

# 日本語ユーザー・マニュアル

## **BeeProg2 / BeeProg2C**

ユニバーサル 48-ピン・ドライブ・プログラマー

USB/LPT インターフェースと ISP 機能

ユニバーサル・プログラマー

ELNEC s.r.o.  
Presov, Slovakia  
August 2014

This document is copyrighted by ELNEC s.r.o., Presov, Slovakia. All rights reserved. This document or any part of it may not be copied, reproduced or translated in any form or in any way without the prior written permission of ELNEC s.r.o.

The control program is copyright ELNEC s.r.o., Presov, Slovakia. The control program or any part of it may not be analyzed, disassembled or modified in any form, on any medium, for any purpose.

Information provided in this manual is intended to be accurate at the moment of release, but we continuously improve all our products. Please consult manual on [www.elnec.com](http://www.elnec.com).

ELNEC s.r.o. assumes no responsibility for misuse of this manual.

ELNEC s.r.o. reserves the right to make changes or improvements to the product described in this manual at any time without notice. This manual contains names of companies, software products, etc., which may be trademarks of their respective owners. ELNEC s.r.o. respects those trademarks.

COPYRIGHT © 1991 - 2014  
ELNEC s.r.o.

安全にご使用頂くための使用上の注意と免責事項 .....	1
推奨と最低必要な PC の環境 .....	3
クイック・スタート .....	4
イントロダクション .....	7
BeeProg2 の構成 .....	11
BeeProg2 プログラマーを PC に接続 .....	12
BeeProg2 によるイン-システム・プログラミング .....	13
BeeProg2 スペシフィケーション .....	16
PG4UW ソフトウェア .....	19
プログラマー・ソフトウェアのインストール .....	20
コントロール・プログラムの実行 .....	22
File[ファイル] .....	25
Buffer[バッファ] .....	34
Buffer/Checksum[バッファ/チェックサム] .....	40
Device[デバイス] .....	46
Programmer[プログラマー] .....	79
Options[オプション] .....	82
PG4UWMC マルチ制御ソフトウェア .....	100

### 安全にご使用頂くための使用上の注意と免責事項:

保証期間は納入後 3 年です。但し、保証期間内においても天災、操作ミス、ZIF ソケット、アダプターの予期せぬ消耗や汚れによる接触不良による不具合には対しては保証しかねます。全ての操作、保守と修理サービスは以下の安全情報に従うことが要件となります。

機器の理解不足や間違った使用により起こる予期しない問題による直接・間接のトラブルの責任を取ることはありません。

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

1. このユーザー・マニュアルは英文マニュアルからの転載であり、且つ、ご使用頂くためのヘルプとして頂くための資料です。記述に誤りがあると思える場合は英文で再確認して下さい。誤記、又は、不十分な記述により損害が発生した場合は ELNEC s.r.o, 有限会社データダイナミクスはその責任を負いません。

2. 以下の注意事項を守ってご使用下さい。

- 誤った操作によりデバイスを破損させる可能性があります。
- 静電気による破損を防止するために付属の静電防止様リストストラップを使用する等の対策を履行して下さい。
- ソケット、アダプター、デバイスのクリーニングに常に注意して下さい。埃、汚れ等はプログラミング・エラーの原因になります。
- 誤って落としたり異物や液体等が内部に入った場合はすぐに AC アダプターを抜いて下さい。
- 高温な場所での長時間なご使用は出来るだけ避けて下さい。

2. プログラマーは USB ケーブルを使用してホスト・コンピュータに接続する必要があります。両方の USB プラグがコンピュータに接続されていることを確認し、利用可能な USB 電源が正常にプログラマーを動作させるのに十分であることを確認して下さい。

3. 可燃性ガスの近くで製品を操作しないで下さい。高温と湿度も避けて下さい。

4. 製品のケースを外したり内部部品の交換をしないで下さい。

5. アダプターはユーザーの取扱いによるピンの損傷や不十分なメンテナンス等の理由から消耗品と考えられています。ソケットが製造業者によって指定された作動の定格数を超過して使用されている場合、ソケットは接触不良に起因するエラーが起こります。これは接触エラー数の増加につながり書き込みの失敗を起きます。

6. 本製品を大量生産やマスターの生産などに使用する場合は完成品を組み立てる前に適切なテスト対策を必ず行ってください。ソフトウェア製品を使用した操作することに起因するすべてのリスクはユーザーが負うものとします。

7. このマニュアルは参考であり、製品の仕様とマニュアルは予告なく変更することがあります。

8. このマニュアルに記載されている会社、又は、商標はそれぞれの所有者に帰属します。

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

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

IC(read / verify / programming / erase / blank check ...等)を操作中に IC や ZIF ソケットに触れないでください。ZIF ソケットに IC を挿入する場合、**基本的な静電気保護対策を取って下さい。**

#### このマニュアルで使用されている表現

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

#### このマニュアルで使用されている用語:

##### **Device - デバイス**

プログラマブル IC 又は、プログラマブル・デバイス

##### **ZIF socket - ZIF ソケット**

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

##### **Buffer - バッファ**

コントロール・ソフトウ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。

##### **LPT ポート - プリンター・ポート \*BeeProg2 のみ**

PC の LPT IEEE1284 ポート。双方向の場合はパラレル・ポートと呼ばれます。

##### **USB ポート - USB プリンター・ポート**

PC の USB ポートに接続して使用されます。

##### **HEX data format - ヘキサ・データ形式**

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

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

**最低必要な PC の環境**

	BeeProg2/ BeeProg2C
OS - Windows	Windows XP,Vista,7,8(32bit & 64bit)
CPU	Pentium 4
RAM [MB]	256
free disk space [MB]	200
USB 2.0 high speed	-
USB 1.1 以降	●
LPT	● *BeeProg2のみ
CDROM	●

**推奨 PC 必要環境**

	BeeProg2/ BeeProg2C
OS - Windows	Windows XP,Vista,7,8(32bit & 64bit)
CPU	Core 2 Duo
RAM [MB]	1000
free disk space [MB]	1000
USB 2.0 high speed	●
LPT IEEE1284	● *BeeProg2のみ

上記は S/W バージョン 3.02 [17/DEC/2013]以降のご使用を前提



---

## クイック・スタート

---

### プログラマー・ハードウェアのインストール

- PC とプログラマーのスイッチをオフにしてください。
- 添付されているケーブルを使って PC の USB ポート、又は、プリンター・ポート[ECP 又は、EPP モード]とプログラマーを接続します。
- PC のスイッチを入れます。
- 電源ケーブルのコネクターをプログラマーに接続します。

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。**

### プログラマー・ソフトウェアのインストール

CD が付属しておりますが、ご購入時のご使用時には

<http://www.elnec.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。

### プログラマー・ソフトウェアの使用

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



即ち、 をダブルクリックして下さい。

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

メニュー”**File**”はソースファイルの操作、設定とディレクトリーを見たり、ドライブを変更、ロードと保存するファイルとプロジェクトのバッファの開始と終了アドレスを変更します。

メニュー”**Buffer**”はバッファ操作、ブロック操作、バッファの一部をストリングスでフィル、イレース、チェ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], ベリファイ[verify], イレース[erase]とプログラミング・プロセス、シリアライゼーションと関連ファイルのコントロールのセッティング に使用します。

メニュー”**Programmer**”はプログラマーで使用する機能のための使用されます。メニュー”**Options**”は各種デフォルト設定の確認や変更のために使用されます。メニュー”**Help**”はそのバージョンでサポートされているデバイスとプログラマーとプログラム・バージョンについての情報を見るために使用されます。

デバイスのプログラム(書込み)

1. デバイスを選択:  をクリック

2. データをバッファへロード:


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




をクリック

\*読み取った後はデバイスは抜いて下さい。

3. ターゲット(書き込みたい)デバイスを ZIF ソケットに装着

4. デバイスがブランクかを確認するために  をクリック

5. デバイスにプログラム(書込み)  をクリック

6. デバイスに書き込んだデータとバッファのデータを照合(Verify)するため

に  をクリック





---

## イントロダクション

---

## イントロダクション

**BeeProg2/BeeProg2C** は Windows 2000/XP/Vista/7/8(32bit と 64bit) ベースのプロフェッショナルなモバイル・アプリケーションのためにも使用できる比較的小さくてパワフルなユニバーサル・プログラマーと TTL/CMOS ロジック IC **テスター** です。さらに、BeeProg2/BeeProg2C は特別なモジュールを使用することなくマイコン、フラッシュ、GAL 等を幅広くサポートしています。供給電源とプログラミング電源がデジタルに制御され、そして、H のレベルを制限することが出来ますので、プログラマーは 1.8V からの真の低電圧デバイスにも使用出来ます。

**BeeProg2/BeeProg2C** のインターフェースは IBM 互換の PC, AT 又は、同等以上のポータブル又は、デスクトップ PC で動作します。プログラマーは IEEE1284 (ECP/EPP) 高速パラレル・ポートをサポートしています。プリンター・ポートか、又は、USB ポートを使用しますので、特別なカードは必要としません。

**BeeProg2/BeeProg2C** の **FPGA** ベース完全コンフィギャラブル 48 **パワフル TTL ピンドライバ** がソケットの各ピンに H/L/プルアップ/プルダウン と 読み取り機能を供給。 **高品質、高速回路** を持った先進のピン・ドライバーが、サポートされた全デバイスのためにオーバーシュートなしで、又は、グラウンド・バウンス無しでシグナルを供給します。ピン・ドライバーは 1.8V まで操作できますので、すべての低電圧デバイスをプログラムすることが出来ます。

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

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

**BeeProg2/BeeProg2C** はプルダウン・メニュー、ホット・キーとオンライン・ヘルプを持った**使い易いコントロール・プログラム**によりデバイスをクラス、マニファクチャー又は、マニファクチャー名をパーツ番号により選択することが出来ます。標準のデバイス操作機能(読み出し、ブランク・チェック、プログラム、ベリファイ) はいくつかのテスト機能と一緒に完了されます。プログラムは自動ファイル・フォーマットの検知と変換を含む、バッファとファイルの使用機能があります。

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

**BeeProg2/BeeProg2C** は自動ファイル形式認識機能で入力ファイルを扱います。



Jam files(JEDEC standard JESD-71)は Jam プレイヤーによってインタプリットされます。Jam files は各プログラム・デバイスのメーカーから供給された設計ソフトウェアによって生成されます。

VME files は VME プレイヤーによってインタプリットされます。

VME file は SVF ファイルの圧縮されたバイナリー・バリエーションでありハイレベル IEEE 1149.1 バス・オペレーションを含んでいます。

VME files は各プログラム・デバイスのメーカーから供給された設計ソフトウェアによって生成されます。

ファイルのローディング中に行われます。ソフトウェアは全ての知られているデータ形式はサポートされています。

バイナリー(RAW), インテル(拡張)HEX(Intel, Intel EXT), モトローラ, MOS テクノロジー, Exormax, Tektronix, ASCII-SPACE-HEX, ASCII HEX Altera POF, JEDEC(ver. 3.0.A), eg. from ABEL, CUPL, PALASM, TANGO PLD, OrCAD PLD, PLD Designer ISDATA 等々, JAM(JEDEC STAPLE Format), JBC(Jam STAPL Byte Code), STAPLE(STAPL File)JEDEC standard JESD-71 VME(ispVME file VME2.0/VME3.0)

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

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

### フリー・アディショナル・サービス:

#### 何故PG4UWの最新バージョンを使うのが重要か?

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

フリー・ソフトウェア・アップデートは  
[www.elnec.com](http://www.elnec.com)からダウンロード出来ます。

ELNECは下記のサービスも提供しております。

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

### BeeProg2/BeeProg2C の構成

1. 48ピン ZIF ソケット
2. LED \* 操作結果を表示
3. LED パワー/スリープ
4. YES! ボタン

\*Automatic YES! は通常 Disable になっていますが、これを Enable に設定しますと Programmer - Automatic YES!を使用時にデバイスを交換した後このボタンで書き込み開始が出来るようになります。

5. ISP コネクター
6. 電源スイッチ
7. GND コネクターと ESD リスト・ストラップ・コネクションのためのコネクター



8. 電源コネクター
9. LPT コネクター PC⇔BeeProg2 通信用ケーブル
- 10.USB コネクター PC⇔BeeProg2/BeeProg2C 通信用ケーブル



## **BeeProg2/BeeProg2C プログラマーを PC に接続**

### **USB ポートを使用**

ソフトウェアが先にインストールされていますと、LPT ポートをハード的にスキャンします。もし、USB でのみご使用の場合は、ソフトウェアをインストールする前に BeeProg2/BeeProg2C を USB ポートに接続して下さい。その場合、LPT ポートのスキャンは行われません。

USB ケーブルと電源ケーブルの接続はどちらが先でもかまいません。

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

### **LPT ポートを使用**

PC とプログラマーのスイッチをオフにして下さい。

PC のプリンター・ポートとプログラマーを付属の平行ケーブルで接続して下さい。付属の 100V 電源ケーブルのプラグをプログラマーのラベルに 15VDC とある電源コネクタに接続して下さい。プログラマーの LED POWER が点灯し、BeeProg2 が使用出来る状態であることを確認して下さい。そして、PC の電源をオンにして、コントロール・プログラムを実行して下さい。プログラマーを機械的なプリンター切り替え器等を経由して接続しないで下さい。

**LPT ポートは必ず LPT1 としてご使用下さい。**

そして、PC の電源をオンにして、コントロール・プログラムを実行して下さい。

**LPT(プリンター)ポートと電源アダプターの接続はどちら先でもかまいません。**

**警告!** もし、PC をスッチ・オフしたくない場合は、下記の手順に従ってください。:

●プログラマーが PC に接続されている場合: 平行ケーブルを最初に、そして、電源ケーブル

●プログラマーが PC に接続されていない場合: 最初に 電源ケーブル、それから、平行ケーブルをはずして下さい。

## BeeProg2/BeeProg2C によるイン-システム・シリアル・プログラミング

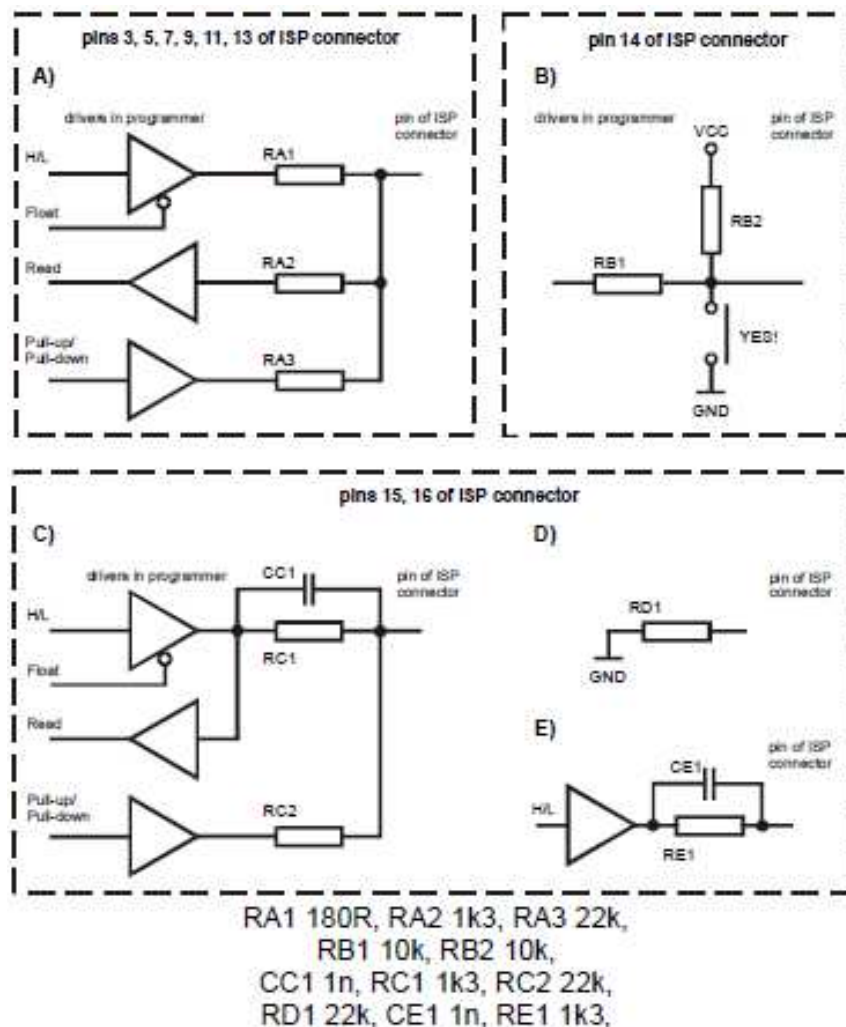
各デバイスを選択の上、Device Info[デバイス情報]でご確認下さい。デバイスのメーカーにより異なります。



プログラマーの ISP コネクタの正面

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

ノート: H/L/read ドライバー



C) ISPプログラミングのために論理シグナルとして設定が必要な時はピン15と16を接続して下さい。

D) E) ピン15と16がLED OKとLED ERRORが状態として構成されている時

D) ISP動作の前のLED回路状態

E) ISP動作の後のLED回路状態

ノート: LED OK又は、LED ERROR ON(輝いている)時、この状態は希望のISPデバイスのHレベルによりロジカルH、Hのレベル1.8V – 5Vを表しています。

LED OK又は、LED ERROR OFF(輝いていない)時、この状態はロジカルL、L

のレベルが0V – 4Vを表しています。この上記の値はプログラムされるチップとターゲット・システムを絶縁している抵抗の正確な値を示しています。

ISP コネクタ・ピンのスペックはデバイスにより異なります。それはソフトウェア (Pg4uw)のメニュー **Device / Device Info (Ctrl+F1)**で見ることが出来ます。それぞれのデバイスの ISP プログラミングの方法が選択されているか注意して下さい。選択されたデバイス名の後のサフィックスにより表示されています。

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

**ノート:** ISP ケーブル・コネクタ上のピン番号 1 はトライアングル・スクラッチにより示されています。

ISP ケーブル・コネクタは 20pin コネクタは Harting 09185207813 が使用されます。

\*ターゲット側は Harting 09 18 520 6324 をお勧めします。



BeeProg2/BeeProg2C ISP ケーブル

**警告:**

- BeeProg2/BeeProg2C を ISP プログラマーとして使用する時は、ZIF ソケットにデバイスを装着しないで下さい。
- ZIF ソケットでデバイスをプログラミングする時は、ISP ケーブルを ISP コネクタに接続しないで下さい。
- 付属の ISP ケーブルのみをご使用下さい。他の ISP ケーブルを使用されると、プログラミングが上手くいかない可能性があります。
- BeeProg2/BeeProg2C はプログラムされたデバイス(ISP コネクタのピン 1)とターゲット・システム(ISP コネクタのピン 5)を制限付きで供給することが出来ます。ターゲットから電源供給は出来ません。
- BeeProg2/BeeProg2C はターゲット・デバイスにプログラミング電圧を印加し、そして、その値をチェックします。(ターゲット・システムはプログラミング電圧を修正することが出来ます。) もし、プログラミング電圧が期待したものと異なる場合は、ターゲット・デバイスは実行されません。



### プログラマーのセルフ・テスト

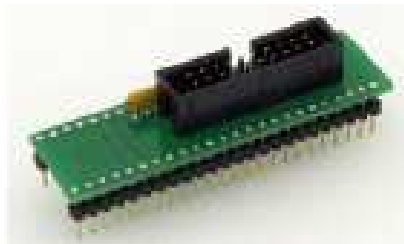
プログラマーが正常に動作していないと思える場合(但し、最低3ヶ月毎)は、診断 [Diagnostic]POD を 48 ピン ZIF ソケットに装着して BeeProg2、又は、BeeProg2C のセルフ・テストを行って下さい。ソフトウェアの Programmer/Selftest でセルフテストが行われます。



### ISP コネクタのセルフ・テスト

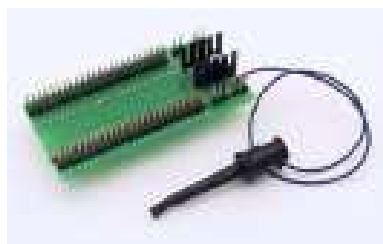
ISP コネクタのための Diagnostic POD[#2]を ZIF ソケットに装着します。20 ピン・コネクタをプログラマーの ISP コネクタケーブルで接続します。※20 ピンが内部的に正しく接続されていることを確認して下さい。(即ち、1-1, 2-2,..., 20-20)

ソフトウェアの Programmer/Selftest ISP connector でセルフテストが行われます。



### カリブレーション・テスト・ポッド ※オプション

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



オプション: 注文番号 **70-0438**

**48 Pins Calibration test POD, Type I**

## BeeProg2/BeeProg2C スペシフィケーション

### ベース・ユニット、DAC

- USB 2.0 ハイ・スピード互換ポート 480Mb/s まで
- FPGA ベース IEEE 1284 スレーブ・プリンター・ポート 1MB/s までの転送レート
- オン・ボード・インテリジェンス: パワフルなマイクロプロセッサと FPGA ベース・ステート・マシーン
- VCCP, VPP1 と VPP2 のための 3 つの D/A コンバータで立ち上がりと下がり時間をコントロール
- VCCP 範囲 0..8V/1A
- VPP1, VPP2 範囲 0..26V/1A
- セルフ・テスト機能
- 電源供給入力とパラレル・ポート接続での静電気と ESD に対するプロテクション
- ESD リストストラップ接続のためのバナナ・ジャック
- グラウンドへ接続のためのバナナ・ジャック

### ソケット, ピンドライバー

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

### ISP コネクター

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

### I.C. テスター

- TTL type: 54,74 S/LS/ALS/H/HC/HCT シリーズ
- CMOS タイプ: 4000, 4500 シリーズ
- static RAM: 6116.. 624000

- ユーザー定義パターン生成

### ソフトウェア

- **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。**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によるインテリジェント・デバイス選択
  - 自動 EPROM/フラッシュ EPROM の 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
- **セキュリティ**
  - 装着テスト
  - 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)
  - 自動デバイス連続番号インクリメント
  - スタティスティクス
  - 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

### 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) from ABEL, CUPL, PALASM,



TANGO PLD, OrCAD PLD, PLD Designer ISDATA 等

- JAM(JEDEC STAPL 形式), JBC(Jam STAPL バイト・コード)、STAPLE(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(ActI STAPL file)

#### **一般仕様**

- 操作電圧 110-250V AC
- 消費電力 最大 20W(アクティブ), 2W(スリープ)
- サイズ 195x140x55 [mm] (7.7x5.5x2.2 [インチ])
- 重さ 0.9kg(1.98lb) \*アダプター類を除く
- 操作温度 5° ÷ 40°C
- 保管湿度 20%..80% 非結露



---

*PG4UW* ソフトウェア

---

### プログラマー・ソフトウェアのインストール

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

### ソフトウェアの新しいバージョン

常にプログラマーの機能を最大にご利用頂くために最新のバージョンを <http://www.elnec.com/download/> から PG4UWarc-OnDemand.exe のソフトウ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。(OnDemand Version)

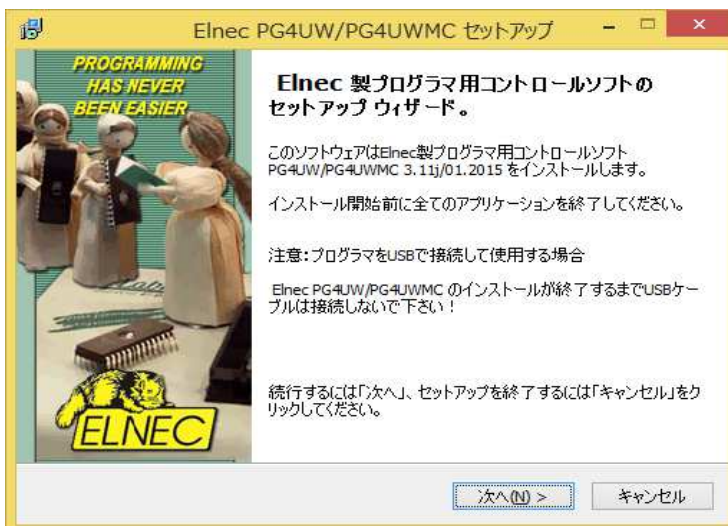
\*参考: PG4UWarc.exe(Regular Version) はレギュラー・バージョンですが、そのバージョンの最初のバージョンに過ぎません。

### プログラマー・ソフトウェアのインストール

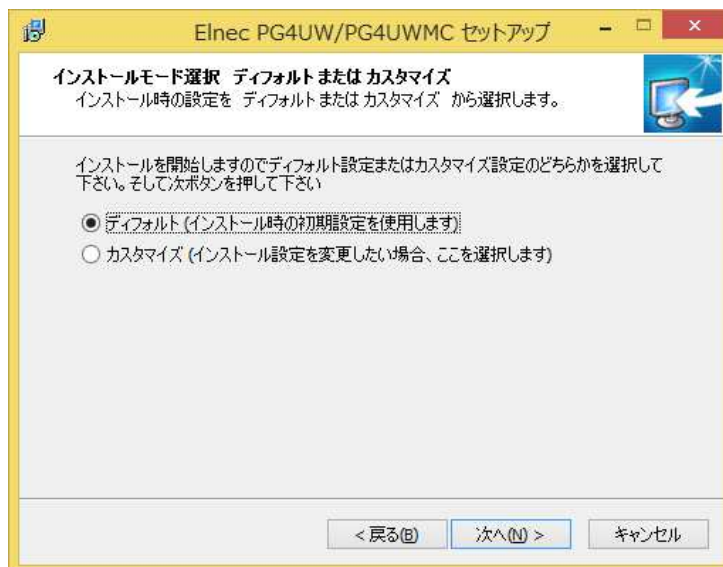
ダウンロード/保存した PG4UWarc-OnDemand.exe をダブルクリックします。



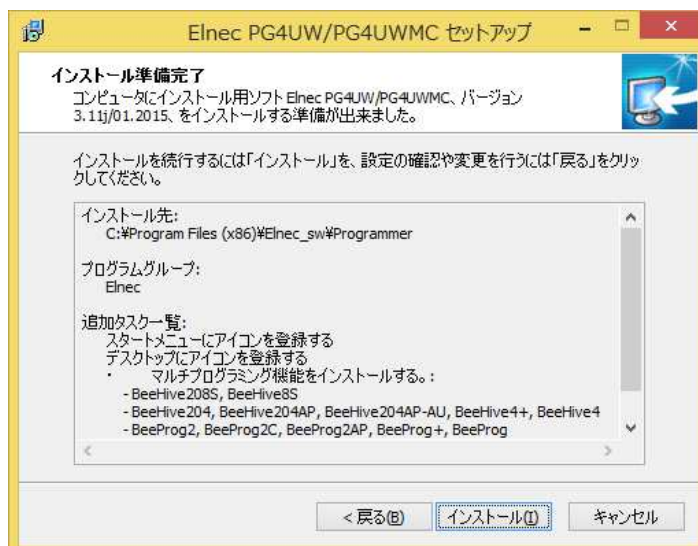
### OK をクリック



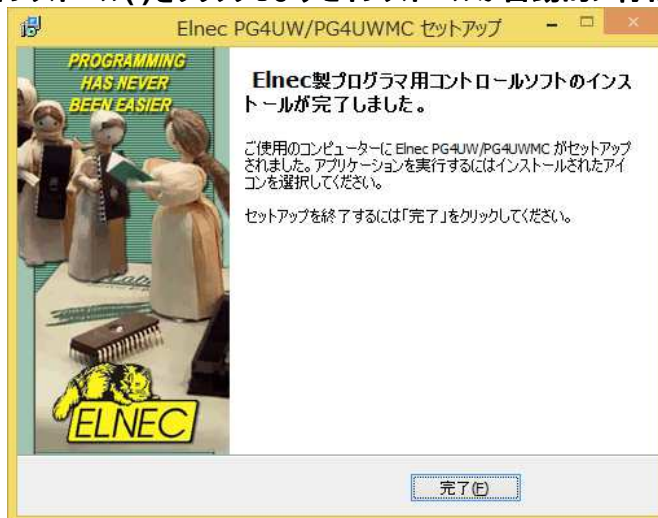
### 次へ(N)をクリック



**次へ(N)をクリック**



**インストール(I)をクリックしますとインストールが自動的に行われます。**



**完了(F)をクリックしますとインストールが完了します。**

## コントロール・プログラムの実行

PG4UW アイコンをダブル・クリックして下さい。



使用しているプログラマーにチェックを入れて下さい。

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

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

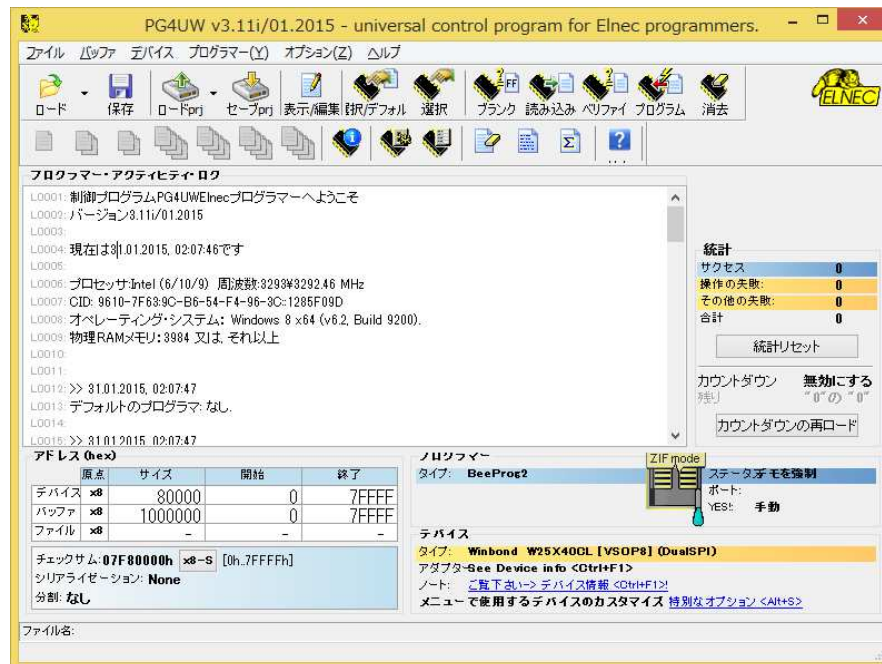
**ノート:** PG4UW プログラムが開始されると、標準のユーザー・メニューが表示されてユーザーからの指示を待ちます。

もし、コントロール・プログラムがプログラマーと通信できないと、スクリーンにエラー・コードと考えられる理由(プログラマーの接続が外れている、間違った接続をしている、電源アダプターの不良、間違ったプリンター・ポート)を含むエラー・メッセージが現れます。再点検の上、問題を取り除いて、そして、いずれかのキーを押してください。

もし、エラーがまだ有る場合は、プログラムはデモ・モードで再開されますので、プログラマーへのアクセスは出来ません。もし、エラーの原因が見つからないときはトラブル・シューティングの指示に従ってください。コントロール・プログラムはデバイスのプログラムの前にプログラマーとの通信をチェックします。



## ユーザー画面の説明



### Toolbars[ツールバー]



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

### Log window[ログ・ウィンドウ]

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

操作:

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

これらの情報は問題が有った時のためにカット&ペースでテキスト・ファイルにコピーすることも出来ますが、通常はヘルプ->プロブレム・レポート作成をクリックしますと"PG4UW\_LOG\_windows\_content\_xxxx.zip"としてデスクトップ上に作成しサポートのためにメールに添付して送ることが出来ます。

### アドレス

アドレス (hex)				
	原点	サイズ	開始	終了
デバイス	x8	80000	0	7FFFF
バッファ	x8	80000	0	7FFFF
ファイル	x8	-	-	-

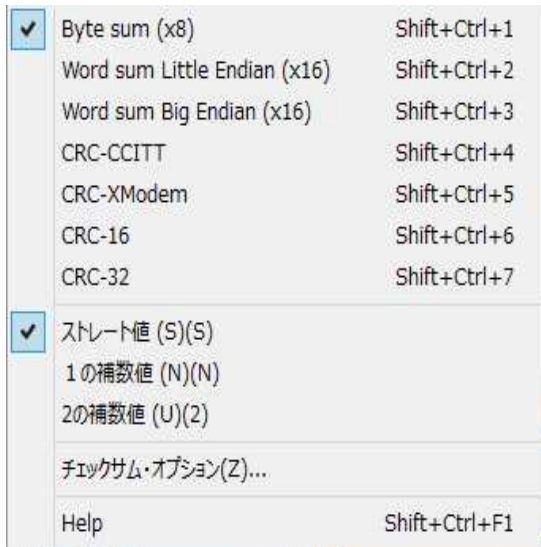
パネル・アドレスは現在選択されているデバイスの実際のアドレス範囲, ロードされたファイルとバッファの開始と終了アドレス設定についての情報を含んでいます。

ある種のデバイスではメニューのデバイス->デバイスオプション->操作オプションによってデフォルト・デバイスとバッファ・アドレス範囲を変更できます。

チェックサムには以下の様な機能も利用出来る様になっています。

チェックサム: **07F80000h** **x8-S** [0h..7FFFFh]  
 シリアライゼーション: **None**  
 分割: **なし**

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



**ノート:** Buffer/View/Edit[バッファ/ビュー/編集]コマンドの ASCII ブロックを除き、キーボードで入力されたコマンド・データは HEX フォーマットです。

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

### File[ファイル]

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

### File / Load[ファイル/ロード]

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

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

### ファイルフォーマットの説明:

#### ASCII HEX フォーマット

各バイトデータは 2 つの 16 進数で表され、他の全てのデータ・バイトから空白スペースによって区切られています。

データ;バイトのためのアドレスは\$Annnn, キャラクターのシーケンスを使ってセットされます。nnnn は 4 つの 16 進数アドレスです。後ろにカンマが必要です。

各データバイトがアドレスを持っているが明白でない場合、明示的なアドレスがデータストリームに含まれていない限りデータ・バイトは連続してアドレス指定されます。最初のデータバイトの前にアドレス部分が設定されていない場合は、ファイルは 0 から開始します。ファイル STX(Control-B) 文字 (0x02) で始まり ETX(Control-C)文字(0x03)で終わります。

ノート: チェックサム部分は\$Sとカンマ文字間の 4 つの 16 進数文字列で構成されます。チェックサムはファイルの最後の部分になります。

ASCII HEX ファイルの例: データ"Hello, World"をアドレスの 0x1000 にロードしています:

```
^B $A1000,  
48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 0A ^C  
$S0452,
```

#### ASCII SPACE フォーマット

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

ASCII SPACE ファイル例: データ"Hello, World"をアドレスの 0x1000 にロードしています:

0001000 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 0A

### Straight HEX フォーマット

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

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

### Samsung HEX フォーマット

Samsung HEX フォーマットは Intel HEX フォーマットを少し修正したもので、ファイルの形式、及び 構成等は Intel HEX フォーマットと略同等ですのでソフトウェアでは 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。

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

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

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

### 追加操作

チェック・ボックス **Erase buffer before loading** にチェックをいれますと入力されたイレース値を使って全てのバッファ・データをイレースします。

バッファ消去はファイルを読み込む前に直ちに実行されます。これはバイナリーと全ての 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は **Hex file options** のメニューの **Options /General options** で **Erase buffer before loading option** を無効にします。

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

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

4F52H がメモリーにストアされる内容を次に示します。:

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

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

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

ローディングのためのバッファ・オフセットはファイルからバッファにデータをロードするためにワン・ショット・オフセット設定が出来ます。

標準設定はいつも"None"になります。ユーザーが使用する時にオフセット値を設定して下さい。設定はバッファリングするために保存するためにロードされたデータのオフセットをオプションで指定するために使用されます。

Load File ダイアログ・ウィンドウが開かれていますオフセットはデフォルトでは常に設定なしです。これはバッファに読み出しデータを格納するために使用されるオフセットはないことを意味します。

利用出来るオフセット・オプション:

**None**[なし] ファイルからバッファへのローディングにオフセットは適応されません。

**Positive offset**[ポジティブ・オフセット] オフセット値のポジティブ・オフセットは現在のアドレスにオフセットを加算してバッファにデータをストアします。このオフセットは全てのフォーマットに利用でき、そして、もし、現在のバッファの構成が x8, 又は、x16 フォーマットの場合は x8 フォーマットが使用されます。

**Negative offset**[ネガティブ・オフセット] モードは 2 つのオプションを持っています:

**Negative offset**[ネガティブ・オフセット]と **Automatic negative offset**[自動ネガティブ・オフセット]-手動、又は、自動の2つの方法でセット:

手動セットはオプション Negative offset を使用し、希望するオフセット値とその編集ボックスに置きます。自動オフセット検出はオプションの Automatic negative offset を使用します。この値はデータをバッファに保存するため現在のアドレスから減算されます。

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

ノート:

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

ネガティブ・オフセットの使用例:

Motorola S - フォーマットによるデータがふくまれるファイル

データ・ブロックがアドレス FFFF0H で開始するファイル

それは 3 バイトのアドレス・アレイの長さを持った S2 フォーマットです。

全てのデータの読み取りで Negative offset オプションを設定し、ネガティブ・オフセットの値を FFFF0H に設定することができます。

これは現在の実際のアドレスからオフセットが差し引かれデータがバッファ・アドレス 0 から書き込まれることを意味します。(即ち、ファイルの読み込みを実行するとアドレス FFFF0H の前のデータは読み飛ばされ FFFF0H 以降のデータがバッファメモリの 0 番地から順にストアされます)

ファイル・フォーマットとエラー・コードのリスト

サポートされたフォーマットの幾つかでファイルのダウンロード中にエラーが起こり得ます。

エラーは LOG ウィンドウに次の様に書かれます "Warning: error #xxy in line rrr", xx はファイル・フォーマット・コード, y はエラー・コードと rrr は 10 進数での行番号。

File format codes:

#00y - binary

#10y - ASCII Space

#20y - Tektronix

#30y - Extended Tektronix  
#40y - Motorola  
#50y - MOS Technology  
#60y - Intel HEX

Load file error codes:

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

### File / Save[ファイル/保存]

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

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

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

### File/Load project[ファイル/ロード・プロジェクト]

このオプションはデバイスの保存されたコンフィギュレーション・バッファ・データとユーザー・インターフェース・コンフィギュレーションを持ったプロジェクト・ファイルをローディングするために使用されます。

標準ダイアログ Load project は追加のウィンドウを含みます - Project description - ダイアログ

このウィンドウはダイアログ Load project で選択されている現在のプロジェクト・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

ノート: シリアライゼーションを持ったプロジェクトはオン

シリアライゼーションは以下の手順でプロジェクト・ファイルから読み込まれます:

1. プロジェクトに記述したシリアライゼーション設定が受け付けられます。
2. 追加のシリアライゼーション・ファイルを検索が実行されます。ファイルが検出された場合、それが読み込まれ、そして、追加ファイルからのシリアライゼーション設定

が受け付けられます。追加のシリアライゼーション・ファイルは常に特定のプロジェクト・ファイルに関連付けられます。追加のシリアライゼーション・ファイルの設定が受け入れられる時、プロジェクトのシリアル化の設定は無視されます。

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

例:

プロジェクト・ファイル名 : my\_work.prj

コントロール・プログラムのディレクトリ: c:\Program Files\Programmer\

追加のシリアライゼーション・ファイルは:

c:\Program Files\Programmer\serialization\my\_work.prj.sn

追加のシリアライゼーション・ファイルはデバイスのプログラム成功後に作成され更新されます。

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

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

### **Enter job identification dialog[入力ジョブ・アイデンティフィケーション・ダイアログ]**

このダイアログはプロテクトされたプロジェクト・ファイルをローディングされて時に表示されます。

2つの編集可能フィールドを持っています:

プロテクトされたプロジェクト・ファイルをリードした時、Job ID コードを入力するダイアログが表示されます。

・オペレーターID - このパラメータはプログラマーのオペレーターを認識するため使用されます。

オペレーターIDは3文字以上でなければいけません。

プロテクトされたプロジェクトの Job Report を作成する時にパラメータが必要ですので、ユーザーはオペレーターIDを入力しなければいけません。

・Job ID 入力 - 現在行っている作業の Job ID 認証を入力します。

ノート:ダイアログ Enter job identification はパスワードのダイアログではありません。オペレーター認証の値と Job ID は情報のために単に 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]

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

ファイル・リストから実際に選択されたプロジェクトの説明

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

セーブされるプロジェクトの説明

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

チェックボックス Encrypt project file (with password)は暗号アルゴリズムを使った特殊なフォーマットでプロジェクトをセーブするために使用されます。これはパスワード無しでソフトウ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、password ダイアログが表示されますので、保存されるプロジェクトのための暗号化パスワードを指定するのに使用されます。

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

推奨: Encrypt project file (with password)と Set Protected mode of software after loading of this project file のためのパスワードは同じものは使用しないで下さい。

チェックボックス Require project file checksum before first programming がアクティブな時、ロード・プロジェクトの後に最初のデバイス・プログラミングの開始前にソフトウ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 モードで使用することをお勧めします。プロジェクト・ファイルの固有の ID の要求がアクティブな場合、ソフトウ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 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 사용되는セコンダリー・バッファ・データからも計算されることです。プロジェクト・ファイルのチェックサムの要求がアクティブな場合、ソフトウェアが制御プログラムのメイン・ウィンドウの一番下のステータス行にプロジェクト・ファイル名の隣にこのラベル(CSum)を表示します。このオプションは Save project ダイアログボックスで使用できなくなりましたが、チェックサム要求が設定を持った古いプロジェクト・ファイルの読み込み後にアクティベートすることができます。

### **File/Reload file[ファイル/ファイルを再ロード]**

最近使用したファイルを再ロードするためにこのオプションを選んで下さい。

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

ファイルの再ロード:

1. ファイル・メニューから Reload ファイルを選択
2. 最後に使用したファイルのリストが表示されます。再ロードしたいファイルをクリック

ノート: ファイルを再ローディングする時、そのファイルは最後にロード/セーブされたファイルで使用されたファイル形式が使用されます。

### **File/Reload project[ファイル/プロジェクトの再ロード]**

最近使用したプロジェクトを再ロードするためにこのオプションを選んで下さい。

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

プロジェクトの再ロード:

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

### **File/Project options[ファイル/プロジェクト・オプション]**

このオプションは実際にロードされたプロジェクトの表示/編集プロジェクト・オプションのために使用されます。プロジェクト・オ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ー
- ・ プロジェクト作成日付

- ・ ユーザー定義プロジェクト説明(任意テキスト), 例えば、更に詳細なプロジェクト説明のための作者と他のテキスト・データ

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

#### **File / Load encryption table{ファイル/暗号テーブルのロード}**

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

#### **File / Save encryption table[ファイル/暗号テーブルの保存]**

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

#### **File / Exit without save[ファイル/保存せずに終了]**

設定を保存せずにプログラムを終了

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

INI ファイルに設定を保存してプログラムを終了

### **Buffer[バッファ]**

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

### **Buffer / View/Edit[バッファ/ビュー/エディット]**

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

<F4> ホット・キーでも使用出来ます。

### **View/Edit Buffer [ビュー/バッファの編集]**

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

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

- |                      |   |
|----------------------|---|
| <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ヘキサ<br>又は、ASCII スtringをセットして下さい。  |
| <b>Ctrl+F2</b>       | 指定したblank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>F3</b>            | ブロックのコピーは新しいアドレス上の現在のバッファのデータの<br>指定されたブロックをコピーするのに使用されます。<br>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<br>必要はありません。   |
| <b>F4</b>            | ブロックの移動は新しいアドレス上の現在のバッファのデータの<br>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<br>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<br>ドレス・ブロック(又は、一部分)はblank・キャラクターにより<br>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・<br>ペアのハイとローの順番をスワップします。<br>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<br>て、バイトの偶数でなければいけません。<br>もし、この条件が満たされないと、プログラムのアドレス自身を<br>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<br>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>            | ストリング検索(最大長 16 ASCII キャラクター)  |
| <b>F8</b>            | ストリングを検索し置き換え (最大 16 ASCII キャラクター)  |
| <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 デイケーター] 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>	カーソル移動
<b>Home/End</b>	開始/終了ヘジャンプ
<b>PgUp/PgDn</b>	前/次ページヘジャンプ
<b>Ctrl+PgUp/PgDn</b>	現在のページの開始/終了ヘジャンプ
<b>Ctrl+Home/End</b>	現在のデバイスの開始/終了ヘジャンプ
<b>Shift+Home/End</b>	現在のバッファの開始/終了ヘジャンプ
<b>Backspace</b>	カーソルを1つ左へバック

**ノート:** キャラクター 20H - FFH (ASCII モード)と番号 0..9, A..F (HEX モード) は即座に編集エリアの内容を変更します。

**警告:** ワード・デバイスへの ASCII キャラクターの編集は出来ません。

#### **Print buffer[プリンター・バッファ]**

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

#### **Print buffer[プリント・バッファ]**

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

ダイアログに次のオプションがあります。:

#### **Block start[ブロック開始]**

バッファ内の選択されたブロックの開始アドレスを定義

#### **Block end[ブロック終了]**

バッファ内の選択されたブロックの終了アドレスを定義

#### **外部エディター**

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

### Find [テキスト検索]ダイアログ・ボックス

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

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

Origin[オリジン]は検索を開始する場所を指定します。

### Find & Replace text[テキストの置換]ダイアログ・ボックス

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

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

**Origin[オリジン]** は検索をどこから開始するか指定します。

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

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

**Replace[置換]** ボタンを押しますと、ダイアログ・ボックスが閉じられ、そして、クエスチョン・ウィンドウが表示されます。このウィンドウは下記の選択を含んでいます。:

<b>Yes</b>	置換と次を検索
<b>No</b>	置換せずに次を検索
<b>Replace All</b>	すべてを置換
<b>Abort search</b>	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]

<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>F9</b>	アドレスへ...
<b>F10</b>	ビュー/編集のモードを変更
<b>F11</b>	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>	カーソル移動
<b>Home/End</b>	現在行の開始/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>	エディット・エリアの開始/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:** 0 と 1 のキャラクターがエディット・エリアの内容を即座に変更

#### **Buffer/Fill block[バッファ / ブロックのフィル]**

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

選択オプション “Allow address history logging”は最近確認された値のセーブをアクティベートします。これらは各デバイスは別個にセーブされます;カウントは最後の 15 個に制限されています。

ノート: アドレス履歴値は全てのバッファ・データ操作ダイアログで共通です。

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

#### **Buffer/Copy block[バッファ / ブロックのコピー]**

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

#### **Buffer/Move block[バッファ / ブロックの移動]**

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

選択オプション “Allow address history logging”は最近確認された値のセーブをアクティベートします。これらは各デバイスは別個にセーブされます;カウントは最後の 15 個に制限されています。

ノート: アドレス履歴値は全てのバッファ・データ操作ダイアログで共通です。

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

### Buffer/ Swap block[バッファ/スワップ・ブロック]

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

次のスワップ・モードが利用出来、ユーザーは選択することが出来ます：

- 1.Swap 2-bytes inside 16-bit words 16-bit ワード内をバイト・ペアでスワップ
- 2.Swap 4-bytes inside 32-bit words 32-bit ワード内をバイト 4 でスワップ
- 3.Swap nibbles inside bytes バイト内をニブルでスワップ
- 4.Mirror bits inside bytes バイト内のビットを逆にします。

バッファ内のスワップ操作の例：

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

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

b0, b1, b2..はアドレス 0, 1, 2..からのオリジナル・バッファ・バイト値

b0n, b1n, b2n..は次のルールによるニブル・スワップ・オリジナル・バイト b0, b1, b2

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

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

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

ノート：アドレス履歴値は全てのバッファ・データ操作ダイアログで共通です。



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

### **Buffer / Erase[バッファイレース]**

このコマンドを選択しますと、バッファの内容を空白でフィルします。

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

ノート:アドレス履歴値は全てのバッファ・データ操作ダイアログで共通です。

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

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

### **Buffer/ Fill random data[バッファ]**

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

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

ノート:アドレス履歴値は全てのバッファ・データ操作ダイアログで共通です。

デフォルトのアドレス範囲は選択されたデバイスのバッファ範囲に従ってセットされます。

オプション"Maintain last inserted values"を選択しますと次回にこのダイアログを開いた時に以前に確認された値がデフォルトとして再ロードされます。

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

### **Buffer /Duplicate buffer contents[バッファ/バッファ内容のコピー]**

このコマンドはコピー先の EPROM の範囲にソースの EPROM の範囲のバッファの内容をコピーします。これは、例えば、27C512 EPROM 位置へ 27C256 データを使用するのに適切です。

ノート:手順は常にバッファ開始アドレス 00000h を使用

### Buffer / Checksum[バッファ/チェックサム]

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

アドレス (hex)

	原点	サイズ	開始	終了
デバイス	x8	80000	0	7FFFF
バッファ	x8	80000	0	7FFFF
ファイル	x8	-	-	-

チェックサム: **07F80000h** x8-S [0h..7FFFFh]  
 シリアライゼーション: **None**  
 分割: なし

ファイル名:  
 Byte sum (x8), Straight, チェックサム・オプションを開くにはクリック.

ここをクリックすると下のダイアログが現れます。

<input checked="" type="checkbox"/>	Byte sum (x8)	Shift+Ctrl+1
	Word sum Little Endian (x16)	Shift+Ctrl+2
	Word sum Big Endian (x16)	Shift+Ctrl+3
	CRC-CCITT	Shift+Ctrl+4
	CRC-XModem	Shift+Ctrl+5
	CRC-16	Shift+Ctrl+6
	CRC-32	Shift+Ctrl+7
<input checked="" type="checkbox"/>	ストレート値 (S)(S)	
	1の補数値 (N)(N)	
	2の補数値 (U)(2)	
	チェックサム・オプション(Z)...	
	Help	Shift+Ctrl+F1

ここをクリックすると”チェックサム・ダイアログが次ページのように現れます。



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

Checksum

Checksum Calculator | Main Checksum Options

Checksum Calculator's custom address range

Effective

Address from: 0 h (8)  
Address to: 4002FF h (8)

Buffer block(s) excluded from checksum calculation

Effective

Block Start End

0 0

Cancel Add

Results

	Start value	1's complement value	2's complement value	MD5 Hashsum:
Byte sum (8)	00000000	00000000	00000000	
Word sum LE (x16)	00000000	00000000	00000000	
Word sum BE (x16)	00000000	00000000	00000000	
CRC-CCITT:	0000	0000	0000	
CRC-XModem:	0000	0000	0000	
CRC-16:	0000	0000	0000	
CRC-32:	00000000	00000000	00000000	

SHA-1 Hashsum:

Device-dependent checksum:

Checksum Options

Checksum type: Byte sum (8) | Start value: | Address: 0 | Size: DWORD

Note 1: All numbers are in hexadecimal format.  
Note 2: "Address from" and "Address to" are usually in bytes.

? Calculate Calculate & Insert Close

- 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)



チェックサム

チェックサム・カリキュレータ    メイン・チェックサム・オプション

メイン・チェックサムのためのカスタム・アドレス範囲

有効にする

アドレスから: 0 h (6)

アドレスまで: 4002FF h (6)

Buffer block(s) excluded from checksum calculation

有効にする

ブロック    開始    終了

削除

0    0    追加

除外されたブロック・リストのコピー

チェックサム・カリキュレータから

チェックサム・カリキュレータへ

チェックサム・タイプ

Byte sum (x8)

Word sum Little Endian (x16)

Word sum Big Endian (x16)

CRC-CCITT

CRC-XModem

CRC-16

CRC-32

チェックサム形式

ストレート値     1の補数値     2の補数値

結果

チェックサム: 3FC2FD00h

大きなデバイス(xメモリ32MB以上)のチェックサムも自動的に計算されます

ノート: "アドレスから"と"アドレスへ"は常にバイトを適応します

適用    閉じる

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

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

- **NEGATED**はチェックサムの反転  $SUM + NEG. = FFFFH$ .
- **SUPPLEMENT**はチェックサムの補数  $SUM + SUPPL. = 0 (+ \text{carry})$ .
- **Insert checksum options[チェックサムの挿入オプション]** ボックス  
– このボックスは**Calculate & insert[計算と挿入]** 操作のための次のオプションを含みます:
- **Insert checksum[チェックサムの挿入]** Calculate & insert[計算と挿入]が実行されたときバッファに書込まれるチェックサムの種類
- **Insert at address[アドレスの挿入]** Calculate & insert[計算と挿入]が実行されたときに選択されたチェックサムの結果を書込むバッファのアドレス。アドレスは <From address> から <To address>の範囲内で指定することが出来ません。アドレスはバイト・アドレスとして定義されます。
- **Size[サイズ]** 選択されたチェックサム結果が書込まれるバッファのサイズ。チェックサムのサイズはByte(8-bit)又は、Word(16-bit)、又は、DWORD(32-bit)です。  
もし、選択されたチェックサム・サイズより小さい場合は、チェックサム値のロー・バイトのみがバッファに書込まれます。  
*ノート: もし、ワード・サイズが選択されると、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を計算します。バッファへの書込みは行われません。
- **Calculate & insert button[計算と挿入]** - Calculate & insert ボタンをクリックしますと、バッファ内の選択されたブロックのチェックサムを計算され、そして、選択されたチェックサムがInsert address[アドレス挿入]で指定されたアドレスでバッファに書込まれます。この機能はByte, Word, CRC-CCITT とCRC-XMODEMチェックサムで利用出来ます。
- **Close button[クローズボタン]** – ダイアログChecksumを閉じます。  
(\*1) これらの値はプロジェクトに保存されません。それぞれの新しいデバイス選択でデフォルトに初期化されます。

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

- **Custom address range for main checksum[メイン・チェックサムのためのメイン・チェックサム]**
- **Enabled[有効にする]** – ユーザー定義アドレスがバッファのデータのチェックサムの計算に使用されます。他方、もし、**Disabled[無効]**の場合はバッファのデータのチェックサムの計算にグローバル・バッファ開始とバッファ終了アドレスが使用されます。
- フィールド**From address[アドレスから]** と**To address[アドレスまで]** はメイン・チェックサム計算のアドレス範囲の入力に使用されます。アドレスはチェックボックス**Enabled[有効にする]**にチェックが入っている時のみ使用されます。
- 選択グループ **Checksum type[チェックサム・タイプ]**はメイン・チェックサムに使用する希望するタイプの選択使用します。詳し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 **buffer block(s) exclude from checksum calculation**  
– はチェックサム計算のタブと同じです。
  - ボタン **Apply[適用]** は **Main checksum options[メイン・チェックサム・オプション]** からのチェックサム設定を確認するために使用します。ノート: 一度ボタンが押されると、前回のチェックサム設定は失われます。
  - ボタン **Close[閉じる]** はチェックサム・ダイアログを閉じるために使用されます。もし、設定で変更を加えた場合、**Apply[適用]** を押すまで変更は反映されません。
- (\*2) それらの値はコフィギュレーション・ファイルとプロジェクト・ファイルにストアされます。プロジェクト・ファイルからの設定が優先されます。

### **チェックサム・タイプ**

#### **Byte sum (x8)**

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

#### **Word sum Little Endian (x16)**

バッファのデータは現在のバッファのビューモードの構成に関係なくワード単位で加算されます。32ビットを超える任意のキャリー・ビットは無視されます。このチェックサム・モードはメイン・プログラム・ウィンドウのチェックサム値の後に表示され文字列(x16LE)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)**

バッファのデータは現在のバッファのビューモードの構成に関係なくワード単位で加算されます。32ビットを超える任意のキャリービットは無視されます。このチェックサム・モードはメイン・プログラム・ウィンドウのチェックサム値の後に表示され文字列(x16BE)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**

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

#### **CRC-XMODEM**

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

#### **CRC-16**

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

#### **CRC-32**

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

#### **MD5**

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

### SHA-1

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

### Checksum forms

**Straight** – 追加の調整無し 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  $SUM + SUPPL. = 0 (+ carry)$ .

**デバイス依存チェックサム** – いくつかのデバイスが適応されます。

例えば、STMicroelectronics's STM8ファミリー。メイン・チェックサムのためのチェックサム・モードはメイン・プログラムのラベル・チェックサム上でクリックすることでポップ・アップ・メニュー(又は、メニュー・ショートカット)でセットすることが出来ます。

**Shift+Ctrl+1** - Byte sum (x8),

**Shift+Ctrl+2** - Word  
sum Little Endian

(x16) **Shift+Ctrl+3** -  
Word sum Big  
Endian (x16) etc...

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

### **Device[デバイス]**

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

### **Device / Select from default devices[デバイス/デフォルト・デバイスから選択]**

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

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

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

### **Device / Select device ...[デバイス/選択されたデバイス...]**

このウィンドウは現在のプログラマーによりサポートされている全てのデバイスのタイプを選択することができます。デバイスを **名前**、**タイプ** 又は、**マニファクチャラー** により選択することが可能です。

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

例えば:

- ・ デバイス *Am27C512-150, Am27C512-200* と *27C512-250* はソフトウェアでは *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。

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



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・アダプタ名で全体のデバイス・リストのフィルターリングのためにマスクを入力することができます。

スペースはフィルター項目の区切り文字として "OR"機能を持っています。スペースを含め正確なフィルタ文字列を入力したい場合はクォテーション・マーク " を使用します。

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

#### **Select device ... / All**[デバイス選択.../全部]

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

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

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

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

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

#### **Select device / Only selected type**[デバイス選択/選択されたタイプのみ]

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

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

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

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

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

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

#### **Device / Select EPROM by ID[デバイス/ID による EPROM 選択]**

このコマンドはデバイス ID を読むことでアクティブ・デバイスとして EPROM を自動選択するのに使用します。プログラマーはチップに焼き付けられているマニファクチャラーとデバイスの ID を読むことで EPROM を自動的に認識します。これは、この機能をサポートしている EPROM のみに適応されます。もし、デバイスがチップ ID とマニファクチャーID をサポートしていないときは UNKOWN 又は、NOT SUPPORTED DEVICE であることを告げるメッセージを表示します。

他に一致したチップ IC とマニファクチャー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:** プログラマーはソケット上の対応ピンに高電圧に適応します。これはデバイス ID を読むために必要です。EPROM でないデバイスをソケットに装着しないで下さい。プログラマーが高電圧に対応しているときは、ダメージを与えることがあります。

このコマンドを 2764 と 27128 EPROM タイプに適応させることはお薦めできません、それらのほとんどで ID がサポートされておりません。

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

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

#### **Device / Device options / Operation options [デバイス/デバイス・オプション/操作オプション]**

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

### 通常使われる項目のリスト:

#### アドレス・グループ

Device start address デバイス開始アドレス (デフォルト 0)

Device end address デバイス終了アドレス (デフォルト・デバイス・サイズ-1)

Buffer start address バッファ開始アドレス (デフォルト 0)

#### スプリット

このオプションはプログラミングやデバイスを読み取る際にバッファの特殊なモードを設定することが出来ます。16-bit、又は、32-bit のアプリケーションを 8bit のデータ・メモリ・デバイスに書き込みに使用する際に分割オプションを使用すると特に有用です。次の表はバッファからデバイスとデバイスからバッファのデータ転送を説明しています。

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

実際のアドレスは次のようになります: (全てのアドレスは16進数です)

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

#### 用語説明:

デバイス・アドレス ADDR へのアクセスは Device[ADDR]と書かれています。

バッファ・アドレス ADDR へのアクセスは Buffer[ADDR]と書かれています。

ADDR 値は 0 からデバイス・サイズ(バイト)にすることが出来ます。

全てのアドレスはバイト指向アドレスです。

### インサクション・テスト・グループ:

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

有効の場合、プログラマーは ZIF ソケットとの接続が正しいかチップの全てのピンをチェックします。プログラマーはチップの誤装着と逆挿しの接触不良を認識しません。

#### Check ID byte チェック ID バイト (デフォルト ENABLE)

プログラマーは選択された各アクションの前に ID チェックを提供します。

これはデバイスの製造業者によって定義された ID コードをデバイスから読み出した ID コードとを比較します。ID エラーの場合、制御プログラムが次のように動作します:

- 項目を ENABLE[有効]にセットしている場合、選択された動作は終了します。
- 項目を DISABLE[無効]にセットしている場合は、選択した動作が続行されます。制御プログラムは単に ID エラーに関する警告メッセージを LOG ウィンドウに書き込みます。

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

ノート 1: ある種の古いチップは電子的な ID 機能を持っていません。

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

### コマンド実行グループ:

プログラミング前にブランクチェック(デフォルト DISABLE)

プログラミング前にイレース (デフォルト DISABLE)

読み込み後のベリファイ (デフォルト ENABLE)

プログラミング後のベリファイ (ONCE, TWICE)

ベリファイ・オプション (nominal VCC 5%,  
nominal VCC 10%,  
VCCmin VCCmax)

### ターゲット・システム電源供給パラメータ

ある種のタイプのデバイスでは ISP モードを利用出来ます。次のセッティングを含んでいます:

**Enable target system power supply[ターゲット・システムの電源供給を有効にする]** - プログラマーからターゲット・システムへの電源供給を可能にします。プログラムされたデバイスとのアクションの前にターゲット・システムのための供給電源がスイッチオンされ、そして、アクションが完了した後にスイッチオフされます。もし、操作が有効にされた後に定義されたレベルでの ISP の信号を保持する場合、プルアップ/プルダウン抵抗が無効にされた後、プログラマーは電源供給のスイッチをオフにします。

**Voltage[電圧]** - ターゲット・システムへの供給電源。供給電圧は 2V~6V です。

**ノート:**ターゲット・システムに供給する電圧値はターゲット・システムへの電流に依存します。ターゲット・システムへの適切な電源供給に達するためには、適切な電

圧と最大電流値が定義される必要があります。最大電流値はターゲット・システムの実際の消費電流と同じで可能な限り正確でなければいけません。

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

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

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

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

**Power down time[パワー・ダウン時間]** - ターゲット・システム内でターゲット・システムの電源供給がスイッチオフされた後、残留供給電圧を維持する時間(例えば充電されたコンデンサから)決定。この時間の経過後ターゲット・システムは供給電源なしでプログラマーから安全に切り離すことができます。

ターゲット・システムのパラメータ

これはある種のデバイスのタイプの ISP モードで利用出来ます。次のセッティングを含みます:

**Oscillator frequency (in Hz)[オシレータ周波数(in Hz)]** - デバイス(ターゲット・システム)のオシレータの周波数。コントロール・プログラムはそれによってプログラミング速度をセットしますので、正しい値をセットする必要があります。

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

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

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

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

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

### **プログラミング・パラメータ**

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

### **イレース・パラメータ**

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

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

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

シリアライゼーションには 3 つのタイプがあります。:

- インCREMENTAL[増加]・モード
- ファイルからのモード
- カスタム・ジェネレーター・モード

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

#### **シリアライゼーションのベーシック・ルール:**

もし、新しいデバイスが選択されますとシリアライゼーション機能はデフォルト状態(即ち、デisable)です。

実際に選択されたデバイスに対する実際のシリアライゼーション設定は **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。

INCREMENTAL・モードがアクティブなとき、次の実際の設定がコンフィギュレーション・ファイルにセーブされます。: アドレス, サイズ, シリアル, 増加ステップとモード ASCII / BIN, DEC / HEX, LS byte / MS Byte first の設定。

フロム・ファイル[ファイルから]・モードがアクティブなとき、次の実際の設定がコンフィギュレーション・ファイルにセーブされます。: インプット・シリアライゼーション・ファイル名と入力ファイルの実際のシリアル番号が表示されている行の実際のラベル。

プログラムがマルチプログラミング・モード(複数のソケット・プログラマーが実際に選択されている)時の独特な選択 - **Action on not programmed serial values due to error** - が **Serialization[シリアライゼーション]** ダイアログに表示されます。このセクションでは2つの選択が利用可能:

- Ignore not programmed serial values[プログラムされていないシリアル値は無視]
- Add not programmed serial values to file[プログラムされていないシリアル値をファイルに追加]

**Ignore not programmed serial values**はプログラムされていないシリアル値は無視されて何のアクションも一緒に行われません。

**Add not programmed serial values to file**はプログラムされていないシリアル値がファイルに追加されることを意味します。プログラムされていないシリアル値のファイルは"From-file"シリアライゼーション・モードのためのシリアライゼーション・ファイルと同じテキスト・フォーマットを持っています。従って、後で"From-file"シリアライゼーション・モードによるそれらのシリアル値をプログラムすることが可能です。

ノート: デバイスのプログラミングがユーザーによって停止された場合、プログラムはデバイスの次のバッチのために準備されているシリアル値に更新しません。同じ状況は、デバイスのプログラムが不完全な場合、例えば、あるデバイスの挿入テストエラーの発生した場合も同様です。

無視、又は、プログラムされなかったシリアル値の書き込みは、マルチプル・ソケット・プログラマー内のデバイスの現在のバッチから、少なくとも1つのデバイスが完全にプログラムされエラーなしでベリファイされた時のみ使用されます。

シリアライゼーションはある種のデバイスのタイプに対してはPG4UWコントロール・プログラムのメイン・バッファ、又は、使用可能な拡張バッファを操作することが出来ます。例えば、データEEPROMメモリーを搭載したマイクロチップPIC16FXXXデバイスです。どのバッファをシリアライゼーション・ルーチンにより使用するかは、ダイアログ"シリアライゼーション"で選択可能です。拡張バッファの選択は"playlist file mode"でのFrom-fileシリアライゼーションでは無視されます。この制限事項の詳細については"**From file mode**シリアライゼーション・モード"をご覧ください。もし、メイン・チェックサム・自動計算からシリアライゼーション・データを除外したい場合は、メニュー/バッファ/チェックサムをクリックし、ダイアログ "Checksum"のタブ "**Main checksum options**" で **Exclude buffer data for checksum** 機能を利用して下さい。

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

インクリメンタル・モードは各プログラム・デバイスに個々のシリアル番号を割り当てることが出来ます。各デバイスのプログラム操作に対してユーザーにより入力され

た開始番号が指定されたステップで増加され、そして、各デバイスのプログラミングに先立ち、選択されたフォーマットで指定されたバッファ・アドレスにロードされます。インクリメンタル・モードのためにユーザーが修正することが出来るオプションには以下の項目があります。

### S/N 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。

Bin(バイナリー) シリアライゼーション・モードの値は 1-8 が有効です。

ASCII シリアライゼーション・モードでは 1-16 の値が S/N サイズに対する有効値です。

### 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(最高又は、最低)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。

### スタート値

スタート値オプションはシリアライゼーションが開始されるイニシャル値を指定します。

一般的にシリアライゼーションの最大値は 32 ビット・ロング・ワードで \$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 つの最も大きなビットがゼロに設定されます。このあと、番号はつねに 0..\$1FFFFFFF 間の中です。(これはオーバー・フロー操作の基本スタイルです。)

### ステップ

ステップ・オプションはシリアル値の増加の増加ステップを指定します。

### 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 ストリング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 – シリアル番号を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の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 進法]

### Hexadecimal[ヘキサデシマル - 16 進法]

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。  
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。すなわち、各ニブルが 0 から 9 までの値を持たなければいけません。A から F の値は BCD 番号のニブルとしては使用出来ません。シリアル開始値とステップの数字を入れるまえに “Style” [スタイル]オプションでベースを選択して下さい。

### Save to buffer[バッファにセーブ]

Save to buffer[バッファにセーブ]オプションはバッファに書込むためのシリアル値のバイト・オーダーを指定します。このオプションは Bin S / N モード(ASCII モードには役立ちません。)に対して使用されます。

2つのオプションが利用出来ます。:

- LSByte first (インテルのプロセッサで使用されています。) はシリアル番号のリスト・シグニフィカント・バイトをバッファの最下位アドレスに置きます。
- MSByte first (モトローラのプロセッサで使用されています。) はモースト・シグニフィカント・バイトをバッファの最下位アドレスに置きます。

### スプリット・シリアル番号

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

次のスプリット・オプションが利用可能:

- チェック・ボックス “**Split serial number**” –スプリット機能のターンオン/オフ
- Split gap** – スプリット・シリアル番号のフラグメント間に置くバイト数を指定
- S/N fragment size** – シリアル番号はこのオプションにより指定されたサイズでフラグメントにスプリットされます。

### サンプル:

#### サンプル 1:

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

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

モード: インクリメンタル・モード



S/N size: 4 bytes  
S/N mode:: Bin Style: Hex  
Save to buffer: LS Byte  
first Address: 7FFFCH  
Start value: 16000000H

#### ステップ: 1

以下の値がデバイスに書き込まれます:

1番目のデバイス

アドレス データ

007FFF0 xx xx xx xx xx xx xx xx xx xx xx 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 etc.

"xx" はデバイスにプログラムされるユーザー・データ

シリアル番号はデバイスのアドレス7FFFCH から7FFFFHに書き込まれます。

シリアル番号のサイズは4バイトです。

#### サンプル 2:

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

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

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分の使用を可能にし、そして、シリアル番号の各バイトの間にはRETLW命令コードが挿入されます。

"From file"シリアライゼーションは独自のシリアル番号ファイルを使用しています。このファイルは色々なシリアル番号で構成することができます。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。ノート:PG4UWのシリアル・ファイル形式はマイクロチップ社のMPLABによって生成されるSQTPシリアル・ファイルとは互換性がありません。

#### サンプル 2a:

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



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

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

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

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

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

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

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

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

RETLW命令を使いたいと仮定しますとバッファは:

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

これは次のステップで行うことができます:

**A)** メイン・バッファのアドレス 40H に 4 つの RETLW 命令を書く(これは手動でバッファを編集するか、又は、適切な内容のファイルをロードすることによって行うことができます)。各 RETLW 命令の下位 8 ビットは重要ではありません、それは各 RETLW 命令の下位 8 ビットには、シリアライゼーションの正しいシリアル番号のバイトが書き込まれる為です。

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

アドレス	データ
0000040	3400 3400 3400 3400 xxxx xxxx xxxx xxxx

各RETLW命令8ビットはゼロです。それらは如何なる値も持つことができます。

**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
Start value:	1234ABCDH



Step: 1  
S/N mode: BIN  
Style: HEX  
Save to buffer: LS Byte first  
Split serial number: checked  
Split gap: 1 byte(s)  
S/N fragment size: 1 byte(s)

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

最初のデバイスがプログラミングされる時のシリアル番号のバッファ内容は:

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

2番目のデバイスは:

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

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

#### サンプル 2b

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

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

次のステップでこれを行うことができます:

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

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

xx – はバイト値を意味します。

**B)** 次のようなシリアライゼーション・オプションをセット:

S/N size: 3 bytes  
Address: 800h  
Start value: 123456h  
Step: 1  
S/N mode: BIN  
Style: HEX



Save to buffer: LS byte first  
Split serial number: checked  
Split gap: 2 byte(s)  
S/N fragment size: 2 byte(s)

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

最初のデバイスがプログラミングされる時のシリアル番号のバッファ内容は:

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

2番目のデバイスは:

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

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

サンプル 3:

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

スプリット・ギャップが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 xxxx

**ノート:** シリアライゼーション・オプションの効果がわからない時は、バッファに書き込まれる実際のシリアル番号をテストすることが可能です。テストは次のステップで行うことができます。

- 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.デバイス操作オプションでインサクション・テストとDevice IDチェックを無効にします。
- 3.プログラマーのZIFソケットにデバイスが装着されていないことを確認して下

さい。

4. デバイス・プログラム操作を実行(ある種のデバイスではプログラミングの開始前にプログラミング・オプションを選択する必要があります)
5. プログラミング操作が終了後(殆どの場合、デバイスが装着されていませんでエラーとなります)、どこにシリアル番号が置くかはアドレスのメイン・バッファ(View/Editバッファ)で見てください。

**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、例えば、16ビット・ワード(x16)より広いならシリアライゼーション・アドレスはワード・アドレスになります。

### **Device / Device options / Serialization / 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/プレイリスト・ファイル・モード]**

**Playlist From-file mode** 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(プログラムされた1つのデバイス)を意味します。Playlist From-file modeはメイン・ウィンドウと"From-file-pl"シリアライゼーションとしてパネル"Serialization"がPG4UWコントロール・プログラムの情報ウィンドウに表示されます。

#### **ファイル・フォーマット**

From-file シリアライゼーションplaylist fileはシリアライゼーション・データを含む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Classic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が、ファイル・フォーマットの違いはplaylist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ーを持つ必要があります。そのヘッダー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、フォーマットにおける個別の行で表わされます。  
`[label x] datafilename`

`labelx` – ラベルを現わします。

ラベルは入力ファイルの各行が、空白でない事を示すための識別子です。これらはファイルの各行をアドレス指定するために使用されます。ラベルはファイル内で固有である必要があります。ファイルのアドレス指定行は、入力ファイルでユーザーが定義した、シリアル値の読み出しを開始する行の必要な開始ラベルを意味します。

**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 -----
```

さらに詳しい全てに機能するFrom-file playlistのシリアライゼーションのサンプルはPG4UWがインストールされたディレクトリー

[Elnec\_sw\Programmer\examples\Serialization]の中の以下をご覧ください:

<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\Elnec\_sw\Programmer\Examples\Serialization\fromfile\_playlist\_example\

次のステップでシリアライゼーションをテストすることが出来ます:

1. PG4UWを実行
2. ELNECプログラマーが接続されて、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、イレーズ可能なメモリー・デバイスをお薦めします。(OTP メモリーではありません)
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。
8. 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ーに移動します。

- **option Delete used file**

プログラムは使用されたシリアライゼーション・データ・ファイルを削除します。

## ディレクトリー

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

次のエラー表示がPlaylist From-fileシリアルライゼーションで使用されます:

- s/n error #3 シリアルライゼーション・データ・ファイルは存在しません。
- s/n error #34 使用されたシリアルライゼーション・データ・ファイルを削除出来ません(シリアルライゼーション・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 使用されたシリアルライゼーション・データ・ファイルを移動出来ません(シリアルライゼーション・ファイルは書き込みプロテクト・ディスクに置かれているか、又は、ターゲット・ディレクトリーが存在しないかも知れません)

## **Device / Device options / Serialization / Custom generator mode[デバイス/デバイス・オプション/シリアルライゼーション/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。

### サンプル:

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

```
<PG4UW_inst_dir>\Examples\Serialization\custom  
generator_example\
```

一般的なパスはこのように見えます:

```
C:\Program  
Files\Elnec_sw\Programmer\Examples\Serialization\customgener  
ator_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[シリアライゼーション・データ・ファイル]**

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

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

### **Serialization generator[シリアライゼーション・ジェネレーター]**

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

### **最初のシリアル番号**

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

### **最後のシリアル番号**

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

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

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

トの番号。TRUEはデバイス操作がOKで終了したことを意味します。FALSEはデバイス操作がエラーで終了したことを意味します。

-N<serial number>

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

### カスタム・ジェネレータ・プログラム

カスタム・ジェネレータ・プログラム、又は、シリアルライゼーション・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・データをシリアルライゼーション.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。シリアルライゼーション・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ルライゼーション・オプションのダイアログで指定する必要があります。

プログラムは新しいシリアル・データが生成される度にPG4UWから呼び出されます。これは通常各デバイスのプログラミング操作の前に行われます。PG4UWコントロール・プログラムはシリアルライゼーション・プログラムにコマンド・ライン・パラメータを渡し、そして、シリアルライゼーション・プログラムはPG4UWコントロール・プログラムによって読み込まれるシリアルライゼーション.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 "シリアルライゼーション"において、最後のシリアル番号の値がゼロで無い時のみ渡されます。シリアルライゼーション・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、シリアルライゼーション.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"シリアルライゼーション.datファイル形式"のセクションを見て下さい。

### シリアルライゼーション .dat ファイル・フォーマット

シリアルライゼーション・ジェネレータによって生成されたシリアルライゼーション.datファイルは、次のテキスト形式でなければいけません。シリアルライゼーション.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ルライゼーションの状態(現在と最後のシリアル番号、シリアルライゼーション・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が次のシリアルライゼーションで使用する次のシリアル番号値を含んでいます。この値はシリアルライゼーション・ジェネレータで生成し、PG4UWに現在のシリアル番号に続くシリアル

- T03:<data format code> 番号を知らせます。  
シリアライゼーション・データ形式を指定。  
次のフォーマットがサポートされています:  
T03:50 又は、T03:55  
ASCIIスペース・データ形式  
T03:99 - Intel Hexデータ形式
- T04: シリアライゼーション・データが次の行からファイルの最後に続くことを示します。シリアライゼーション・データは例えばIntel Hex, ASCII Space等々の標準のASCIIデータ・ファイル形式の1つで保存されます。データに使用するフォーマットはレコードT03で指定する必要があります。

**サンプル:** 典型的なシリアライゼーション・データ・ファイル:

```
T01:000005 T02:001006 T03:99
T04: :0300000000096B89 :03000300000
005F5 :02000C005A0197 :01003F004F7
1 :00000001FF
```

ファイルは以下の情報で構成されています:

line T01 - 現在のシリアル番号 000005h

line T02 - 最終(最後)のシリアル番号 001006h

line T03 - 行 T04の後のシリアライゼーション・データ・フォーマットはIntel Hexです。

line T04 - デバイス・プログラミングの前にPG4UWのバッファにロードされるシリアライゼーション・データ、データはインテルHEXフォーマットで表現されます。

**オプション・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ライゼーションを停止し、PG4UWソフトウェアにより警告、又は、エラー・メッセージが表示される原因となります。シリアライゼーションをオフにする理由は現在のシリアル番号が許可された最大値の最終シリアル番号より大きいからです。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、それはシリアライゼーション・ダイアログでシリアル値の指定がゼロでないことを意味します。

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

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

カスタム・ジェネレータのシリアライゼーションが使用される場合、各デバイスのプログラミングが開始される前に、シリアライゼーション・エンジンがシリアル.datファイルを生成するために実行可能なシリアライゼーションを呼び出します。PG4UWのシリアライゼーション・エンジンはシリアライゼーション・ジェネレーターを呼び出すために

適切なコマンド・ライン・パラメータを管理します。.datファイルからのデータは直ちに内部プログラマー・バッファに読み出され、そして、プログラミング・デバイス用のデータとして使用されます。また、次のシリアル番号情報(レコード T02)はPG4UWに記憶されています。

#### デバイス・プログラミングの典型的なフローチャートは次の通りです:

1. プログラミング・バッチを開始
2. デバイス装着テスト
3. シリアライゼーションのシーケンスは4つのステップからなります:
  - 1) シリアライゼーション .datファイルを発生させるために適切なコマンド・ライン・パラメータでシリアライゼーション・ジェネレーターを呼び出す
  - 2) 利用可能なシリアライゼーション.datファイルを待つ
  - 3) シリアライゼーション .datファイルのデータをプログラマー・バッファへ読み込む(データはプログラミング・デバイスのために使用)
  - 4) データを読み込んだ後シリアライゼーション.datファイルを削除

#### 4. デバイス・プログラミング

#### 5. デバイス・ベリフィケーション

#### 6. 操作結果のチェック。

これは全てPG4UWコントロール・プログラムによって管理されます。シリアライゼーション・ジェネレーターの操作結果はどの操作とも関係がありません。コントロール・プログラムは要求されたコマンド・ライン・パラメータでシリアライゼーション・ジェネレーターを呼び出します。

**OK** - PG4UWは次のシリアル番号の要求をします。次のシリアル番号はステップ3で .datファイルから読み込まれています。

シリアライゼーション・ジェネレーターの呼び出しにより、コマンド・ラインで指定された次のシリアル番号を持ちます。

**ERROR** - PG4UWは新しいシリアル番号の要求をしません。最新のシリアル番号は次のデバイスで使用されます。

次のシリアライゼーション・ジェネレーターの呼び出しはコマンド・ラインで指定された最新のシリアル番号を持ちます。

#### 7. 次のデバイスへのプログラミングを繰り返しますか?

Yes ステップ2へ行く

No ステップ8を継続

#### 8. プログラミング・バッチの終了

#### ノート:

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

#### Device / Device options / Statistics[デバイスデバイス・オプション/スタティクス(統計)]

スタティクスは選択されたタイプのデバイスで処理されるデバイス操作の実際のカウン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、すなわち、プログラミング、デバイス操作の数がプログラムされるデバイスと同じ場合です。

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

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

チェック・ボックス - **Program[プログラム]**, **Verify[ベリファイ]**, **Blank[ブランク]**, **Erase[イレース]** と **Read[リード]** はスタティクス値がインク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(成功または失敗)に応じて**Success[成功]**又は、**Failure[失敗]**になります。

部分操作の組み合わせも1つの操作としてカウントされます。例えば、Readの後のVerifyを含むRead操作は1つの操作です。Eraseと/又は、Verify操作を含むプログラム操作も1つの操作としてカウントされます。

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

**Statistics[スタティクス]** ダイアログは **Statistics** パネルで右マウス・ボタンを押して、そして、表示されている項目 **Statics** はクリックすることで開くことができます。実際のスタティクス値は **Statistics[スタティクス]**パネルのコントロール・プログラムのメイン・ウィンドウに表示されます。

スタティクス・ダイアログは 7 つの値が含まれます - **Success[成功]**, **Operational Failure[操作失敗]**, **Adapter test failure[アダプター・テスト失敗]**, **ID check failure[ID チェック失敗]** と他の **Failure[失敗]**(prog. SW, HW)と **Total[合計]**

値の意味は:

<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

**Total** 全操作数

実際の **Statistics[統計]**値はメイン・ウィンドウの **Statistics[統計]**パネルに表示されます。

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

値の意味は:

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

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

#### **Device / Device options / Associated file[デバイス/デバイス・オプション/アソシエーテッド・ファイル(関連ファイル)]**

このコマンドはターゲット・デバイスの関連ファイルを設定するために使用されます。これはデフォルト・デバイス選択リスト又は、コントロール・プログラムをスタートした後バツファに自動的にロードすることが出来るファイルです。

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

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

#### **Device / Blank check[デバイス/ブランク・チェック]**

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

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

#### **Device / Read[デバイス/リード]**

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

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

### Device / Verify[デバイスベリファイ]

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

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

タブ Error は PG4UW メニュー・コマンド Device / Device options / Operation options [デバイス/デバイス・オプション/操作オプション] で標準としてその他のワーキング・エリアを設定することが出来ます。

### ノート:

- ベリファイ操作はソフトウェアでデータを持って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、ソロ(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あります。

### Device / Program[デバイスプログラム]

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

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

### Device / Erase[デバイスイレース]

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

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

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

#### **Device / Test[デバイス/テスト]**

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

#### **Device / IC Test[デバイス/テスト]**

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



## Device / Device info [デバイス/デバイス情報]

コマンドはターゲット・デバイスについての追加情報を提供します。 - デバイスのサイズ、オーガナイゼーション、プログラミング・アルゴリズムと、そのデバイスをサポートしているプログラマーのリスト(モジュールを含む)。

ここでパッケージ情報とその他の一般情報も見つけることができます。ISP プログラミングに関する情報も見ることができます。

デバイス情報: Microchip 25AA512 (ISP)

共通の情報    ISP接続詳細    パーツ番号検索

プログラムのISPコネクタ: (コネクタを正面から見た図)

ISPコネクタピンの記述

- 1 - ターゲットVCC検出のみ
- 2 - GND (VSS)
- 3 - ターゲット WP# チェックのみ
- 4 - GND (VSS)
- 5 - SO
- 6 - GND (VSS)
- 7 - CS#
- 8 - GND (VSS)
- 9 - SI
- 10 - GND (VSS)
- 11 - SCK
- 12 - GND (VSS)
- 13 - 接続しないでください!
- 14 - YES!
- 15 - OK
- 16 - エラー
- 17 - GND (VSS)
- 18 - GND (VSS)
- 19 - Target system power supply \*1
- 20 - Target system power supply \*1

ノート

\*1 プログラマーはターゲットシステムに電源を供給できます  
ヘルプの「デバイス・オプション/操作オプション」メニューの細部の項目を見て下さい。

推奨されるターゲット回路設計

R1,R4 抵抗の目的はターゲットシステムからプログラムチップを分離する事です。 BeeProg2プログラマーにおける推奨される R1,R4 の値は 1kΩ 以上です。 抵抗の代わりにショットバーを使うことも出来ます。  
VCC: アプリケーションの供給電圧。ターゲット・アプリケーションに電源が投入されているとプログラマーはこの信号をテストに用います  
GND: プログラマーとアプリケーションの共通グランド  
SI, SO, SCK: SPI バスの制御信号  
CS#: チップ選択入力  
WP#: 書き込みプロテクト・ピンはデバイスのプログラミング中はインアクティブなレベルにセットしなければなりません。プログラマーはチップの書き込み前にWPピンのレベルをテストするためにこの信号を使用します  
HOLD: 入力信号はコミュニケーションの一時停止に使用されます。デバイスを操作中にインアクティブ・レベル(logical HI)に接続されなければなりません。プログラマーはこの信号のレベルをチェックしません。  
更なる情報のためにELNECのウェブサイトから入手可能なISPアプリケーション・ノートを参照して下さい。  
ノート: ISPコネクタからプログラムされるチップのISPケーブルの長さ30cmを越えない様にして下さい。

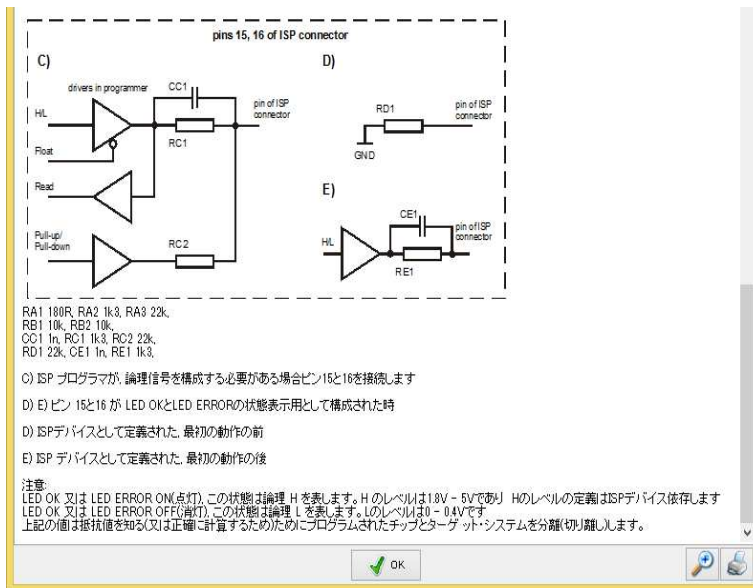
BeeProg2プログラマーのハードウェア内部を見てどのようにISPコネクタ信号が駆動されているのかを知りましょう:

pins 3, 5, 7, 9, 11, 13 of ISP connector

A)

pin 14 of ISP connector

B)



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

## Device /Jam/VME/SVF/STAPL/Mdoc...Player [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)製造業社、プログラミング装置メーカーとテスト装置製造業者のコンソーシアムによりサポートされています。

Jam™ 標準テストとプログラミング言語(STAPL), JEDEC standard JESD-71 は ISP(イン-システム・プログラミング)の目的のための標準ファイル・フォーマットです。

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、ベリファイ出来ますが、Jam STAPL はデバイスのリードのような他の機能は一般的には許可されていません。

Jam STAPL プログラミング・ソリューションは 2 つのコンポーネントから構成されています: Jam Composer と Jam Player.

Jam Composer はプログラムで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。

デバイスはプログラマーの ZIF ソケット、又は、ISP コネクタ経由のターゲット・システムでプログラムすることが出来ます。

それはコントロール・プログラムで選択されるデバイスの名前後に [PLCC44](Jam)又は、(ISP-Jam)のサフィックスにより表示されています。

マルチ・デバイスを JTAG chain: JTAG chain (ISP-Jam) 経由でプログラムとテストをすることが可能です。

さらに詳しい情報はウェブサイト: <http://www.altera.com> を見て下さい。

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",

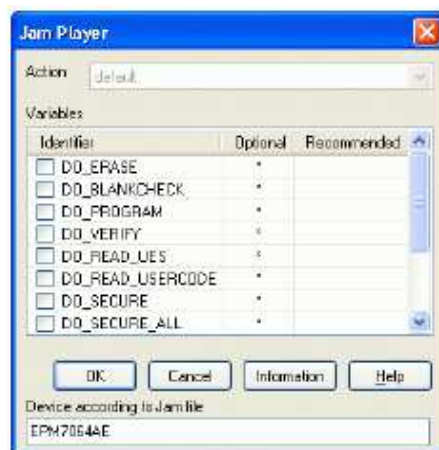
"AN 122: 組み込みプロセッサ経由で ISP & ICR のために Jam STAPL を使用" と関連アプリケーション・ノート.

ソフトウェア・ツール:

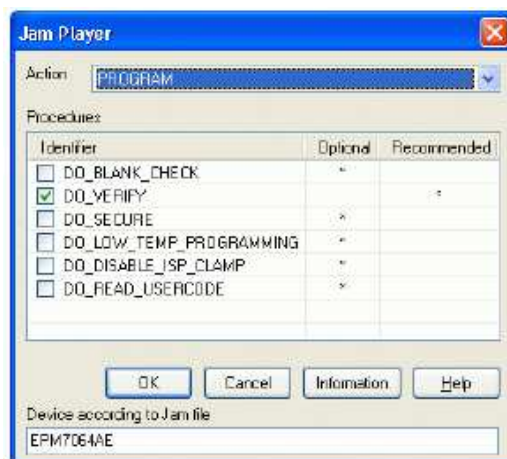
Altera: MAX+plus II, Quartus II, SVF2Jam utility (シリアル・ベクター・ファイルを Jam file に変換), LAT2Jam utility (ispLSI3256A JEDEC ファイルを Jam ファイルに変換);

Xilinx: Xilinx ISE Webpack or 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) (generates 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 Version1



JAM Player Version2

JAM Player ボタン、ダイアログ等の関連メニューは JTAG 対応デバイスを選択しないと表示されません。

#### Action

実行したい動作を選択して下さい。

version 2 の Jam file は各種の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・フロー開始

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

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。手順はオプションですが recommended 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

Jam file version 1 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)の数は一定であり、それは Jam ファイルに依存しません。もし、プリフィックス DO\_something を持った新しいブーリアン変数が必要な場合はご連絡下さい。

#### OK

マークされた適切な手順で選択されたアクションを受付ます。

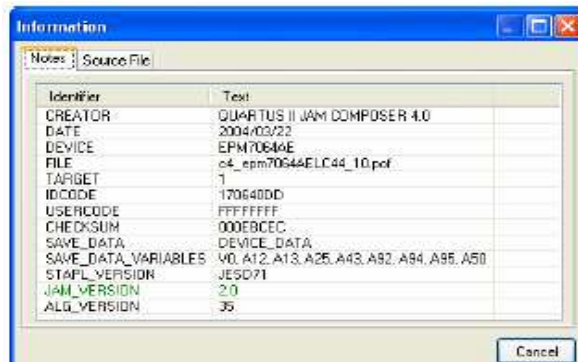
#### Information

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

#### Jam ファイルに従ったデバイス

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 の開始時に警告メッセージによってこの状況を示します。

## JAM ファイル情報ダイアログ



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でラベル、命令と引数を含みセミコロン(;)で終了します。

引数は目的のデータ・タイプ(即ち、ブーリアン、又は、整数)になるリテラル定数、変数、又は、式になります。各ステートメントは通常 Jam プログラムの 1 行を占有しますが、これは必須ではありません。改行は終了コメントを除いて Jam 言語の構文に重要ではありません。アポストロフィ分(')はインタプリタで無視されることを除いてはコメントを表わすために使用することが出来ます。言語は行の長さ、ステートメントの長さ、又は、プログラム・サイズのための任意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 ファイルへの変換についての情報:

- ・インストール Xilinx Integrated Software Environment(ISE) 6.3i ソフトウェアのフリー・ダウンロード: WebPACK\_63\_fcfull\_i.exe + 6\_3\_02i\_pc.exe (315MB 又は、概ね)
- ・起動 Xilinx ISE 6/Accessories/iMPACT
- ・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 で Jam ファイル名を入力,
- ・ダイアログ “Add Device” 拡張子 .jed の JED ファイルを選択,

- ・ JTAG チェーンで作成されたデバイスを選択 例えば: XC2C32A そして、シーケンス操作(即ち: Erase, Blank, Program, Verify; right mouse button),
- ・ メニューの選択項目で“Output/Stapl file/Stop writing to Stapl file”を選択
- ・ PG4UW を実行, デバイスを選択、即ち: Xilinx XC2x32A [QFG32](Jam), Jam ファイルをロード(Files of type: select STAPL File)
- ・ “Device operation options Alt+O”を選択、“Jam configuration”ボタンを押します。

警告 メニューからのデバイス・選択"Select Devices"と Jam ファイルは恐らく違います! 続けますか?"

はいを選択 (Xilinx ソフトウエアは行: NOTE "DEVICE" "XC2x32A";を Jam ファイルに含んでいません).

ダイアログ“Jam player”でアクションと手順を選択、ダイアログと終了、ツール・バーから“Play Jam”をクリック、そして、Log ウィンドウを読みます。

STAPLE ファイルを使用して 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 STAPL ボタンをクリックして下さい。操作(即ち、プログラミング)が完了しますと Log ウィンドウに“Exit Code = 0... Success”と表示されます。

## ACTEL デバイスのために PDB ファイルを JAM STAPLE に変換に付いての 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 ファイルに変換する:

- ・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・アローン・バージョンとして<<http://www.actel.com>>からダウンロード)
- ・FlashPro を起動
- ・New Project[新規プロジェクト]ボタンをクリック、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・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下さい。この値は手動で STAPL ファイル(情報ウィンドウでも見ることが出来ます)のヘッダーの値と比較することが出来ます。

注意: プログラムされたデバイスのチェックサムの値は実際の(壊れているかも知れません)デバイス・データの内容からカウントされませんが、この値はプログラミング中にスペシャル・メモリ・ローカライゼーションに保存され、そして、それだけ読み取っています!



VERIFY\_DEVICE\_INFO: 前のオプションと類似していますが、違いはプログラムされたデバイスのチェックサムと STAPLE ファイルのチェックサムを自動で比較します。比較の結果はメッセージ・ウィンドウに success 又は、error の何れかで表示されます。

VERIFY: STAPLE ファイルの内容とプログラムされたデバイス内容のデータの比較に関して、最も安全ですが、最も遅い(オプション 1 と 2 の約 1 秒に比べデバイスの要領に依存しますが数 10 秒)オプションです。選択されたファミリー機能の比較(FPGA Array, targeted FlashROM pages, security setting...)は bit ごとに実行されます。そして、もし、データの不整合が起こりエラー・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ビルト-インのマルチ-プロジェクト

機能を持っています。例としてデータ・コンテンツ(最初の STAPLE ファイル)と一緒にセキュリティ暗号化キー(2 番目の STAPL ファイル)をプログラムすることが出来ます。

#### IspVM Virtual Machine

IspVM Virtual Machine はバウンダリー・スキャン・テストのための IEEE 1149.1 Standard と互換性のあるプログラミング・デバイスのための仮想マシンです。

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。

デバイスはプログラマーの ZIF ソケット、又は、ISP コネクタ経由でターゲット・システムをプログラムすることで来ます。

それはコントロール・プログラムの選択デバイスの名前の後に[PLCC44](VME) 又は、(ISP-VME)サフィックスで示されています。マルチプル・デバイスは JTAG チェーン: JTAG chain (ISP-VME)経由でプログラムとテストが可能です。

さらに詳しい情報はウェブサイト: <http://www.latticesemi.com>

Software tools: Lattice: ispLEVER, IspVM System ISP Programming Software, PAC-Designer Software, svf2vme utility

(converts a serial vector file to a VME file)をご覧ください。



### **Programmer[プログラマー]**

#### **Programmer / Automatic YES! [プログラマー/オートマチック 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、プログラマー上の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ーのZIFソケットにある新しいデバイスの1つ又は、それ以上のピンを示した後、BUSY LEDは連続して点灯します。ここでプログラムは新しいデバイスの残りのピンが挿入されるための要求された時間を待ちます。もし、要求された時間(デバイス挿入完了時間)が過ぎたり、デバイスが正しく挿入されない場合、プログラムはこのステータスをERROR LEDで示します。

新しいデバイスが正しく挿入された後、プログラムはBUSY以外のステータスLEDをオフにします。そして、新しいデバイスで操作を開始します。

このモードは **Automatic YES!** モードにより有効、又は、無効にすることが出来ます。もし、新しいプログラマーが **Options / Find programmer** で選択されると、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 programmer's ZIF excluded from sensing[感知から除外されたプログラマーのZIF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ーの接続になります。

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)を検出し、それらのピンを前記の感知から除外されたピンのリストをセットするためのウィザードを実行します。

デバイスの選択の後、除外されたピンのリスト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・アダプタにバイパス・コンデンサが追加された場

合は、デフォルト・パラメータを無視又は、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** 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** 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。

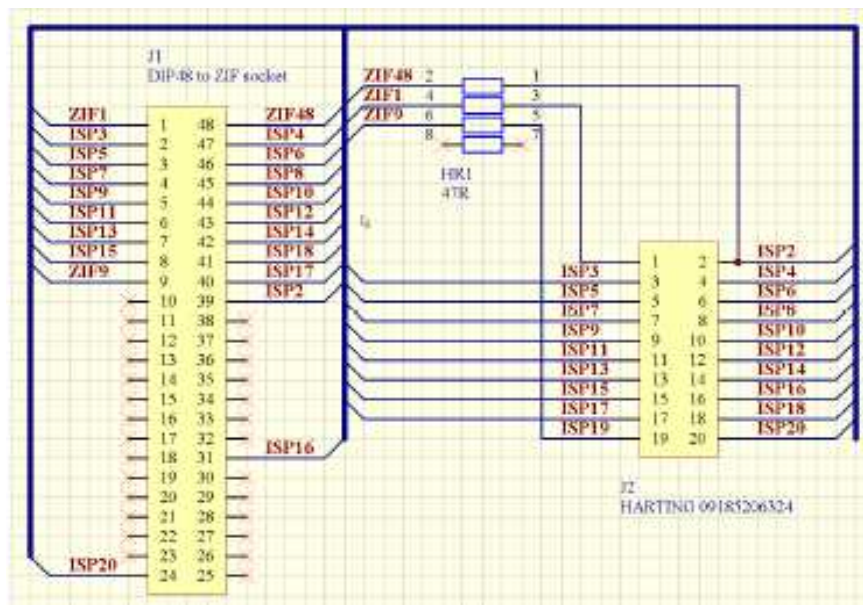
ノート: ある種の受動部品のあるソケット・アダプターを使用している時、例えば、供給電源をバイパスするためにコンデンサーを使用時、Automatic YES!機能はそれらのピンを知る必要があります。それはSetting Automatic YES! parameters[Automatic YES!パラメータのセッティング] ウィザードで行われます。これはAutomatic YES! 機能が正しく動作するために必要です。さもないと Automatic YES! 機能はピンが未だ接続されていると考え、ユーザーが新しいデバイスを挿入して新しいプログラミングを開始できません。

### **Programmer / Selftest ISP connector**[プログラマー/セルフテストISPコネクター]

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

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

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



### 20ピンISPコネクタのテストするためのシーケンス:

1. ISPコネクタ#2のためのISP用のDiagnostic POD[診断POD]を48ピン・デバイスとしてプログラマーの48pin ZIFソケットに装着します。
2. プログラマーに標準付属の20ピンISPケーブルをプログラマーのISPコネクタに接続し、フラット・ケーブルの他方を48ピンZIFソケット上のISPコネクタ#2のためのDiagnostic PODの20ピンの角形コネクタに接続します。ピンが相互に正しく接続されていることを確認して下さい(即ち、1-1, 2-2, ..., 20-20)。
3. PG4UW (Programmer/Selftest ISP connector[プログラマー/セルフテストISPコネクタ]...)でISPコネクタのセルフテストを実行して下さい。

このテストを6か月毎に実行されることをお勧めします。

## Options[オプション]

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

## Options / General options[オプション/一般オプション]

一般オプション・ダイアログはユーザーがプログラムの下記のオプションをコントロールすることが出来ます。

## File options [ファイル・オプション]

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

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

グループ内の現在のファイルが別のプロセスによって変更された場合は、実際にロード(現在の)ファイルの再ロードのモードを設定することができます。3つの選択があります:

- ・再ローディング・ファイル前のプロンプト
- ・自動的に再ロード
- ・現在ファイルの変更スキヤニングを無視  
ファイルの修正がテストされた時の3つのシチュエーションがあります:
- ・他のアプリケーションからコントロール・プログラムへの切り替え
- ・デバイス操作 Vefiry、又は、Program の選択
- ・最後のデバイス操作のリポートがダイアログ"Repeat?"で選択された時

Load file format ロード・ファイル・フォーマット]はファイルのロードのためのファイル・フォーマット認識のモードをセッティング。自動ファイル形式が選択された時、プログラムで利用可能なサポートされたフォーマットの各々に対してローディング・ファイルとテスト・ファイルのプログラム・フォーマットを解析します。

ファイル・フォーマットがサポートされている形式のいずれかと一致する場合はファイルが検出された形式でバッファリングするために読み込まれます。

マニュアル・ファイル形式はユーザーがサポートされているファイル形式のリストから明示的に望んだファイル形式を選択することができます。

ファイル形式がユーザが選択した形式と一致しない場合、ファイルは不完全、又は不正にロードされます。

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

## File extensions[ファイル拡張子]

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

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

ノート: 各ファイル・フォーマットに対して複数のマスクを指定することができます。  
セミコロンは拡張子のための区切り文字として使用されます。

例: Motorola: \*.MOT;\*.S19

2つのファイル・マスクを定義 \*.MOT と \*.S19 を定義 - Motorola ファイル形式。

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

### Buffer[バッファ]

このオプション設定により、デバイスを選択した時にバッファメモリ内容を自動で指定 データにクリアすることができます。

消去時データ

- 自動設定 を選択した時は使用するデバイスの BLANK 状態にクリアします。
- ユーザーが指定する を選択した時はその後ろにある - 消去データを指定する部分に入力したデータでクリアします。

### 指定したバッファファイルのバスを有効にする

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

### Erase buffer before selecting of new device[新しいデバイスの選択前にバッファをイレース]

動作を選択することができます。  
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)することができます。これはグループボックス Erase value and Custom erase value edit field[イレース値とカスタム・イレース値編集フィールド]で制御することができます。

ノート: バッファの消去を行うために多くの時間を消費しますで大規模なデバイス(8MB 以上メガバイト)の場合この機能を使用することはお勧めしません。  
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)上のアクティビティ終了後、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。

次のオプションを含む次のアクションのためにサウンドを鳴らします:

- ・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。
- ・チェックボックス エラーの場合  
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。

## Colors and LEDs

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

## Errors[エラー]

このオプションはデバイス・ベリファイ・エラーをファイルにセーブする設定を行います。ベリファイ・エラーした時、45 個まで Log ウィンドウに書かれます。もし、ユーザーがベリファイ・エラー(データの差異)とファイルにセーブしたい場合、セクション Save device verify errors to file[デバイス・ベリファイ・エラーをファイルにセーブ]のオプションで 2 つの方法の何れかでセットすることが出来ます: 同じファイルに対する全てのベリファイ動作からの累積エラー、又は、最後のベリファイ動作からのみファイルにエラーをセーブ。

ベリファイ・エラーは指定された名前で Error file name edit box[エラー・ファイル名編集ボックス]によりファイルにセーブされます。

次のエラー・レポート・ファイル・オプションが利用出来ます:

- ・オプション No[いいえ](デフォルト)はベリファイ・エラーをファイルにセーブするのを無効にします。

- ・オプション New save[新規セーブ]はベリファイ・エラーLog ウィンドウに表示されると同時に最後のベリファイ動作からのみファイルに書かれます。新しいベリファイ動作書き込み前にファイルが削除されて新しいものが作成されます。
- ・オプション Append[追加]は全てのベリファイ動作からのベリファイ・エラーが同じファイルに累積されます。もし、ファイルが存在しない場合は、新しいファイルが作成されます。ボックス Error report file size limit[エラー・レポート・ファイル・サイズ制限]はファイルにセーブされるベリファイ・エラーの最大数をセッティング出来ます。

次のオプションが含まれています：

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

#### Log file[ログファイル]

このオプションは Log window[ログ・ウィンドウ]の使用と関連しています。このウィンドウに関するすべてのレポートは Log file[ログ・ファイル]にも書込まれます。ログ・ファイル名はデフォルトで"Report.rep"です。

コントロール・プログラムが Log file name edit box[Log ファイル名編集ボックス]で指定された名前と指定されたディレクトリーにこのファイルを作成します。

次のログ・ファイル操作が利用出来ます：

- ・No[いいえ] デフォルトは Log ウィンドウの内容を Log ファイルにコピーしません。即ち、全レポートは Log ウィンドウのみに表示されます。
- ・New[新規]にセットしますと古いログ・ファイルがある場合は削除され、コントロール・プログラム開始毎に 新しいファイルが作成されます。
- ・Append[追加] をセットしますと既存のログ・ファイルに全レポートを追加します。ファイルがない場合は新しいファイルを作成します。設定はプログラムのスタート時のみ適応されます。

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 edit box[Log ファイル名編集ボックス]でユーザーが指定した日付情報をセットすることが出来ます。

チェックボックスにチェックが入れられている時、プログラムは自動的に次のルールでユーザー指定の Log ファイル名に現在の日付を追加します：

ユーザーが指定したログ・ファイル名がフォーマット持っている場合：

<user\_log\_file\_name>.<log\_file\_extension>

日付が追加された名前は：

<user\_log\_file\_name><-yyyy-mmm-dd>.<log\_file\_extension>

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

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

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

.<log\_file\_extension> - .(dot)はファイル名の最初

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

アドバンスド・オプション Log ファイル・サイズ制限の利用について:

- ・ オプション Use Log file text truncating when file size limit is reached[ファイルサイズの上限に達したときにログ・テキスト切り捨てのファイルを使用] - チェックが入っている時、ログ・ファイルのサイズ制限がオン。これはログ・ファイルのサイズが指定された値に達した時にログファイルに含まれているテキストの一部が切り捨てられることを意味します。オプションにチェックが入っていない時、ログ・ファイルのサイズは無制限で、PC のフリー・ディスク容量のみにより制限されます。
- ・ オプション Maximum Log file size[最大ログ・ファイル・サイズ]は kB 単位でログ・ファイルの最大サイズを指定します。
- ・ オプション Amount of truncated text[切り捨てられたテキストの量]は最大ログ・ファイル・サイズに達した後に切り捨てられるログ・ファイル・テキストの%を指定することができます。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できます。

### Job Report[ジョブ・レポート]

ジョブ・レポートは最近のデバイスで行った操作の概要説明を表します。Job はプロジェクト・ファイルに関連付けられロード・プロジェクトで開始された操作から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
- ・ 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
- ・閉じる、又は、切断されるプログラマーのサイトが選択された時
- ・PG4UW を終了した時
- ・デバイス・カウント・ダウンが 0 になった時(スタート完了)
- ・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

<ordnum> はファイルの 10 進数の順序です。もし、同じ名前の既存のレポート・ファイルが存在する場合、新しいレポート・ファイルの順序は既存のファイルにインクリメントされます。

<prjname> は最近使用したプロジェクトのプロジェクト・ファイル名で、プロジェクト・ファイル名の拡張子はありません。

Example 1:

プロジェクト・ファイル c:\myproject.eprj を使用し、ジョブ・レポートのためのディレクトリーが d:\job\_reports\ にセットされている場合。

ジョブ・レポート・ディレクトリーにレポート・ファイルが無い場合、最後のジョブ・レポートのファイル名は: d:\job\_reports\job\_report\_000\_myproject.jrp

Example 2:

Example 1 からの条件を使用し、しかし、1 つのレポート・ファイルが既にあると仮定します。

このファイルの名前は d:\job\_reports\job\_report\_000\_myproject.jrp

最終的に新しいレポートのジョブ・レポート・ファイル名は:

d:\job\_reports\job\_report\_001\_myproject.jrp

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

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

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

Job Report ダイアログでユーザーはジョブ・レポートで行う操作を選択することができます。もし、何も選択しない場合(ボタンを閉じる)、ジョブ・レポートは PG4UW ログ・ウィンドウにのみ書かれます。

### **Automatic YES![オートマチック YES!]**

ユーザーはプログラマーとソフトウェアがアクティブ 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・セッティング(PG4UW ソフトウェアのプリセットとして)を無視することができます。

デフォルト・セッティング(PG4UW ソフトウェアのプリセットとして) - プログラマーはデバイスがプログラムされた時とプログラマーが新しいデバイスに交換されるのを待っている時にその状態を表示します。を点滅させます。

マルチ・ソケット・プログラマーではこの表示はされません(quiet mode)。シングル・ソケット・プログラマーは LED ビジー点滅によりこの状態を表示します。By LED Busy blinking セッティングをご覧ください。

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

By LED Busy blinking[LED ビジー点滅] - プログラマーは ZIF ソケットの数に関係なく(マルチ・プログラマー、シングル・ソケット・プログラマー)共にデバイスがプログラムされて新しいデバイスの装着を待つ時に LED ビジーで点滅することで状態を表示します。

デバイス操作の後、その操作の結果によりステータス LED error 又は、OK の何れか1つのみが点灯し、そして、LED ビジーが点滅します。もし、プログラムが ZIF ソケットからデバイスが取り除かれるのを検知しますと LED はオフになりますが、新しいデバイスで操作が繰り返すためにプログラムが用意していることを示すために LED ビジーは点滅します。プログラムが ZIF ソケットで(新しい)デバイスの

1 つ以上のピンを検知しますと、LED ビジーは連続して点灯します。この時点からプログラムは新しいデバイスの残りのピンが装着されるために要求された時間待ちます。

もし、要求された時間(デバイス装着完了時間)がオーバーフローしたり、デバイスが正しく装着されなかった場合、プログラムはこの状態を示すために LED Error を点灯します。デバイスが正しく装着された時、ステータス LED はオフになり、デバイスでの新しい操作が開始されます。

#### Remote control[リモートコントロール]

PG4UW コントロール・プログラムのリモート・コントロールは他のアプリケーションによる PG4UW アプリケーションのいくつかの機能を制御することができます。

これはマス・プロダクション・ハンドラー、又は、他のアプリケーションにデバイス・プログラマーを統合するために大変適した機能です。

PG4UW を制御するリモート・アプリケーションはサーバーとして動作します。プログラム PG4UW はクライアントとして動作します。PG4UW とリモート・コントロール・プログラムは TCP プロトコル経由で通信されます。

- これは PG4UW が 1 つの PC にインストールされ、そして、リモート・コントロール・アプリケーションが他の PC にインストールされ、そして、これらの PC がネットワーク経由で接続します。

リモート・コントロールのためのデフォルト TCP 通信設定は:

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

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

デフォルト設定は 1 台の PC(アドレス・ローカルホスト)上のリモートコントロールを使用を許可します。

PG4UW(クライアント)とリモート・コントロール・サーバーを同じ PC にインストールする必要があります。

ノート: もし、ファイアウォールがシステムにインストールされている場合、ファイアウォールはリモート・コントロール・サーバー、又は、クライアントがスタートする時に警告メッセージを表示することができます。

ファイアウォールがリモート・サーバー、又は、クライアントのネットワーク・アクセスの強化、又は、拒否を尋ねる質問を警告表示された時、'Allow'オプションを選択して下さい、そうでないとリモート・コントロールが動作しません。勿論、指定されたアドレスとポートのみでリモート・サーバー/クライアント・アクセスを許可するためにより厳格な権限をファイアウォール・オプションで指定することができます。

PG4UW のリモート制御アプリケーションとデモンストレーション・リモート・コントロール・アプリケーションの詳細については、PG4UW がインストールされたサブ・ディレクトリー\remotectrl に置かれ remotemanual.pdf を参照して下さい。

\*Remote control of PG4UW, user's manual

5. Using remote control with multiply programmers (multiprogramming)

Restrictions:を特に注意してご覧下さい。弊社でのサポートは出来ません。

### Save options[オプションの保存]

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

Don't save プログラム終了中にオプションをセーブしません、そして、保存オプションも聞いてきません。

Auto save 保存オプションを聞くことなくプログラム終了中にオプションをセーブ

Prompt for save プログラム終了前に保存オプションを聞いてきます。ユーザーはオプションをセーブするか、又は、しないかを選択することが出来ます。

### Other[その他]

プログラムの処理優先レベルをセットするために使用します。システム内のより要求の厳しいアプリケーションの実行がある場合は優先レベルの設定はプログラマーのパフォーマンス(デバイス・プログラミング時間)に影響を与えます。

アプリケーション優先レベルを Low にセッティングした場合書き込み時間が非常に遅くなりますので注意して下さい。

Tool buttons[ツール・ボタン], メイン・プログラム・ウィンドウでのツール・ボタン hint display[ヒントの表示]

オプションを変更することが出来ます。

パネル Start-up directory[起動ディレクトリ]はプログラムの起動時にディレクトリを選択するモードを選択することが出来ます。

デフォルトの起動ディレクトリはプログラムが呼び出されるディレクトリを意味します。プログラムは最後に終了されたディレクトリはプログラムの最後に終了した最後のカレント・ディレクトリを意味します。

このディレクトリはディレクトリ履歴リストから最初のディレクトリを前提としています。

### Options / View[オプション/ビュー]

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

### Options / View / Main Toolbar[オプション/ビュー/メイン・ツールバー]

メイン・ツール・バーの表示又は、非表示はこのコマンドを選びます。

### Options / View / Additional Toolbar[オプション/ビュー/アディショナル・ツールバー]

アディショナル・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 / Display errors[オプション/表示エラー]

このオプションはプログラムされたデータのペリファイの結果としてのエラー表示の形式をセットすることが出来ます。エラーをスクリーンに表示(最大 45)したり、カレント・ディレクトリーの **VERIFY.ERR** ファイルに保存したり、又は、表示をオフにする

ることが出来ます。エラー表示がターン・オフにされると、コントロール・プログラムは INFO ウィンドウのみに警告メッセージをレポートします。

この設定は **options / Save options [オプション/セーブ・オプション]** コマンドでディスクにセーブすることが出来ます。

### **Programmer / Find programmer[オプション/プログラマー検索]**

新しいプログラマーのタイプとコミュニケーション・パラメータを選択します。このコマンドは下記の項目を含んでいます。:

**Programmer[プログラマー]** – 新しいプログラマーのタイプをセットします。もし、Search all[すべてを検索]が選択されると、コントロール・プログラムはサポートされているすべてのプログラマーを検索します。

**Establish communication[通信の確立]** – 新しいプログラマーのための手動又は、自動通信の設定を行います。

**Speed[スピード]** – 通信速度を設定。もし、マニュアル通信設定が選択されますと、速度は最大速度からパーセンテージとして表示されます。

通信速度修正は PC<->プログラマーのケーブルに対して十分なパワーを持っていない、遅い LPT ポートを持った PC に対しては重要です。

もし、PC の LPT ポートとプログラマーの接続で通信の問題がある場合は、このコマンドを使用してください。コントロール・プログラムはプログラムがない、プログラマーとの通信が不安定等をレポートします。

Automatic establishing communication[自動通信設定]を選択しますと、コントロール・プログラムが最大通信速度を設定します。

**ポート** –LPT ポートを選択しますと、プログラマーが接続されているポートをスキャンします。もし、すべてのポートが選択されると、コントロール・プログラムは標準アドレスとして利用出来るすべてのポートをスキャンします。

**特別なポートのためのアドレス** – もし、特別なポートが選ばれた場合、LPT ポートのアドレスを設定します。

設定パラメータによるプログラマーのスキャンを開始するときは<Enter>キー、又は、OK ボタンを押して下さい。

### **Options / Protected mode[オプション/プロテクト・モード]**

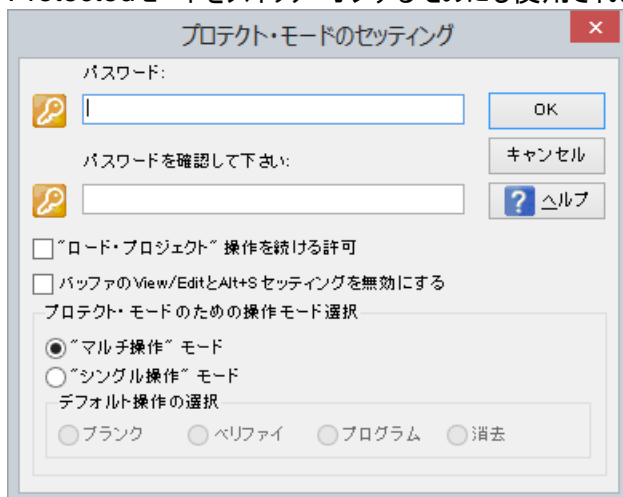
**Protected mode[プロテクト・モード]** はプログラムの特別なモードです。プログラムがProtectedモードの時は特定のプログラム操作とコマンドのバッファ、又は、デバイス設定の変更が無効にされます。Protectedモードは不用意にバッファ、又は、デバイス設定を変更することを防ぐために使用されます。Protectedモードは同じ種類のデバイスの大量のプログラミングに適しています。

Protectedモード機能はシングル・プログラミング制御ソフトウェアPG4UWとマルチ・プログラミング制御ソフトウェア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。もし、オプションが有効(チェックされている)になっている場合、Load project operation[ロード・プロジェクト操作]ボタンとメニューはProtectedモードで許可されます。

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

Protectedモードでバッファの内容を見れないようにしたい場合はこのオプションをアクティブにしてください。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]もアクティブに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

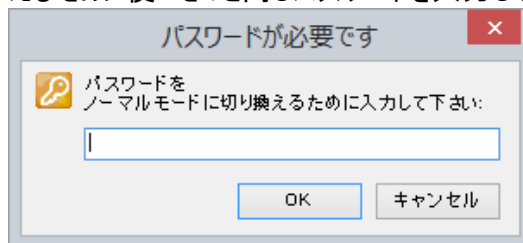
オプション **"Multi operation[マルチ操作]" モード** - リード以外のデバイス操作 (blank, verify, program, erase)の全てが利用できるプロテクト・モードの基本を現わしています。これは偶発的、又は、意図的なリード操作によってバッファ・デ

ータを変更してしまうことを防ぐ確実性を提供します。1つのプロジェクト(マルチプロジェクト)で全てのサポートされているデバイス操作を行いたいときに便利です。

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

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

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



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

Protectedモードがアクティブな時、ソフトウ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[プロテクト]モードが表示されます。



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

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

プログラムPG4UWMCはデフォルトでAdministrator Modelにセットされています。これはユーザーのためにブロックしている操作が適用されないことを意味します。し

かし生産で、いくつかのメニュー・コマンドをブロックするのに適しており、確実にするために、ユーザーは重要なプログラムの設定や構成を変更しません。オペレーター・モードはこの目的のために使用されます。

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

プログラムPG4UWMCはプログラムPG4UWに非常によく似たProtectedモードを持っています。違いはProtectedモードはメニュー・コマンドによりアクティベートすることが出来ますがプロジェクト・ファイルによりアクティベートすることが出来ません。

もう1つの違いは、PG4UWMCのProtectedモード設定はPG4UWMCが閉じられている間にPG4UWMCのコフィギュレーション .iniファイルにセーブされます。次のPG4UWMC開始の間 .iniファイルから取得した最新のProtectedモードの設定が使用されます。

PG4UWMCで使用できる1つのメニュー・コマンドがあります -

**Options/Protected mode**[オプション/Protectedモード]

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

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

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

プログラムをProtectedモードからNormalモードに切り替えるためにはメニュー・コマンド**Options/Normal mode**[オプション/通常モード]を使って下さい。

**"Password required"** ダイアログが現れます。Protectedモードに切り替えるために使ったのと同じパスワードを入力しなければいけません。

Protectedモードがアクティブな時、ラベル "Protected mode"がPG4UWMCメイン・ウィンドウのLog windowsのトップの近くに見えます。

ノート: 時にProtectedモードがアクティブ状態からインアクティブ状態(Normal mode)に切り替えられる時、ある種のコマンド(例えば、"Load project")は無効のまま保持されます。これはボタン **Stop ALL** をクリックすることで解決することが出来ます。



## Multi-projects[マルチプロジェクト]

**Multi-project**は**sub-projects**と**multi-project**の作成中にセーブされた情報をベースにした如何なるデバイスでの操作のシーケンスを実行するための特別な機能です。

実際にマルチプロジェクトを使用する:

- マルチチップ・デバイスのプログラム
- デバイス操作のシーケンスの実行(即ち、Program + Verify + Verify + Verify)

更に詳しくは操作モードをご覧ください。

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

### 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-デバイス操作にはマスター-デバイスのsub-devices(chips)に関連した部分的なsub-projectsが含まれているマルチ-プロジェクト-ファイルが必要です。マルチ-プロジェクト-ファイルはマルチ-プロジェクト-ウィザードで作成することが出来ます。

ウィザードは次の主機能を持っています:

- sub-projectsを選択、そして、最終的なマルチ-プロジェクトをビルドします。
- 既存のマルチ-プロジェクト-ファイルを読み \*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コマンドを読みすることでマルチ-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-ファイルを読み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。

**ノート:**

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

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

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

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

マルチ-プロジェクト・ファイルの作成時は以下のステップを推奨します:

- "classic"プロジェクトを作成, マルチチップ・デバイスの各sub-deviceのための1つのプロジェクト。ジェネリック・デバイスのためのプロジェクトと同じ方法でプロジェクトを作成:
- マルチチップ・デバイスに搭載されている希望するチップに従いsub-deviceを選択 \*1
- デバイスのパラメータをセット、設定し、そして、希望のデバイスのデータをPG4UWのLoad Fileコマンドでバッファにロード
- 確認のためにデバイス上でデバイス操作を実行することでデバイスのテストを行って下さい。
- 全てOKの場合、Save projectコマンドでプロジェクト・ファイルを作成することが出来ます。
- そのマルチ-プロジェクトを使用すべきマスター・マルチチップ・デバイスを選択します。マルチチップ・デバイスを選択後、マルチ-プロジェクト・ウィザードは自動的に開かれます。
- マルチ-プロジェクト・ウィザードのAdd projectボタンにより必要なプロジェクトを追加します。各プロジェクトはマルチチップ・デバイスの対応した1つのsub-deviceを表します。
- sub-project選択の完了後、最終的なマルチ-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す。マスター・デバイスとの関連付けは必須ではありません。ユーザーへの配慮のため1つのマルチ-プロジェクト内に正しいサブ・デバイス(sub-project)を結合する方法をサポートしているだけです。この機能はJTAGチェーンでISPプログラミングを使用して異って定義されたプロジェクトを使用する時には特に便利です。

マルチ-プロジェクト・ウィザードは次の動作の1つにより開くことが出来ます:

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

\*1 PG4UWデバイス・リストでのマスター・デバイスとサブ・デバイス・パーツの名前

Master-device: マルチチップのオリジナル部品名[パッケージ・タイプ]

Sub-devices: マルチチップのオリジナル部品名[パッケージ・タイプ] (part1)  
マルチチップのオリジナル部品名[パッケージ・タイプ] (part2) .

.....

マルチチップのオリジナル部品名[パッケージ・タイプ] (part n)

サンプル:

Master-device: TV0057A002CAGD [FBGA 107]

Sub-devices #1 TV0057A002CAGD [FBGA 107] (NAND)

#2 TV0057A002CAGD [FBGA 107] (NOR)

## デバイス操作実行のためのマルチ-プロジェクトの使用

既存マルチ-プロジェクト・ファイルの典型的な使用方法は次の順序です。

PG4UWでのシングル・プログラミング:

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

ノート:

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

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

- Load projectメニューでマルチ-プロジェクトをロード
- 利用出来るデバイス操作ボタン(Blank, Verify, Program, Erase)の1つを使って希望するデバイス操作を実行、最も使用されるのはプログラム・デバイス操作です。選択されたデバイス操作はマルチ-プロジェクトで定義された各sub-deviceのためのsub-projectのロードとその結果としてのsub-deviceのプログラミングのシーケンスとして実行されます。そして、これはマルチ-プロジェクトの主目的です - マルチチップ・デバイスの各チップのためのデバイス操作のシーケンスを自動で実行。このコンセプトのサイド・イフェクトは、そのデバイスの進捗インジケータは各sub-device操作の開始時に0にリセットされますので、マルチチップ操作の実行中にプログレス・バーが0数回に"ジャンプ"されているように見える動作になります。

- 全てのsub-devicesのプログラミングが完了(又は、エラー)後、PG4UWMCにデバイス操作の結果情報が表示されます。プログラマーのソケットからプログラムされたデバイスを取り除き、そして、新しいデバイスを挿着することが出来ます。サイトに対する操作ボタンを押すか、又は、プログラマー・サイトのYES!ボタンを押しますとマルチチップ・デバイスのプログラミング・シーケンスが再開されます。  
\*もしAutomatic YES!機能がオンされている場合、プログラマー・ソケット状態とプログラムされたデバイスの取り除きと新しいデバイスを装着後デバイス操作シーケンスが自動的に開始します。Automatic YES!の詳細はProgrammer / Automatic YES! を参照して下さい。

**ノート:**

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

**Options / Save options [オプション/セーブ・オプション]**

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



---

**PG4UWMC マルチ制御プログラム**

---

プログラムPG4UWMCは同一PCにUSBで接続された複数のプログラマー、又は、1つのマルチプログラミング・プログラマーで完全に並列同時デバイス・マルチプログラミングを行うために使用されます。

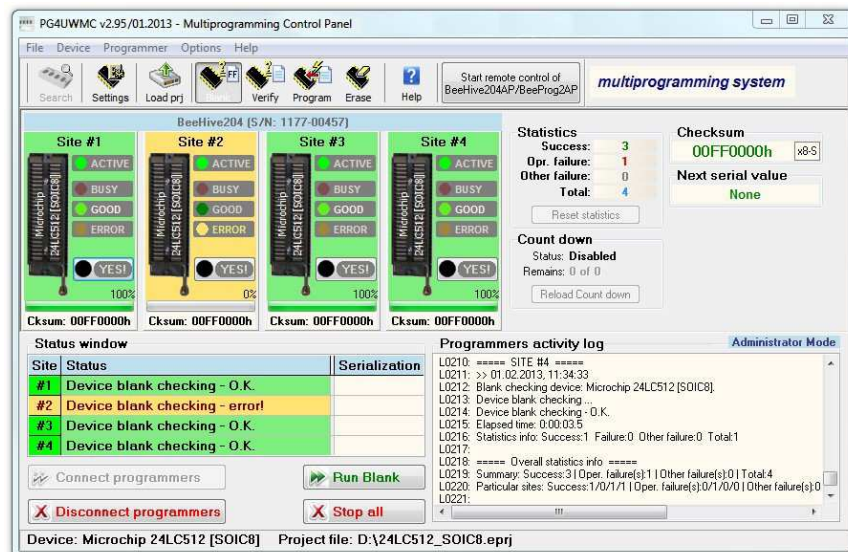
PG4UWMCは大量生産の操作の監視を簡単にするためにフォーカスされています。PG4UWMCのユーザー・フレンドリーなユーザー・インターフェイスは使いやすさで多くの強力な機能を組み合わせて重要でない詳細に負担を掛けることなく全ての重要な動作と操作結果を提供します。

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
- セッティング・ダイアログ・ウィンドウ
- "Search for Programmers" ダイアログ・ウィンドウ

### PG4UWMCのメイン・パートの基本説明



PG4UWMC メイン・ウィンドウ

PG4UWMCのメイン・ウィンドウは次のパーツで構成されます：

#### メニューとツール・ボタン

#### ツール・ボタン・セッティング

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

**パネル Site #1, Site #2,...** パネルは以下について表示します:

- 選択されたプログラマーのサイト
- プログラマー・サイトの動作状況
- 現在のデバイス操作状態と/又は、結果

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

### ボックス Statistics

Statistics[統計]はプログラムされたデバイスと成功と失敗したデバイスの数について通知します。

**ノート:** 'Reset statistics' ボタンはサイト上で操作が実行されるまで無効にされます。操作を中止したい場合はボタン'Stop All'を押して下さい。

### Checksum

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

### パネル Status window

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

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

**Log window** ステータス・ウィンドウの右側

Log windowはProgrammers activity logと表示され、プログラマーの接続/非接続、デバイス操作結果やその他の情報が含まれています。

### ボタン Connect programmers

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

### ボタン Disconnect programmers

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

### ボタン Run <operation>

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

### ボタン Stop ALL

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

### ボタン Help

ヘルプを表示するために使用します。

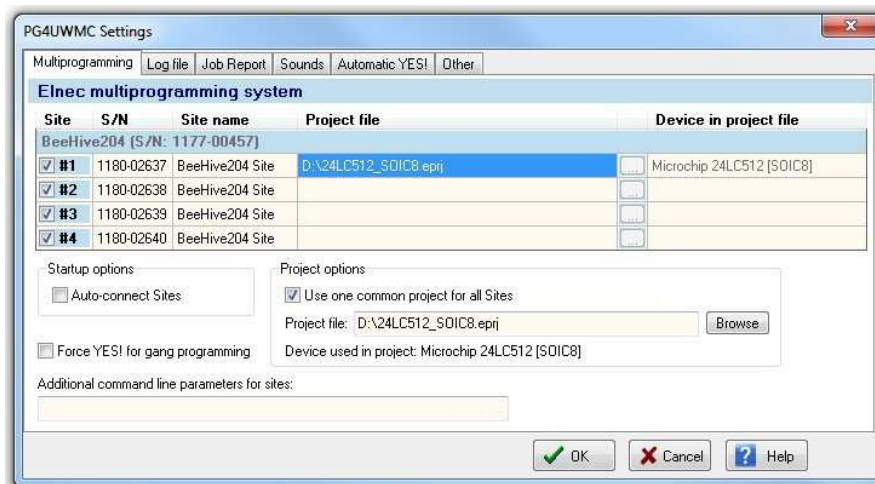


### ボタン Start remote control of BeeHive204AP/BeeProg2AP

このボタンは自動化プログラマーBeeHive204AP/BeeProg2APでのみ利用可能で、PG4UWMCの設定 ダイアログ/マルチプログラミング/プロジェクトのオプションで**Use Site #1 project for all Sites**[全てのサイトのためにサイト#1のプロジェクトを使用]をチェックにされている場合。これはPG4UWMCインターフェースのリモート・コントロールを開始するために使用されます。

ボタンのキャプションでプログラマーは最近接続に使用したプログラマーの名前に置き換えられます。オプションはサイト上でロードされたプロジェクトに応じてモジュールの検出をアクティブにします。

### PG4UWMC Settings ダイアログ



PG4UWMC Settings ダイアログは次のオプションをセット、又は、表示するために使用されます:

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

### パネル Multiprogramming

PG4UWMCの最初のコントロール・パネルは次を含んでいます:

- 指定されたサイト番号で個々のプログラマー・サイトを使ってサイト番号 #1, #2, #3, #4を有効/無効にするチェック・ボックスを含みます。
- プログラマー・サイトのシリアル番号に付いての情報
- Project fileは各PG4UWの実行にロードする個々のプロジェクトのセッティングのための編集行Project: #1, Project: #2, ...Project: #4を含みます。プロジェクト・ファイル名は手動で入力出来ますが、又は、ダイアログ**Select project file**によって各プロジェクト編集行の右側におかれたボタン"..."上でクリックするこ

とで各サイトのために開くことができます。もし、プロジェクト名編集行が空白の場合、自動プロジェクト・ロードは行われません。

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

同じデータで同じデバイス・タイプをプログラムする必要がある時、チェックボックスはチェックにされていなければいけません。

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

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

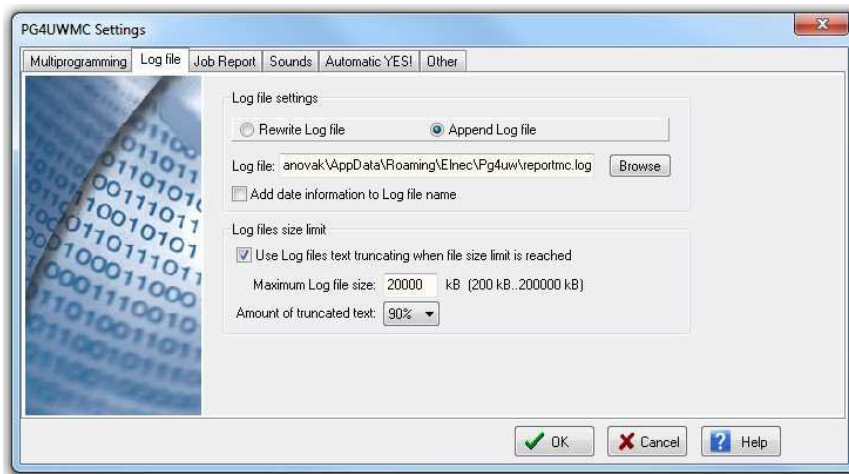
**Force YES for gang multiprogramming mode**[ギャング・マルチプログラミング・モードのための強制YES]

各プログラミング・サイトが独立して動作し、他のプログラミング・サイトが実行されている間にオペレーターがプログラムされたデバイスを再ロードできる時、マルチプログラマーでのマルチプログラミング操作の標準モードは**concurrent multiprogramming mode**[同時マルチプログラミング・モード]です。**gang multiprogramming mode**[ギャング・マルチプログラミング・モード]ではあらかじめ定義された操作を任意のYES!ボタンを押すことで**同時に全てのプログラミング・サイト上で操作を開始**します。

ノート:

- 全てのアクティブ(現存し有効な)サイトで動作
- 何れかのサイトがビジーな間は開始はブロックされます。
- このモードではAutomatic YES!は無効にされます。

パネル **Log file セットアップ**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制御プログラムの動作フローについての情報を含むテキスト・ファイルで、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・ファイルと実行している各プログラマー・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>のフォームでサイト数が続きます。

#### 例えば:

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

次のオプションをLogファイル作成のために設定することが出来ます。

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

チェックボックス **Add date information to Log file name**[日付の情報をログファイル名に追加]は Log file name [Log ファイル名] 編集ボックスでユーザーが指定した日付情報をセットすることが出来ます。チェックボックスにチェックが入

れられている時、プログラムは自動的に次のルールでユーザー指定の Log ファイル名に現在の日付を追加します:

ユーザーが指定したログ・ファイル名がフォーマット持っている場合:

**<user\_log\_file\_name>.<log\_file\_extension>**

日付が追加された名前は:

**<user\_log\_file\_name><-yyyy-mmm-dd>.<log\_file\_extension>**

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

**例えば:** ユーザー指定ログ・ファイル名: c:\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・ファイル名を付けたい場合、次の様にログ・ファイル名をとして指定することができます:

**.<log\_file\_extension> - dotはファイル名の最初**

**例えば:** ユーザー指定ログ・ファイル名: c:\Vogs\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\2006-nov-07.log

#### **アドバンスド・オプション Logファイル・サイズ制限の利用について:**

•オプション Use Log file text truncating when file size limit is reached - チェックが入っている時、ログ・ファイルのサイズ制限がオン。これはログ・ファイルのサイズが指定された値に達した時にログファイルに含まれているテキストの一部が切り捨てられることを意味します。オプションにチェックが入っていない時、ログ・ファイルのサイズは無制限で、PC のフリー・ディスク容量のみにより制限されます。

•オプション Maximum Log file size[最大ログ・ファイル・サイズ]は kB 単位でログ・ファイルの最大サイズを指定します。

•オプション Amount of truncated text[切り捨てられたテキストの量]は最大ログ・ファイル・サイズに達した後に切り捨てられるログ・ファイル・テキストの%を指定することが出来ます。大きい値はより多くのテキストがログ・ファイルから切り捨てられる(削除)されることを意味します。

**ノート:** '+'で開始される行はログファイルに表示されますが、画面のログには表示されません。

#### **共通情報:**

**Index of Programmer Site**[プログラマー・サイトのインデックス]は明確に実行中の各プログラマー・サイトを定義する1から8の整数番号です。

**Serial number of Programmer Site**[プログラマー・サイトのシリアル番号]は明確に使用されているプログラマー、又は、プログラマー・サイトを定義しています。

シリアル番号とプログラマー(サイト)を見つけるまでUSBバス上に接続された全てのプログラマーを検索します。PG4UWMCはプログラマー(サイト)が見つからない場合は"Not found"の表示と共にDEMOモードに設定されます。1つのコンピューターで8つの同じタイプのプログラマーをサイトとして同時に実行することが出来ます。

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

**Job Report** は次の情報を含んでいます:

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

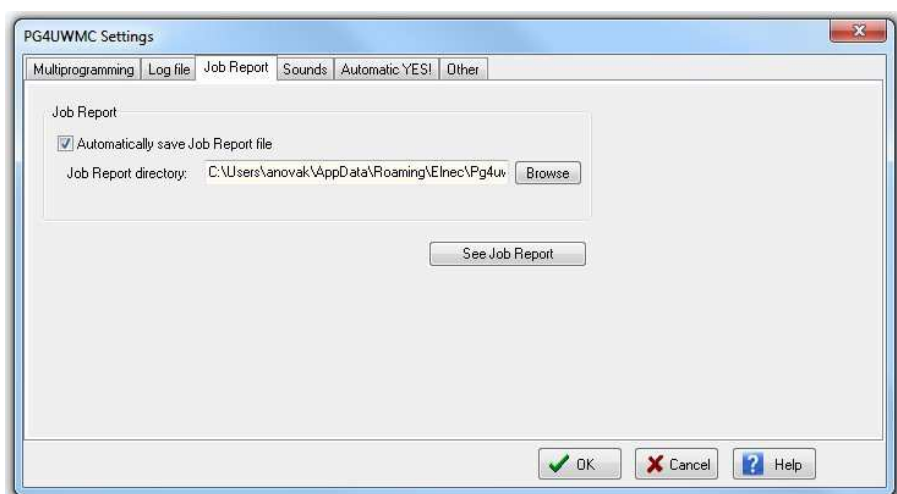
Job Reportは次の場合に生成されます:

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

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

Job Reportダイアログの設定はタブJob ReportのダイアログPG4UWMC Settings (メニューOptions / Settings)です。

次のオプションがJob Reportで利用出来ます:



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

*job\_report\_<ordnum>\_<prjname>.jrp*

<ordnum> はファイルの 10 進数の順序です。もし、同じ名前の既存のレポート・ファイルが存在する場合、新しいレポート・ファイルの順序は既存のファイルにインクリメントされます。

<prjname> は最近使用したプロジェクトのプロジェクト・ファイル名で、プロジェクト・ファイル名の拡張子はありません。

**例 1:** プロジェクト・ファイル c:\myproject.eprj を使用し、ジョブ・レポートのためのディレクトリーが d:\job\_reports\ にセットされている場合。

ジョブ・レポート・ディレクトリーにレポート・ファイルが無い場合、最後のジョブ・レポートのファイル名は:

d:\job\_reports\job\_report\_000\_myproject.jrp

**例 2:** Example 1 からの条件を使用し、しかし、1 つのレポート・ファイルが既にありと仮定します。

このファイルの名前は d:\job\_reports\job\_report\_000\_myproject.jrp

最終的に新しいレポートのジョブ・レポート・ファイル名は:

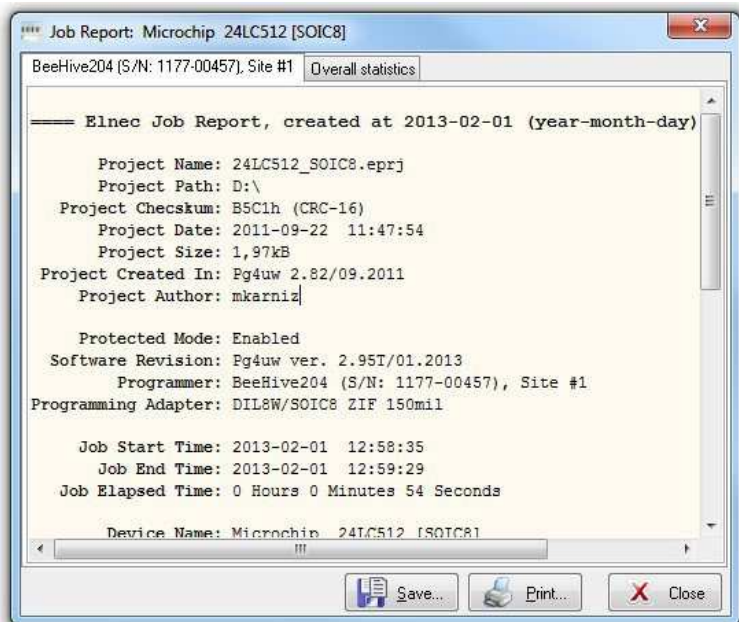
d:\job\_reports\job\_report\_001\_myproject.jrp

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

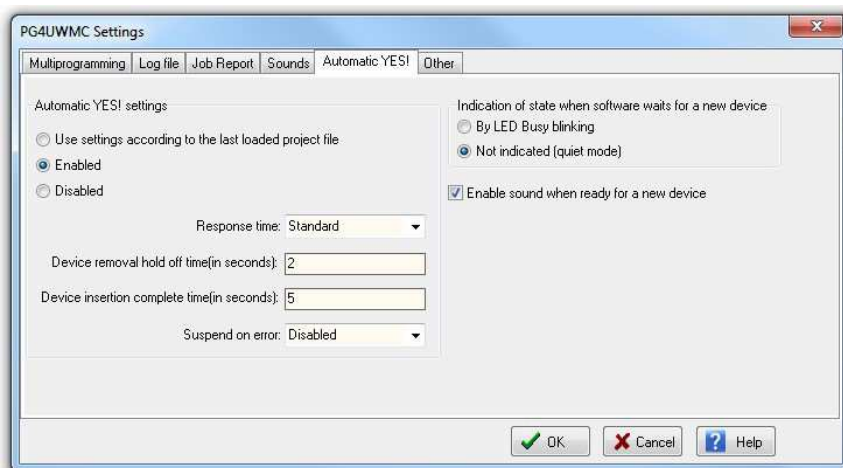
Automatically save Job Report file[自動的にジョブ・レポート・ファイルをセーブ]セッティングに設定されている時、ジョブ・レポートを生成する時に Job Report ダイアログは表示されません。新しく生成されたジョブ・レポートはダイアログやメッセージ無しでセーブされます(ファイルのセーブ中にエラーが起こらない場合)。

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

Job Reportダイアログでユーザーはジョブ・レポートで行う操作を選択することができます。もし、何も選択しない場合(ボタンを閉じる)、ジョブ・レポートは PG4UWログ・ウィンドウにのみ書かれます。:



## Automatic YES! セッティング



このモードではプログラムされたデバイスを取り除いて新しいデバイスをZIF ソケットに装着しますと最後の操作が自動的にリポートされます。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。リ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ーでは利用できません。

**Use settings according to the last loaded project file[最後にロードされたプロジェクトによるセッティングを使用]** – Automatic YES!オプションはプロジェクト・ファイルのセッティングによって設定されます。Automatic YES!の設定の項目の1つは使用されるプログラミング・アダプターに依存する'Pins of programmer's ZIF excluded from sensing'です。これは別のプログラマーが同じデバイスで異なるプログラミング・アダプターを使用することが可能であるため、この設定はこの場合無視されログ・ウィンドウには次のメッセージを見つけることができます:

"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付られません。Automatic YESウィザードを再度使用して下さい]。もし、これがマスター・プログラミング・サイト(オプション'Use Site #1 project for all Sites'でPG4UWMCを実行している場合)、又は、前述のメッセージがログに書かれたプログラミング・サイトで起こった場合はProgrammer/Automatic YES! [プログラマー/Automatic YES!]オプションでボタン'Setting Automatic YES! parameters'をクリックして下さい。'Indication of state when software waits for a new device'と'Enable sound when ready for a new device'のセッティングはプロジェクト・ファイルにストアされません。

**Enabled** - Automatic YES! 機能はPG4UWMCによりパラメータ・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い様にするために最初のピン(複数)が検知された後に全てのピンが適切に挿入されなければならない時間をセットすることが可能です。この時間は秒間隔で1~120 (デフォルト値は2 秒)でなければいけません。

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

ソフトウェアが新しいデバイスを待つ時の状態表示:

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



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

**Enable sound when ready for a new device**[新しいデバイスの用意がされた時にサウンドを有効にする] -

チェックされている時、もし、SWが完全な空のZIFソケットを検出し、そして、ZIFソケットに新しいデバイスを受け入れる準備ができている場合には音が発せられます。

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

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

**Other**

プログラマーの動作結果のLEDの色:

•標準カラー・スキーム (ERROR=red, BUSY=yellow)

•旧タイプ・カラー・スキーム (ERROR=yellow, BUSY=red)

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

**Timer refresh rate**[タイマー・リフレッシュ・レート] PG4UWMCプログラムが実行しているプログラマー・サイトからステータス情報を要求する頻度を定義します。ステータス情報には現在のデバイス操作タイプ、進行、結果等を意味します。現在のステータス情報がPG4UWMCのメイン・ウィンドウに表示されます。デフォルトのタイマーのリフレッシュ・レート値は200msです。もし、PG4UWMCの操作パネルに表示されるステータス情報の表示をより早くリフレッシュしたい場合、短いリフレッシュ間隔を選択して下さい。より速いリフレッシュを使用している時に、もし、システムのパフォーマンスのスローダウンに気付いた場合、リフレッシュ頻度が少なくするためにより高いリフレッシュ値を選択します。Pentium 4コンピューターではタイマ

ーのリフレッシュ・レートに依存するパフォーマンスの低下は殆どありませんが、遅いコンピュータでは時々長い(より少ない)タイマー間隔を選択することは有用です。

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

#### ローカル・コンピュータ上をサーチ

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

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

