための高い要求に完全に適合するために設計されました。

**BeeProg3プログラマ**・コアは最先端の**FPGA**と強力な**ARM**プロセッサと内部SSDをベースに理論上可能な速度でデバイスをプログラムする用意ができています。達成された**超高速プログラミング速度** - 22.5 MB/秒以上を持続可能 - はこれまでサポートされていた実際のデバイスよりも実際には高くなっています。これは非常に短いプログラミング時間として反映されます。例えば、8 GB eMMC NAND Flashはプログラムされたメモリがその速度を許容する場合、250秒未満[2 GB eMMC NAND Flashで**100秒未満**]で実行できます。テストでは、BeeProg3は現在この価格カテゴリーで全ての競合他社よりも高速です。

**BeeProg3**は現在の**あらゆる種類**のシリコン・テクノロジーと将来の**プログラマブル・デバイス**をサポートします。古いデバイスも部分的にサポートしています。新規のデバイスのサポートでは殆どの場合(必要に応じて)シンプルなプログラミング・モジュールでソフトウェアの更新のみが必要となるため**所有コストを最小限に抑えることが出来ます**。設計に最適なデバイスを自由に選択できます。

超高速メモリの適切で信頼性の高いプログラミングのためにBeeProg3はプログラムされるデバイスのニーズに従って特定のデバイス・ファミリー用に**最適に設計された特殊なモジュール**を使用しますが可能な限りICパッケージ・タイプ専用の汎用プログラミング・モジュールが使用されます。プログラミング・モジュールはBeeProg3プログラミング・コアに基づきBeeHive304マルチ・プログラマで共通に使用できます。

プログラミング・モジュールの構造はプログラマの上部で**完全な安定性を確保するように設計**されておりメカニカル・アームによるチップの挿入/交換を行うのに十分に頑丈であり、モジュール交換後もZIFソケットの同一位置を保つことができます。

**BeeProg3**はUSB(2.0ハイスピード)ポート、又は、100Mb LANポート(スイッチ、又は、ルータ経由)を介してMS Windows OSを実行しているIBM PC互換のパーソナル・コンピュータとインターフェースします。

非常に競争力のある価格と信頼性の高いプログラミングのための優れたハードウェア設計により、おそらくこのクラスの最高の"価格に見合った価値のある"プログラマです。

64ピンの豊富な機能を備えた**BeeProg3**プログラミング・コアの**精密でパワフルなピン・ドライバ**は、ノイズ、グラウンド・バウンス、オーバーシュートを排除することで、デバイスの高速、高精度、クリーンな波形信号を提供し、プログラミング歩留まりを最大限に高めています。これにより、プログラマブル・デバイス - (E)EPROM, Flash, MRAM, PCM, ... に使用される事実上すべての不揮発性テクノロジーを単一のデバイス・プログラマで確実にサポートすることができます。

**FPGAベースの完全再構成可能なTTLピンドライバ**はデバイスの各ピンにH/L/pull\_up/pull\_down、及び、読み出し機能を提供します。デュアルH/Lドライバは追加ロジックなしでプログラムされたデバイスのコア信号とI/O信号の両方に2つの異なるHレベルを提供することができます。

プログラムのピン・ドライバは0.8Vまで動作するためプログラマは最新、及び、将来の高度な超低電圧デバイス of フルレンジをプログラムする準備が来ています。

FPGAベースのステートマシン、高速プロセッサ、及び、SSDを使用することによって達成される**非常に高速なプログラミング**によりプログラムの内部で時間に拘らず重要なルーチン、及び、データ転送が実行されます。

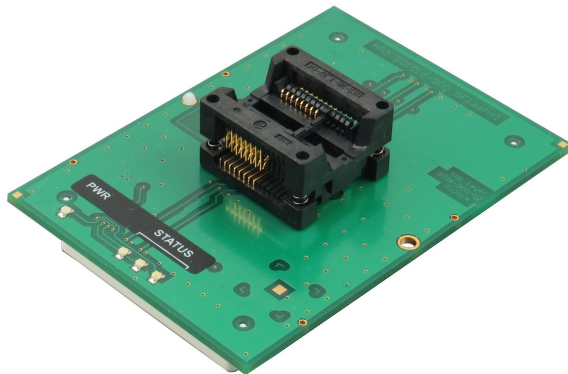
プログラマは各デバイスをプログラムする前にプログラマとプログラムされるデバイスとの間の適切な信号経路のチェックに基づいてデバイス挿入テストを実行します。プログラミング構成に依ってプログラムされるデバイスとプログラミング・モジュールのZIFソケットとの間の接触ミス、接触不良、プログラミング・モジュールとプログラマとの間の接触不良、又は、非接触を識別、ZIFソケット、又は、プログラミング・モジュール内のデバイスの位置(移動、回転、逆方向)が間違っていることを識別します。これらの機能は、**過電流保護とシグネチャバイト・チェック**によってサポートされオペレータ・エラーによるチップの損傷を防止します。

セルフテスト機能によりソフトウェアの診断部分を実行してプログラマの状態を完全にチェックすることができます。

**内蔵保護回路**は環境やオペレータの失敗によるプログラマ、及び/又は、プログラムされたデバイスの損傷を排除します。BeeProg3プログラミングコアの全入力 - プログラミング・モジュール・インタフェースのピン(ピン・ドライバ信号、及び、サポート信号)、PC、及び、電源入力への接続は最大15kVの**ESD**に対して**保護**されています。

静電気防止用リストストラップ接続用バナナジャックはESD保護制御を容易に実装できます。

プログラミング・モジュールは PDIP, PLCC, JLCC, SOIC, SDIP, SOP, PSOP, SSOP, TSOP, TSOPII, TSSOP, QFP, PQFP, TQFP, VQFP, QFN(MLF), SON, BGA, EBGA, FBGA, VFBGA, UBGA, FTBGA, LAP, CSP, SCSP, LQFP, MQFP, HVQFN, QLP, QIP や他のパッケージを含みます。。



**ノート:** ZIFソケットとデバイス・リファレンス・ピンの向きは完璧な機能と人間工学に基づいて設定されているためAP3プログラミング・モジュールで異なる場合があります。

プログラマはプルダウン・メニュー、ホット・キー、オンライン・ヘルプを備えた**快適で使いやすい**制御プログラムによって駆動されます。このソフトウェアはWindows XPからWindows 10(32bit、及び、64bit)までのバージョンのMS Windows上で動作します。

デバイスの選択はそのクラス、製造元、又は、単にベンダ名及び/又は、部品番号の一部を入力することによって実行されます。いくつかの**テスト機能**(挿入テスト、シグネチャバイト・チェック)といくつかの**特殊機能**(自動インクリメント、生産モード - ソケットへのチップ挿入直後のプログラミングの開始)によって標準デバイス関連コマンド(読み取り、ブランク・チェック、プログラム、ベリファイ、イレース)が強化されています。

全ての既知のデータ形式がサポートされています。自動ファイル形式の検出とファイル読み込み中の変換が実行されます。

豊富な機能の1つである**シリアルライゼーション機能**によりプログラムされた各デバイスに個々のシリアル番号を割り当てることができます。- 又は、単にシリアル番号をインクリメントするか、又は、その機能によりシリアル番号、又は、プログラムされたデバイス識別シグネチャをファイルから読み取り、プログラムされるデバイスにプログラムすることができます。

ソフトウェアは - デバイス情報セクション - でプログラムされるデバイスに関する多くの情報を提供します。特に、**全ての利用可能なパッケージの図面**が提供されています。ソフトウェアはまたサポートされている**各チップのチップ・マーキング(チップのプリフィックスとサフィックスの意味)**の説明も提供しています。

JEDEC標準JESD-71のJamファイルはJam Playerによって解釈されます。Jamファイルはそれぞれのプログラマブル・デバイスの製造元によって提供される設計ソフトウェアによって生成されます。チップはJTAG (IEEE 1149.1 Joint Test Action Group)インターフェースを介してプログラミングされます。

**VMEファイル**はVME Playerによって解釈されます。VMEファイルはSVFファイルの圧縮されたバイナリ形式であり、高水準のIEEE 1149.1バス操作を含んでいます。SVFファイルはSVF Playerによって解釈されます。SVFファイル(シリアル・ベクトル形式)には高水準のIEEE 1149.1バス操作が含まれています。SVFファイルはそれぞれのプログラマブル・デバイスの製造元によって提供される設計ソフトウェアによって生成されます。チップはJTAGインターフェースを介してプログラムされます。VMEファイルはそれぞれのプログラマブル・デバイスの製造元によって提供される設計ソフトウェアによって生成されます。

複数のBeeProg3プログラマを同じPC(USBポート経由で)に接続することで**BeeProg3と同じ数のチップをサポートするパワフルなマルチ・プログラミング・システム**を作成することができます。同時マルチ・プログラミング - 各プログラマは独立して動作し必要に応じて各プログラマは異なるチップをプログラムできます。

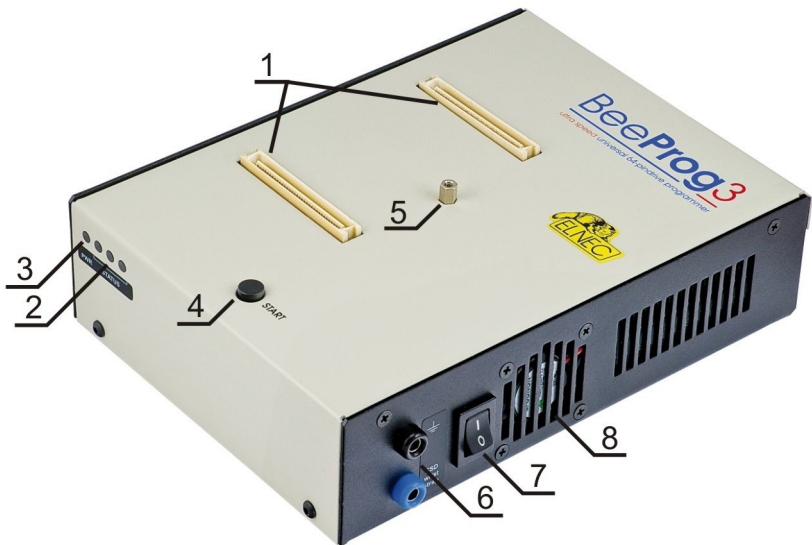
保護回路、オリジナル・ブランド・コンポーネント、慎重な製造と焼き付けを含む**BeeProg3**ユニバーサル・プログラマの先進的な設計により、プログラマの部品や製造技術に対する**3年間の保証**を提供しております。

保証は購入日から有効です。製品の登録は修理依頼を迅速化します。登録は購入日から60日以内に行う必要があります。



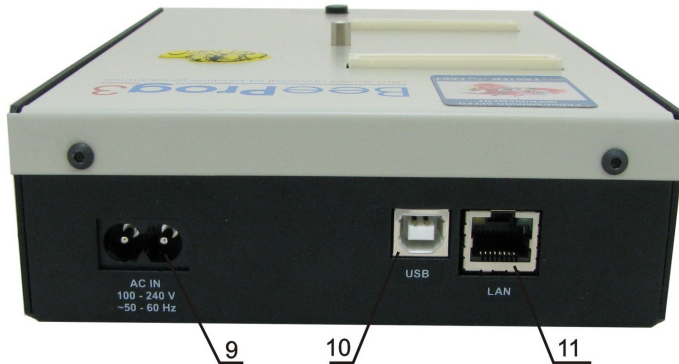
## BeeProg3 構成要素

- 1) プログラミング・モジュール・インターフェース(PMI)コネクタ
- 2) 動作結果 LED
- 3) サイトのパワー/スリープ LED
- 4) STARTボタン
- 5) プログラミング・モジュール固定ネジ
- 6) "GND"バナナ・ジャックはプログラマの接地用に使用できる"ESDリスト・ストラップ"バナナ・ジャックは静電気防止用リストストラップを取り付ける場所です。
- 7) パワー・スイッチ
- 8) 温度制御クーリング・ファン



BeeProg3の右上ビュー

- 9) 電源供給コネクタ
- 10) type B USB コネクタ PC ↔ BeeProg3 通信ケーブル
- 11) ネットワーク用LAN コネクタ ↔ BeeProg3通信ケーブル



BeeProg3の背面ビュー

## BeeProg3 を PC に接続

### USBポートの使用

プログラマとPCの接続に関する推奨事項:

1. プログラマとPC、又は、他のグランド間を接地接続
2. USBケーブルでプログラマをPCと接続
3. プログラマに電源を接続

### LANポートの使用

プログラマとPCの接続に関する推奨事項:

1. プログラマとPC、又は、他のグランド間を接地接続
2. Cat5eイーサネット・ケーブルを使用してプログラマを最寄りのネットワーク・デバイス(スイッチ、ハブ、又は、ルータ)に接続
3. ネットワーク内のDHCPサーバーが設定されていることを確認
4. プログラマに電源を接続

## プログラムされるデバイスの操作

制御プログラムでDevice/Select device (Alt+F5)を選択します。デバイス名の一部、又は、全部を入力して目的のデバイスを選択します。右側の列には希望するデバイスでの作業に必要なAP3モジュールの名前が表示されます。

AP3プログラミング・モジュールをプログラミング・モジュール・インターフェイス(PMI)コネクタに挿入します。開いているZIFソケット(ZIFの上部を押して)に目的のデバイスを挿入し、ソケット(ZIFの上部を離して)を閉じてください。ZIFソケットのプログラムされたデバイスの正しい方向はZIFソケットの近くに示されています。

#### ノート:

プログラムの電源を切る必要はなく、またソフトウェアが次の場合に実行されている可能性があります:

- プログラミング・モジュール・インターフェイス(PMI)コネクタへからのAP3プログラミング・モジュールの挿入/取り外し
- AP3プログラミング・モジュールZIFソケットへからターゲット・デバイスの挿入/取り外しを行います。ターゲット・デバイスの操作は無効です(LED BUSYライトが消灯)。

プログラムの電子保護はターゲット・デバイスとプログラム自身を短期、又は、長期の停電から保護し一部はPCの障害からも保護します。しかし、誤ったユーザー選択したプログラミング・パラメータにターゲット・デバイスの完全性を与えることは不可能です。ターゲット・デバイスはプログラムへの物理的な接続を取り除くことによって制御プログラムの強制的な中断(PCのリセット、又は、スイッチオフ)によって破壊されることはありませんが、実際にプログラムされたセルの内容は不定義のままです。

## BeeProg3 によるマルチ・プログラミング

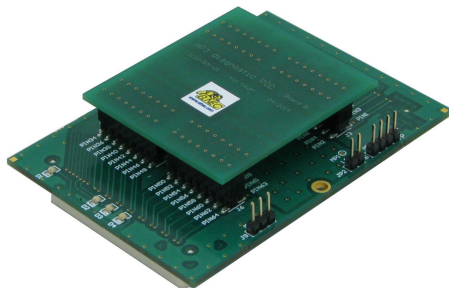
PG4UWのインストール中に[Select Additional Tasks]ウィンドウをチェックし、BeeProg3マルチ・プログラミング制御サポートをインストールすることが許可されているかどうかを確認します。BeeProg3マルチ・プログラミングを開始するには特別な制御プログラムpg4uwmc.exeを実行する必要があります。このプログラムでユーザはプログラムを制御するためにBeeProg3を割り当て、全てのBeeProg3のプロジェクトをロードし接続され割り当てられたBeeProg3ごとにPG4UWを実行することができます。

## セルフテストとカリブレーション・チェック

プログラマが期待通りに動作しないと感じる場合は、標準パッケージに同梱されているAP3診断PODを使用してプログラマの"Selftest"を実行してください。

### プログラマ・サイトのセルフテスト

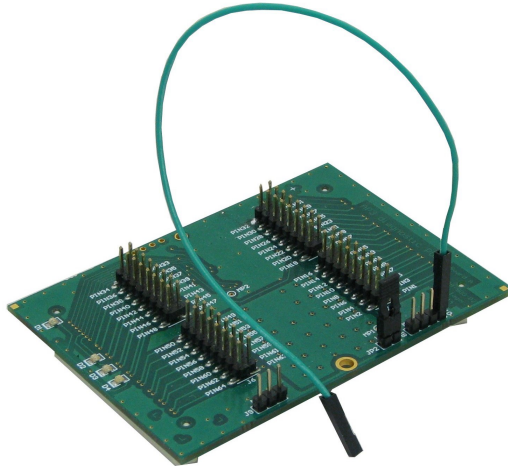
- ボードAをAP3診断PODのボードBに接続
- プログラマ・サイトのプログラミング・モジュール・インタフェース(PMI)コネクタに**AP3診断POD**を挿入します。
- プログラマのセルフテストをPG4UW(メニュー Programmer/Selftest)で実行します。



Selftest plus[セルフテスト・プラス]のためのAP3 diagnostic POD

### カリブレーション・テスト

- ボードAをAP3診断PODのボードBから外します。
- プログラム・サイトのプログラミング・モジュール・インタフェース(PMI)コネクタにAP3診断PODのボードAを挿入します。
- PG4UW(メニュー Programmer/Calibration test)でプログラムの校正テストを実行します。



カリブレーション・テストのためのAP3診断POD

## テクニカル・スペシフィケーション

### ハードウェア

#### ベース・ユニット, DACs

- USB 2.0ハイスピード互換ポート、最大480Mb/s転送レート
- 100Mbit LANポート
- オンボード・インテリジェンス: 強力なプロセッサ(ARM9 400MHz)とFPGAベースのステートマシン(基本クロック50MHzプラスPLL)
- 内部バッファ(1282GB、大容量にアップグレード可能)として内蔵mSATA SSD (\*)
- VCC1, VCC2とVPP用の3つのD/Aコンバータ、制御可能な立上りと立下り時間
- VCC1, VCC2 範囲 0.8V..7V/1A (10mVステップ)
- VPP 範囲 0V..25V/1A (25mVステップ)
- 温度制御されたファン
- セルフテスト機能
- 電源入力に対するサージとESDに対する保護、プログラミング・モジュール・インタフェース(PMI)(IEC1000-4-2: 15kV空気、8kV接点)のUSBとLANポートとピン
- ESD[静電気防止用]リスト・ストラップ接続用のバナナジャック
- グランド接続のためのバナナ・ジャック

**(\*) ノート:**

- バッファ内のデータはフラグメントに格納され、そして、また部分的に圧縮されるため、バッファは現在のバッファ・サイズより大きなサイズのデバイスについても標準データを持てます。
- このアップグレードは機械的に複雑でEl necでのみ実行できます。
- s/n 1200-00053(を含む)までのプログラマは内部バッファとして32GBのSSDを提供されました。
- 128GB SSDは比例してより大きいサイズのSSDに対して約110GBのランダム・データ(32GB SSDの場合は28GB)をバッファとして保存できます。デバイス・サイズ的全範囲でランダムなデータを含む可能性があるeMMC/NANDデバイスを作業/コピーする予定の場合は、プログラマに等しい容量のバッファを使用することをお勧めします。

**ピン・ドライバ(プログラミング・モジュールのインターフェイスで利用可能 - プログラミング・モジュール用のコネクタ)**

- ピン・ドライバ: 64 ユニバーサル
- VCC1/VCC2とVPPは各ピンに接続することが出来ます。
- 各ピンの完璧なグラウンド
- 2つの独立したFPGAベースのTTLドライバは全てのピンドライバ・ピンでH, L, CLK, プルアップ, プルダウン, ロジック・レベル 0.75V - 5V(IOLとIOH電流20mA)を提供します。
- ロジック信号周波数: 最大125MHz (3.3V), 80MHz (5V)
- 0.8V~25Vのアナログ・ピンドライバ出力レベルを選択可能
- 電流制限、過電流シャットダウン、停電シャットダウン
- 導通テスト: 各ピンはプログラミング動作の前にテストされます。

**デバイス・サポート****プログラマ, プログラミング・モジュールを使用:**

- NAND FLASH: Samsung K9xxx, KFxxx, SK Hynix (ex Hynix) HY27xxx, H27xxx, Toshiba TC58xxx, TH58xxx, Micron MT29Fxxx, (ex Numonyx ex STM) NANDxxx, Spansion S30Mxxx, S34xxx, 3D-Plus 3DFNxxx, ATO Solution AFNDxxx, Fidelix FMNDxxx, Eon Silicon Sol. EN27xxx, ESMT F59xxx, LBA-NAND Toshiba THGVNxxx
- serial NAND FLASH: Micron MT29Fxxx, GigaDevice GD5Fxxx
- eMMC: Hynix H26Mxxxxxxxx, Kingston KE44B-xxxx/xxx, Micron MTFCxxxxxx, Numonyx NANDxxxxxxxx, Phison PSM4A11-xx, Samsung KLMxxxxxxxx, SanDisk SDINxxx-xx, Toshiba THGBMxxxxxxxx
- Multi-chip devices: NAND+RAM, NOR+RAM, NOR+NOR+RAM, NAND+NOR+RAM
- Serial Flash: standard SPI, high performance Dual I/O SPI and Quad I/O SPI (25Bxxx, 25Dxxx, 25Exxx, 25Fxxx, 25Lxxx, 25Mxxx, 25Pxxx, 25Qxxx, 25Sxxx, 25Txxx, 25Uxxx, 25Vxxx, 25Wxxx, 25Xxxx, 26Vxxx, 45PExx, MX66Lxxx, S70FLxxx), DataFlash (AT45Dxxx, AT26Dxxx)
- parallel NOR Flash: 28Fxxx, 29Cxxx, 29Fxxx, 29GLxxx, 29BVxxx, 29LVxxx, 29Wxxx, 49Fxxx series, Samsung's K8Fxxxx, K8Cxxxx, K8Sxxxx, K8Pxxxx series, ...
- EPROM: NMOS/CMOS, 27xxx and 27Cxxx series
- EEPROM: NMOS/CMOS, 28xxx, 28Cxxx, 27EExxx series, 3D Plus 3DEExxxxxxxx
- mDOC H3: SanDisk (ex M-Systems) SDED5xxx, SDED7xxx, MD2533xxx, MD2534xxx, Hynix HY23xxx
- FRAM: Ramtron
- MRAM: Everspin MRxxxxx8x, 3D Plus 3DMRxxxxxxxx
- NV RAM: Dallas DSxxx, SGS/Inmos MKxxx, SIMTEK STKxxx, XICOR 2xxx, ZMD

## U63x series

- Serial E(E)PROM: Serial E(E)PROM: 11LCxxx, 24Cxxx, 24Fxxx, 25Cxxx, 30TSExxx, 34Cxxx, 34TSxx, 59Cxxx, 85xxx, 93Cxxx, NVM3060, MDAxxx series, full support for LV series, AT88SCxxx
- Serial FRAM: Cypress(Ramtron): FM24xxxxxx, FM25xxxxxx, Fujitsu: MB85RCxxxx, MB85RSxxxx, Lapis(OKI, Rohm): MR44xxxxx, MR45xxxxx
- Serial MRAM: Everspin MH20xxx, MH25xxx
- Configuration (EE)PROM: XCFxxx, XC17xxxx, XC18Vxxx, EPCxxx, EPCSxxx, AT17xxx, AT18Fxxx, 37LVxx
- 1-Wire E(E)PROM: DS1xxx, DS2xxx
- PLD Altera: MAX 3000A, MAX 7000A, MAX 7000B, MAX 7000S, MAX7000AE, MAX II/G/Z, MAX V
- PLD Lattice: ispGAL22V10x, ispLSI1xxx, ispLSI1xxxEA, ispLSI2xxx, ispLSI2xxxA, ispLSI2xxxE, ispLSI2xxxV, ispLSI2xxxVE, ispLSI2xxxVL, LC4xxxB/C/V/ZC/ZE, M4-xx/xx, M4A3-xx/xx, M4A5-xx/xx, M4LV-xx/xx, ispCLOCK, Power Manager/II, ProcessorPM
- PLD Xilinx: XC9500, XC9500XL, XC9500XV, CoolRunner XPLA3, CoolRunner-II
- SPLD/CPLD series: AMD, AML, Atmel, Cypress, Gould, ICT, Lattice, National Semicond., Philips, STMicroelectronics, TI (TMS), Vantis, VLSI
- FPGA: FPGA: Microsemi(Actel): ProASIC3, IGL00, Fusion, ProASICplus, SmartFusion
- FPGA: Lattice: MachXO, MachXO2, LatticeXP, LatticeXP2, ispXPGA
- FPGA: Xilinx: Spartan-3AN
- Clocks: TI(TMS), Cypress
- Special chips: Atmel Tire Pressure Monitoring ATA6285N, ATA6286N; PWM controllers: Zilker Labs, Analog Devices; Multi-Phase ICs: IR(Chil Semiconductor); Gamma buffers: AUO, Maxim, TI, ...
- Microcontrollers MCS51 series: 87Cxxx, 87LVxx, 89Cxxx, 89Sxxx, 89Fxxx, 89LVxxx, 89LSxxx, 89LPxxx, 89Exxx, 89Lxxx, all manufacturers, Philips LPC series
- Microcontrollers Atmel ARM. AT91SAM7Sxx, AT91SAM7Lxx, AT91SAM7Xxx, AT91SAM7XCxx, AT91SAM7SExx series;
- Microcontrollers Atmel ARM9: AT91SAM9xxx series;
- Microcontrollers ARM Cortex-M3: ATSAM3Axxx, ATSAM3Uxxx, ATSAM3Nxxx, ATSAM3Sxxx, ATSAM3D20, ATSAM3Xxxx series
- Microcontrollers ARM Cortex-M4: ATSAM4Sxxx series
- Microcontrollers Atmel AVR 8bit/16bit: AT90Sxxxx, AT90pwm, AT90can, AT90usb, ATtiny, ATmega, ATxmega series
- Microcontrollers Atmel AVR32: AT32UC3xxxx, ATUCxxxD3/D4/L3U/L4U series
- Microcontrollers TI (Chipcon): CC11xx, CC24xx, CC25xx, CC85xx series
- Microcontrollers Coreriver: Atom 1.0, MiDAS1.0, 1.1, 2.0, 2.1, 2.2, 3.0 series
- Microcontrollers Cypress: CY7Cxxxxx, CY8Cxxxxx
- Microcontrollers ELAN: EM78Pxxx
- Microcontrollers EPSON: S1C17 series
- Microcontrollers Explore Microelectronic: EPF01x, EPF02x series
- Microcontrollers Generalplus: GPM8Fxxx series
- Microcontrollers GreenPeak: GPxxx series
- Microcontrollers Infineon(Siemens): XC800, C500, XC166, C166 series
- Microcontrollers MDT 1xxx and 2xxx series
- Microcontrollers Megawin: MG87xxx, MPC82xxx series
- Microcontrollers Microchip PICmicro: PIC10xxx, PIC12xxx, PIC16xxx, PIC17Cxxx, PIC18xxx, PIC24xxx, dsPIC, PIC32xxx series
- Microcontrollers Motorola/Freescale: HC05, HC08, HC11, HC12, HCS08, RS08, S12, S12X, MC56F, MCF51, MCF52 series, Kinetis (K,L), Qorivva/5xxx Power Architecture
- Microcontrollers Myson MTV2xx, 3xx, 4xx, 5xx, CS89xx series

- Microcontrollers National: COP8xxx series
- Microcontrollers NEC: uPD70Fxxx, uPD78Fxxx series
- Microcontrollers Novatek: NT68xxx series
- Microcontrollers Nordic Semiconductor: nRF24LExxx, nRF24LUxxx, nRF315xx, nRF51xxx Flash and OTP series
- Microcontrollers Nuvoton ARM Cortex-Mx: NUC1xx, M05x, Mini51, Nano1xx series
- Microcontrollers Nuvoton (Winbond): N79xxx, W77xxx, W78xxx, W79xxx, W83xxx series
- Microcontrollers NXP (Philips) ARM Cortex-Mx: LPC11xx, LPC11Cxx, LPC11Dxx, LPC11Uxx, LPC12xx, LPC12Dxx, LPC13xx, LPC17xx, LPC11Axx, LPC11Exx, LPC11xxLV, LPC18xx, LPC43xx, LPC8xx, EM7xx, series
- Microcontrollers NXP (Philips) UOC series: UOCIII, UOC-TOP, UOC-Fighter (TDA1xxxx) series
- Microcontrollers NXP (Philips) ARM7: LPC2xxx, MPT6xx, PCD807xx, SAF7780xxx series
- Microcontrollers NXP (Philips) ARM9: LPC31xx series
- Microcontrollers Pasat: TinyModule DIL40, DIL50 series
- Microcontrollers Scenix (Uvicom): SXxxx series
- Microcontrollers Syntek: STK6xxx series
- Microcontrollers Renesas: R8C/Tiny series, RX series, uPD70Fxxx, uPD78Fxxx series, RL78 series, R32C series
- Microcontrollers SyncMOS: SM39xxx, SM59xxx, SM73xxx, SM79xxx, SM89xxx series
- Microcontrollers & Programmable System Memory STMicroelectronics: uPSD, PSD series
- Microcontrollers STM (ex SGS-Thomson): ST6xx, ST7xx, ST10xx, STR7xx, STR9xx, STM32F/L/W, STM8A/S/L series, SPC5 (Power Architecture)
- Microcontrollers Silicon Laboratories(Cygnal): C8051 series
- Microcontrollers Silicon Laboratories(Energy Micro): EFM32Gxx, EFM32GGxx, EFM32LGxx, EFM32TGxx, EFM32WGxx series
- Microcontrollers Silicon Laboratories: SiM3Cxxx, SiM3Lxxx, SiM3Uxxx series
- Microcontrollers Texas Instruments: MSP430 series, MSC12xx series, TMS320F series, CC430 series,
- Microcontrollers Texas Instruments (ex Luminary Micro): LM3Sxxx, LM3Sxxxx series, LM4Fxxxx series, TM4C series
- Microcontrollers ZILOG: Z86/Z89xxx and Z8Fxxxx, Z8FMCxxxxx, Z16Fxxxx, ZGP323xxxxxx, ZLF645xxxxxx, ZLP12840xxxxx, ZLP323xxxxxx series
- Microcontrollers other: EM Microelectronic, Spansion(Fujitsu), Goal Semiconductor, Hitachi, Holtek, Novatek, Macronix, Princeton, Winbond, Samsung, Toshiba, Mitsubishi, Realtek, M-Square, ASP, Coreriver, Gencore, EXODUS Microelectronic, Topro, TinyARM, VersaChips, SunplusIT, M-Square, QIXIN, Signetic, Tekmos, Weltrend, Amic, Cyrod Technologies, Ember, Ramtron, Nordic Semiconductor, Samsung, ABOV Semiconductor...

#### ノート:

- サポートされているその時点でのデバイスについては

<https://www.eltec.com/en/search/device-list/beeprog3/> の実際のデバイス・リストを参照してください

## パッケージ・サポート

•DIP,SDIP,PDIP,PLCC,JLCC,SOIC,SDIP,SOP,PSOP,SSOP,TSOP,TSOPII,TSSOP,QFP,PQFP,TQFP,VQFP,QFN(MLF),SON,BGA,EBGA,FBGA,VFBGA,UBGA,FTBGA,LA P,CSP,SCSP,LQFP,MQFP,HVQFN,QLP,QIPや他のパッケージを含みます。

## プログラミング・スピード

### ノート:

- プログラムの速度テストには常に乱数データ・パターンを使用することが重要です。競合会社の中には少量の非ランク・データのみがプログラムされているか、又は、僅かなビット数(FE、EF等)のデータが使用されている"スカース・データ・パターン"を使用するものがあります。この不正行為のアプローチはプログラミング時間をかなり短縮することができます。プログラマーの速度を比較したい場合は常にどのパターンを使用するかを尋ねておいて下さい。
- プログラムのためのデータとプログラミング・アルゴリズムの主要部分がプログラムの内部に格納されているためプログラミング速度は実際にはPCの種類には余り依存しません。
- 2つのNAND Flashを含む下記の全てのデバイスが最大速度でプログラムされプログラミング時間は短くできません。
- 申し訳ありませんが、現時点では20+MB/sのBeeHive304プログラミング速度を利用できるデバイスは余り多くはありません。

Device	Size [bits]	Operation	Time
JS28F00AM29EWH (parallel NOR Flash)	4000080hx16 (1 Giga)	programming and verify	10 sec
MT29F1G08ABAEAWP (parallel NAND Flash)	8400000hx8 (1 Giga)	programming and verify	19.5 sec
SDIN7DP2-8G (eMMC NAND FLASH)	1D2000000hx8 (64Giga)	programming *1	342 sec
S25FL164K (serial Flash)	800300hx8 (64 Mega)	programming and verify	30.7 sec
AT89LP51RD2 (microcontroller)	10000hx8	programming and verify	5.2 sec
PIC32MX360F512L (microcontroller)	80000hx8	programming and verify	8.9 sec

条件: Core2 Duo, 3.16 GHz, 1G RAM, USB 2.0 HS, Windows 7. Version of SW: 3.09, 10/2014.

\*1 プログラムの検証は、eMMCデバイスの内部コントローラによって行われます。デバイスはデータとCRCを加えたブロックを受信し、一致した場合、内部コントローラは適切なプログラミングを確認します。

## ソフトウェア

- **アルゴリズム:** マニ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:** ソフトウェ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。**OnDemand[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は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**
  - インサクション・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(Act1 STAPL file)

## 一般仕様

- 操作電圧 100-240V AC, 90-264 VAC 最大, 47-63Hz.
- 消費電力 最大. 25W アクティブ

- **BeeProg3の寸法** プログラム: 139.5 x 189.5 x 48.4 mm (5.5 x 7.6 x 1.9 inch). 寸法は

プログラミング・モジュールを挿入しない状態での測定、プロジェクションを含みません。プログラミング・モジュールを挿入したBeeProg3プログラマの全高は、ZIFソケットの高さに依存し66～78mmの間で変化します。

- 重さ(プログラミング・モジュールを含まない): 1.25kg (2.7 lb)
- 操作温度: 5°C ÷ 40°C (41°F ÷ 104°F)
- 操作湿度: 20%..80%, 非結露

セットアップ

プログラマ・パッケージには制御プログラム、有用なユーティリティーと追加情報を含むCDが入っています。El necのプログラマの働きを示すためにデモンストレーションするためにCDの内容を自由にコピーする許可が与えられています。

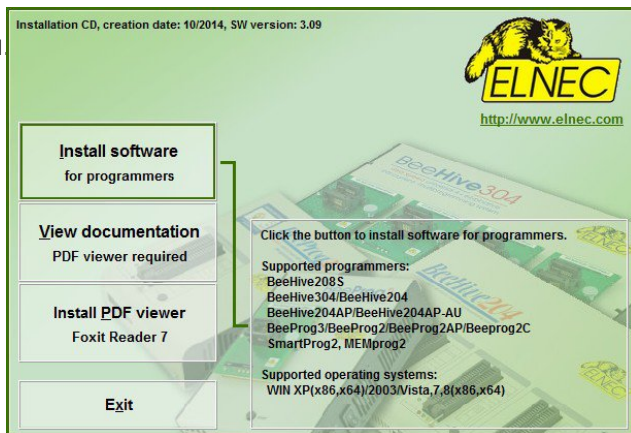
USBポート経由で接続されたプログラマの場合、制御プログラムには正しくインストールされたUSBドライバが必要です。

インストール時に不要な複雑さを避けるためにプログラマをPCに接続する前にソフトウェアをインストールすることを推奨します。

### ソフトウェア・セットアップ

配布されたCDをCDドライブに挿入しますとプログラムは自動的(インストールされていない場合はsetup.exeを実行します)にインストールします。インストールプログラムはインストールプロセスをガイドし、最初にコントロール・プログラムを実行する前に必要なすべての手順を実行します。

#### Step 1.



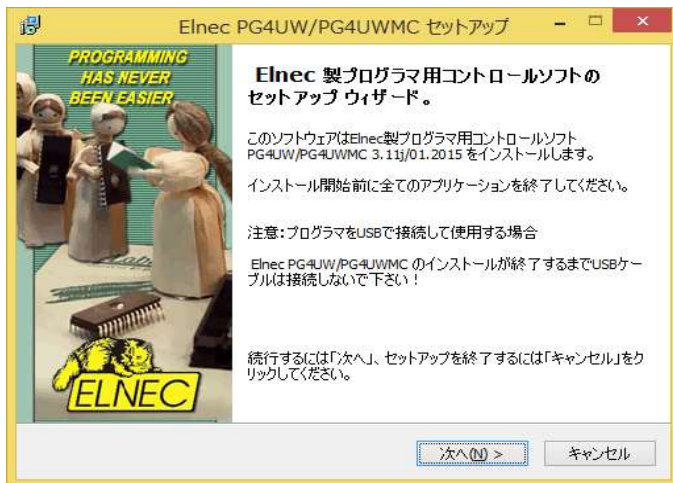
“Install software for programmutton“をクリック

#### Step 2.



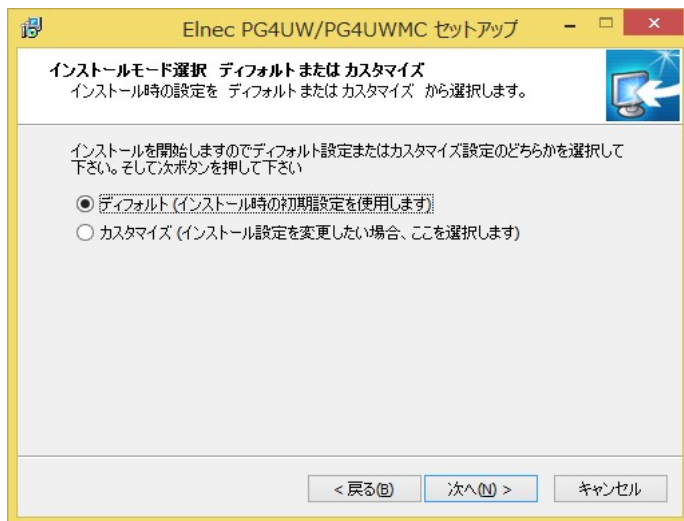
言語を選択し“OK”ボタンをクリック

Step 3.

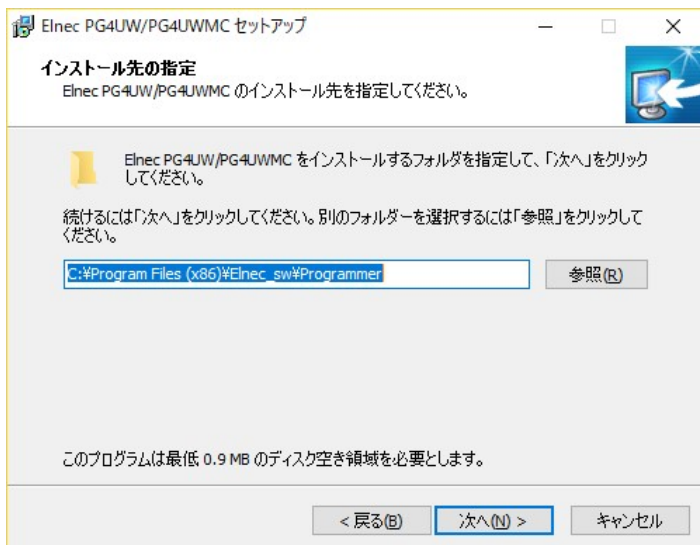


“Next”ボタンをクリック

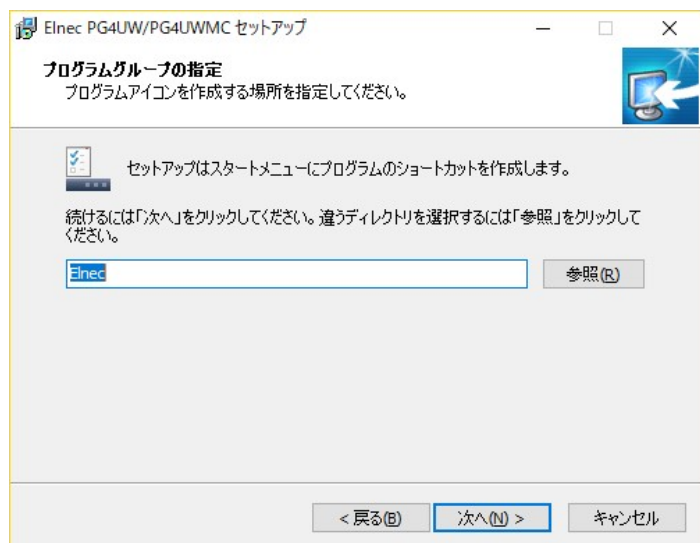
Step 4.



デフォルト設定では“次へ”ボタンをクリックします。セットアップはStep 6に進みます。デフォルト設定を変更するには“カスタム”をクリックしてから“次へ”ボタンをクリックします。

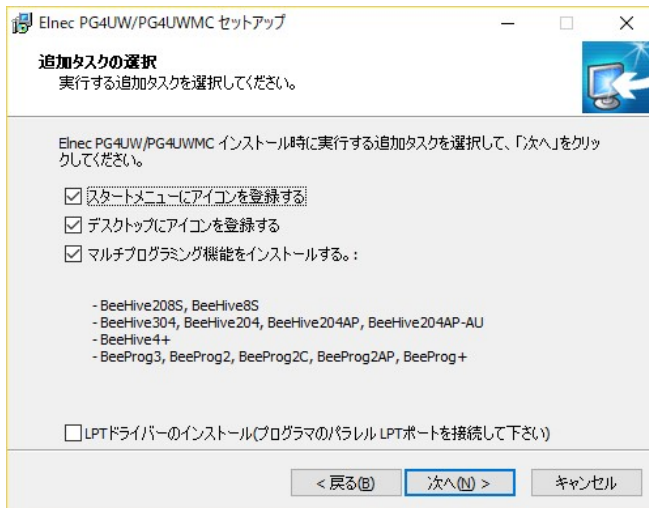
**Step 5.**

デフォルトのフォルダを変更するには"参照"ボタンをクリックして下さい。  
そして、"次へ"ボタンをクリックします。

**Step 6.**

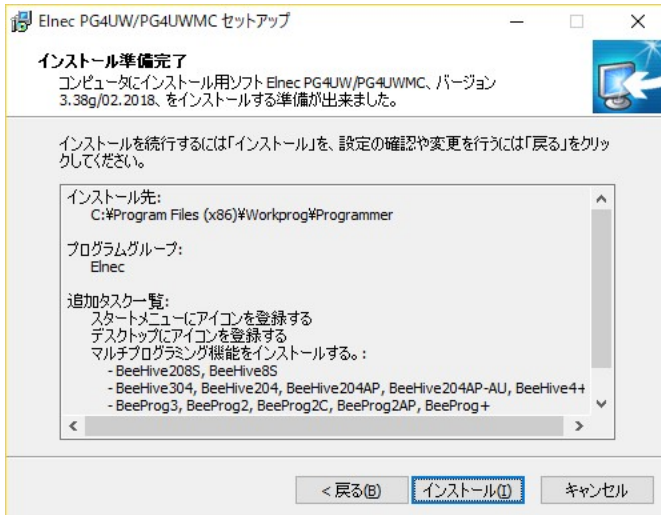
デフォルトのフォルダを変更するには"参照"ボタンをクリックして下さい。そして、"次へ"ボタンをクリックします。

### Step 7.



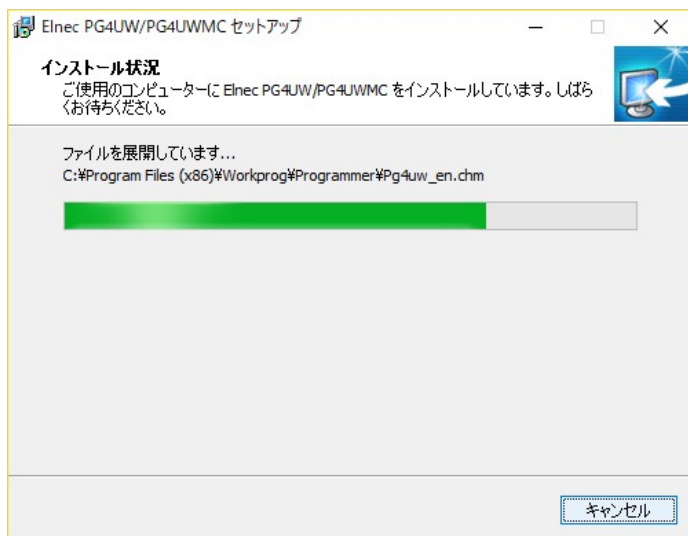
"マルチ・プログラミング機能をインストールする"が選択されていることを確認します。必要に応じてデフォルト設定を変更します。そして、"次へ"ボタンをクリックします。

### Step 8.



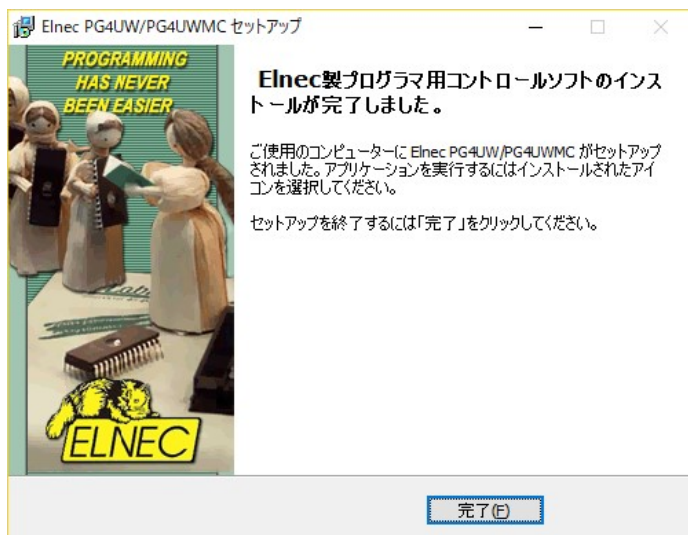
設定を確認の上、“インストール”ボタンをクリックして下さい。

## Step 9.



インストール・プロセスの開始

## Step 10.



セットアップを終了するには「完了」ボタンをクリックして下さい。



## ソフトウェアの新しいバージョン

常にプログラム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  
のソフトウェアをダウンロードされることをお勧めします。  
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ですが、そのバージョンの最初のバージョンに過ぎません。

ダウンロードしたソフトウェア・ファイルを一時ディレクトリにコピーしPCからプログラムを切断して起動します。  
セットアップは前の章の**Step 2**から開始します。

## ハードウェア・セットアップ

**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ムを 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: プログラムの通信量が多いため各プログラムを 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ムを分離した EHCI に接続できるようにします。一般的にはプログラムを 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。

プログラムは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 が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。プログラム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 ケーブルをプログラムの B タイプ USB ポートに直接接続します。
- Step 2.  
USB ケーブルを PC のタイプ A USB2.0 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ム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が約 30 秒点灯した後 LEDが消灯します。

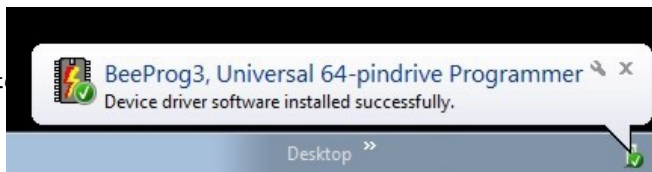
### Windows 7, 8, 10

#### Step 5.

タスク・バーの通知領域



プログラマのた



*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」[デバイス・ドライバ・ソフトウェアのインストール]が再度起動します。同じプログラマが他の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.

Windowsは「新しいハードウェアの検出ウィザード」から始まります。



"No, not this time[いいえ、今ではありません]"を選択し、"Next[次へ]"ボタンをクリックして下さい。

### Step 6.



“Install the software automatically[ソフトウェアを自動的にインストールする]”を選択し、“Next[次へ]”ボタンをクリックします。

### Step 7.



ウィザードがプログラムを検索しドライバーのインストールを自動的に開始します。

**Step 8.**

プログラマが正常にインストールされると、次のウィンドウが表示されます：



セットアップを終了するために"Finish"ボタンをクリック

**Step 9.**

"Found new hardware wizard"が各プログラマ(プログラマ・サイト)に対して1回起動されます (BeeHive304の場合は4回)。Step 5でハードウェアの設定を続けます。

**ノート:**

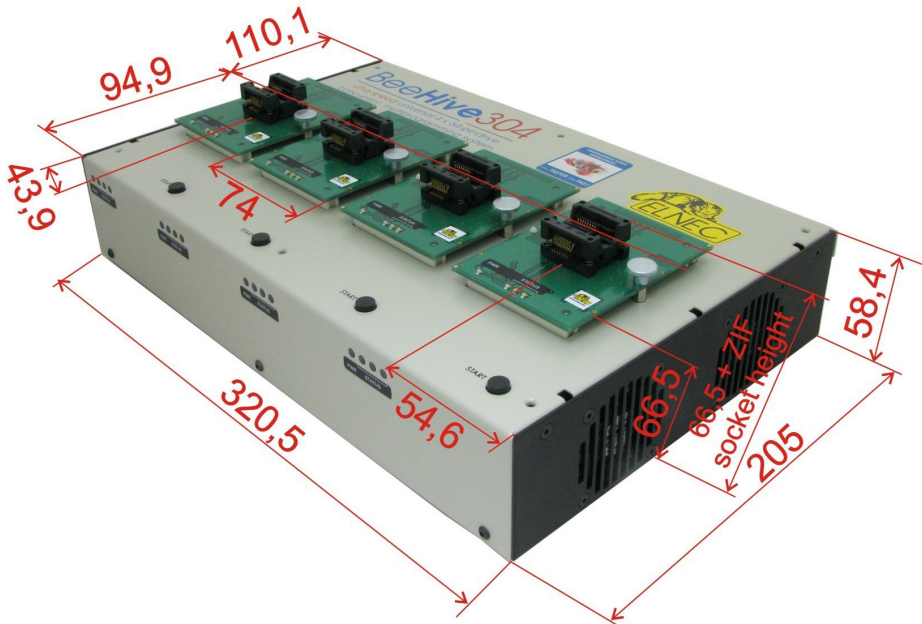
Windows XPではPC上の別のUSBポートをプログラマの次の接続に使用すると、"Found new hardware wizard"が再度起動し新しいUSBドライバがインストールされます。

## ターゲット・システムへのBeeHive304/BeeProg3プログラムのインストール

この章ではBeeHive304とBeeProg3のメカニカルな図とターゲット・システムへのBeeHive304とBeeProg3のインストールに必要なその他の情報を含みます。

### BeeHive304

以下の図はプログラミング・モジュールを使用したBeeHive304の全体的な寸法です。プログラミング・モジュールがインストールされているプログラムの全高はプログラミング・モジュールのZIFソケットの高さによって異なります。合計の高さは**66.5mm+ZIFソケットの高さ**によって決めることができます。



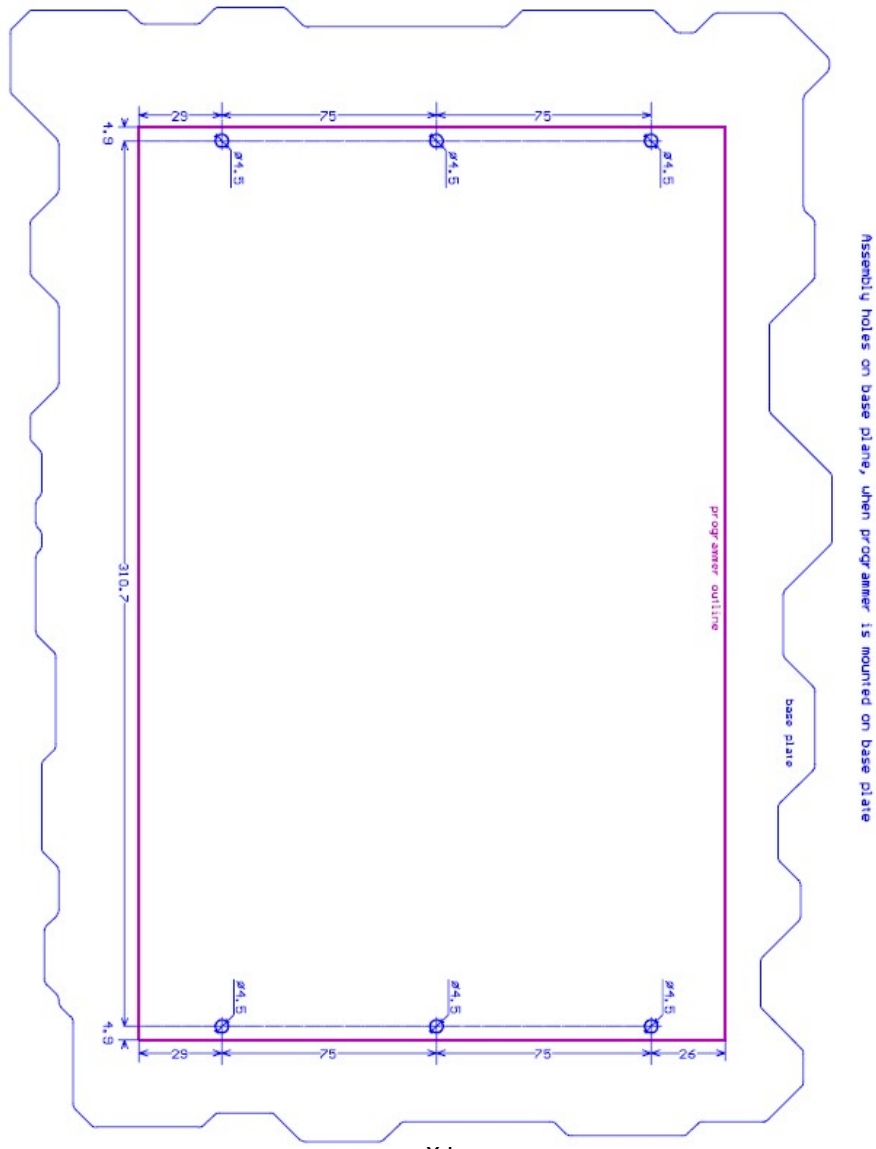
BeeHive304寸法 右上図

BeeHive304と他のオブジェクトとの最小距離は2cmです。BeeHive304の良好な冷却とその周囲の換気が必要です。

X軸とY軸でZIFの中心はプログラミング・モジュールの中心と同じです。

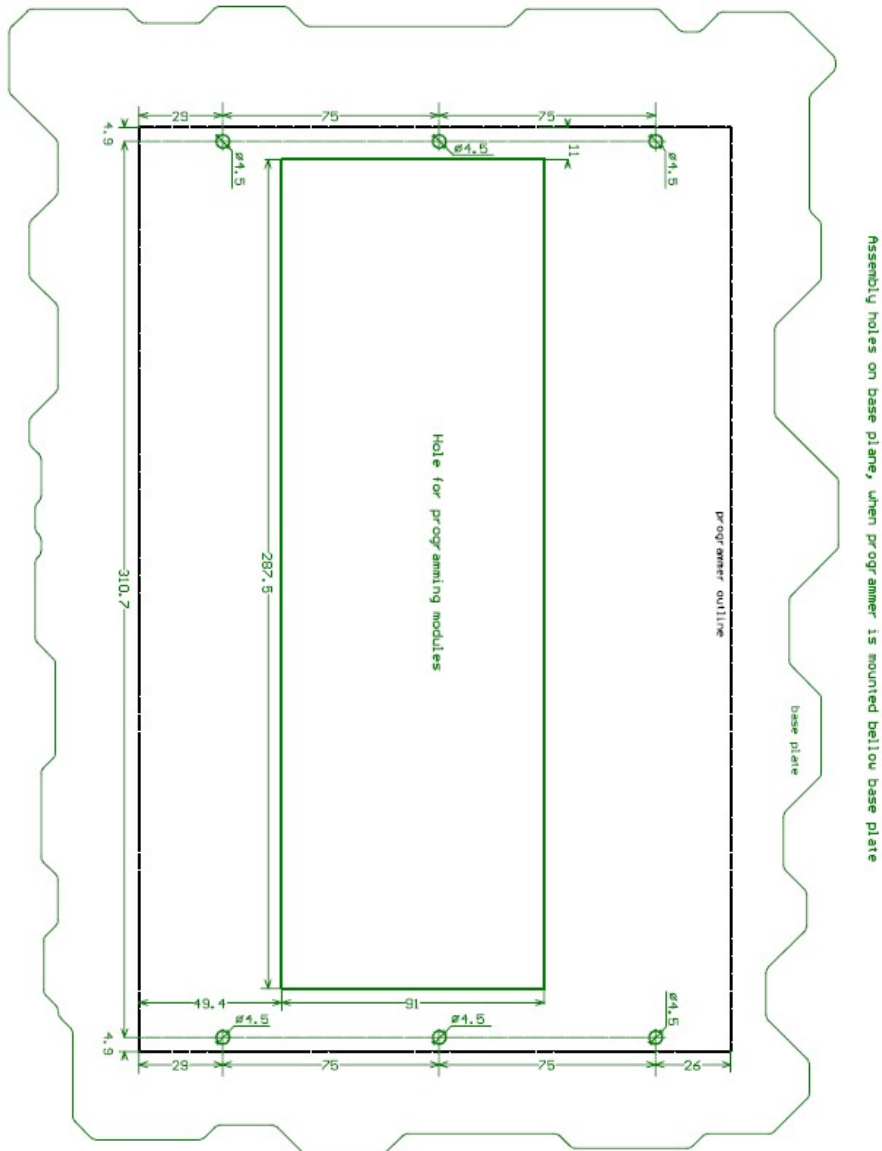
**BeeHive304**はナット付き6個のネジM4でベースプレートに取り付けることができます。ネジの長さはベースプレートの厚さによって異なります。ベースプレートにはφ4,5mmの穴をM4の内部ネジで置き換えることができます。

プログラマをベースプレートに取り付けるための図面:



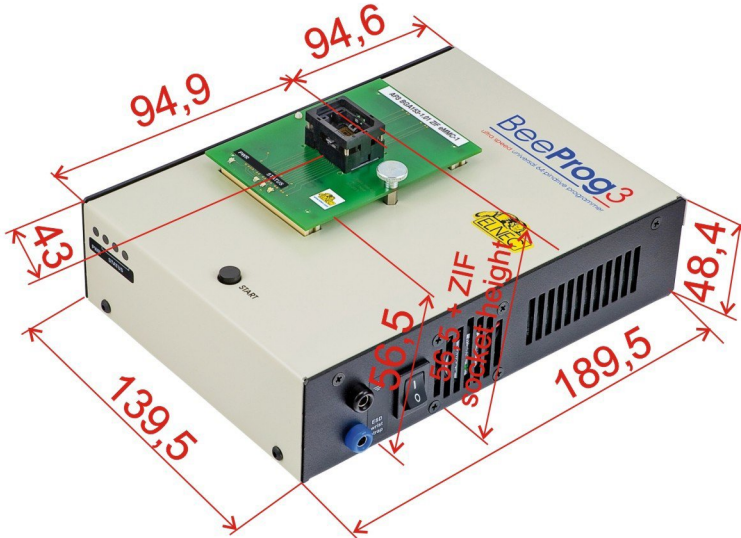
**BeeHive304** はベースプレート6個のネジM4の下にナットで取り付けることができます。ネジの長さはベースプレートの厚さによって異なります。ベースプレートにはφ4,5mmの穴をM4の内部ネジで置き換えることができます。

ベースプレートの下にプログラマを実装するための図面:



### BeeProg3

以下の図はプログラミング・モジュールを使用したBeeProg3の全体的な寸法です。プログラミング・モジュールがインストールされているプログラムの全高はプログラミング・モジュールのZIFソケットの高さによって異なります。合計の高さは**56.5mm + ZIFソケットの高さ**で決定できます。



BeeProg3の寸法 右上図

BeeProg3と他のオブジェクトとの最小距離は2cmです。BeeHive304の良好な冷却とその周囲の換気が必要です。

X軸とY軸でZIFの中心はプログラミング・モジュールの中心と同じです。

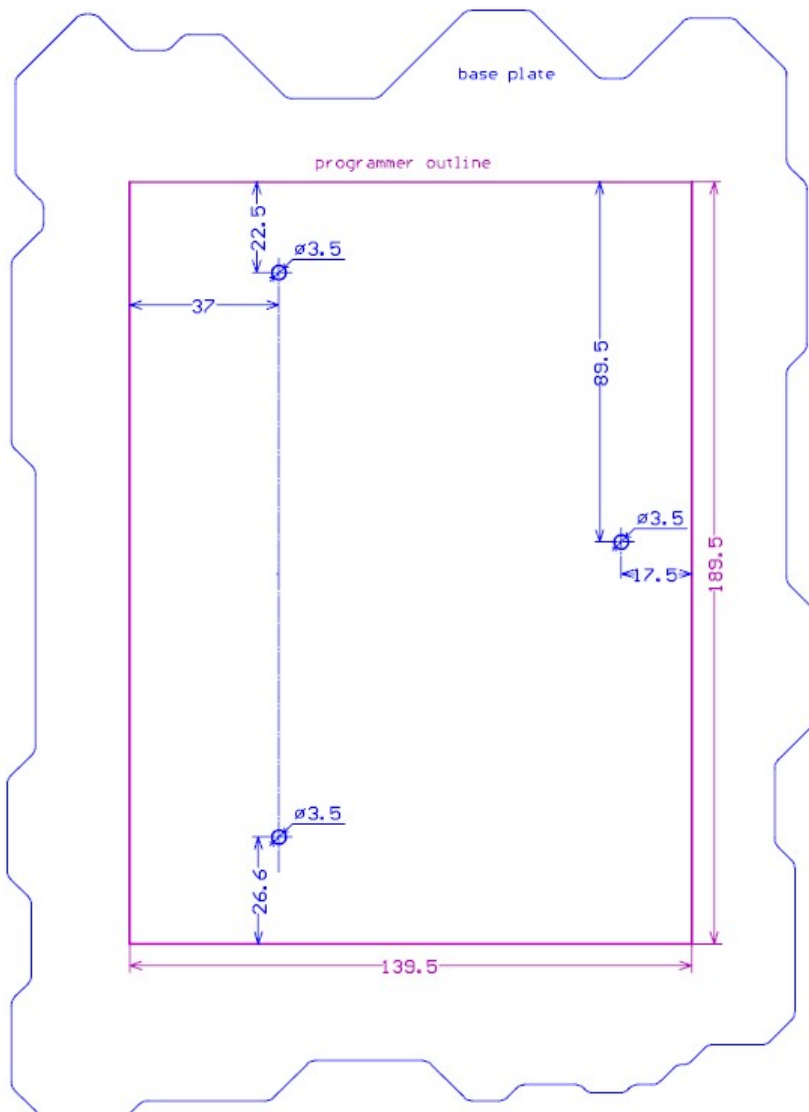


**BeeProg3** は3本のネジM3でベースプレートに取り付けることができます。ネジの長さはベースプレートの厚さによって異なります。

**注意:** ベースプレートにプログラマを正しく取り付けるにはプログラマに取り付けネジの深さを最小3mmにする必要があります。プログラマのPCBの損傷を避けるために取り付けネジはプログラマで最大8mmの深さにする必要があります。

プログラマをベースプレートに取り付けるための図面:

Assembly holes on base plane, when programmer is mounted on base plate



---

---

*PG4UWソフトウェア*

## PG4UWプログラマ・ソフトウェア

プログラムPG4UWはエルネックの全てプログラマにとって共通の制御プログラムです。前述の全てのオペレーティング・システムでこれらのプログラマの実行を保証します。Windowsでのバック・グラウンド操作にエラーはありません。

## プログラマ・ソフトウェアの使用

パッケージのCDに含まれているElneqによって提供されたコントロール・プログラムは納品時にウイルスがないことを確認されています。安全性を高めるためにプログラムにはウイルス感染の可能性を検出する特別なアルゴリズムが含まれています。

## コントロール・プログラマの実行

Windows 環境で: PG4UWアイコンをダブル・クリックします。



起動後、制御プログラムPG4UWは自動的に既存の全てのポートをスキャンし接続されたElneqプログラマを検索します。プログラムPG4UWは全てのElneqプログラマに共通しているためサポートされている全てのプログラマを検索します。

**ノート:** PG4UWが起動すると、プログラマは完全性のために自己チェックを実行します。成功するとプログラマは標準のユーザー・メニューを表示し、指示を待ちます。

制御プログラムがプログラマと通信できない場合は、エラーコードと考えられる理由(プログラマの切断、接続不良、電源障害、互換性のないポート等)の説明を含むエラー・メッセージが画面に表示されます。エラーの原因を取り除き、いずれかのキーを押します。エラー状態が依然として存在する場合、プログラマはデモ・モードで動作を再開し、プログラマへのアクセスは不可能です。エラーの原因が見つからない場合は、**トラブル・シューティング**のセクションの手順に従ってください。さらに、制御プログラムはプログラマされるデバイスでの操作の前にプログラマとの通信をチェックします。

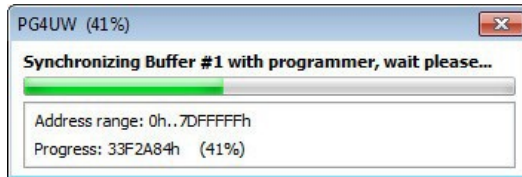
## 内部データ・ディスクを持つプログラマのためのバッファ同期

バッファ同期操作は以下の状況で内部データディスクを持つ高度なプログラマに対して自動的に実行されます:

- 最初のデバイス操作の前
- デバイス読み出し動作後
- 新しいデバイスを選択した後
- プロジェクトをロードした後
- PG4UW制御プログラムでプログラマを見つけた後

**バッファ同期**はPC(PG4UWを実行しているホスト・コンピュータ)のバッファから内部プログラマ・ディスク(又は、デバイスを読み込んだ後、プログラマ・ディスクからPCへ)デバイスに必要なデータをコピーします。この機能はデバイス動作中にデータが転送されないためプログラマ全体でプログラミング/ベリファイ動作が実行されているためデバイス動作のプログラマ/ベリファイのパフォーマンスを大幅に向上させます。

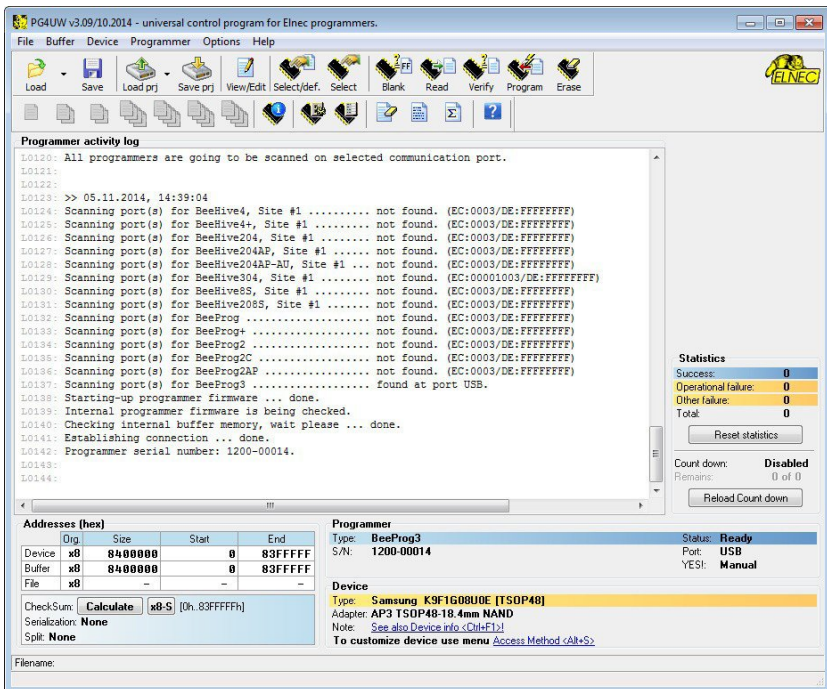
バッファ同期はPG4UW制御プログラムに**"Synchronizing buffers with programmer..."**というメッセージで示されます。



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。

## ユーザー画面の説明

ウィンドウズ・プログラム PG4UW



## Toolbars[ツールバー]

メイン・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->ビュー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
- ・ プログラム・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)
- ・ リモートコントロール・アプリケーションの接続と切断
- ・ 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

### 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。

- ・ プログラム・タイプ
  - ・ コンピュータ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)でデモを選択時
  - ・ 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	デバイスバ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]

メニューメニューはソース・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。

## File formats[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ェアでは 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。

**Automatic file format recognition[自動ファイル形式認識]** チェック・ボックスにチェックをいれますと自動的にプログラムがファイル形式を検出します。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。

**[自動ファイル形式認識]** チェックボックスをオフにすると**[選択されたファイル形式]** パネルの利用可能なファイル形式の一覧から手動で希望のファイル形式を選択できます。デフォルトは**[オプション/一般オプション]**の**[ファイル・オプション]**タブの**[ロード・ファイル形式]**になります。

**ノート:** プログラムはASCII Hexフォーマットのファイルを自動的に認識出来ません、バイナリーとして認識されます。自動ファイル形式認識のためのオプションを無効にしてASCII Hexフォーマットのダウンロードを行って下さい。

### Panel Additional operation[パネル追加操作]

チェック・ボックス **Erase buffer before loading**[ロードする前にバッファを消去]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・データをイレースするようにプログラムに指示します。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。これはバイナリーと全ての 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すると現在の **Hex file options** のメニューの **Options /General options** で **Erase buffer before loading** オプションを無効にします。

チェック・ボックスに**Swap bytes**[スワップ・バイト] 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・バイト機能をアクティベート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(最下位のストレージ・アドレス)にストアされます。リトル・エンディアンはリトル・エンド(シーケンスの最下位の値)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・デシマル値 4F52 は2バイトが要求されストレージ・アドレス 1000H には 4F52H としてストアされます。4FH はストレージ・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 (1000H が 52H、そして、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。

### Panel Buffer offset for loading[ローディングのためのバッファ・オフセット]

パネル **Buffer offset for loading**[ロード用のバッファ・オフセット]はファイルからバッファヘデータをロードするためのワンショット・オフセット設定が含まれています。この設定はバッファに格納するためにロードされたデータのオプションのオフセットを指定するために使用されます。Load file[ロード・ファイル]ダイアログ・ウィンドウを開くと、オフセットは常にデフォルト設定Noneになります。これは、バッファに読み込みデータを格納するためにオフセットが使用され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] バッファにデータを格納するために現在のアドレスに追加されるオフセット値のセット。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: 手動設定の場合は、オプションの負のオフセット[Negative offset]を使用し、希望のオフセット値を編集ボックスに入力します。自動オフセット検出はオプション[自動



負のオフセット[Automatic negative offset]]を使用してください。この値は、現在のアドレスからバッファにデータを保存するために減算されます。

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・オフセットは全てのHEXファイルにのみ適応され、そして、常にx8フォーマットを使用します。ネガティブ・オフセット設定はバイナリ・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際の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みません。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 - 例えば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:

ファイルはMotorola S - フォーマットによるデータを含みます。

アドレス FFFF0Hで開始されたデータ・ブロック。

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。

全てのデータ読み取りに対して負のオフセットオプションと負のオフセット値をFFFF0Hに設定できます。

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。

#### 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 rr", 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

##### ファイル・フォーマット・コード:

#00y - binary  
#10y - ASCII Space  
#20y - Tektronix  
#30y - Extended Tektronix  
#40y - Motorola  
#50y - MOS Technology  
#60y - Intel HEX

##### ロード・ファイル・エラー・コード:

#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]

作成された、変更された、又は、指定されたディスクにデバイスから読み取られたデータをバッファに保存します。保存されたファイルのファイル形式は、サポートされている形式のリスト・ボックスから選択できます。ファイルに保存するバッファの一部を正確に指定するバッファ開始アドレスとバッファ終了アドレスを入力することもできます。現在サポートされているファイル形式は**binary**、**MOTOROLA**、**MOS Technology**、**Tektronix**、**Intel(extended) HEX**、**ASCII space**、**JEDEC**と**POF**。

**Swap bytes[スワップ・バイト]** チェック・ボックスが表示されている場合、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・バイト機能をアクティブすることが出来ます。この機能はファイルをもとローラ表現のバイト形式(ビッグ・エンディアン)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ています。

リザーブ・キー<F2> 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含んだプロジェクト・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**[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するためのものです。プロジェクト情報は下記を含みます:

- 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(それは任意のテキスト、通常はプロジェクトの著者とある種の注釈にすることができます)

**ノート:** シリアライゼーションがオンになっているプロジェクト

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・ファイルを検索実行します。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ライゼーション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プロジェクト・ファイル名に拡張子 ".sn" を付けることによってプロジェクト・ファイル名から導出されます。

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"serialization\"に置かれます。

サンプル:

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。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 [ジョブ・アイデンティフィケーション入力ダイアログ]

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た時に表示されます。

2つの編集可能フィールドを持っています：

- オペレーターID このパラメータはプログラム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は情報のために単に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]

このオプションはプロジェクト・ファイルの保存に使用されます。プロジェクト・ファイルには保存されたデバイス構成とバッファ・データの設定が含まれます。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**[ファイル/ロード・プロジェクト]を使用していつでも復元できます。

### 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

Save project[セーブ・プロジェクト]ダイアログ内に現在選択されているプロジェクト・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書き込みは出来ません。

### 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が内容等の変更は出来ません。これらの実際のプログラム設定は、プロジェクト記述ヘッダの作成に使用されます。

下半分はユーザー編集可能で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)**[暗号化プロジェクト・ファイル(パスワード付き)]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ェアにプロジェクト・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、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るpasswordダイアログが表示されます。

チェックボックス **Set Protected mode of software after loading of this project file**[このプロジェクト・ファイルを読み込んだ後、保護モードを設定する]はProtected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、保存するプロジェクトの保護モードパスワードを指定するためのpasswordダイアログと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読み込みを無効にする、デバイス操作制限)が表示されます。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**[プロテクトモード・プロジェクト]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**をご覧ください。Protected modeがアクティブな時、ソフトウェアはプログラマ・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)**[暗号化プロジェクト・ファイル(パスワード付き)]と **Set**

**Protected mode of software after loading of this project file[このプロジェクト・ファイルを読み込んだ後、保護モードを設定する]のためのパスワードは同じものではありません。**

チェックボックス **Require project file checksum before first programming[最初のプログラミングの前の必要プロジェクト・ファイルのチェックサム]**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ェ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 mode[プロテクトモード]**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の必要がアクティブな場合、ソフトウェ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[必要プロジェクト・ファイルのチェックサム]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カンダリー・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必要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[プロジェクトの保存]ダイアログで使用できなくなりましたが、チェックサム要求がオンになっている古いプロジェクト・ファイルを読み込んだ後にアクティブに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 file[再ロード・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 file[再ロード・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[再ロード・プロジェクト]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 project[再ロード・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・オプ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:

- デバイス名とマニファクチャラ
- 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, マニファクチャラ, 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)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]**

このコマンドはヒープの割り当てを解除しディスク上のバッファを破棄して(存在する場合)、操作システムに戻します。

### **File / Exit and save [ファイル/終了と保存]**

このコマンドはヒープの割り当てを解除しディスク上のバッファを破棄して(存在する場合)、最後に選択されたデバイスの現在の設定を保存し操作システムに戻ります。

## **Buffer[バッファ]**

メニュー **Buffer**[バッファ]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の **view**(ビュー・モード) 又は、**edit**(編集モード)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強調されます。

<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。

<b>F1</b>	ヘルプの表示
<b>F2</b>	フィルのための開始と終了ブロックと要求されるhex(又は、ASCII)文字列をセットして下さい。フィル・ブロックは選択されたバッファのブロックを要求されたhex(又は、ASCII)文字列で埋め尽くします。
<b>Ctrl+F2</b>	指定したブランク値でバッファを消去
<b>Ctrl+Shift+F2</b>	ランダム・データでバッファをフィル
<b>Shift+F2</b>	バッファ・データをバイナリ・ファイルに保存します。このコマンドは2次バッファでのみ使用できます。セカンダリ・バッファはMicrochip PICmicroデバイス用のData EEPROM等の一部のデバイスで使用される特別な領域です。Load/Save data to/from Main buffer[メイン・バッファへ/からデータのロード/セーブ]コマンドはメイン・メニュー "File"で利用できます。メイン・アプリケーション・ウィンドウにロード、セーブ・ボタンもあります。
<b>F3</b>	コピー・ブロック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新しいアドレスへ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・アドレスの外である必要はありません。
<b>Shift+F3</b>	バイナリ・ファイルからバッファにデータをロードします。このコマンドはセカンダリ・バッファでのみ使用できます。詳細については上記のSave Buffer Dataコマンド(Shift + F2)の注意事項を参照してください。
<b>F4</b>	ブロックの移動は新しいアドレス上へ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。
<b>F5</b>	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、プログラムは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)。
<b>F6</b>	プリント・バッファ
<b>F7</b>	文字列検索(最大長 16 ASCII キャラクタ)
<b>F8</b>	文字列を検索し置き換え(最大16 ASCII キャラクタ)
<b>F9</b>	現在のアドレスを変更
<b>F10</b>	ビュー/編集モードを変更
<b>F11</b>	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。
<b>F12</b>	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。
<b>Arrow keys</b>	カーソル上下左右移動
<b>Home/End</b>	現在の行の開始/終了へジャンプ
<b>PgUp/PgDn</b>	前/次ページへジャンプ
<b>Ctrl+PgUp/PgDn</b>	現在のページの開始/終了へジャンプ
<b>Ctrl+Home/End</b>	現在のデバイスの開始/終了へジャンプ
<b>Shift+Home/End</b>	現在のバッファの開始/終了へジャンプ
<b>Backspace</b>	カーソルを1つ左へバック

ノート: キャラクタ 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・デバイスに対するASCIIキャラクタの編集は出来ません。

## 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。

Print bufferダイアログには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**

この項目は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notepad.exeに設定されています。ユーザーは大きなテキスト・ファイルで使用できるwordpad.exe等のテキスト・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文字列を入力し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(編集モード)。

**Forward**[フォワード](現在の位置、又は、バッファの最初からバッファの終了)がデフォルトです。

**Backward**[バックワード]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 dialog box [テキストの検索と置換]ダイアログ・ボックス**

**Text to find**[検索のためのテキスト]文字列入力ボックスに検索のための文字列を入力し、そして、**Replace with**[に置換]入力ボックスに置換のための文字列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:プログラムが例を検出した場合、プログラムを変更する前に尋ねられます。

**Origin**[オリジン]は検索を開始する場所を指定します。

**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(編集モード)。

**Forward**[フォワード](現在の位置、又は、バッファの最初からバッファの終了)がデフォルトです。

**Backward**[バックワード]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> 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、クエスション・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** 見つかった項目の置換と次を検索

**No** 現在のものを置換ずに次を検索

**Replace All** 見つかった全ての項目を置換

**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。
<b>Arrow keys</b>	カーソル上下左右移動
<b>Home/End</b>	現在の行の開始/終了へジャンプ
<b>PgUp/PgDn</b>	前/次ページへジャンプ
<b>Ctrl+PgUp/PgDn</b>	現在のページの開始/終了へジャンプ
<b>Ctrl+Home/End</b>	編集エリアの開始/終了へジャンプ
<b>Backspace</b>	カーソルを1つ左(戻る)へ移動
ノート: 文字列0と1は編集領域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hex(又は、ASCII)文字列によりバッファの選択されたブロックをフィルします。

オプション“Allow address history logging”選択は最後に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 / 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blank・キャラクターによりフィルされます。

オプション“Allow address history logging”選択は最後に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]**

このコマンドはユーザーが選択したスワップ・モードに応じてバイトペアのハイ-とロー-の順番、バイト内の4ビット、又は、ニブルをスワップします。スワップ操作は開始アドレスと終了アドレスで指定されたバッファ・ブロック行われます。このブロックは偶数アドレスで開始され、そして、偶数のバイト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、プログラムはアドレス自身を修正します。(開始アドレスは下位の偶数アドレスに移動されるか、又は、終了アドレスが上位の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ワード内の4バイトをスワップ   |
| 3. Swap nibbles inside bytes        | 各バイト内でハイとローのニブルをスワップ   |
| 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 inside bytes	Mirror bits 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・スワップされたオリジナル・バイト b0, b1, b2:

Original Byte bits	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Nibble-swapped Byte Bits	bit 3	bit 2	bit 1	bit 0	bit 7	bit 6	bit 5	bit 4

Original Byte bits	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Mirrored Byte Bits	bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7

オプション"Allow address history logging"を選択しますと最後に確認された値のセーブをアクティブにします。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。

## Buffer / 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トピカル・ブランクでフィルします。

オプション"Allow address history logging"選択は最後に確認された値のセーブをアクティブにします。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。

<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できます。

## Buffer / Fill random data[バッファランダム・データをフィル]

このコマンドを選択しますとバッファの内容をランダム・データでフィルします。

オプション“Allow address history logging”選択は最後に確認された値のセーブをアクティブにします。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。

<Shift+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 / Duplicate buffer content[バッファバッファ内容のコピー]

このコマンドはソースEPROMの範囲内のバッファ内容を宛先EPROMの範囲にコピーします。この手順は例えば27C512 EPROMから27C256 EPROM位置に使用される場合に適しています。  
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:

- 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**[アドレスへ]が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・アドレスとして定義されます。
- 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或るアドレスのデータ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に使用されたバッファ(データ・ブロック)の一部がチェックサム計算が除外された場合はバッファ・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つまたは複数の除外ブロックを指定できます。
- 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ください。
- STRAIGHTは追加の調整無しチェックサム計算の結果
- NEGATEDはチェックサムの反転、従って  $SUM + NEG. = FFFFH$ .
- SUPPLEMENTはチェックサムの補数、従って  $SUM + SUPPL. = 0 (+ carry)$ .
- **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]操作が実行された時に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。アドレスは常にバイト・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)、Word(16-bit)、又は、DWORD(32-bit)です。  
選択されたチェックサム・サイズより小さい場合は、チェックサム値の下位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、チェックサム値のロー・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つずつインクリ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の計算します。バッファへの書込みは行われません。
    - **Calculate & insert button[計算と挿入ボタン]** - 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[チェックサム]を閉じます。
- (\*) 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)

- グループ Custom address range for main checksum[メイン・チェックサムのためのカスタム・アドレス]
- **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、Disabled[無効]の場合、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チェックサムの種類を選択出来ます。詳しくは下記のチェックサム・タイプをご覧ください。
- 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Exclude buffer block(s) from checksum calculation[チェックサムからバッファ・ブロックを除外] - 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。

## Checksum types[チェックサム・タイプ]

### Byte sum (x8)

バッファ・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・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)によって示されます。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・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)によって示されます。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

バッファ・データは多項式  $x^{16}+x^{12}+x^5+1(0x1021)$  を使って byte を word で加算, 初期値 0, そして、XOR out 0, reflexions in/out は off.

### CRC-XMODEM

バッファ・データは多項式  $x^{16}+x^{15}+x^2+1(0x8005)$  を使って byte を word で加算, 初期値 0.

### CRC-16

バッファ・データは多項式  $x^{16}+x^{15}+x^2+1(0x8005)$  を持った標準CRC-16 アルゴリズムを使って byte を word で加算, 初期値 0, そして、XOR out 0.

### CRC-32

バッファ・データは多項式  $0x04C11DB7$  を持った標準CRC-32 アルゴリズムを使って byte を DWORD で加算, 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(128 bits)で表示されます。

### SHA-1

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

## Checksum forms[チェックサム形式]

**Straight** – 追加の調整無し of 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).

**Device dependent checksum[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。メイン・チェックサムのチェックサム・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)でセレク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]

メニュー **Device[デバイス]**はデフォルトのリストから希望するデバイス・タイプを選択出来ます - 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, ベリファイヒイレース

## 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いるプログラマー(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/デバイス選択...]

このウィンドウは現在のプログラマ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、データシートのパーツ番号のセクション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はソフトウェアではAm27C512と表示されます。
- S29GL064N11TF1010デバイスはソフトウェアでは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。

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。

**Search mask [検索マスク]** フィールドではデバイス名、マニファクチャラー及び/又は、プログラミング・アダプタ名によるデバイス・リスト全体のフィルタリングのためのマスクを入力できます。フィルタ項目(フラグメント)の区切り文字としてのスペースには"OR"機能があります。スペースを含む正確なフィルタ文字列を入力する場合は引用符文字 " を使用してください。

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いるプログラマー(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。

現在表示されているデバイス・リストはボタン Save currently displayed list to file[現在表示されているリストをファイルに保存]を押すことでテキスト・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、希望のマ

ニファクチャー名と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。

### **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希望するデバイス・タイプを選択することが出来ます。最初に、マウス、又は、カーソル・キーを使ってデバイス・タイプ(例、EPROM)を選択、そして、デバイスのサブ・タイプ(例、64Kx8 (27512))を選択して下さい。そうしますとマニファクチャーのリストとデバイスが表示されます。

デバイスは希望のマニファクチャ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、希望のマニファクチャー名と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。

### **Select device... /Only selected manufacturer**[デバイス選択.../選択マニファクチャーのみ]

このウィンドウはマニファクチャ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ファクチャー・ボックスで希望するマニファクチャーを選択します。選択されたマニファクチャ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いるプログラマー(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、又は、Flash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IDとマニファクチャーIDが検知されると、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。どのプログラマーもピン番号を自動的に決定します。他のプログラマーはこの番号を手動で入力する必要があります。

プログラマー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。プログラマー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 EPROMとFlashメモリ

### Device / Device options[デバイス/デバイス・オプション]

このメニューの全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。これは現在の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。項目は現在のデバイスに対しては有効ですが、現在のプログラマーでサポートされていないものは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。

一般的に使用されている用語はプログラマーのユーザー・マニュアルでも説明されています。ここで使用される特別な用語はそれぞれのチップの製造元が使用する用語です。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。

一般的に使用される項目のリスト:

#### 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)
<b>Split[スプリット]</b>	(デフォルト なし)

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]として書き込まれます。

バッファ・アドレス ADDR へのアクセスは Buffer[ADDR]として書き込まれます。

ADDR 値は 0 からデバイス・サイズ(バイト)です。

全てのアドレスはバイト指向アドレスです。

グループ **Insertion test**[インサージョン・テスト]:

#### **Insertion test**[装着テスト] (デフォルト ENABLE)

有効の場合、プログラムはZIFソケット(導通試験)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。プログラム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)

プログラムは各選択アクションの前に 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、プログラムはプログラムされるチップの電子 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ID機能を持っていません。

**ノート 2:** 一部の特殊なケースでは、制御プログラムのデバイスIDチェック設定が"有効"に設定されていても、チップ内のコピー保護機能が設定されている場合はマイクロコントローラはIDを提供しません。

#### グループ **Command execution**[コマンド実行]:

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)



ベリファイ	(ONCE, TWICE)
ベリファイ・オプション	(nominal VCC $\pm 5\%$ , nominal VCC $\pm 10\%$ , VCCmin VCCmax)

#### グループ Programming parameters[プログラミング・パラメータ]

このグループ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・パーツ、又は、領域のセッティングを含みます。

#### グループ Erase parameters[イレース・パラメータ]

このグループ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・モードの特殊な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/シリアルライゼーション]

シリアルライゼーションはプログラムの特殊なモードです。シリアルライゼーション・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。

シリアル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[シリアルライゼーション]** はシリアル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ル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“シリアルライゼーションとプロジェクト”をご覧ください。

#### シリアルライゼーションのベーシック・ルール:

- シリアル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、シリアルライゼーション設定がリセットされます(シリアル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ル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。
- シリアルライゼーション・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。

シリアル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

クロック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ルライゼーション・ルーチンにより使用するかはダイアログ `Serialization[シリアルライゼーション]` 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ルライゼーション・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・オプションシリアルライゼーションインCREMENT・モード & SQTP]**

**Incremental mode & SQTP** [インCREMENTAL・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CREMENTAL・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) シリアルライゼーション・モードの値は1-8が有効で、そして、ASCII シリアルライゼーション・モードでは1-16の値が有効値です。

#### **Address[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ル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。一般的にシリアル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[バッファへセーブ]

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で使用されています)はシリアル番号の最下位バイトを最初に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で使用されています)は最上位バイトを最初に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分割し、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、又は、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分割されます。

### サンプル:

サンプル 1:

アドレス7FFF0H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、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: 7FFF0H

Start value: 16000000H

Step: 1

次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 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 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 02 00 00 16等々

"xx" はデバイスにプログラムされるユーザー・データ

シリアル番号はシリアル数サイズが4バイトのためデバイスのアドレス7FFFCH から7FFFFHに書き込まれません。

#### 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できます。

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できます。

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数を使用することが出来、シリアル番号の各バイトの間に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できます。この番号はSQTPに適した形式を持つことができます。たとえばRETLW b1 RETLW b2等です。注意: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	リテラルをWで返す	11	01xx kkkk	kkkk

xx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、シリアル番号バイト

RETLW命令のオペコードはKKはデータ・バイト(シリアル番号バイト)であるヘキサデシマル 34KKH です。

4つのRETLW命令の一部としてシリアル番号1234ABCDHをデバイスPICに書き込むと仮定しましょう。シリアル番号の最も高いバイトが最上位バイトです。アドレス40Hのデバイス・プログラム・メモリにシリアル番号を書きたいとします。シリアル番号はこの状況で非常に便利に分割しました。シリアル番号分割のないシリアライゼーションは次の番号をバッファとデバイスに書き込みます:

アドレス	データ
0000080	CD AB 34 12 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、アドレスは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

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できます:

**A)** アドレス40Hに4つのRETLW命令をメイン・バッファに書き込みます(これは手作業によるバッファの編集、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命令の8ビットはゼロです。それらは如何なる値も持つことができます。

**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  
Star 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)

上述のスプリット設定はシリアル番号をバイト単位で分割して2バイト毎にバッファすることを意味します。正しいシリアル番号はデバイス・プログラミング動作が開始する前に厳密に設定されます。

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。

### Example 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できます:

**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できます。PG4UW/Bバッファ内のアドレス800hはPIC24Fxxxプログラム・メモリのアドレス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。

デバイスのプログラムを開始する前のアドレス800hのNOPで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  
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) 次のようなシリアルライゼーション・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

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

サンプル 3:

次のサンプルはシリアル番号スプリット・ギャップが2と3に設定されている代わりサンプル2aと同じシリアルライゼーション・オプションを使用しています。

スプリット・ギャップが2バイトにセットされている時、バッファ内容は次のように見えます:

バイト・バッファ構成:

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

ワード16 バッファ構成:

アドレス      データ  
0000040      xxCD ABxx xxxx xx34 12xx xxxx xxxx xxxx

スプリット・ギャップが3バイトにセットされている時、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 xxx

ノート: シリアライゼーション・オプションの効果が変わらない時は、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できます:

- 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.ダイアログ"デバイス操作オプション"でインサージョン・テストとDevice IDチェックを無効にします。
- 3.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けませんのでエラーとなります)、どこにシリアル番号が置かれるかはアドレスのメイン・バッファ(View/Editバッファ)で見てください。

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、即ち、16ビット・ワード(x16)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・ウィンドウとj情報ウィンドウに"**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。

#### 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。

#### 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:

```
[label 1]  addr byte0 byte1 ..  
byten ...  
[label n]  addr byte0 byte1 .. bytem , addr byte0 byte1 ... bytek
```

└──────────────────┬──────────────────┘  
                        |                        |  
                        ベーシック・パート                        オプション・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。

#### オプション・パート

オプション部分は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。オプション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。

### label1, labeln – 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。

### byte0..byten, byte0..bytem, byte0..bytek -

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：

```
byte0 to addr
byte1 to addr + 1
byte2 to addr + 2
....
byten to addr + n
```

**Optional part[オプション部分]**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。

' ' - データのベースック・パート[基本部分]とオプション部分を区切る文字

',' - セミicolon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、又は、定義行の最後に置くことができます。

#### ノート:

- ラベル名は 'I' と 'l' 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]:

```
[nav1] A7890 78 89 56 02 AB CD ; comment1
[nav2] A7890 02 02 04 06 08 0A
[nav3] A7890 08 09 0A 0B A0 C0 ; comment2
[nav4] A7890 68 87 50 02 0B 8D
[nav5] A7890 A8 88 59 02 AB 7D
```



; 次の行には2番目の定義も含まれます

```
[nav6] 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・バ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`

*label x* - 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インストール・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)\El nec_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レポ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

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/シリアルライゼーション/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ルライゼーション・システムを全て持つ場合は、カスタム・ジェネレーター・シリアルライゼーション・モードが最もフレキシブルなシリアルライゼーション・モードを提供します。

シリアルライゼーションのCustom generator mode[カスタム・ジェネレーター]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・モードのシリアル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。

#### Examples:

また、.exeとC/C++のソースファイルも利用可能です。ファイルは次のPG4UWインストール・ディレクトリ-Examples\subdirectoryに有ります:

```
<PG4UW_inst_dir>\Examples\Serialization\customgenerator_example
```

一般的なパスはこのように見えます:

```
C:\Program Files (x86)\Elnecc_sw\Programmer\Examples\Serialization\customgenerator_example\
```

PG4UWコントロール・ソフトウェアの**Custom generator serialization**[カスタム・ジェネレーター・シリアルライゼーション]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[シリアルライゼーション・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社のプログラマも使用しているためユーザーは普通同じシリアルライゼーション・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、シリアルライゼーション・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ル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[シリアルライゼーション・ジェネレーター]

シリアルライゼーション・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・シリアルライゼーション・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。番号は入力されると16進数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、シリアルライゼーション・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、そのシリアルライゼーション.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ンなプログラマ・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・ジェネレータの通常の呼び出しと同様で、現在のシリアル番号を指定

### Custom generator program[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・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・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必要ある度に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制御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制御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ソフトウェアのダイアログ "シリアライゼーション"で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。

### Serialization .dat file format[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 -N<serial number>によってジェネレータ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PG4UWに現在のシリアル番号の次のシリアル番号を知らせシリアライゼーション・ジェネレータで生成されます。

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。

**Example:** 典型的なシリアルライゼーション・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の後のシリアルライゼーション・データ形式は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ルライゼーション・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。

**オプション・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ルライゼーションが中止されたために起こり、そして、PG4UWソフトウェアでワーニング、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ル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。シリアル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からです。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、それはシリアルライゼーション・ダイ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。シリアル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・シリアルライゼーションでのデバイス・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ル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、シリアルライゼーション・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ルライゼーションを呼び出します。PG4UWのシリアルライゼーション・エンジンはシリアルライゼーション・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. シリアルライゼーション・シーケンスは4つのステップからなります:
  - ・ シリアルライゼーション .datファイルを発生させるために適切なコマンド・ライン・パラメータでシリアルライゼーション・ジェネレーターを呼び出す
  - ・ 利用可能なシリアルライゼーション.datファイルを待つ
  - ・ シリアルライゼーション .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ルライゼーション.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。シリアルライゼーション・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  
ラメータでシリアルライゼーション・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。シリアルライゼーション・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。最後のシリアル番号は次のデバイスで  
使用されます。次のシリアルライゼーション・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後  
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、最後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、シリアルライゼーション .datファイルのエラーが検出された場合、プログラムPG4UWはシリアルライゼー  
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/スタティステクス(統計)]

スタティステ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タ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、即ち、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。

スタティステクス[統計]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]**はスタティステクス値がインクリ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、又は、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[スタティステクス]** 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Statisticsはクリックすることで開くことができます。

**Statistics[スタティステクス]** 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]**、**Insertion test failure[インサー  
ション・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:

<b>Success</b>	成功して完了した操作の数
<b>Operational failure</b>	デバイス・エラーで失敗した操作の数
<b>Adapter test failure</b>	アダプターによる失敗した操作の数
<b>Insertion test failure</b>	アダプターの誤った位置により失敗した操作の数
<b>ID check failure</b>	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
<b>Other failure(prog. SW, HW)</b>	ハードウェア・エラー、又は、制御ソフトウェアのエラーにより失敗した操作の数
<b>Total</b>	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]**と2つのカウントダウン情報値**Count down[カウント・ダウン]**と**Remains[残り]**

値の意味は:

<b>Success</b>	成功して完了した操作の数
<b>Operational failure</b>	デバイス・エラーで失敗した操作の数
<b>Other failure</b>	デバイス・エラー以外の理由で失敗した操作の数
<b>Total</b>	全操作数
<b>Count down</b>	カウント・ダウン(有効、又は、無効)の情報
<b>Remains</b>	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・セルフテスト・サマリー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有効、又は、無効も変更出来ます。

**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有効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]

このコマンドはデバイスのブランク・チェックを行いません。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デバイス、又は、その一部分を読み込むことが出来ます。また、リードはチップのコンフィギュレーション(もし、存在し、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ーView/Edit bufferとメニュー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ムの内蔵SSDディスクからPCI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]** は標準で他のワーキング・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]

このコマンドはその使用可能な特殊領域を含むデバイス全体の内容をプログラムの内部ストレージディスクに格納されたバッファ内のデータと比較します。従って、データバッファからのデータはPCからプログラムに転送されます。バッファ同期の進捗状況は別ウィンドウで示されます。同期処理は<Esc>キーを押すか、又は、ウィンドウの終了ボタンを押すことでキャンセルできます。制御プログラムはベリファイ結果を情報ウィンドウとLogウィンドウに報告します。

PG4UWのメニュー・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。

PG4UWのダイアログ **General options (Options/General options [オプション/一般オプション])** のタブ **Errors** は見つかったエラーをユーザーが指定したレポート・ファイルに書き込む方法を制御できます。

ノード:

- ベリファイ操作はソフトウェアでデータをチップ全体の内容と比較しますので、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、ソコ(単体)ベリファイ操作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出来ます。
- 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



スタート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]

このコマンドはデバイスのプログラミングを実行します。コントロール・プログラムは情報ウィンドウとLOGのメッセージによりこのアクションの結果を報告します。

PG4UWのメニュー・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] は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できます。

プログラミング手順の前にPCに格納されたデータがプログラムの内部ストレージディスクに転送されます。バッファ同期の進捗状況は別ウィンドウで示されます。同期処理は<Esc>キーを押すか、又は、ウィンドウの終了ボタンを押すことでキャンセルできます。

### Device / Erase[デバイスイレース]

このコマンドはデバイスのイレースを実行します。コントロール・プログラムは情報ウィンドウとLOGのメッセージによりこのアクションの結果を報告します。

PG4UWのメニュー・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] は利用できる操作オプションをカスタマイズすることができます。

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されたデバイスのリストから選択されたデバイス(スタティックRAM等のこのテストをサポートしている)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・パート

プログラム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(オプション)

"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できます。

<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。

- プログラム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せん
- 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テストを使用してください。

**Device / IC test[デバイス/IC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ブにします。主に標準のロジックIC。ICはテクノロジーの種類ごとにグループ/ライブラリに分類されます。

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Altera® 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)製造業者、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・フォーマットです。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、ペリファイ出来ませんが、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で構成されています: Jam Composer と Jam Player

Jam Composer はプログラムであり、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。

**ソフトウェア・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);

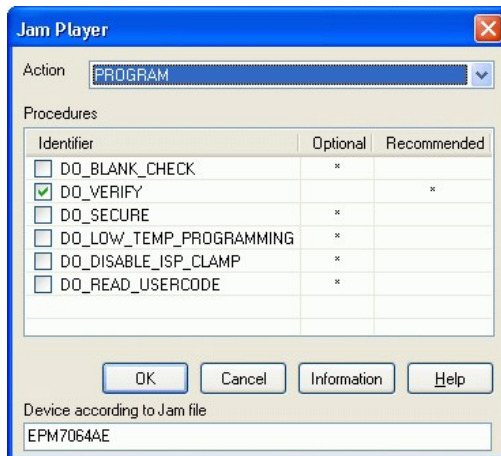
**Xilinx:** Xilinx ISE Webpack又は、Foundation software(ユティリティ SVF2Jamで使用するためにSTAPLファイル、又は、SVFファイルを生成);;

**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けけます。

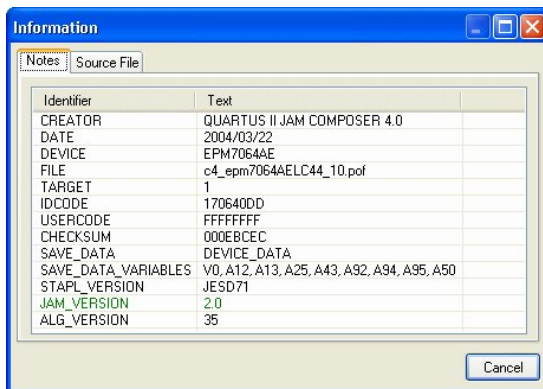
### Information

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できます。

### Device according to Jam file

フ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(')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すために使用することが出来ます。文法は行の長さ、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 STAPLE 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 STAPLE ファイルへの変換:**

- ・ フリー・ダウンロード Xilinx Integrated Software Environment (ISE) 6.3i ソフトウェアのインストール: WebPACK\_63\_fcfull\_i.exe + 6\_3\_02l\_pc.exe (315MB 程)
- ・ 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 ソフトウェアは行: NOTE “DEVICE” “XC2x32A”; 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。

### **STAPLE 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 (STAPLE 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**

STAPLE 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 Play 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 デバイス用のPDBファイルからJAM STAPLEへの変換に関する情報

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:

- ・ ソフトウェア・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・アロン・バージョンとして <<http://www.actel.com>> からダウンロード)
- ・ FlashPro を実行
- ・ New Project [新規プロジェクト] ボタンをクリック、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 [シングル・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。
- ・ 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の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ください。この値は手動で STAPL ファイル (情報ウィンドウでも見る事が出来ます) 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 つの異なる 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 バーチャル・マシーン]

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>をご覧ください。

### ソフトウェア・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、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。



リザーブ・キー<Ctrl+F1>如何なるメニューの時も直ちに表示します。

## Programmer[プログラマ]

メニュープログラマにはプログラマでの作業に使用されるコマンドが含まれています。

## 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。:

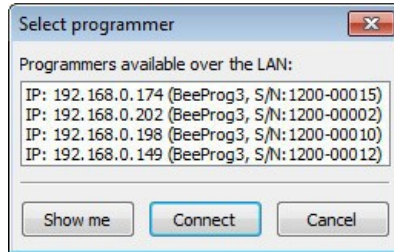
**Programmer[プログラマ]** - 検出のための新しいタイプのプログラマを設定します。**Search all[すべて検索]**を選択すると制御プログラマはサポートされているすべてのプログラマを検出します。

**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。

自動通信が選択されると制御プログラマは最大通信速度をセット。

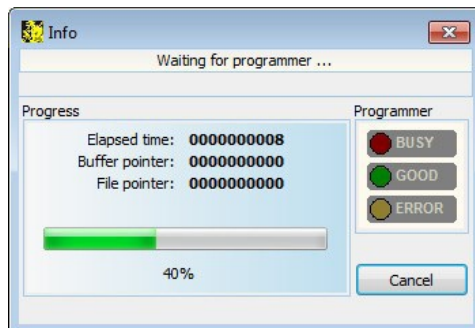
**Port** - 要求されたプログラマのためにスキャンされるポートを選択。All portが選択されている場合、制御プログラマは標準のアドレスで使用可能な全てのポートをスキャンします。LANポートが接続に必要なポートとして選択されている場合は、追加のオプションを使用できます。利用可能なプログラマのために希望のプログラマのIPアドレス、又は、ネットワーク・スキャンを入力することができます。自動スキャン・プロセスは最初にネットワークのプロパティを取得します。複数のネットワークが存在する場合は、どのネットワークをプログラマにスキャンするかを選択することができます。プログラマのスキャンは**Find**ボタンを押すことによって開始されます。



スキャン・プロセスの後、利用可能なプログラマのテーブルが表示されます。テーブルには非アクティブなプログラマしか存在しません。希望のプログラマと接続するには、テーブルでプログラマを選択し、**Connect**[接続]ボタンを押します。視覚的に異なるプログラマの場合は、ボタン **Show me**を押します。選択されたプログラマは数秒間すべてのLEDを点滅させます。

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。制御プログラマ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そ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。

選択されたプログラマとの接続時間が異なる場合があります。接続中にプログラマのファームウェアが確認され、プログラマが再起動される可能性があります。この場合は、別のウィンドウでプログラマの再起動が進行します。





再起動シーケンス中にプログラムのすべてのLEDが点滅します。再起動シーケンスが終了すると、電源LEDだけが不完全に点灯します。接続プロセスの一部はプログラムの内部データストレージ、SSDディスクのテストです。テスト・プロセスの時間はディスク容量、ファイル・システム・エラーの量によって異なる場合があります...内部バッファ・テストの進捗状況は別ウィンドウで表示されます。



このセッティングはコマンド **Options / Save options**[オプション/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ムを再検出(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ムを選択するには、プログラムの通信パラメータと新たに選択されたプログラム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せん。

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。現在のハンドラーの設定は、コマンド**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 / Module options**[プログラマ/モジュール・オプション]

このオプションはMASTERソケットと各ソケットの動作を定義するためにマルチプル・ソケットプログラムに使用されます。**MASTERソケット**・グループ・ボックスはユーザーがデバイスの読み取り操作に優先的に使用されるソケットを設定できます。**Enable/Disable socket** [ソケット有効/無効] チェック・ボックス配列はユーザーが各ソケットの有効化と無効化を個別に設定できます。無効なソケットはどのデバイス操作でも無視されます。

### **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うちの場合、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・アダプタに他のパイパス・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ェ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]**

コマンドは標準装備の**AP3 diagnostic POD**(Board AはBoard Bに接続)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。

### **Programmer / Calibration test[プログラマ/カリブレーション・テスト]**

コマンドは標準装備の**AP3 diagnostic POD** (Board Aのみ) を使用してプログラマの校正値のテストを実行します。

ZIFソケットの各ピンのTTLドライバ、VCCP、VPP1とVPP2電圧のテストされる電圧レベルがあります。カリブレーション・テストの結果はファイルに保存・印刷することができます(次の使用のために)。

## Options[オプション]

このオプション・メニューは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・ファイルのコマンド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。

グループ内で**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:**

- ファイルを再ロードする前にメッセージを表示[プロンプト]
- 自動的に再ロード
- 現在のファイルの変更スキャンを無視

ファイルの修正がテストされる時の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・プログラムへ切り替えられた時
- デバイス操作 Vefiry、又は、Programの選択された時
- ダイアログ"Repeat?"で最後のデバイス操作のリポートが選択された時

**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ー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。

**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。

**サンプル:** 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]

**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** と **Custom erase value**[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の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]

プログラムのサウンド・モードを選択するために使用します。プログラムは動作、即ち、デバイス(programming, verifying, reading, 等々)上の動作終了後、サウンドを鳴らします。プログラムは警告、又は、エラー・メッセージを表示した時にも鳴らします。ユーザーはウィンドウズ・システム・サウンド(サウンド・カードがインストールされている必要があります)、PC スピーカーからサウンドを選択することが出来ます。

パネル **Allow sound for following actions**は次のオプションを含んでいます:

- チェックボックス**操作完了**  
チェックされている時、サウンドはデバイス操作が成功完了後に鳴らします。

チェックされていない時、サウンドは鳴りません。

- チェックされていない時、サウンドは鳴りません。  
チェックされている時、サウンドはデバイス操作がエラーで終わった後に鳴らします。チェックされていない時、サウンドは鳴りません。

## パネルProgrammer internal speaker sound settings[プログラマ内部スピーカースOUND設定]

は内蔵スピーカを内蔵しているプログラマ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・スピーカから音が鳴り結果が良いか悪いかが表示されます。

## Colors and LEDs[プログラマのカラーと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マ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ム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。

#### ソフトウェ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)
- プログラムの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ー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ー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(プログラマー・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。

### 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ように設定できます。ベリファイ・エラーが発生した時、45個までLogウィンドウに書き込まれます。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ーブ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[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:

- オプション No(デフォルト) 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。エラーは画面に表示されるだけです。
- オプション New 最後のベリファイ・アクションからファイルにベリファイ・エラーを保存します。最初に書き込む前に新しいベリファイ・アクションの書き込み前にファイルが削除され新しいものとして作成されます。
- オプション Append 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**最大数**に達した後にはベリファイ・アクションが完了しエラー数がファイルに書き込まれます。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 [エラーの最大数]**は1回のベリファイ操作でエラー・ファイルに書き込むことができるベリファイ・エラーの数を指定します。

### Log file[ログ・ファイル]

このオプションはLogウィンドウの使用に関連付けられます。ログ・ウィンドウの全てのレポートもログ・ファイルに書き込むことができます。ログファイル名はデフォルトで "Report.rep"です。制御プログラムはこのファイルをログファイル名編集ボックスで指定した名前とディレクトリで作成します。

以下のログ・ファイル・オプションが利用できます:

- **No** デフォルトではLogウィンドウの内容はLogファイルにコピーされません。即ち、全レポートはLogウィンドウのみに表示されます。
- **New** 毎コントロール・プログラム開始の間に古いログ・ファイルは削除され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LOGウィンドウ・リポートを追加し、ファイルがない場合は新しいファイル

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[ログファイル名] 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:

`<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と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(2016年11月7日の場合):

c:\logs\myfile-2016-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、ログ・ファイル名を以下として指定することができます:

`<log_file_extension> - .(ドット)`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(2016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**[切り捨てられるテキストの量]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、そして、ロード・プロジェクトで開始された操作から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ソフトウェア・バージョンn
- 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
- シリアル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・プロジェクトが選択された時
- 閉じられる、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を終了する時
- デバイス・カウント・ダウンが0になった時(スタート完了)
- 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
```

この場合

<ordnum>はファイルの10進数の順序です。もし、同じ名前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: \myproject.eprj を使用し、ジョブ・レポートのディレクトリーを d:\job\_reports\ に設定してみましょう。ジョブレポート・ディレクトリーにはレポート・ファイルはありません。

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。

**Automatically save Job Report file**[**自動的にジョブ・レポート・ファイルをセーブ**]にセッティングされている場合、新しくジョブ・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ムとソフトウェアがアクティブ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設定(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)** - プログラムはデバイスがプログラムされた時とプログラム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。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ください。

**表示無し(Quiet mode)** - プログラムは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ビジーは継続的に点灯します。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。

## Save options[オプションの保存]

プログラム終了時のプログラム・オプション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]

**Other[その他]** は他のプログラム・セッティングを管理出来ます。

パネル **Application priority**[アプリケーションの優先度]によりユーザーはプログラムの優先度を設定できます。優先順位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)に影響を与える可能性があります。アプリケーションの優先度をLow[低]に設定するとプログラムが大幅に

遅くなることに注意してください。

パネル **Tool buttons**[ツール・ボタン] は、メイン・プログラム・ウィンドウのツール・ボタン hint display [ヒントの表示] オプションを変更することができます。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できます。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が最後に終了した最後の現在の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。

### **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。

次のツールバーが現在利用可能:

### **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 モードの時はバッファ、又は、デバイス設定の変更が出来る特定のプログラム操作とコマンドが無効にされます。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ェア 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。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せんが、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勧めします。詳しくはFile/Save project[ファイル/プロジェクトをセーブ]をご覧ください。

### 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、意図的なりード操作によってバッファ・データを変更してしまう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**を表します。このモードでは使用可能なすべての操作から**1つの操作のみが有効**にされます。オペレータが間違ったタイプのデバイス操作を実行できないようにするため確実性が向上します。

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ることです。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、ソフトウェアはプログラマー・アクティビティ・ログの右上に**ラベル Protected mode[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。

## PG4UWMCでのProtectedモード

PG4UWMCのAdministrator Mode と Operator Mode (以前のProtected 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、幾つかのメニュー・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は付いての更に詳しい情報は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。

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。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**の作成中に保存された情報や**マルチプロジェクト**自体に基づいて、**任意のデバイスで任意のシーケンス**操作を実行できる特別な機能です。

実際にマルチプロジェクトを使用すると次のことが出来ます:

- 快適にマルチチップ・デバイスをプログラム
- 1個のデバイスでデバイス操作の任意のシーケンスを構成、実行(即ち、Program+Verify+Verify+Verify)

更に詳しくは操作モードの説明をご覧ください。

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・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・オプションとあるレベル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度選択されると、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で構成されています。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保存/ロード/実行することが出来ます。シングルチップ・デバイスからビルド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、Menu/Device/Device[メニュー/デバイス/デバイス]オブ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・ウィザードをご覧ください。

## 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・デバイスの操作には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・ファイルにはマスター・デバイスのsub-devices(chips)に関連する部分的なサブ・プロジェクトが含まれています。マルチプロジェクト・ウィザードでマルチプロジェクト・ファイルを作成することができます。ウィザードには以下のメイン機能があります:

- 複数のsub-projectを選択、そして、最終的なマルチプロジェクトをビルド
- 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。

マルチチップ・デバイス(シングルチップ・デバイスと同様)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

## Multi-project[マルチプロジェクト](又は、Multi-project file [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、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。*

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・パーツの名前規則

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Multi-project[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

ブ・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。プログラム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、プログラムの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。
- もし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・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。

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。プログラム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・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。
- もしAutomatic YES!機能がオンされている場合、プログラマ・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auto-save[自動保存]がオフになっていても、現在保存がサポートされているすべての設定を保存します。次のオプションが保存されます: Optionsメニューのオプション、最後に選択された10個のデバイス、ファイル履歴、メイン・プログラマ・ウィンドウの位置とサイズ。

## Help[ヘルプ(英文)]

メニューHelp[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文章[英文]
- 他の全ての重要な単語[英文]
- 現在のクロス・リファレンス[相互参照]。このクロス・リファレンスをクリックして詳細情報を入力してください。\*英文

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: 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。ELNEC社ウェブ・サイト [www.elnec.com](http://www.elnec.com) 又は、最新バージョンのHelpファイルを参照して下さい。

## Help / Supported devices[ヘルプサポートされているデバイス]

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、ソフトウェア同様に全ての製品を継続的に改善しています。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。

このコマンドはサポートされている全てのプログラムの少なくとも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ム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の"g\_"という接頭辞が付いていればデバイスはマルチソケット・プログラムでサポートされています。

## Help / Supported programmers[ヘルプサポートされているプログラマ]

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ムの全てのデバイス・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。キャラクター?????は現在のプログラム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 (all programmers) [ヘルプデバイス・リスト(全プログラマ)]

このコマンドは全プログラムの全てのデバイス・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。キャラクター?????は現在のプログラムの略称に置き換えてデバイス・リストが生成されます。

ノート: このコマンドが実行された後、制御プログラムは現在のデバイスに関する全ての情報を失います。DEVICE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 (cross reference)[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:

- サポートされているデバイス製造元がリストされている1つのメインHTMLファイルTOP\_DEV.htm
- 各デバイス・メーカーのサポートされるデバイスのリストを含む部分的なHTMLファイル。

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。



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

コマンド プロブレム・レポート作成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クリップボードにコピーします。ログ・ウィンドウの内容は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、ユーザー顧客が自分の問題について製造元にメッセージを送るときにはプロブレム・レポートも送る必要があります。プロブレム・レポートはメーカー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必須のものです。

### **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 マルチ制御ソフトウェア**

---

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。

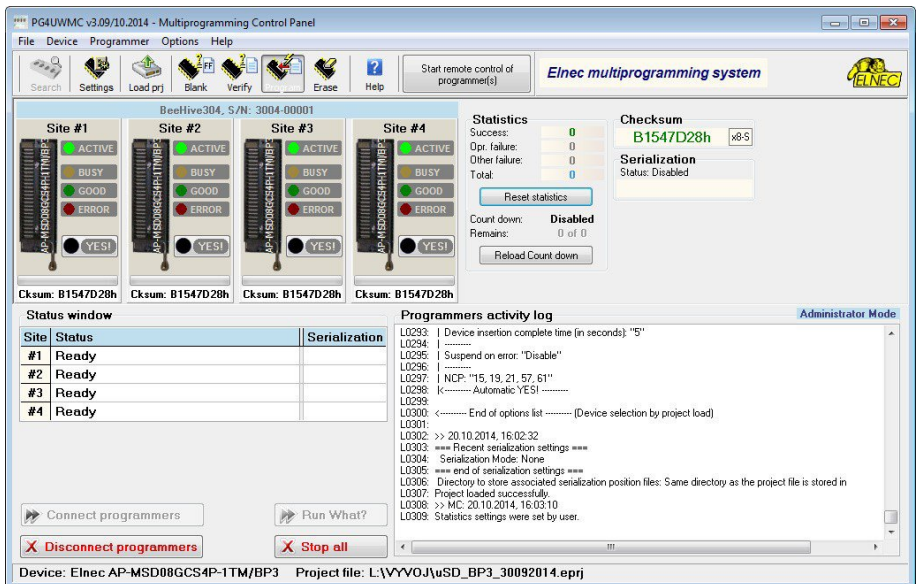
PG4UWMCは大量生産オペレーションの容易な監視を簡単に行うことに焦点を当てています。PG4UWMCのオペレーター・フレンドリーなユーザー・インターフェイスは多くの強力な機能と使い易さを組み合わせて重要でない詳細にオペレーターに負担をかけることなく全ての重要な動作と操作結果の概要を提供します。

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。オプションのプロテクト・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



Site	Status	Serialization
#1	Ready	
#2	Ready	
#3	Ready	
#4	Ready	

At the bottom of the interface, there are buttons for 'Connect programmers', 'Run What?', 'Disconnect programmers', and 'Stop all'. The status bar shows: 'Device: Elneec AP-MSD08GCS4P-1TM/BP3 Project file: L:\VYVOJ\JuSD\_BP3\_30092014.epri'.

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

メニューとツールボタンは殆どのPG4UWMC機能にアクセスできます。

### 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ムのサイト
- プログラム・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。プログラム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。

**and other information[その他の情報]** 現在実行されているデバイス操作、結果、プログラム接続状態等々

**Log window** ステータス・ウィンドウの右側

Log windowはプログラムの接続/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/プログラム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・サイトを切断し、プログラム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・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

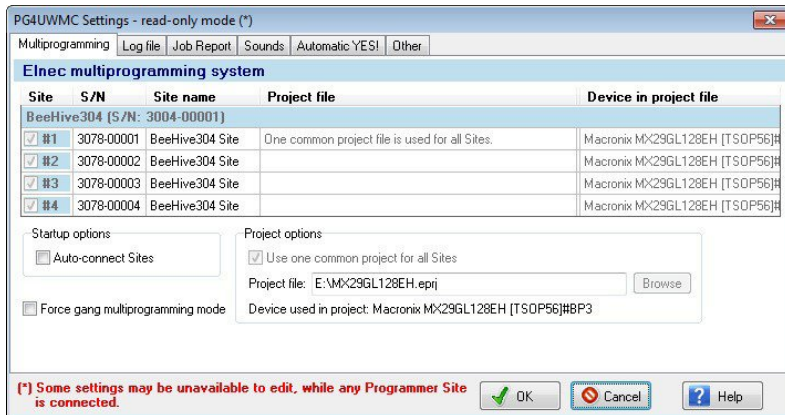
ヘルプを表示するために使用します。

### ボタン Start remote control of Programmer(s)

このボタンは自動化プログラマでのみ利用可能で、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・キャプションでプログラマは最近接続に使用したプログラマ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セッティング・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・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[強制ギヤング・マルチ・プログラミング・モード]
- パネル Log fileセッティング
- パネル Job Reportセッティング
- パネル Automatic YES!セッティング
- パネル Other[その他]

### パネル Multiprogramming

PG4UWMCの上のコントロール・パネルは次の3つのコラムを含んでいます:

- コラム**Sites**は指定されたサイト番号で個々のプログラマ・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。
- コラム**Serial number**はプログラマ・サイトのシリアル番号に付いての情報

• コラム**Project file**は各PG4UWの実行後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、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・テーブルの下に置かれます。

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。

- 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・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。
- 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 Project file[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

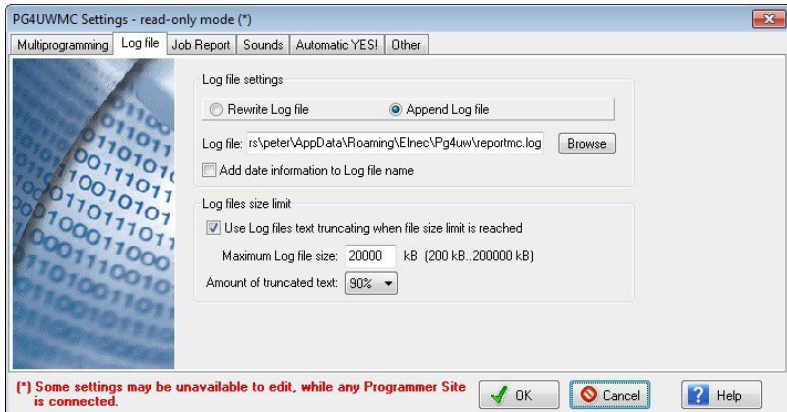
### Force YES for gang multiprogramming mode[ギャング・マルチプログラミング・モードのための強制YES]

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 settings[ログ・ファイル・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:

ユーザーにより指定されたLogファイル名は: "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の次の起動時には、既存のログ・ファイルが書き換えられ新しいログ・ファイルが作成されます。以前のログ・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] 追加するログ・ファイル名の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。このチェック・ボックスにチェックが入れている時、現在の日付文字列をユーザーが指定したログ・ファイル名に次の規則で追加します:

ユーザーが指定したログ・ファイル名の形式が次の場合:

**<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:\Vogs\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:\Vogs\myfile-2006-nov-07.log

日付情報の前にプリフィックス無しでログ・ファイル名を使用したい場合、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:\Vogs\log

日付の追加された最後のログ・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:\Vogs\2006-nov-07.log

#### Logファイル・サイズ制限に関する詳細オプションも利用可能:

- 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・サイトのインデックス]は明確に実行中の各プログラマ・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・サイトのシリアル番号]は使用されているプログラマ、又は、プログラマ・サイトを定義します。実際は希望するシリアル番号を持つプログラマ(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を検索します。シリアル番号の異なるプログラマー、又は、プログラマ・サイトは無視されます。PG4UWMCはプログラマ(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をサイトとして同時に実行することができます。

**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新しいプロジェクトのローディング、又は、プログラムPG4UWMC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ソフトウェア・バージョン
- プログラム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
- プログラム・サイトを閉じる、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

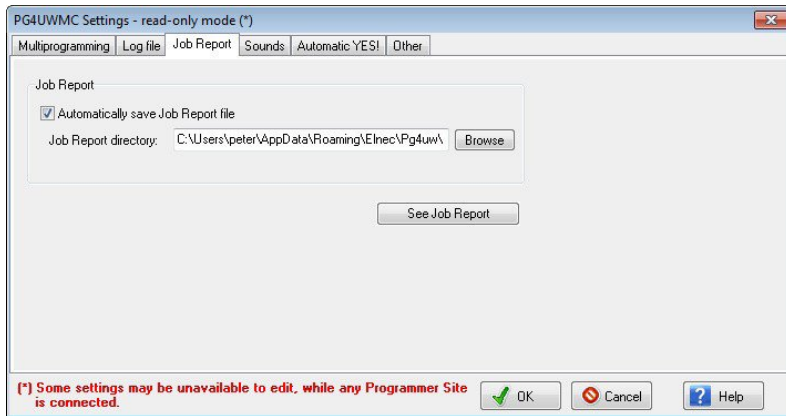
Job Reportは最近ロードされたプロジェクト・ファイルのために、合計の統計値が0より大きい時のみ生成されます。

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は d:\job\_reports\ にセットされます。

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*

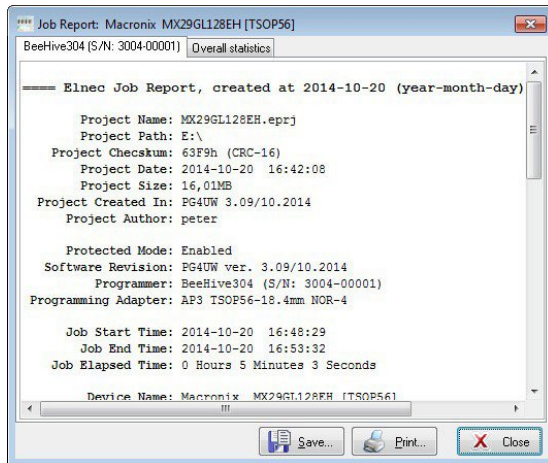
ノート: ファイル名に含まれている番号の順番が 1 つインクリメントされます。

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)。

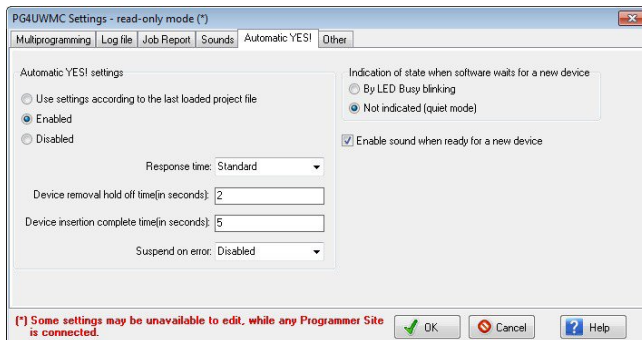
もし、チェックボックスが Automatically save Job Report file[自動的にジョブ・レポート・ファイルをセーブ]のチェックが外されていますと、PG4UW は Job Report ダイアログは必要なら毎回表示されます。

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。何も選択しない場合(ボタンを閉じる)、ジョブ・レポートは PG4UWMC ログ・ウィンドウにのみ書かれます:

典型的な Job Report ダイアログの例を以下に示します:



## Automatic YES! セットアップ



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レポートされます。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。レポート操作の実行は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'[検出から除外されたプログラマのZIFのピン]です。これは別のプログラマ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できます：

"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け付けられませんでした。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

[Automatic YES!]オプションでボタン'Setting Automatic YES! parameters'[Automatic YES!]パラメータのセッティングをクリックして下さい。

'Indication of state when software waits for a new device'[ソフトウェアが新しいデバイスを待っているときの状態の表示]と'Enable sound when ready for a new device'[新しいデバイスの準備ができたならサウンドを有効にする]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ェ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く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5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

ソフトウェアが新しいデバイスを待つ時の状態表示:

**Not indicated (quiet mode) - プログラムはZIFソケットの数に関係なく、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すると直ちにオフになります。**

**By LED Busy blinking(LEDビジー点滅) - プログラムはZIFソケット、プログラムの数に関係なく(マルチプログラム、シングル・ソケット・プログラム)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すると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すると、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ない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/オートマチックYES!].  
をご覧ください

### その他

プログラムの動作結果の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ム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。

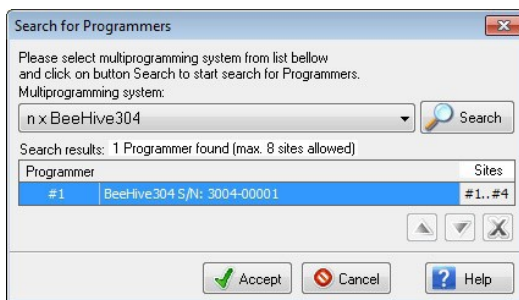
**Timer refresh rate[タイマー・リフレッシュ・レート]** はPG4UWMCプログラムが実行しているプログラマ・サイトからステータス情報を要求する頻度を定義します。ステータス情報には現在のデバイス操作タイプ、進行状況、結果等を意味します。現在のステータス情報がPG4UWMCのメイン・ウィンドウに表示されます。デフォルトのタイマーのリフレッシュ・レート値は200msです。もし、PG4UWMCの操作パネルに表示されるステータス情報の表示をより早くリフレッシュしたい場合、短いリフレッシュ間隔を選択して下さい。より速いリフレッシュを使用している時に、もし、システムのパフォーマンスが低下したことがわかった場合は、高速リフレッシュを使用する場合はリフレッシュ値を高く設定してリフレッシュを少なくします。Pentium 4コンピュータ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[プログラマのサーチ]"

### ローカル・コンピュータ上をサーチ

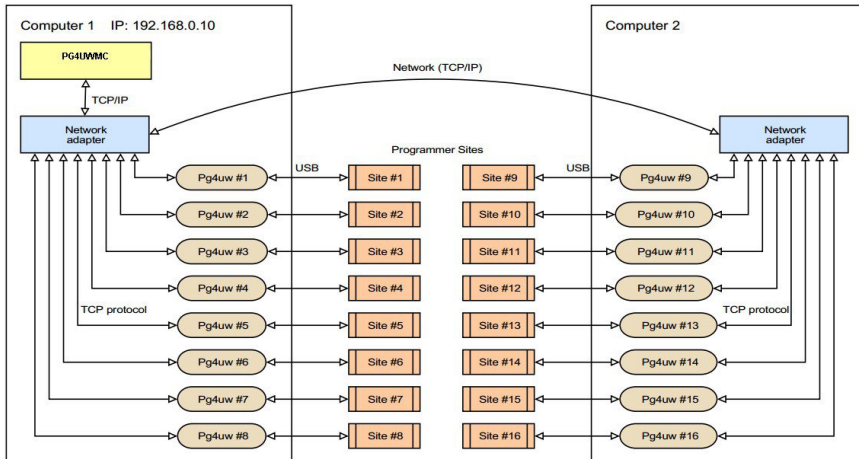
プログラマ検索のこのモードはデフォルトでPG4UWMCのインストール後にアクティブです。もし、ネットワーク経由で別のコンピュータに接続されたプログラマで操作したい場合はネットワーク・モードを試してみてください。

赤色のプログラマ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は非表示にされています



### Search in defined Programmers group on network[ネットワーク上の定義されたプログラマ・グループでサーチ]

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。ネットワーク上の全ての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、主にハンドラー・マシンで使用することを意図しています

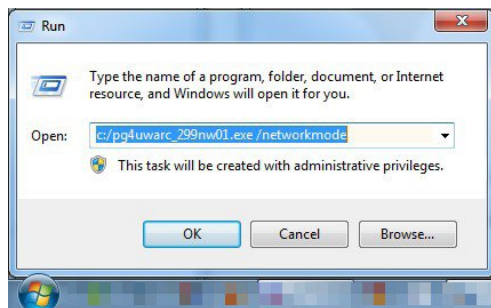
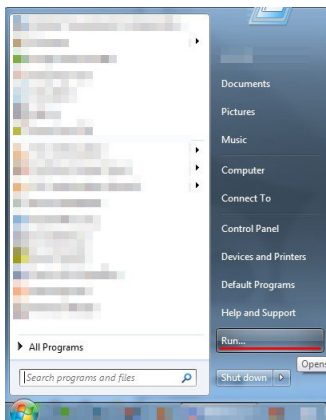


2台のコンピュータ上で実行されるリモート制御されたマルチプログラミング・システムの典型的な構成

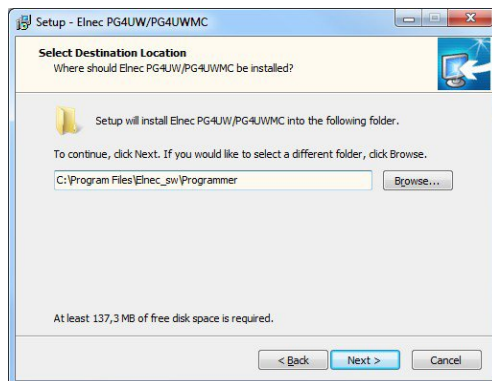
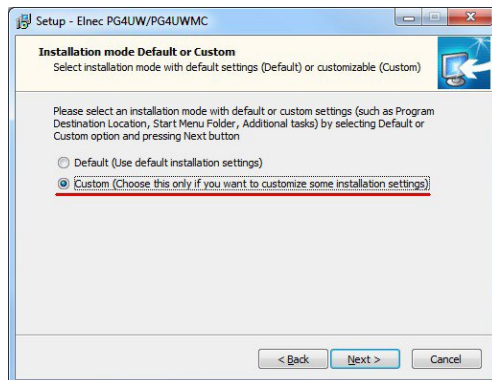
### インストール

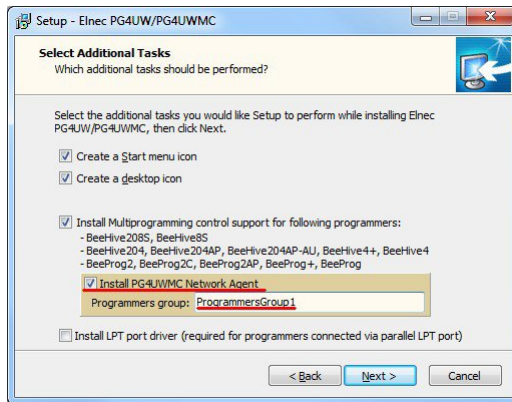
インストール中、Network Mode[ネットワーク・モード]機能はデフォルトではインストールされません。コマンドライン・パラメータ `/networkmode` を使用してインストール手順を実行することでアクティブする必要があります(例: `Start/Run/C:\pg4uwarc.exe/networkmode`)。

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` でのインストール手順

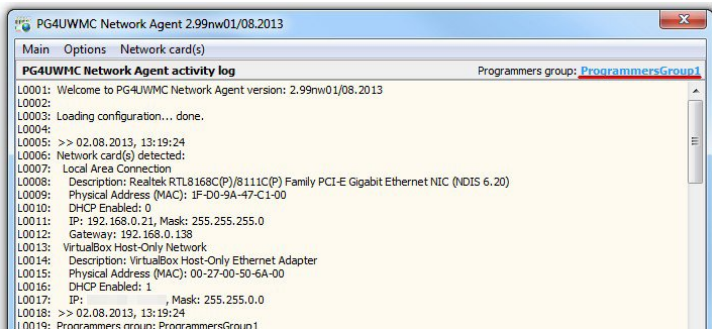




PG4UWMC Network AgentのInstallationがチェックにされProgrammers group名を使用したインストール手順

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...から実行して下さい。



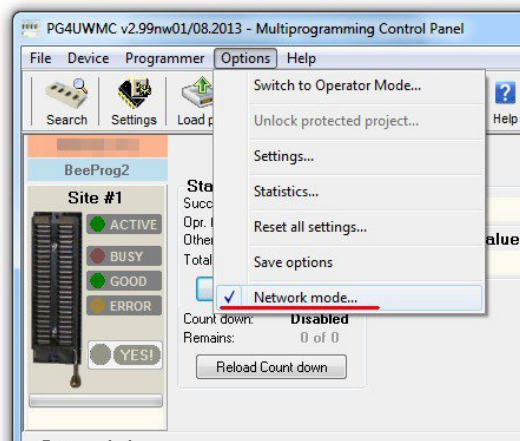
PG4UWMC Network AgentのInstallationがチェックにされProgrammers group名を使用したインストール手順

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

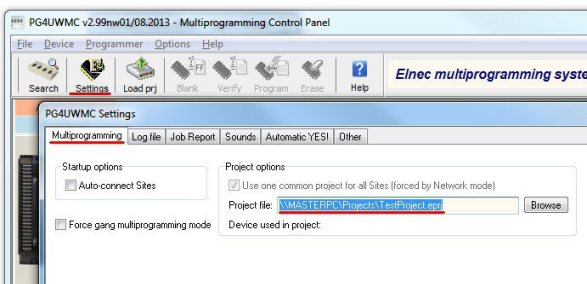
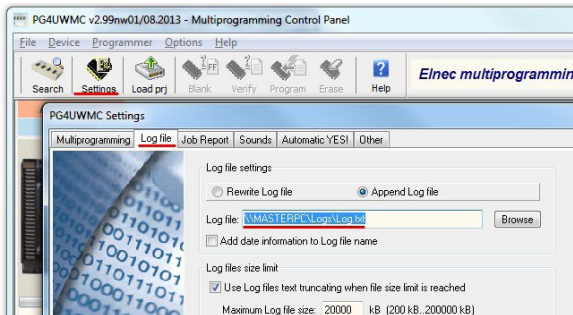
プログラミング・プロセス全体を制御するコンピュータでPG4UWMCを実行します。Menu /Optionsで Network modeをチェックします。





PG4UWMC Network AgentのInstallationがチェックにされ、Programmers group名を使用し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。

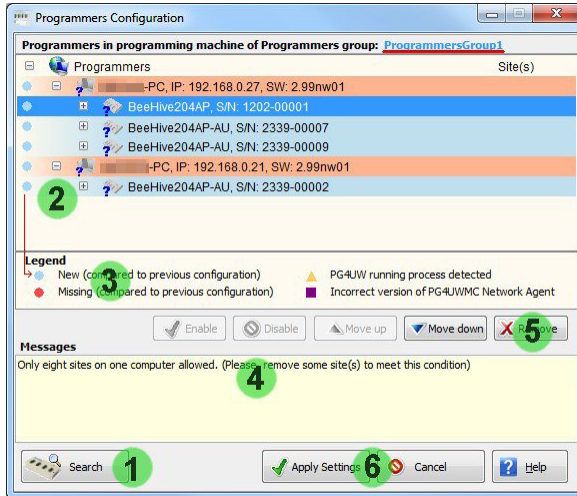


ネットワークからPG4UWMCプロジェクト読み込みの設定、ネットワーク・パスへのログの保存



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。

- プログラマを探す
- 見つかったものを評価する
- 凡例をチェックする(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
- 希望のプログラマを有効、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・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。

## **Programmers supported by PG4UWMC[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、又は、LANインタフェースを備えた48ピン・ドライバと64ピン・ドライバ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・サイトを扱うことが出来ます。1つのプログラマ・サイトは1つのZIFソケットモジュールを意味します。

## **Troubleshooting[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・サイトに対してアプリケーションPG4UWI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)。

### **Communication error(s) while searching for programmers[プログラマ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*

---

## Maintenance[メンテナンス]

プログラム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。

プログラム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。何れにしても、以下の推奨事項が一般的に受け入れられて実行されるべきです:

- プログラム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ソケットとプログラミング・モジュール・インターフェイス(PMI)・コネクタ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ムのZIFソケットとプログラミング・モジュール・インターフェイス・コネクタにカバーして下さい。
- プログラム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

プログラミング・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

全てのプログラム、又は、プログラミング・サイト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、又は、工業用アルコールでプログラムの表面を静かに清掃します。

清潔で乾燥した圧縮空気をを使ってプログラミング・モジュール・インターフェイス(PMI)・コネクタからゴミや埃を取り除きます。

カリブレーション・テストを実行して下さい。

### 日々の使用 (開発研究室、オフィス)

#### 日々のメンテナンス

作業が終了したらプログラミング・モジュールのZIFソケットにカバーします。また、プログラミング・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

プログラミング・モジュールのZIFソケットの状態と装着状態を確認してください。清潔で乾燥した圧縮空気でZIFソケットからゴミやほこりを取り除きます。閉じた状態と開いた状態の両方でZIFソケットをクリーニングします。

#### 四半期毎のメンテナンス

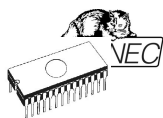
プログラム、又は、プログラミング・サイトに対してセルフテストを実行します。

#### 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、又は、工業用アルコールでプログラムの表面を静かに清掃します。

清潔で乾燥した圧縮空気をを使ってプログラミング・モジュール・インターフェイス(PMI)・コネクタからゴミや埃を取り除きます。

カリブレーション・テストを実行して下さい。



## 時々の使用

### 日々のメンテナンス

作業が終了したらプログラミング・モジュールのZIFソケットにカバーします。また、プログラミング・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

プログラミング・モジュールのZIFソケットの状態と装着状態を確認してください。清潔で乾燥した圧縮空気ですらZIFソケットからゴミやほこりを取り除きます。閉じた状態と開いた状態の両方でZIFソケットをクリーニングします

### 半年毎のメンテナンス

プログラマ、又は、プログラミング・サイト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、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。

清潔で乾燥した圧縮空気を使ってプログラミング・モジュール・インターフェイス(PMI)・コネクタからゴミや埃を取り除きます。

カリブレーション・テストを実行して下さい。

### 警告:

プログラミング・モジュールのZIFソケットは消耗品とみなされます。ZIFソケットの製造業者によって与えられたプログラミング・モジュールZIFソケットの保証された機械的ライフサイクルは一般的に5,000~10,000機械的サイクル(アクチュエーション)であり、いくつかの特殊なBGZIFソケットのライフサイクルは約500,000機械的サイクルまで可能な場合があります。プログラムされるデバイス、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ください。プログラミングの失敗回数が増えていることに気付いた場合は、ZIFソケット、又は、ソケットコンパータを交換して下さい。

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[ソフトウェア]

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。従って、操作中に現在選択されているプログラマを参照しない項目が発生する可能性があります。BeeProg3、BeeHive304も内容的には同じですがPG4UWarc3、PG4UWarc-3-OnDemand.exeをダウンロードしてご使用下さい。

## コマンド・ライン・パラメータ

### 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"  
                    パッファに"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 'noanquest' 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:noanquest
```

/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。

イグゼクティブ・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)

## 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ム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。

## Hardware[ハードウェア]

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。

BeeProg3, BeeHive304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。

## Other[その他]

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[safe mode]に切り替えることができます。

---

トラブルシューティングと保証

---



## Troubleshooting[トラブルシューティング]

Elneq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。

- プログラム、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ムとPG4UWを別のコンピュータにインストールしてみてください。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。
  - 既にプログラム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。その上でプログラムを購入した販売店に連絡するかElneq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: ！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)を添付して下さい。プログラムのモデル名、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、**elneqドットコム**で(迷惑メールにならないように)に送付してください
  - プログラムが不良品であると診断された場合は、お住まいの国のElneqの代理店、又は、Elneq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下さい:
    - 問題の製品
    - プロブレム・レポートを事前にメール添付するか、又は、"**DEVICE PROBLEM REPORT**" フォームに入力
    - 購入日証明になる何らかのコピー

上記項目なしではプログラムの修理を 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.elneq.com](http://www.elneq.com)) で Support -> Problem Reportからフォームを使用出来ます。

### ***If you have an unsupported target device[サポートされていないターゲット・デバイスがある場合]***

プログラム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、プログラム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Elneq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。

## Warranty terms[保証期間]

製造元、Elneec s.r.o. Presov, Slovakiaは購入日から3年間(BeeHive304、BeeProg3)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。製品が欠陥品であると診断された場合、Elneec 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。

プログラミング・モジュールの保証はZIFソケットの機械的寿命によって制限されます。特定のアダプタでZIFソケットの寿命は異なります；特定のアダプタのZIFソケットの機械的寿命についての情報はElneec社のWebサイトで見つけることができます：製品 - アダプタのプログラミング... - 希望の種類のアダプタをクリックします。この保証はZIFソケットの機械的寿命のみを対象としています。この保証は損耗には適用されません。例えば、埃によるより高い/未定義/接触抵抗の変動、又は、プログラムされたデバイスの鉛からの擦れによる接触抵抗。

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。

保証期間はElneec社から直接プログラマを購入するお客様に有効です。Elneec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Elneec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。

Elneec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。Elneec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。

Elneec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、改造しようとする事による損害
- ウィルス、ルートキット等による損害
- 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

**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**

### 製造元:

✉ Elneec s. r. o., Jana Bottu 5, SK - 08001 Presov, Slovakia

☎ +42151/77 34 328, 77 31 007, fax 77 32 797

[www.elneec.com](http://www.elneec.com), e-mail (nospam version): [elneec@elneec.com](mailto:elneec@elneec.com)

**Elnec s.r.o.**

Jana Bottu 5  
SK – 080 01 Presov  
Slovakia

[www.elnec.com](http://www.elnec.com)

ZLI-0330