



COPYRIGHT & LEGAL DISCLAIMER

11-Jul-05

BP Microsystems, LP. BPWin software version 4.52.0 (this number is subject to change as new software updates are available).

©2005 by:

BP Microsystems, LP.

1000 N. Post Oak Blvd.

Suite 225

Houston, TX 77055-7237

PHONE: 713-688-4600, 800-225-2102

FAX: 713-688-0920

www.bpmicro.com

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transcribed, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of BP Microsystems, LP.

BP-1000™, BP-2000™, BP-3000™, BP-4000™, BPWin and Concurrent Programming System® are registered trademarks of BP Microsystems, LP.

IBM and PS/2 are registered trademarks of International Business Machines.

MSDOS, Windows 98®, 2000®, and XP® are trademarks of the Microsoft Corporation.

The information in this manual is subject to change without notice and, except in the warranty, does not represent a commitment on the part of BP Microsystems, LP. BP Microsystems, LP. believes the information in this manual to be correct at the time of publication; however, BP Microsystems, LP. cannot be held liable for any mistakes in this manual and reserves the right to make changes to the product in order to make improvements.

Any mention of third-party products is for reference only, and does not constitute a recommendation or endorsement of these products.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION.....	1-1
BP MICROSYSTEMS' PROGRAMMERS	1-1
ABOUT THIS MANUAL.....	1-2
CHAPTER 2 GETTING STARTED	2-4
COMPATIBILITY	2-4
SOFTWARE INSTALLATION	2-4
HARDWARE INSTALLATION	2-11
1000 and 2000 Series.....	2-11
3000, 4000 and 6000 Series.....	2-12
INITIAL SOFTWARE STARTUP	2-16
RUNNING THE SELF-TEST.....	2-18
CHAPTER 3 PROGRAMMING WITH BPWIN.....	3-1
SELECTING A CHIP	3-1
LOADING A DATA PATTERN.....	3-4
To load from directory (network or local).....	3-4
To load from another chip	3-6
PLACING A CHIP IN THE SOCKET	3-7
SETTING UP THE HANDLER	3-8
TEACHING LOCATIONS	3-11
RUNNING A JOB.....	3-11
STOPPING A JOB	3-11
FINISHING A JOB.....	3-12
CHAPTER 4 COMMAND REFERENCE	4-1
BPWIN WINDOW	4-1
Pull-down menus	4-1
Action Buttons.....	4-17
Main Screen File Tabs.....	4-27
HANDLER WINDOW	4-28
Auto Handler Settings Tab.....	4-28
Services Tab.....	4-30
KEYBOARD USAGE AND HOT KEYS	4-32
CHAPTER 5 USING JOBMASER	5-1
OVERVIEW	5-1
FEATURES	5-1
.bp Files and .abp Files	5-1
Simple and Secure.....	5-2
INSTALLATION	5-4
CREATING A NEW JOB	5-4
COPYING AN EXISTING JOB.....	5-4
LOCKING THE PROGRAMMER IN OPERATOR MODE	5-5
Password Protection.....	5-5
RUNNING A JOB (OPERATOR MODE).....	5-6

Table of Contents

RETURNING TO NORMAL MODE.....	5-6
CHAPTER 6 BPWIN SERIALIZATION	6-1
OPERATION.....	6-1
<i>Algorithm Protocols</i>	6-5
<i>Command-line Parameters</i>	6-5
OPTIONAL CODES:	6-7
EXAMPLE PROGRAM	6-10
CHAPTER 7 TROUBLESHOOTING	7-1
HOW TO REACH US.....	7-1
<i>Customer Service</i>	7-1
<i>Sales Support</i>	7-1
<i>Calling Customer Service</i>	7-1
<i>Testing the Hardware</i>	7-2
SOFTWARE UPDATES	7-2
FAQ's	7-3
ERRORS WHILE PROGRAMMING	7-3
CLEANING A DIRTY DIP SOCKET.....	7-4
ERROR MESSAGES	7-4
WARNING MESSAGES.....	7-10
GLOSSARY	1
DEFINITIONS	1
ACRONYMS.....	21
INDEX	31

CHAPTER 1

INTRODUCTION

BP MICROSYSTEMS' PROGRAMMERS

Congratulations on selecting a BP Microsystems device programmer designed to perform the most demanding programming tasks. BP Microsystems programmers are used for design engineering, production programming, device testing, and field service. You can continue to program the newest devices with your programmer's flexible hardware by obtaining software updates from BP Microsystems. BP Microsystems stands behind the programmers it manufactures to ensure your continued satisfaction.

BP Microsystems is committed to providing the worldwide market with high performance innovative device programmers and great customer support at the lowest price possible. Our programming platforms span the spectrum from low-cost single-family programmers to fully universal and automated programmers that support virtually every device available today. Every programmer designed by BP Microsystems is built to the highest standards. Each programmer offers a wide range of device support, flexible pin drivers and protection from damage caused by operator errors and defective devices. It has stable hardware that does not require periodic maintenance or recalibration, and software that can be easily updated in the field to support the newest programmable devices. The programmers that support memory devices will accept all popular file formats, use intelligent identifiers to auto-select proper programming algorithms, provide selectable range programming on many chips, and have a special set programming mode for automatic collation of data. The programmers that support PLDs accept JEDEC (Joint Electronic Device Engineering Council) files with editing ability for both fuse data and test vectors. The PLD programmers also support Altera's compressed Programmer Object File (POF) format used with their MAX family of devices.

All of our programmers attach easily to a personal computer (PC) for ease of use. The menu-driven Windows software provides an efficient, user-friendly interface that allows you to quickly maneuver through the device list, select files from any location (including data read from another chip), utilize a full screen editor to view and modify your data, and virtual memory management to deal with very large files.

The programmers easily connect to the parallel printer port or USB 2.0 port of any PC depending on programmer model. The BPWin software operates on the Windows 98 SE, Windows 2000, and Windows XP platforms.

ABOUT THIS MANUAL

This manual has been written to support any of the following products using the BPWin software.

BP-1148	BP-2000	BP-2600M	BP-3100	BP-4100
BP-1200	BP-2100	BP-2700	BP-3500	BP-4500
BP-1400	BP-2200	BP-2700M	BP-3510	BP4510
BP-1600	BP-2500	BP-2710	BP-3600	BP4600
BP-1700	BP-2510	BP-2710M	BP-3700MK2	BP-4700
BP-1710	BP-2600		BP-3710MK2	BP-4710

All of BP Microsystems device programmers are easy to operate. You may want to skip most of the manual if you've used a BP Microsystems programmer before. Either way, you should sit down at your computer and read the next chapter, *Getting Started*. It will help you install the programmer and make sure it is correctly connected. The rest of the manual is primarily for reference.

- Chapter 1 is an overview of BP Microsystems programmers.
- Chapter 2 is a getting started section for programming devices using the BPWin software.
- Chapter 3 details the entire programming process, from choosing a device and loading the buffer to running and monitoring the job.
- Chapter 4 is organized by function and command – listing details and usage for each.
- Chapter 5 covers the functionality of JobMaster.
- Chapter 6 is an overview of the Serialization utility under the Windows platform.
- Chapter 7 explains some error messages and helps you identify problems and mistakes.

Certain character formats have been used to allow us to assist you, the reader, with messages and instructions. Below is a list of the codes that pertain to this manual.

- Warnings are set off by a lightning bolt (⚡) icon and indented, **bolded** text.
- Notes are set off by a notepad (📝) icon and indented, *italicized* text.
- Examples are set off by the bolded word “**Example**” and indented, *italicized* text.

Action words within the manual are set off as listed below:

- **<Bold>** – actual keystrokes on the keyboard, i.e. press **<Esc>**
- **Bold** – important text found within a paragraph or statement and options found within the software, i.e. **YES**.
- ***Bold Italics*** – actual phrases, keywords and buttons found within the software, i.e. ***Device Selector*** , ***Select*** button.

- *Italics* – software fields and drop-down menus, chapter announcements, i.e. *Operator Mode*, *Chapter 5* or *JobMaster/Configure*.
- ☐ Exceptions: if an action/function item is listed in a note or example, the text will be bolded but not italicized. If the item is found in a warning, it will be italicized but not bolded. If a software field or window title is found in a note or example, the title will be regular text (without format). If the title is found in a warning, it will be italicized but not bolded.
- ☐ If you are viewing this manual in Adobe PDF format, you may choose to view the bookmarks that have been set up to assist in usability.

CHAPTER 2

GETTING STARTED

If you are using a BP Microsystems programmer for the first time, this section will help you install the programmer's software on your computer and make sure it is properly connected. It also contains important operating information that you should know before programming chips. Detailed information about each command referenced herein appears in *Chapter 4, Command Reference*.

COMPATIBILITY

BP Microsystems programmers are designed to operate with any PC running Microsoft Windows 98 SE, 2000, or XP®.

For programmer models that support parallel port, the computer needs a parallel printer port (LPT1, LPT2, or LPT3), at least 64MB of available memory, a CD-ROM disk drive, 50MB of available hard disk space (or networked file server), VGA or higher resolution video adapter (SVGA 256-color and 800x600 or higher resolution recommended), and Microsoft Mouse or compatible pointing device. Timing is not derived from the computer, so any speed CPU will work correctly with the software.

For programmer models that support USB 2.0, the computer should have at least one available USB 2.0 port, System RAM that is twice the size of the largest device to be programmed but at least 64MB of available memory, either Pentium III or faster CPU, SVGA 256-color and 800x600 or higher resolution video monitor, and Microsoft Mouse or compatible pointing device.

The software is not copy protected, so you can make backup copies and install it on your hard disk. However, the software is copyrighted; therefore, you may not copy and distribute it.

SOFTWARE INSTALLATION

To install the software from the Internet download:

Go to www.bpmicro.com website. Select the Software download page from the uppermost selection bar. For new users, you will need to register for a username and password for software downloads.

Select the BPWin button when ready to download.

If you have questions, scroll down the page using the right-hand scroll bar. You will also find duplicated download instructions further down the page.

When prompted, choose the drive and directory the self-extracting .EXE file is to be saved in.

 *This software can be installed over a network onto a user's local PC.*

Once the download is complete, double-click on the filename to install the software.

To install the software from CD ROM:

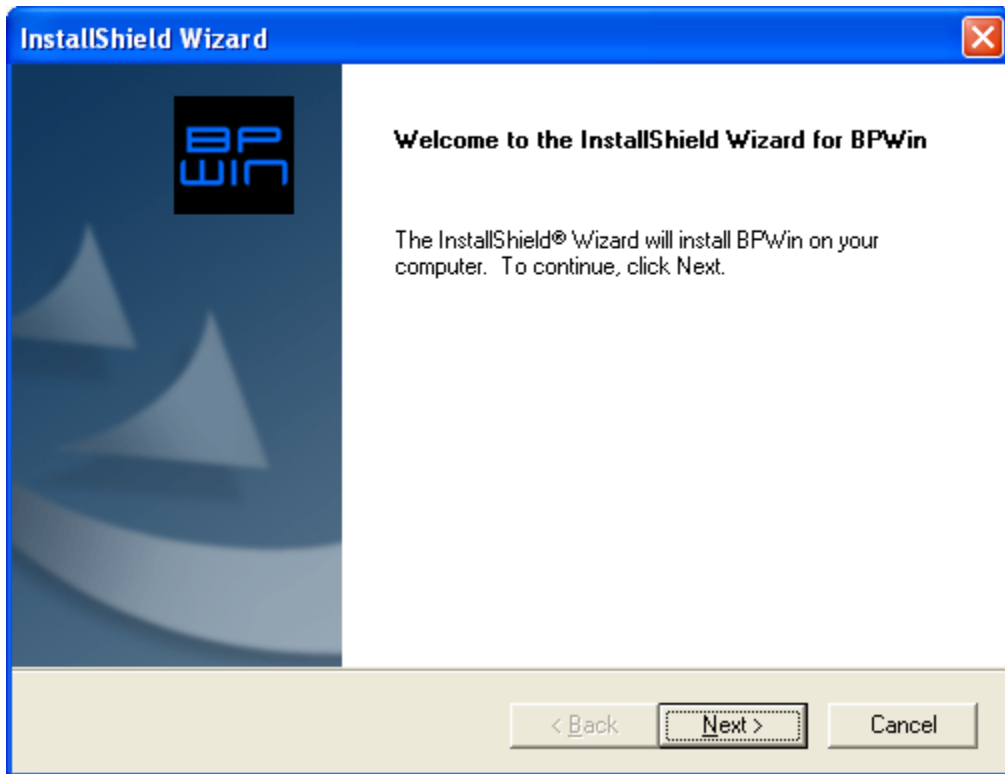
Place the CD in the CD ROM drive.

If auto-launch has been selected, the installation will automatically run a step-by-step procedure to install the software.

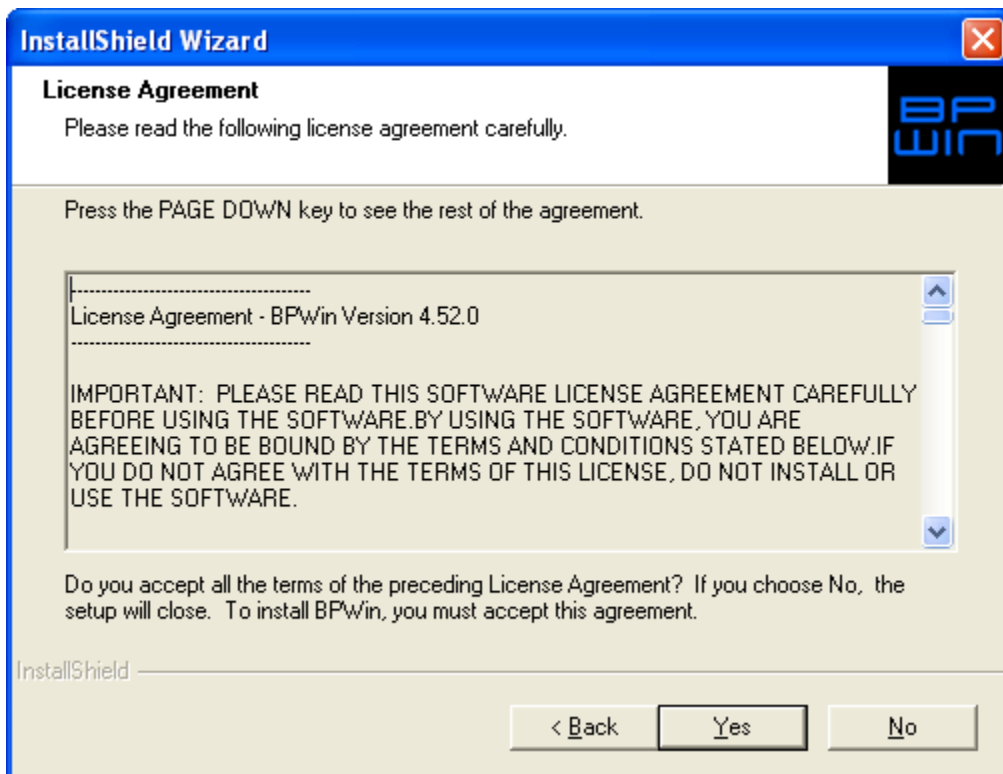
If a manual setup is preferred, choose SETUP.EXE from the list of files located on the CD. The SETUP program will launch the installation procedure.



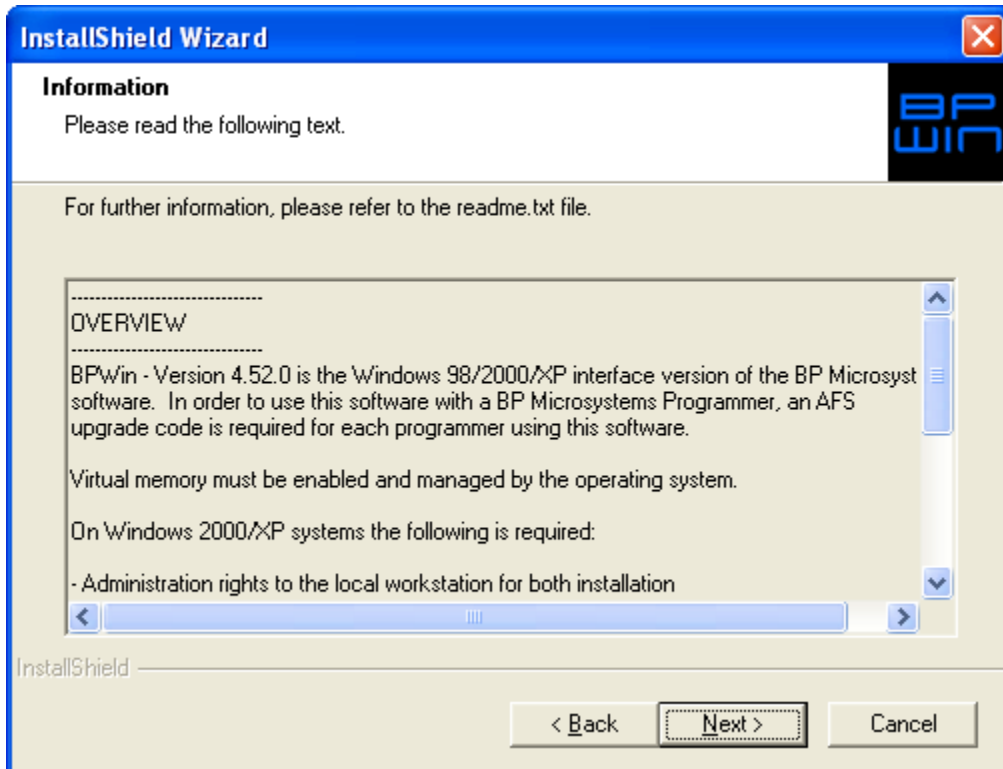
Once activated, the InstallShield Wizard for the software will be displayed. Follow the on-screen instructions for installing BPWin.



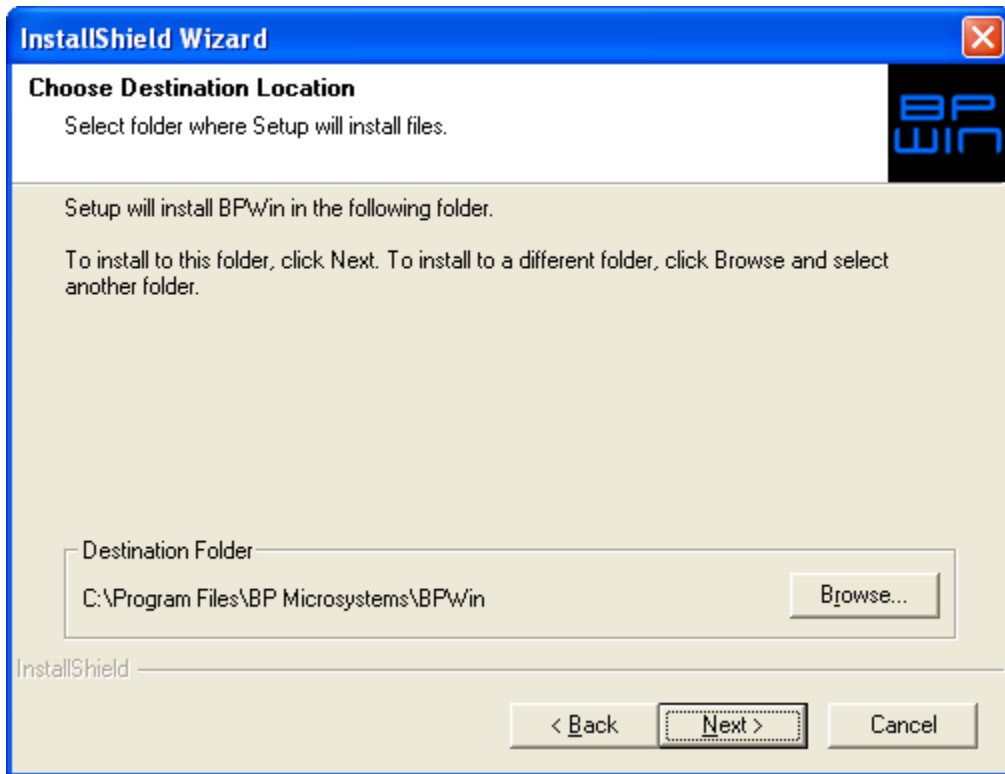
Read the license agreement carefully and decide if you wish to proceed.



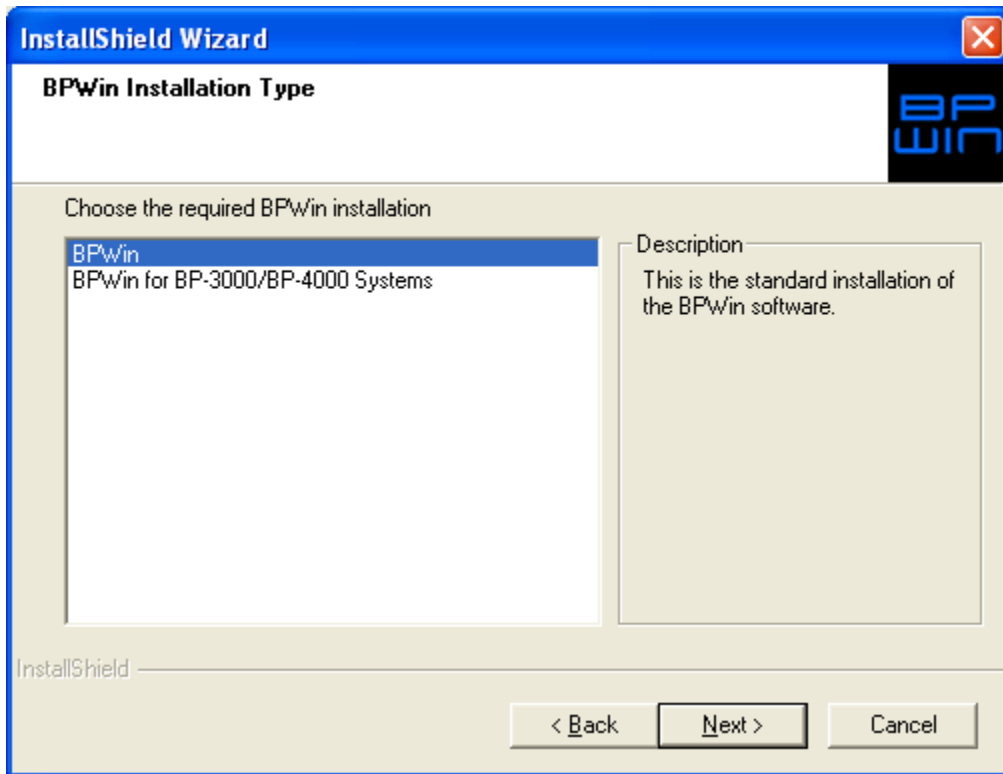
Read the information overview for computer requirements and click Next.



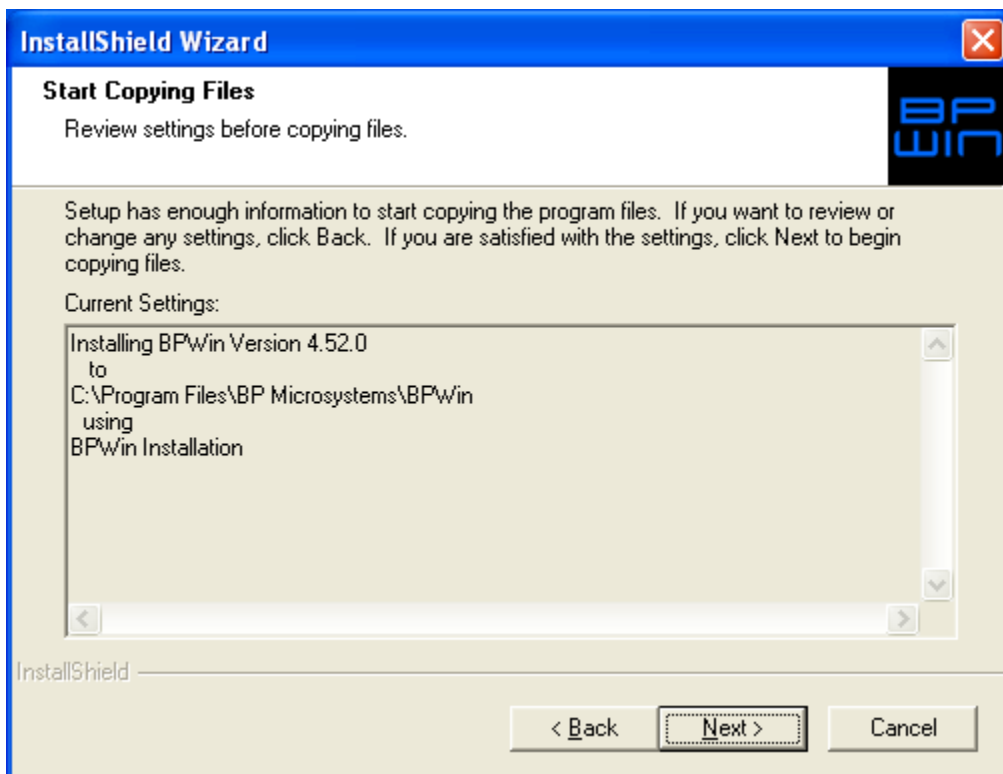
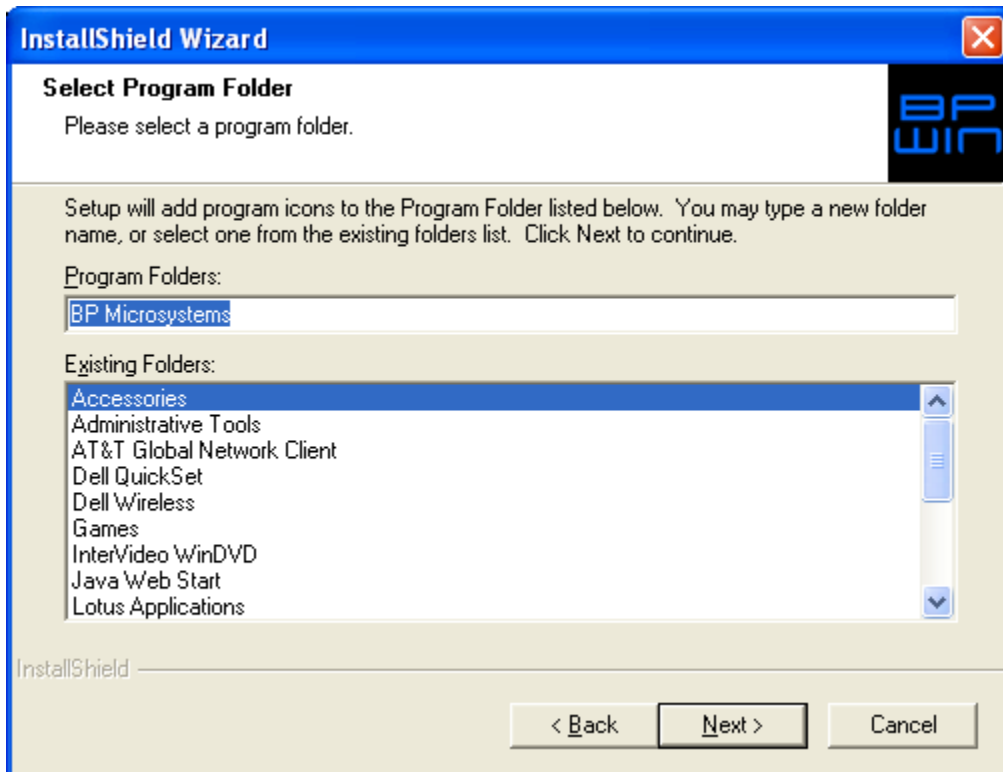
The installation will prompt you to choose a folder name and a location for the BPWin program files. The default has been provided for you, but should you choose to place these files in an alternate location, simply click in the field provided and type the location address or click the **Browse** button to find an alternative. Click Next when done.



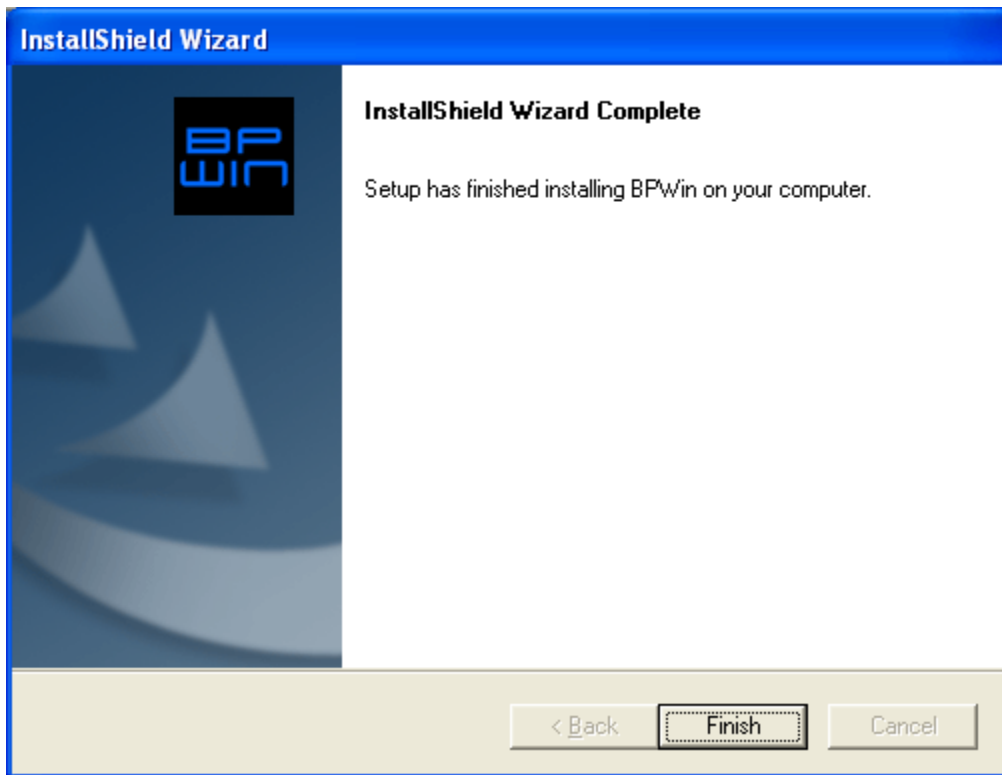
When prompted for installation type, you can choose standard installation, or if you are using an autohandler you can choose BPWin for 3000/4000 Series Systems and click Next.



Select program folder and click Next.



Follow the remaining onscreen instructions to complete the software installation and click Finish when done.



 *If you have a previous version of the software installed on your computer, the installation procedure will also prompt you to uninstall the older version before continuing the new installation procedures. We recommend that the BP Microsystems software should be removed using the Windows Add or Remove programs utility located in the control panel.*

HARDWARE INSTALLATION

1000 AND 2000 SERIES

Connect power cord from the BP Microsystems manual programmer to the wall outlet. Then connect either the parallel printer port or USB 2.0 port cable from the programmer to the PC. It is preferable to dedicate a port to the programmer, but you may plug and unplug the cables.

For programmer models that support parallel port:

- ✎ **Use the 25-conductor cable provided with your programmer. Do not use a ribbon cable or an RS-232 cable that has fewer than 25 conductors. You can extend the cable up to 12 feet, but be certain to use only a 25-conductor shielded cable (available from BP Microsystems).**
- ✎ **Ribbon cables work well installed inside a chassis, but often make poor connections when subjected to the flexing that occurs when used improperly.**

☐ *The 1200 and newer models have a power supply that operates from 90 to 250 VAC for simplified worldwide use. Plug the programmer AC power cord into a source of power. Connect one end of the DB-25 cable provided to the programmer connectors and tighten the screws. Connect the other end of the cable to your computer's parallel printer port.*

⚡ **Do not attempt to use any printer buffers, electronic switches, or software copy protection keys on the same port as the programmer. Verify you have connected to the correct (parallel) port on your computer – connecting to a serial port or a third party card may damage the programmer and is not covered by the warranty.**

Once all components have been setup, turn on the computer and the programmer. You will see the LEDs become active. The LEDs are labeled as Pass, Active and Fail. The Pass LED should come on, the Active LED may come on for a short period but should turn off, and the Error LED might be on or off (the Error indicator is only valid while the software is running).

3000 AND 4000 SERIES

For our larger, automated device programmers, a field service engineer will come out to your facility to install the unit. Any questions, concerns or problems regarding the installation, or stemming from, should be directed to the on-site service engineer or Customer Service (contact information listed in *Chapter 7, Troubleshooting*).

USB 2.0 INITIAL SETUP

USB based programmer models are officially supported in the Windows 2000 and Windows XP operating systems. The 1710 and 2710 products were tested in the Windows 98 SE OS but support is not guaranteed. Users can operate the programmer in the Windows 98 SE environment at their own risk.

The USB 2.0 drivers are automatically installed during the BPWin software installation in Windows 98 SE. If users experience communications problems, check to see if the USB 2.0 drivers are installed using the Device Manager setup. If the USB 2.0 drivers were not installed correctly, then the user must reinstall the BPWin software.

☐ *If the communication problems still persist in Windows 98, then we encourage all users to upgrade their PC to Windows 2000 or Windows XP.*

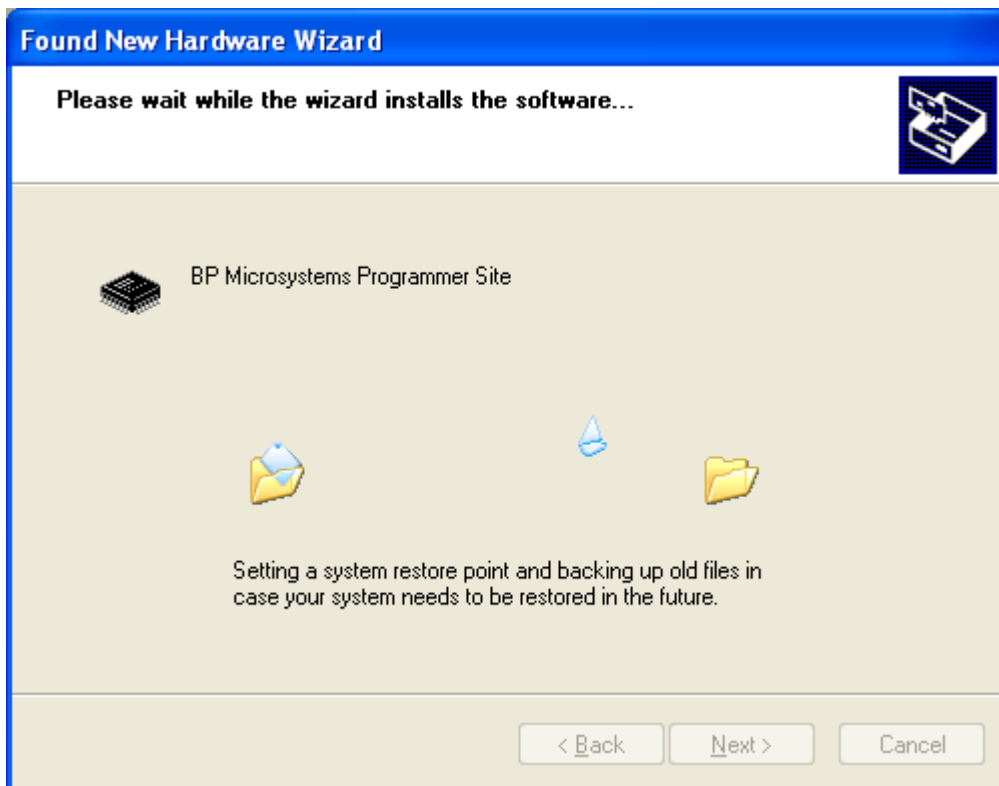
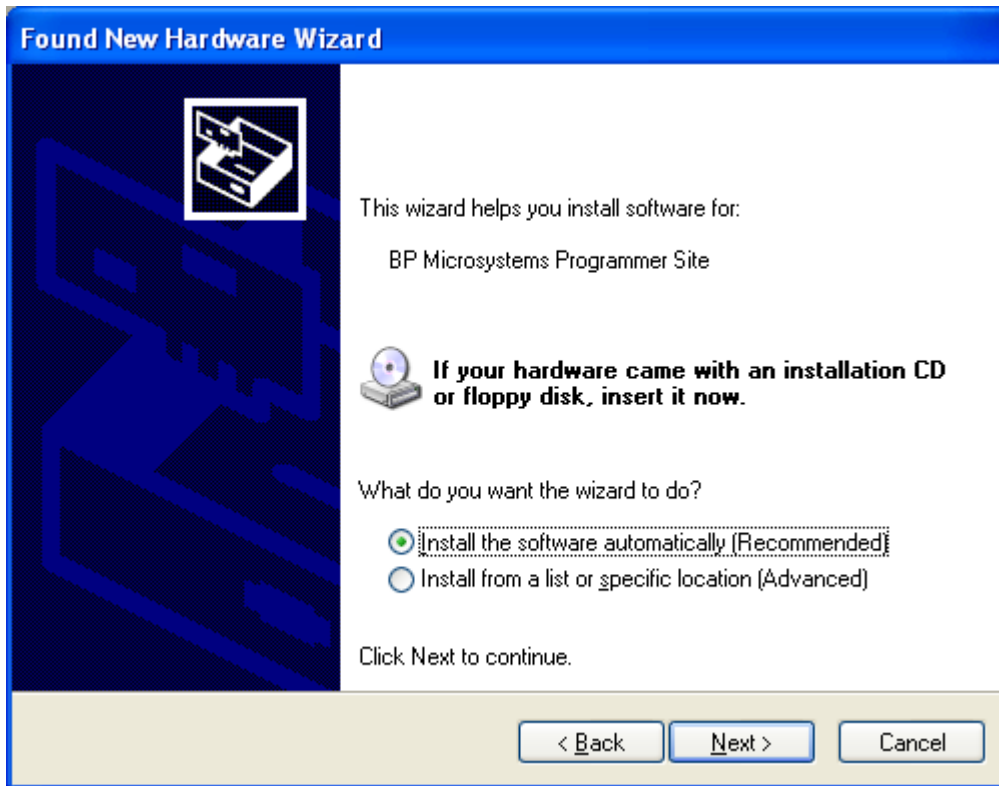
After turning the programmer on, the Windows software will recognize that new hardware is found and will prompt you to install the drivers in Windows XP. Follow the on screen installation instructions provided by the Windows OS Wizard.

The user must install the necessary USB 2.0 software drivers to ensure communication with the programmer.

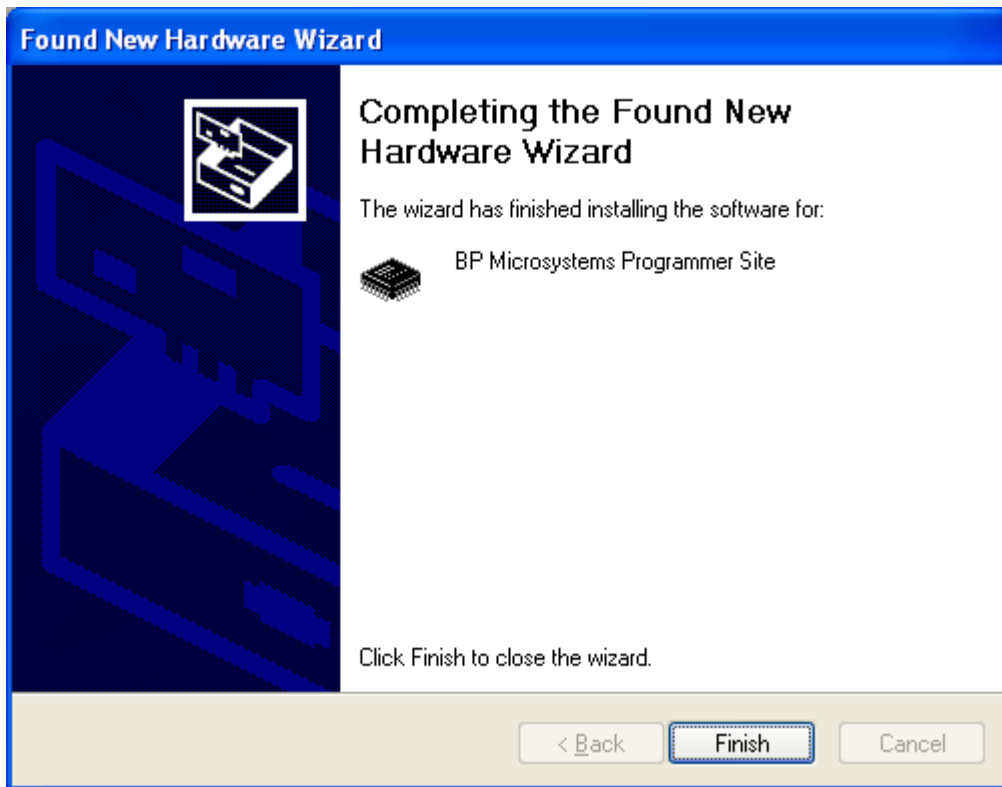
For Windows XP users, select “No, not at this time” and Click Next to continue



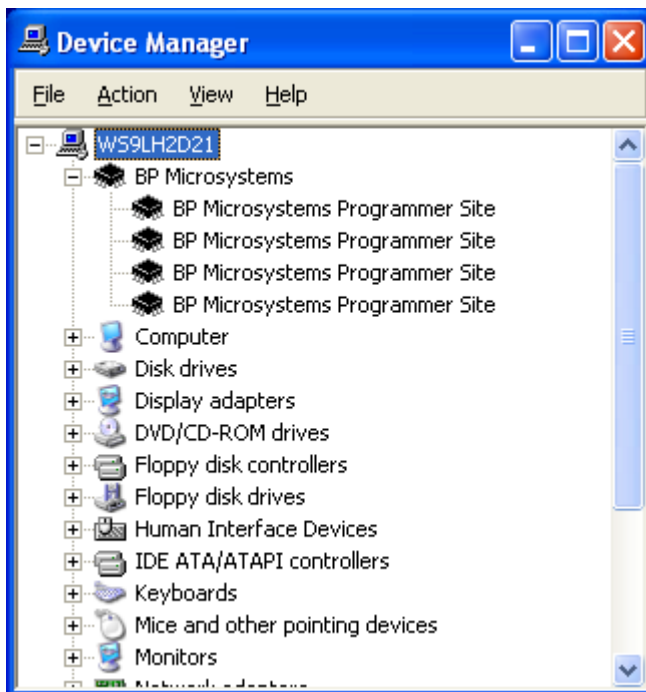
Select the “Install the software automatically” option and click Next to continue



Click Finish to complete installation



- ☐ There should be a **BP Microsystems Programmer Site** for each programmer site attached and turned on. Verify using the Windows Device Manager



INITIAL SOFTWARE STARTUP

BPWin utilizes all the standard functions of a windows-based format including pull-down (drop-down) menus, action buttons, and hot keys along with on-line Help files. These functions are described, in depth, in the following *Command Reference* chapter.

To run the software, select the BPWin shortcut in the Windows Start Menu/Programs list.

As the BP Microsystems software is launched the following screen is displayed. The software is establishing communications with the programmer.



Figure 1 – Launch software

The initial screen of the BPWin software is divided into two sections as shown below. The uppermost section, the *Main Program* window, contains the software announcement including version number, as well as the pull-down functionality menus, buttons for opening and saving files and JobMaster, and the tracking system for the amount of devices to be programmed (*Quantity*).

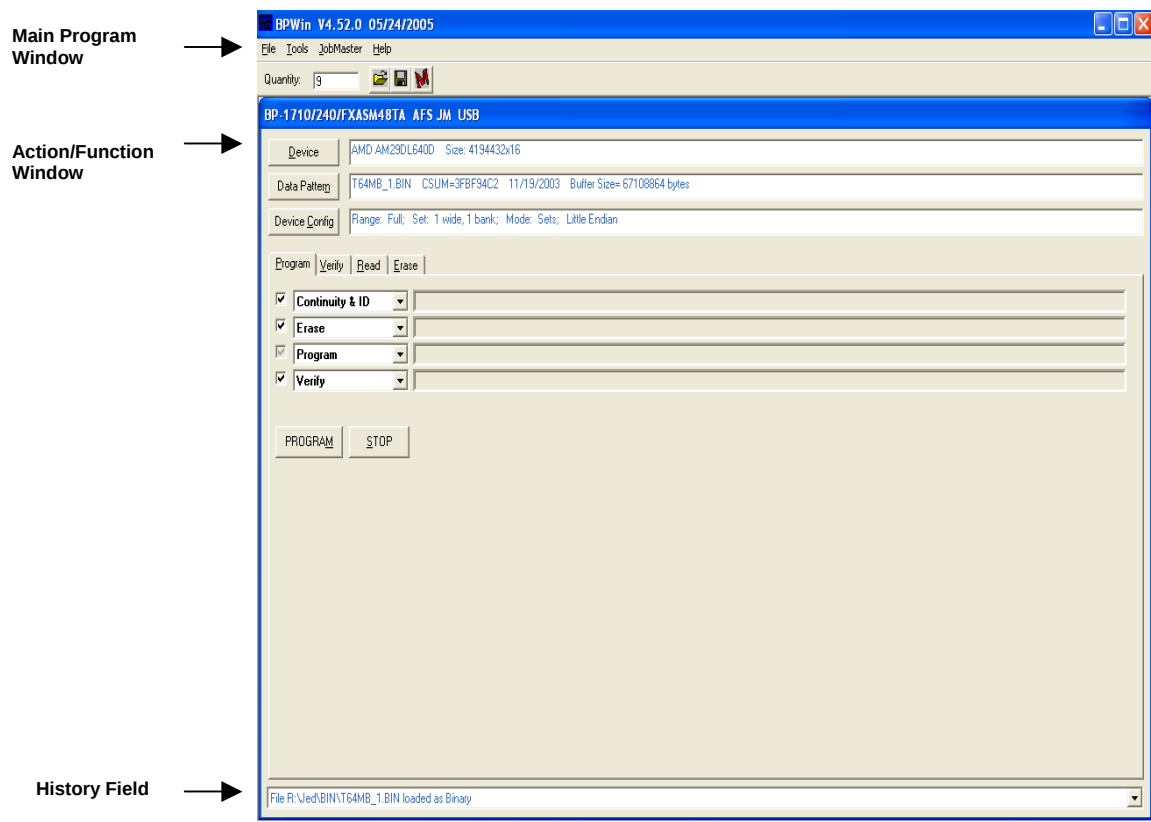


Figure 2 – BPWin Main Screen

The secondary, floating window (known as the *Action/Function* window) is where the user can designate devices and data patterns as well as start and stop programming actions which are covered in *Chapter 4, Command Reference*. The title bar of the *Action/Function* window includes the following information:

- Programmer detected or selected.
If a programmer has not been detected, the software will automatically start up in DEMO mode, allowing the user all functionality possible without a device programmer. A programmer can be added after startup in DEMO mode through the **File/Configure pull-down menu** option. All actions/functions found in the software can be utilized within DEMO except programming a device.
- The number of pin drivers and socket module detected.
- Any Advanced Feature Software detected.
- Which communications port the programmer is detected at.

The above figure depicts a model 1710 manual device programmer, FXASM48TA socket module with AFS JobMaster software on the USB port.

The bottom pull-down menu, visible only when the program window is fully expanded, tracks the history of the current job, including buffer activity, what devices are chosen, etc. This menu will keep a running list of all activity during active use of the software. This data is stored to a log file and is reset each time the software is restarted.

RUNNING THE SELF-TEST

⚡ **There must not be a chip in the socket while performing this test.**

The BPWin software also enables the user to test all applicable parts within the device programmer for accuracy. This test helps to ensure that the programmer is running at top performance standards.

To begin running the self-test on your device programmer, select **Programmer Diagnostics** from the *Tools* pull-down menu. Once selected, the *BP Microsystems Diagnostics* window and a *Test* window will appear. This *Test* window allows the user the ability to designate certain options for the test. To begin testing the programmer, choose the options needed from the *Test* window and click **OK**.

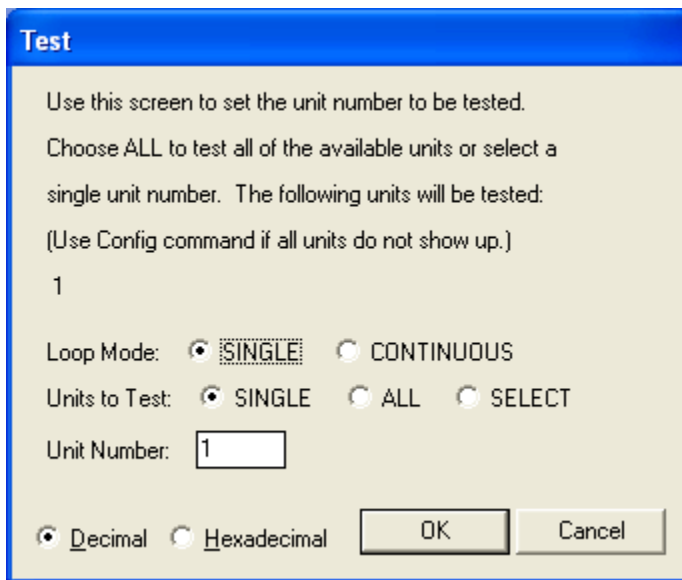


Figure 3 – Test window

The BP Diagnostics window will show the progress of the test.

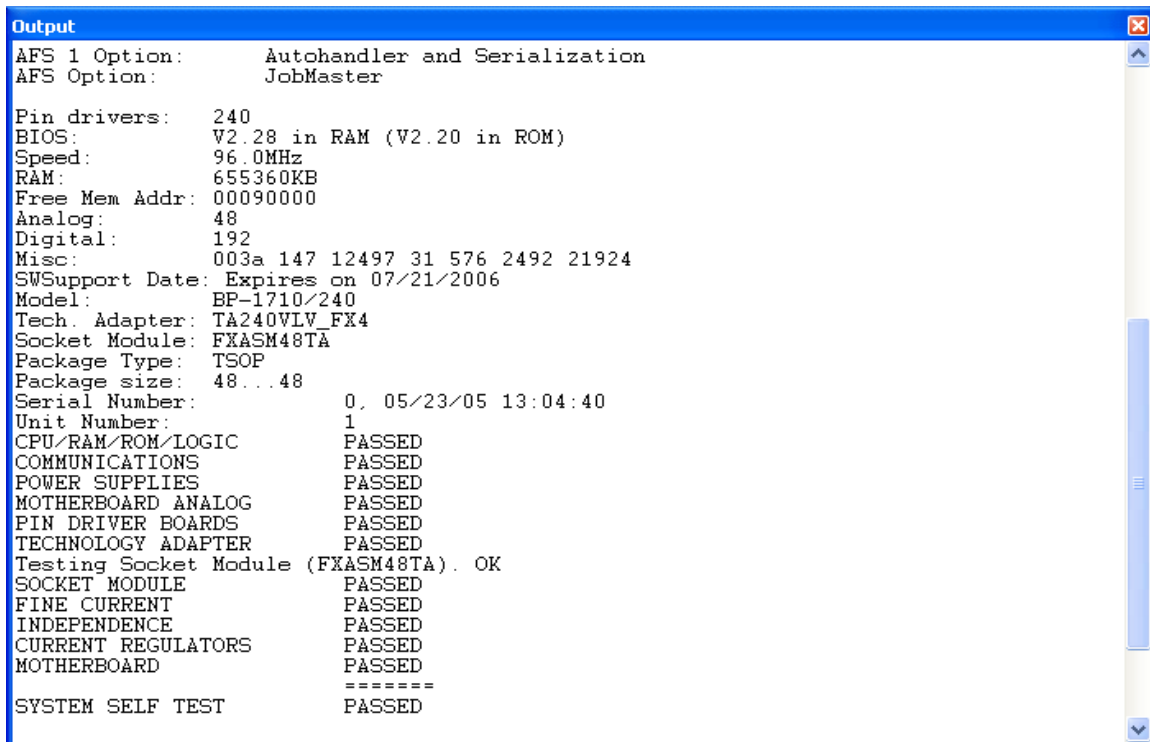


Figure 4 – BPWin main window showing self-test information

A window will appear to prompt you to verify that there is no chip in the socket and whether or not you would like to perform a verify on the calibration with a voltmeter.

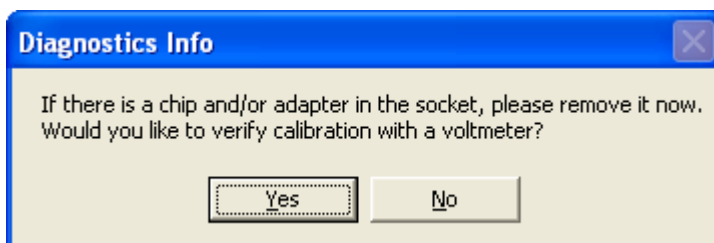


Figure 5 – Warning window for self-test

The test will begin and continue running until it is completed or until interrupted via the **Stop** button. To cancel the self-test, click on the **Stop** button at any time during the procedure. A window will appear to acknowledge the operation was aborted. To re-execute the test, simply click the **Test** button again.

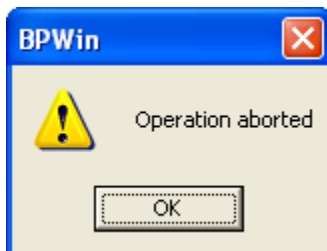


Figure 6 – Self-test abort window

Once your programmer passes the test it can be verified by checking the green PASS LED on the chassis of the programmer. If the programmer does not pass the test, the red FAIL LED will be on and an error message will be presented on-screen.

Error 47: Self-test failed. This unit may need service. Please call Customer Service.

If this should happen, double-check the fidelity of the cable connections and try again. Note the exact error message and call us for technical assistance if you are still having trouble (contact information for Customer Service is listed in [Chapter 7, Troubleshooting, page](#))

CHAPTER 3

PROGRAMMING WITH BPWin

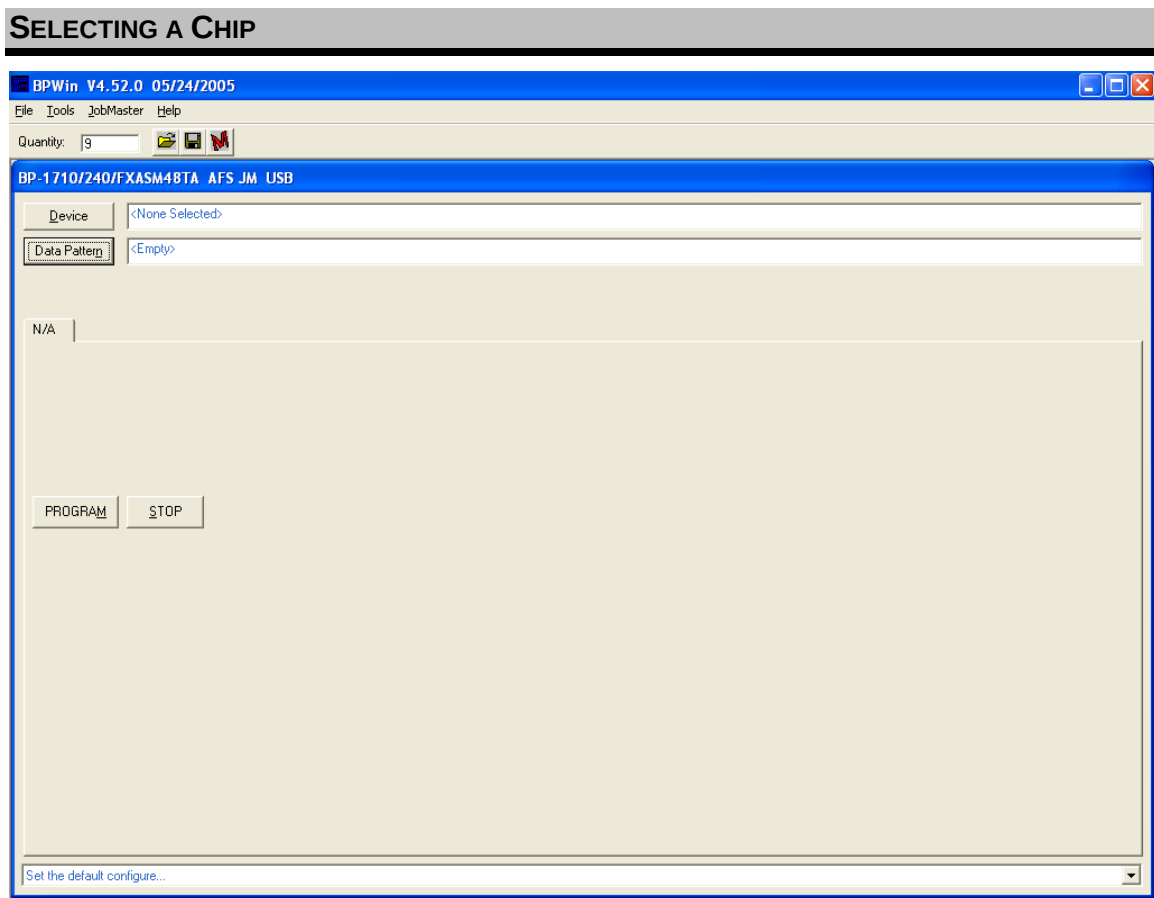


Figure 7 – Select Device window from Main window

You are now ready to begin the programming process by selecting a chip. To do so, click on the **Device** button in the *Main Screen*. The list of parts that pops up contains every device currently supported by your programmer. You may choose a device by scrolling down the available list using the right-hand scroll bar, or utilize the search engine by typing directly into the name field provided. Since there are thousands devices supported, the easiest and quickest way to locate the part you are looking for is to use this search engine.

The search engine works by associating characters typed in the field with the characters that may be found within one or more string items listed in the software's device inventory. These characters do not have to be the beginning of the string line, i.e. – the manufacturer's name.

Example: If you type in a few letters, the software will narrow the list down to any parts listed with those letters in that particular order, showing the shortest name first. As you continue to type, the list will become more defined until a particular device is chosen.

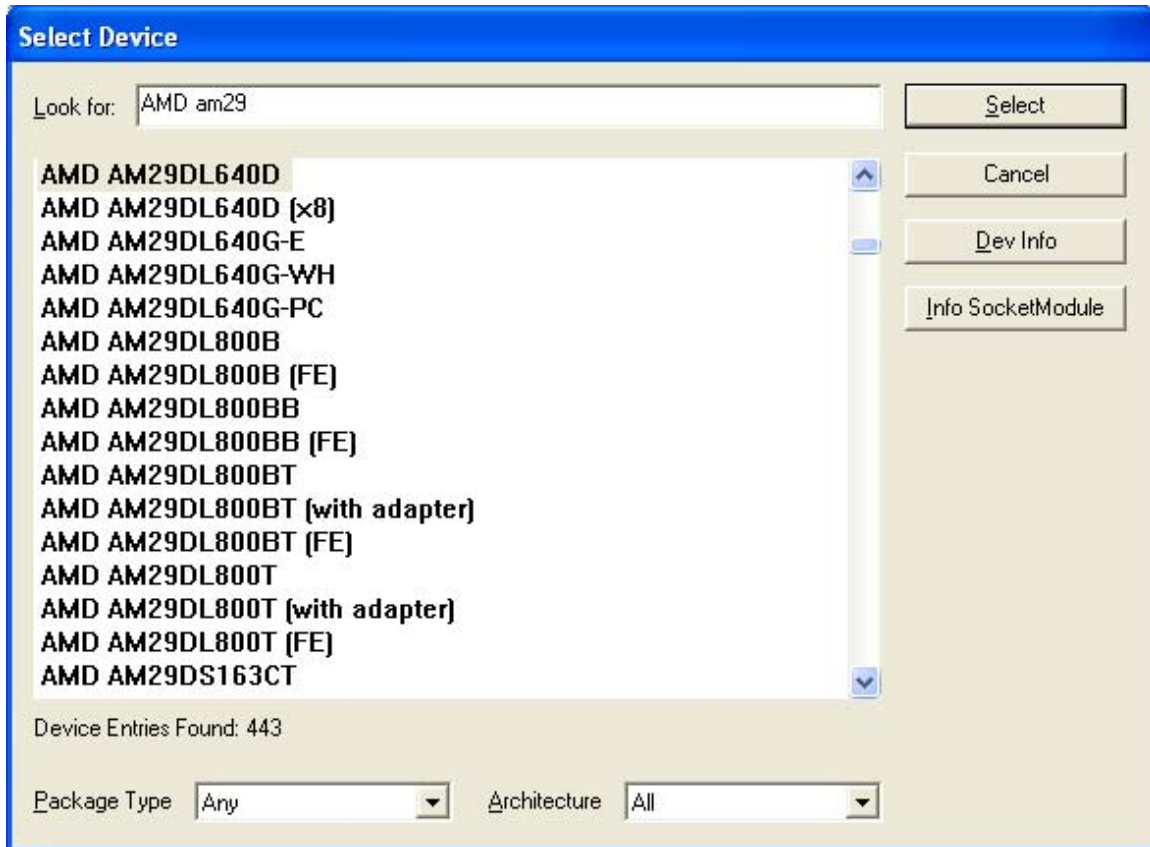


Figure 8 – BPWin Search engine function

You may also use the short-cut feature provided with the search engine. To do so, type a few letters of the manufacturer's name and a few letters of the part number separated by a comma (,).

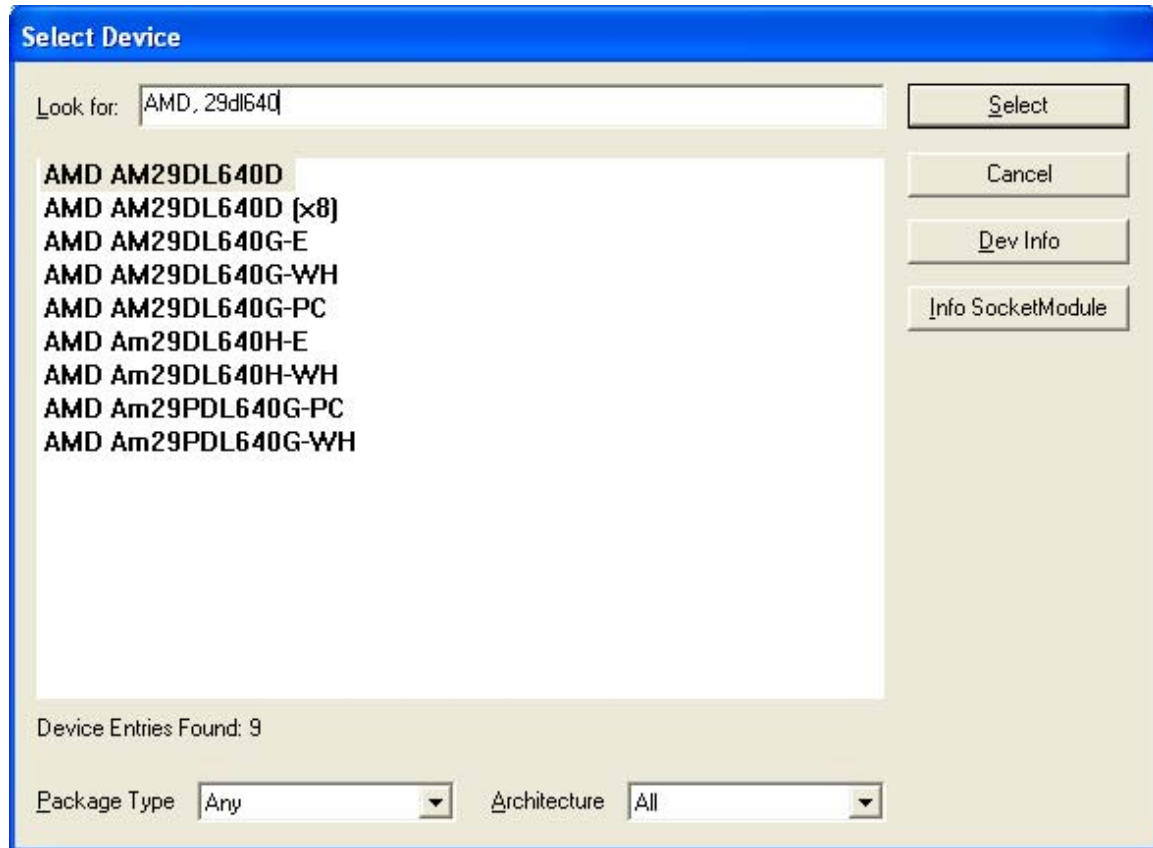


Figure 9 – Shortcut function in search engine using a comma (,)

Highlight the device by clicking on the name. To choose the highlighted device, either double-click on the specified name, or click on the **Select** button. You may want to designate architecture (to list choices available to only one particular type of architecture) or choose a package type to assist you in selecting a device. These options are listed at the bottom of the device window.

- ☐ *It is not required to choose a package before programming a chip with a different package type, however, it can be used to narrow your search. We recommend that you leave it in the “All” position, unless you are strictly using one package type.*

Device information will now be visible in the *Device* and *Device Config* fields within the main screen. Options that pertain to the specific device chosen will now be tabbed at the bottom of the active window.

- ☐ *Only those options available to the current chip will be accessible.*
- ☐ *See the **Select** command in Chapter 4, Command Reference, for more details on the device selector.*

If your device is not shown in the device list, please contact us. Your device may have been recently added, or we may be able to add it. If you would like to request support for a new device

algorithm, please visit www.bpmicro.com and complete the device request form. BP Microsystems will be in contact with you within 48 hours. Customer Service numbers are listed in *Chapter 7, Troubleshooting*.

- ✦ It is essential to choose the correct entry for the device you want to program. Programming algorithms vary widely between different semiconductor manufacturers and even between parts from one manufacturer with different speed ratings! Selecting the wrong algorithm can destroy your chip! The part number on your chip may have package code and temperature rating letters following the part number shown in the menu. When you select a chip, it is a good idea to specify both the manufacturer and the part number of the device you want to program. You must be careful to select the part number from the menu list that exactly or most closely matches your device part number, including the suffix that represents programming voltage and speed.
- ✦ Almost all programmable memory devices manufactured today support an electronic code that can be read to identify the correct programming algorithm. Most likely, your chip supports this feature and the code will be read before programming to ensure that the correct algorithm is being used. However, if you are in possession of a part that does not utilize this electronic code, selecting the correct chip from the menu is imperative.

LOADING A DATA PATTERN

Once you have chosen a device, you are ready to load a data pattern into the buffer.

TO LOAD FROM DIRECTORY (NETWORK OR LOCAL)

Click the **Data Pattern** button from the main screen. This activates the *Data Pattern* window.

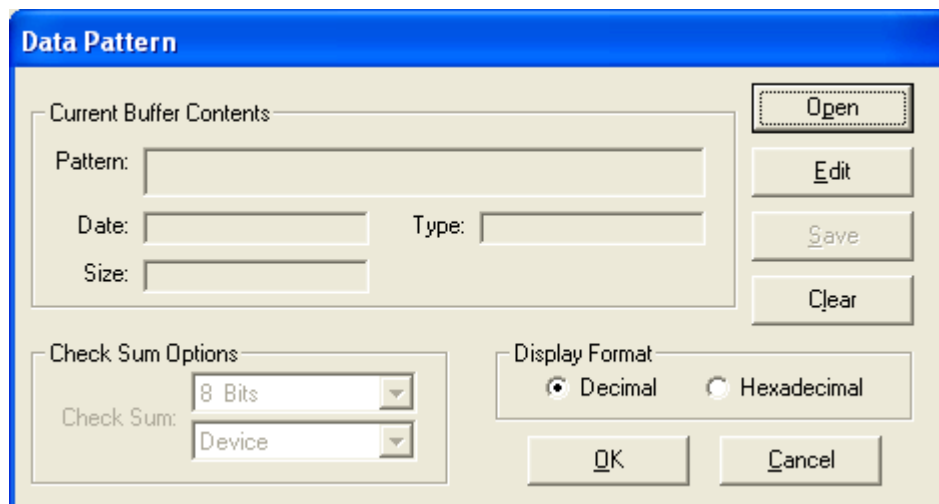


Figure 10 – Data Pattern window

If the buffer is not already loaded, no information will be reflected in the *Current Buffer Contents* group box. To choose a file to load, click the **Open** button to bring up the *Load File into Buffer* screen.

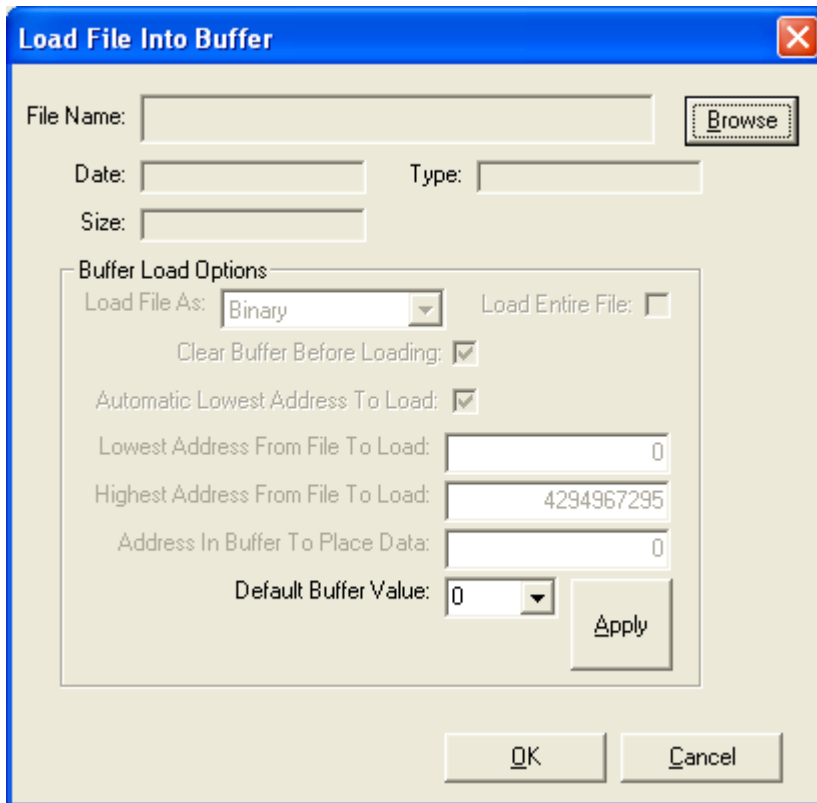


Figure 11 – Load File in Buffer window

Click on the **Browse** button to locate the desired file.

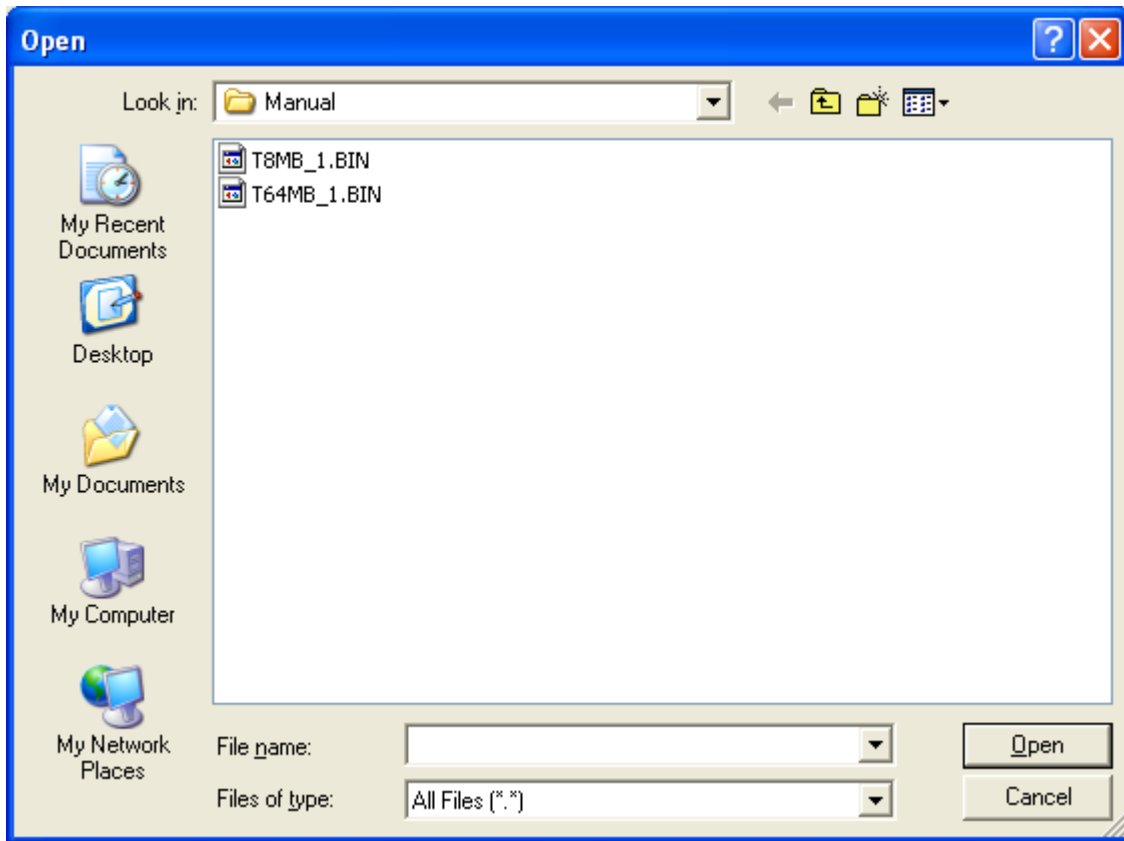


Figure 12 – File Open window

To choose one of the files listed, highlight the file and click on the **Open** button. The software will automatically identify the file type and display the information in a secondary window for your verification. Once the file information has been confirmed, click the **OK** button and the file will be loaded into the data buffer.

The fields within the *Current Buffer Contents* group box should now reflect the loaded file information. At this time, you may set the checksum options including the status windows listing the *CSUM* (checksum), *Size*, *Date* and *Type* of the file highlighted. You may also change the *Checksum* from 8 to 16 bits and designate *Device*, *Buffer* or *Compatibility* through drop-down menu options.

TO LOAD FROM ANOTHER CHIP

To load a data pattern from another chip, place the original chip on the socket module and verify the correct part number (for the device). Select the *Read* file tab then click the **Read** button to load the buffer with the data. Once a device is read, a *Job Summary* report will appear on screen.

Click **OK** to be returned to the *BPWin Main Screen*. The *Data Pattern* field should reflect the buffer contents. The data can be viewed in the buffer through the *Data Pattern* window by clicking on the **Edit** button and can be saved to file using the **Save** button.

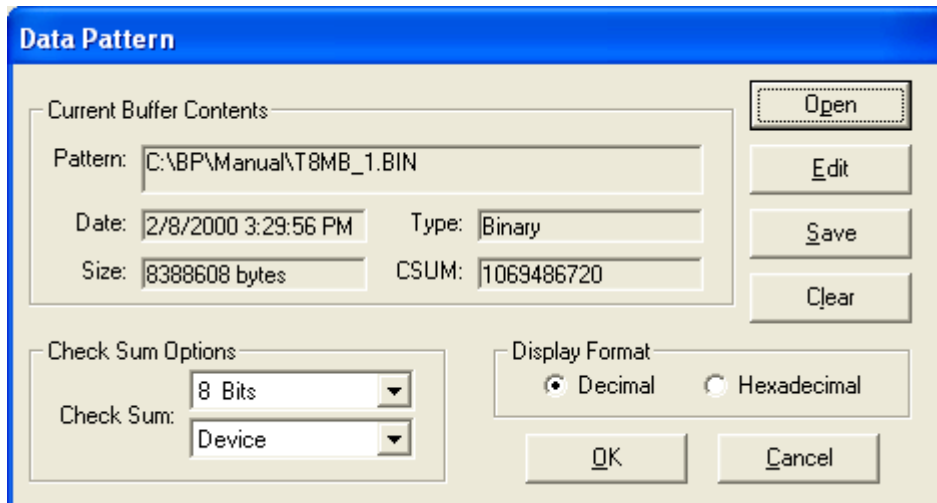


Figure 13 – Device Read Data Pattern window

PLACING A CHIP IN THE SOCKET

To place a device in the socket module, you will need to refer the appropriate device programmer manual.

For manual and concurrent programmers, place the chip in the socket in accordance with the pin placement pertaining to the device.

For automated programmers, you will need to teach the programmer where to place the chips. Refer to the appropriate automated User's Guide to teach these locations.

✍ **Parts must *not* be inserted or removed when the ACTIVE LED is on.**

SETTING UP THE HANDLER

If you are using an automated programmer, the next step in preparing a new job is to set up the Handler through the Handler's software.

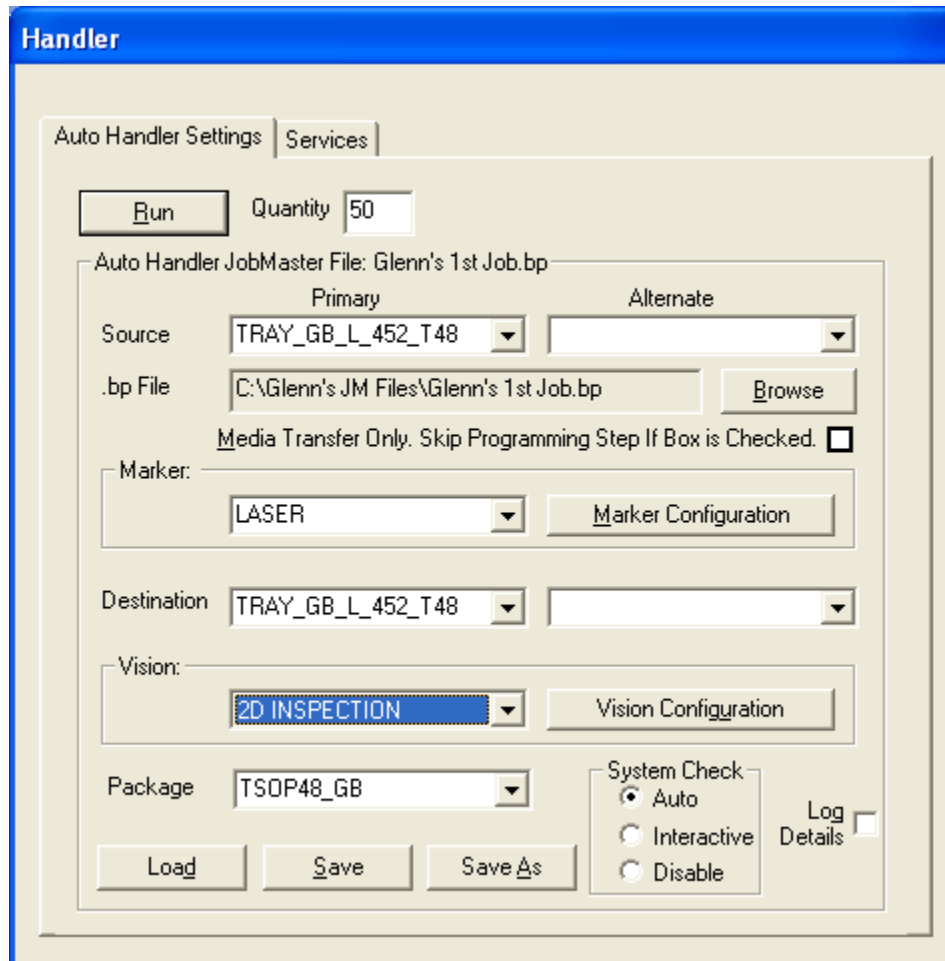


Figure 14 – Handler window

1. Activate the Handler software by clicking on the *Handler* window.
 2. Designate the primary and alternate locations for the *Source* and *Destination* fields.
- ☐ *These fields will reflect whatever was chosen during the last job run before exiting out of the software.*

An example of these settings is as follows:

- **Source:** Primary, TubeIn1; Secondary, TubeIn2
- **Destination:** Primary, TapeOut1; Secondary, (blank)

Devices will be picked up from TubeIn1 and after programming, will be placed into Tapeout1. Then, when TubeIn1 is empty, the Handler will start picking up from TubeIn2.

After programming, the devices will then be deposited into TapeOut1, since there is no secondary destination.

3. The *Marker* field should contain the information regarding marking. If the laser nozzle locations have been taught before and are saved under a specific name, then those coordinates will be automatically recalled.
4. In the *Quantity* field, enter the number of passed devices required for your job.
5. The Cyberoptic laser align sensor is the heart of the vision centering system and it needs to know the dimensions of your device. Under the *Auto Handler Settings* tab in the *Handler* window and enter a package type in the *Package* field. The dimensions of the device will be stored under this name for future jobs.

Example: We used TSOP48, but you can use any naming convention you wish.

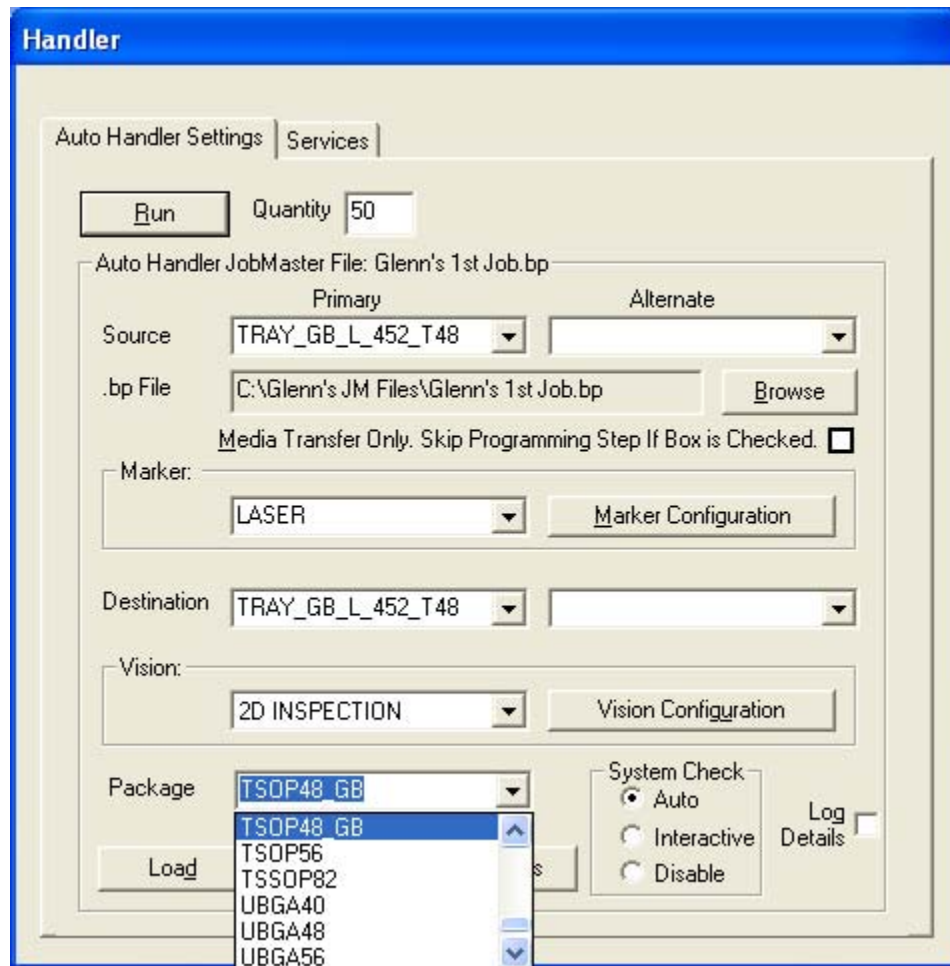


Figure 15 – Handler window, Package Type

6. Now select the *Services* tab and click the **Initialize** button to return the head to the Home position.

☐ This is advised before running an existing job or teaching a new job.



Figure 16 – Handler services window

7. Click the **Teach** button and select the package and location to teach then, click **OK**.
 - *Device thickness, length, and width* can be found on the manufacturer's data sheets or spec. sheets or they can be measured with calipers.
 - The *Drop distance* is how far the device will be dropped into a socket or pockets and is normally set at 50.
 - The *Pickup offset* is how much 'overdrive' of the pickup nozzle will occur when picking up a device. This makes up for slight variations in how level the trays are. The normal value is 45.
 - The *Alignment offset* is where on the body of the device the laser alignment measurements will be taken by the Cyberoptic sensor. A value of about 70% the thickness of the device is preferred.
 - For the *Tolerance* field, always use 20. This stands for 20% of the length and width measurements allowable before the device is rejected.
8. Next, select your package type under the *Pin Numbering Style*. Click **OK** when finished.
9. If the socket modules slated for use have been taught before and the trays you are using have been taught before, return to the *Auto Handler Settings* tab and click the **Run** button to start the job. If not, proceed to the next section, teaching locations.

TEACHING LOCATIONS

When using an automated programmer, i.e. 3000 or 4000 series, it is necessary to teach the different locations for pickup and placement, such as Site (socket module) locations, Primary and Alternate Input Pickup, Laser locations if utilized, Primary and Alternate Output Placement, and Reject location.

1. To teach a location, click on the **Teach** button under the *Services* tab of the *Handler* window.
2. Highlight the location for the teach from the drop-down menu provided and click **OK**.
3. Each location has different requirements for performing the teach. Follow the on screen instructions and answer the questions when prompted to position the head appropriately and save the location correctly.
4. When completed you will be returned to the *Handler* window.
5. Click the **Run** button to start the job.

RUNNING A JOB ON THE HANDLER

Now that you have selected a device, loaded the buffer, setup the Handler, if necessary, and designated the parameters of the job, you are ready to begin programming.

1. Enter the quantity of devices to program in the **Copies** field on the BPWin *Action/Function* window.
2. On the BPWin *Action/Function* window, click on the tab that most accurately describes what you intend to do (i.e., **Program**, **Verify**, **Read**, **Test**).
3. Select the functions to be performed by checking/unchecking the boxes next to the list of available actions.
4. On the *Handler* window click the **Run** button.

STOPPING A JOB ON THE HANDLER

To stop a job, simply click the **Stop** button on the BPWin *Main Screen* or *Handler Window*.

You will be presented with three options:

- **Terminate Immediately** to halt the job, leaving partially programmed devices in an automated programmer's sockets;
- **Terminate Gracefully**, completing the programming for any devices in the sockets; and
- **Resume** the job to allow the job to proceed.

FINISHING A JOB

A job is finished when the amount of programmed parts specified in the **Copies** field has been successfully reached.

However, if you have any parts that have been rejected, it is best to attempt to program those rejected parts again. In many cases, rejected parts merely failed to achieve continuity and will “pass” if run through the programmer again. Those that are rejected a second time are probably defective. To maximize product yield:

1. Place any rejected devices into one of the input source locations.
2. Click **Run** on either the *BPWin Main Screen* or *Handler Window*.
3. Once the last device is run, the job is completed and a summary will appear on screen.

CHAPTER 4

COMMAND REFERENCE

The BPWin software allows for the ease of functionality by placing standard, device-specific functions at the user's command. Commands are located in drop-down menus and file tabs in the main screen as well as hot-link buttons on each active window. To execute a command, select the appropriate execution button or option. There are two things to remember while working with functions or attempting to execute commands within the BPWin software.

- Some commands may not be functional on your particular programmer model.

***Example:** If your programmer only supports PLDs, the commands used strictly for memory devices will be grayed out or not visible on the screen and the converse is true if your programmer only supports memory devices.*

- Sometimes, identical commands invoke different routines when programming memory devices versus PLDs.

Prior to Windows, DOS programs limited the user with the necessity to type in a string command or utilize inline menus to access features and functions of programs. The BPWin software, through the Windows media, includes action button links (commonly known as "buttons") as well as user-input fields, as well as pull-down menus and file tabs, which enable the user to direct the program to specific locations and tasks.

BPWIN WINDOW

PULL-DOWN MENUS

File Menu

Configure

Configure commands the software to re-establish communications with the programmer. This allows for the changing of programmers, listing several options to choose from. This window also provides for handler type to be switched from manual to auto-handler status as well as novice or experienced mode.

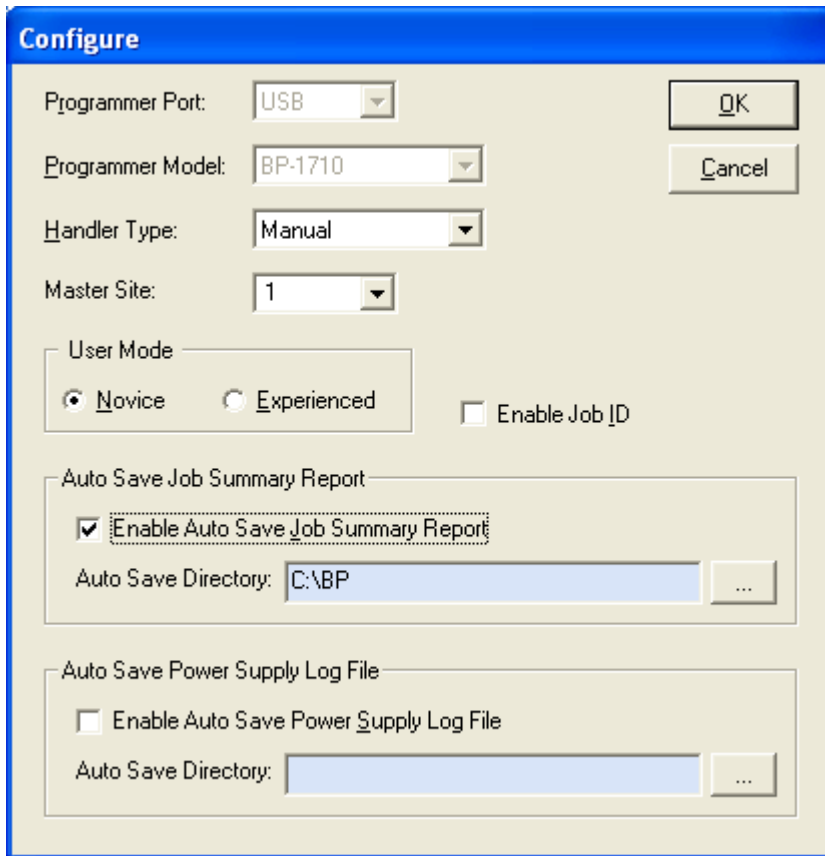


Figure 17 – File/Configure window for BPWin software

Programmer Port selects which port is connected to the programmer. Choices are **DEMO**, **LPT1**, **LPT2**, **LPT3**, or **USB**. Selecting **DEMO** will disable communications with the programmer. You may utilize any of the software's functions while in DEMO mode, except performing any device operations. **DEMO** will automatically be selected if no programmer is detected.

Programmer Model selects, which model programmer, is being used. Select any programmer in DEMO mode and view the list of devices supported on that particular model in the *Device Select Window*.

Handler Type indicates whether or not an auto-handler is being used with the programmer. **Manual** is the setting to use with all programmers not interfaced to a handler.

User Mode distinguishes between **Experienced** and **Novice**. Experienced mode will permit options that are not allowed in Novice mode.

Example: Clearing the buffer and displaying ICC measurements on **Verify and Test**.

Selecting **Novice** will signal the program to prompt the user to accept or cancel before each operation is performed. This is the default choice.

Upgrade (Control U)

The *Upgrade* window allows you to input a special code, which will enable various Advanced Features (AFS) in the software. This code is required in order to enable the BPWin program to

continue communications with the device programmer. For more information, see *Chapter 7, Troubleshooting\Software Updates* (page 7-2).

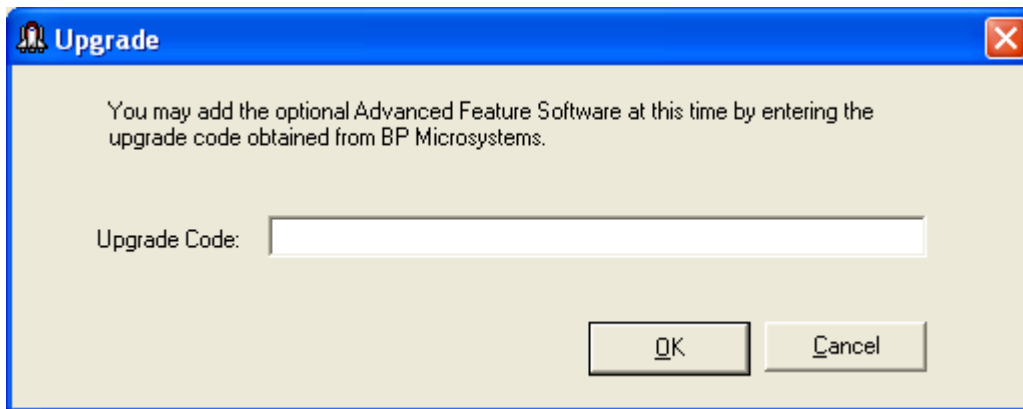


Figure 18 – Upgrade window

✎ To receive an upgrade code, contact Customer Service

In the US: 800-225-2102
Outside the US: 713-668-4600
FAX: 713-688-0920

Exit

Selecting **Exit** from the *File* pull-down menu will close out the software.

Tools

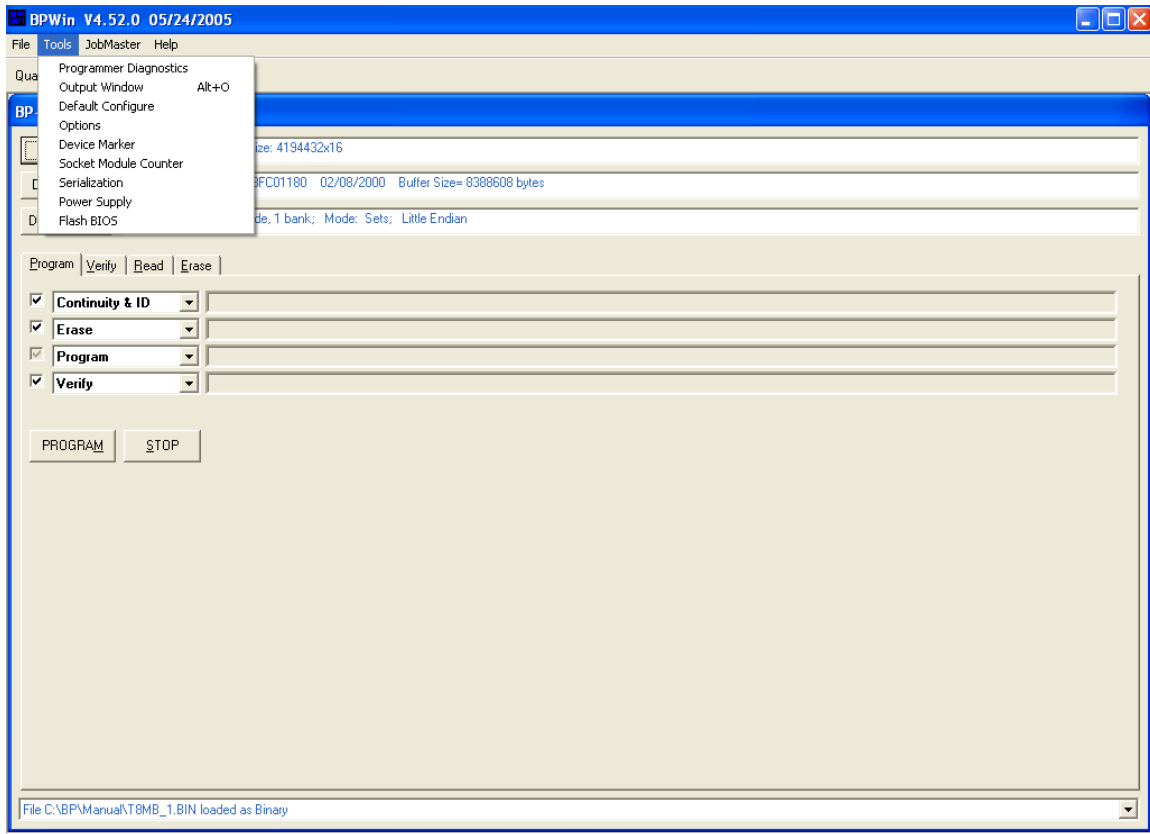


Figure 19 – Tools pull-down menu

Programmer Diagnostics

Programmer Diagnostics performs the routine self-test diagnostics to make sure the programmer and software are communicating and functioning properly.

Output Window (Alt + O)

The **Output Window** displays information about the operations performed and the options selected. Selecting this option again will hide this window

Default Configure

Default Configure allows you to reset all the settings to the default configuration. When this option is selected, you are asked to confirm whether or not you want to reset to the default configuration. This option will clear any device selection and the buffer.

Example: If the users want to start over and select a new device, the *Default Configure* option will allow the user to change the settings all at once.

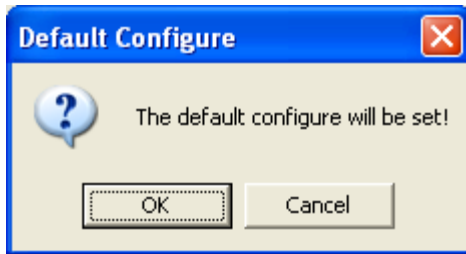


Figure 20 – Default Configure window

Options

Options allow you to set certain global options.

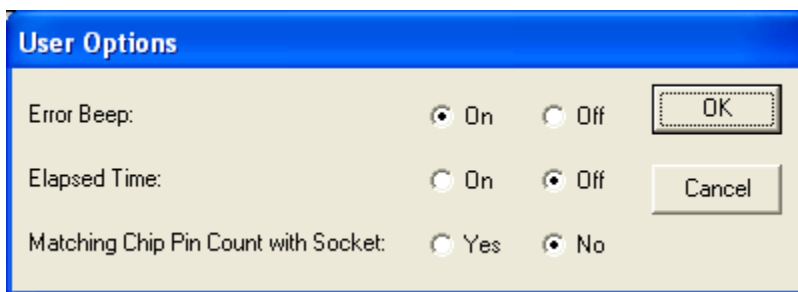


Figure 21 – User Options window

Error Beep allows you to turn on or off the sounds created by the BPWin software that can indicate completion, errors, end of job, etc.

Elapsed Time displays the duration of time of a specific operation, i.e. Read, Program, etc.

Device Marker

This feature can only be used when running an automated programmer. **Device Marker** allows the user select the Marker Package type, either Plastic or Ceramic and designate up to three (3) lines of text.

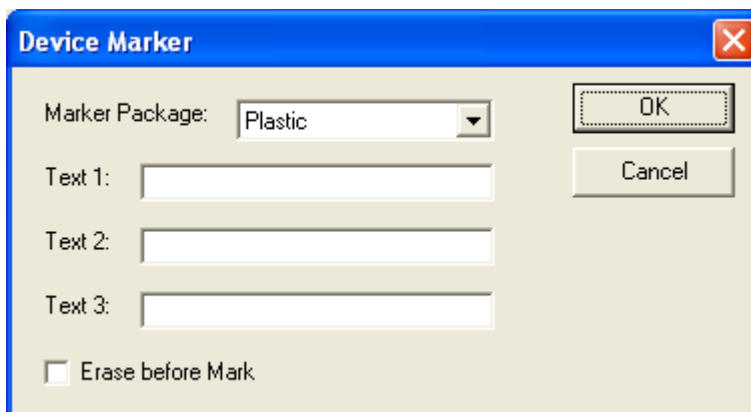


Figure 22 – Device Marker window

Socket Module Counter

This feature keeps track of the number of operations performed using a particular module. The user can view the number of operations on the screen by selecting a site number. The software tracks the number of operations performed by the socket. Any normal operation will turn the counter, and a combination operation only counts as one. For example, a Read or a Compare count as one operation, and an Erase/Program/Verify twice counts as one operation. A verification on its own counts as one operation. This information is also viewable in the Job Summary report.

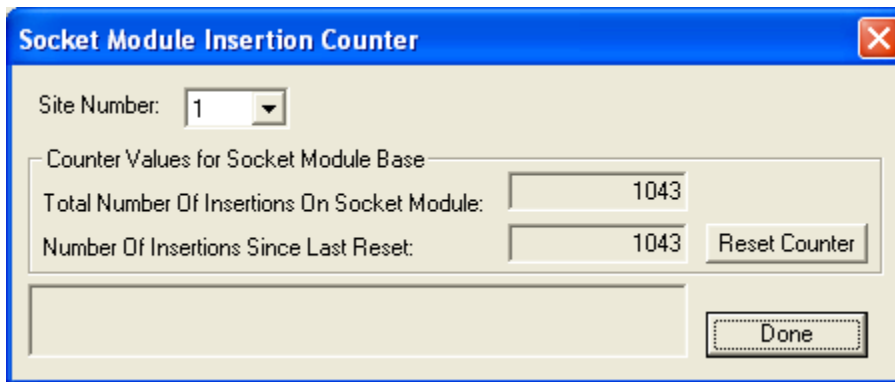


Figure 23 – Device Marker window

There is also a separate “trip counter” that can be reset by the user. This is used when replacement sockets are installed. Resetting the trip counter is only available in Supervisor Mode.

Serialization

This feature supports Simple serialization, a linear sequence of numbers incremented by one, and Complex serialization, for users who wish to create a unique sequence of numbers by writing an algorithm.

For more information on serialization, see *Chapter 6, BPWin Serialization*, page 6-1.

Power Supply

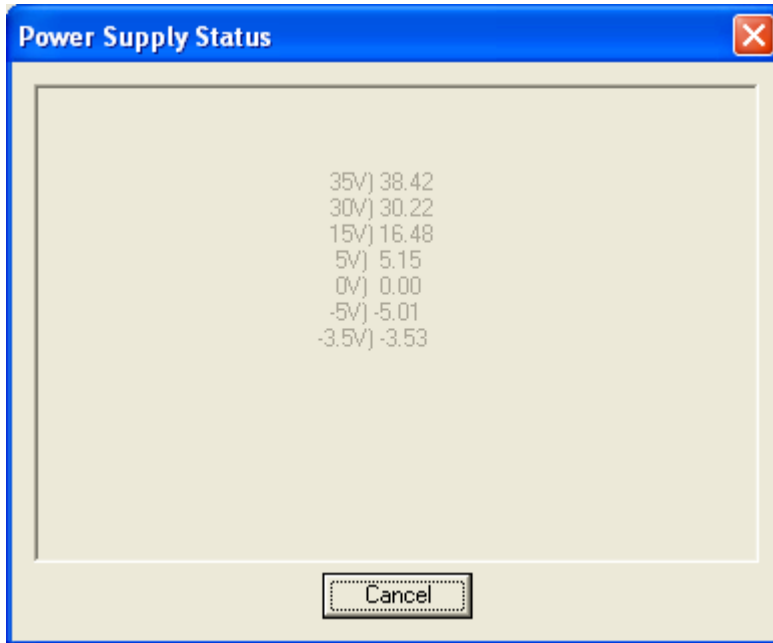


Figure 24 – Power Supply Status Window

This feature is a real time display of the power supply readings for the programmer master site. Actual voltage values are shown by the software. Values marked with an asterisk (*) may indicate a power supply out of range. Perform Programmer Diagnostics in order to fully test a site.

The programming site can be changed by reselecting the Master site number in the Configure window located in the File pull-down menu.

Power Supply values can be automatically recorded in a log file. See *Auto Save Power Supply Log* in the Configure window.

JobMaster

Once JobMaster is installed, the *JobMaster* option will appear on the *Pull-down Menu Task Bar* at the top of the software *BPWin Main Screen*. Selecting the *JobMaster* pull-down menu will allow you access to the following options when in *Supervisor*, or *Normal Access, Mode*.

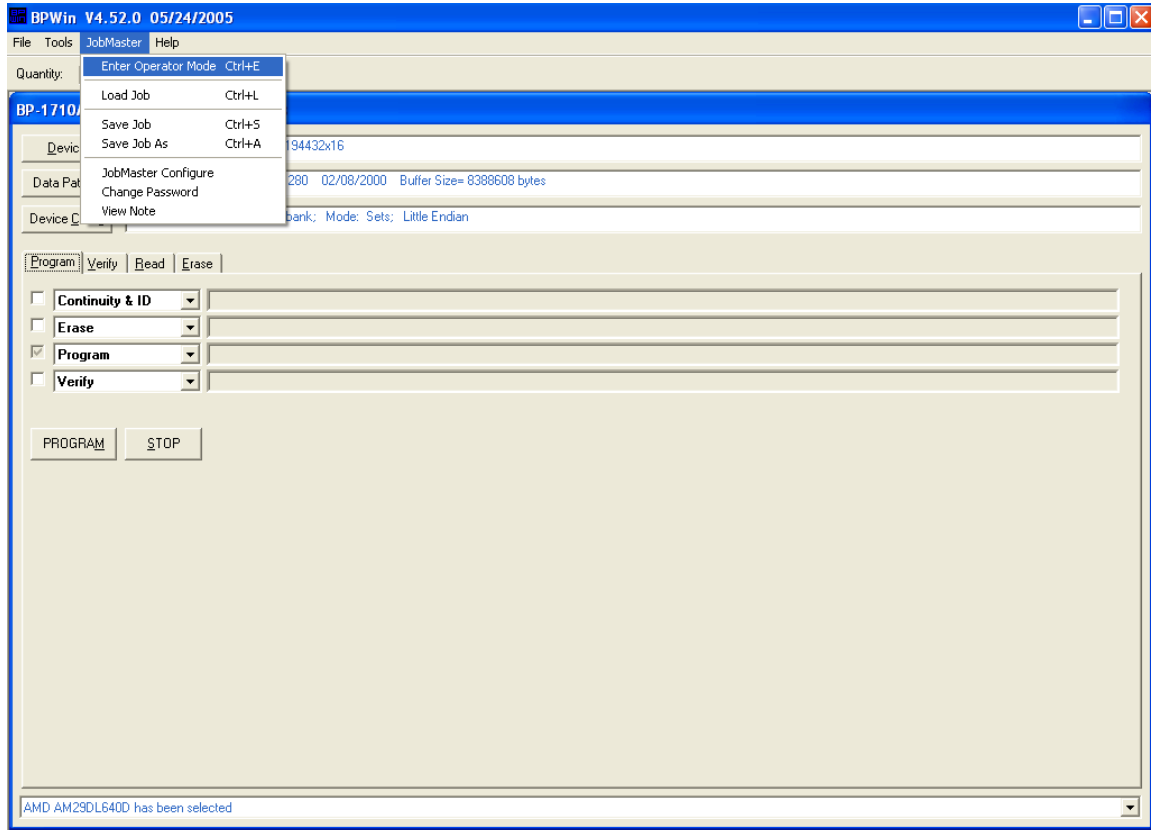


Figure 25 – JobMaster pull-down menu

- ☐ When working in *Operator Mode* only the *Load Job* and *View Note* options are available in the *JobMaster* pull-down menu.

Enter Supervisor or Operator Mode

Use this function to set up BPWin to run in *Operator Mode*, which limits access to functionality within the software, or *Supervisor Mode*, which allows the software to run in normal access mode.

Load Job

Use the *Load Job* command in JobMaster to load existing jobs (.bp files) for programming. Select the location and file and click the **OK** button. The job will load the device and data pattern information, configuration options and any notes applicable. Then, just program, test or verify, depending on what is needed.

- ☐ JobMaster keeps track of the date, time and author of each job's creation, and of every subsequent revision. This information is displayed whenever you select the *JobMaster/Load* option.

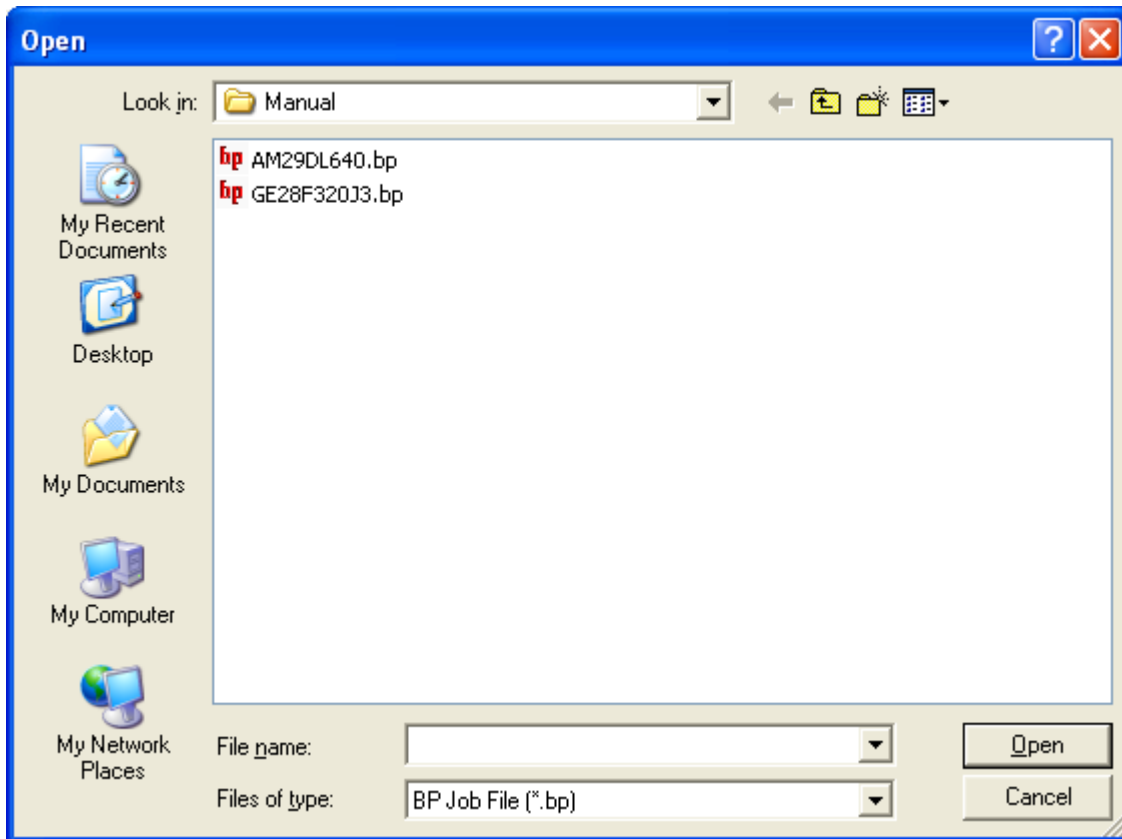


Figure 26 – JobMaster load menu

Save Job

Save Job saves the information chosen within the software (device, data pattern and configure options) under a specific job file to be run while programming.

Once a device has been chosen, the buffer loaded via the **Data Pattern** button and all configurations needed have been set within the **Configure** window, click on the **Save Job** button on the Main Screen or select **JobMaster/Save Job** from the pull-down menu options. In the **Save Job** window, choose a location to store the file and type a name in the **File Name** field. The **Note** window will pop for any messages that need to be shown to the Operator while using this job file. Type in the note and click the **OK** button, or click the **Cancel** to save the job without a note attached.

The **Save Job** function can also be used to add or revise notes already attached to existing jobs. While a job is loaded, go to the **JobMaster/Save Job** option and click. The **Notes** window will automatically come up, allowing for revisions to be made to any existing notes along with a revision history of all notes applied to that particular job and providing an additional space for any new notes to be added.

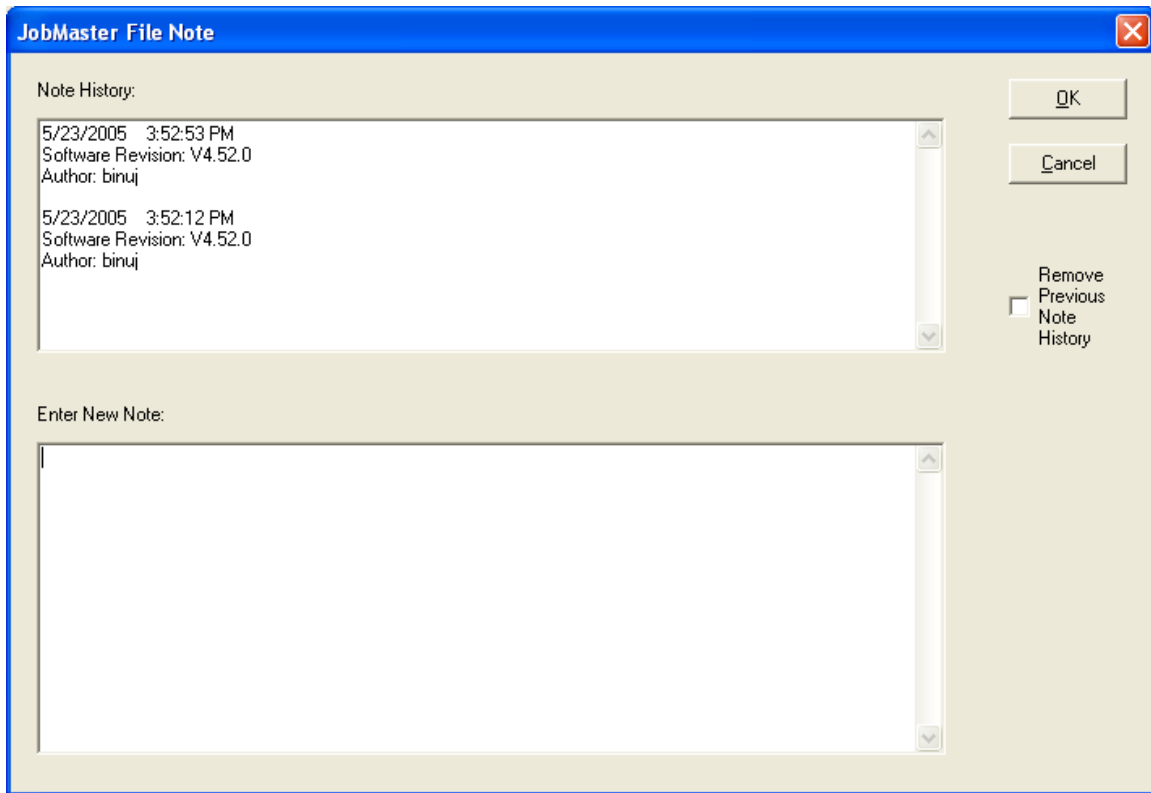


Figure 27 – JobMaster/Note window

Save Job As

Save Job As function is used to create, update, or copy a job file.

After loading the job file, select *Save Job* from the *JobMaster* pull-down menu. Select location and type in job file name in the *File Name* field. Click the **Save** button when done.

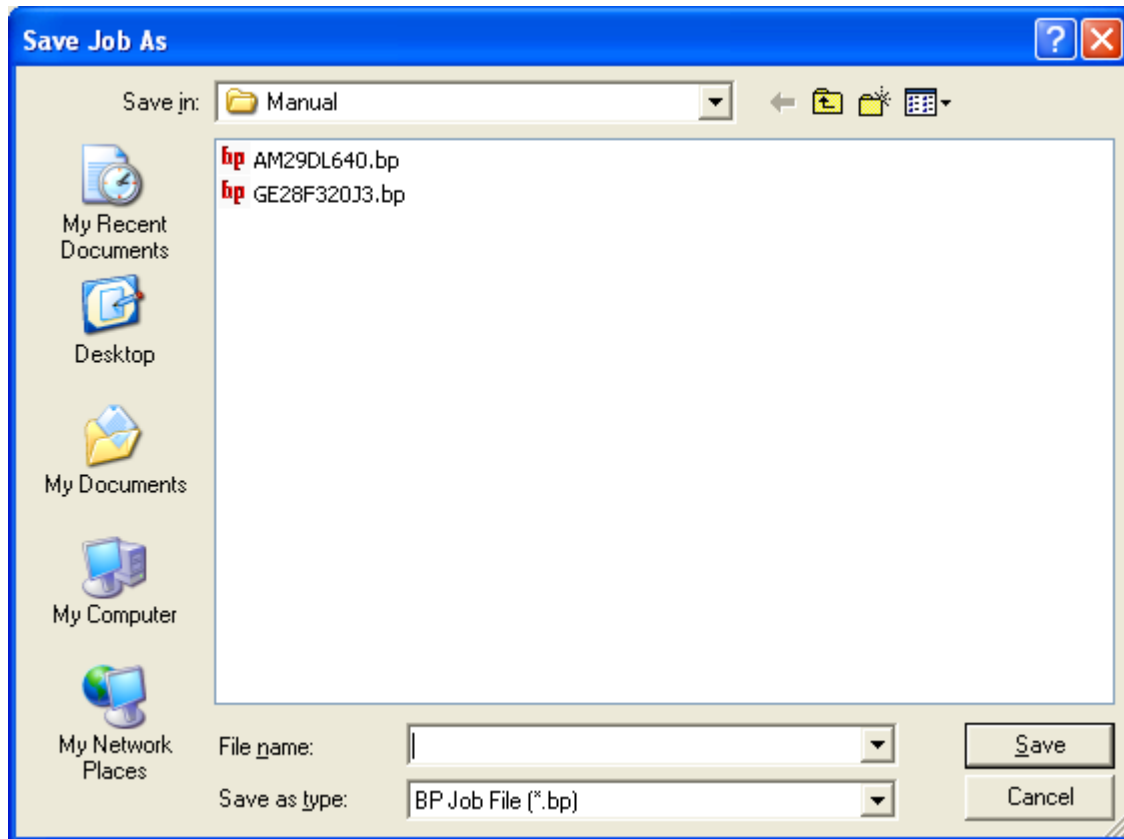


Figure 28 - JobMaster Save As window

JobMaster Configure

Configures BPWin to start up JobMaster in *Operator Mode*.

 You can configure the programmer to start up in Operator Mode automatically.

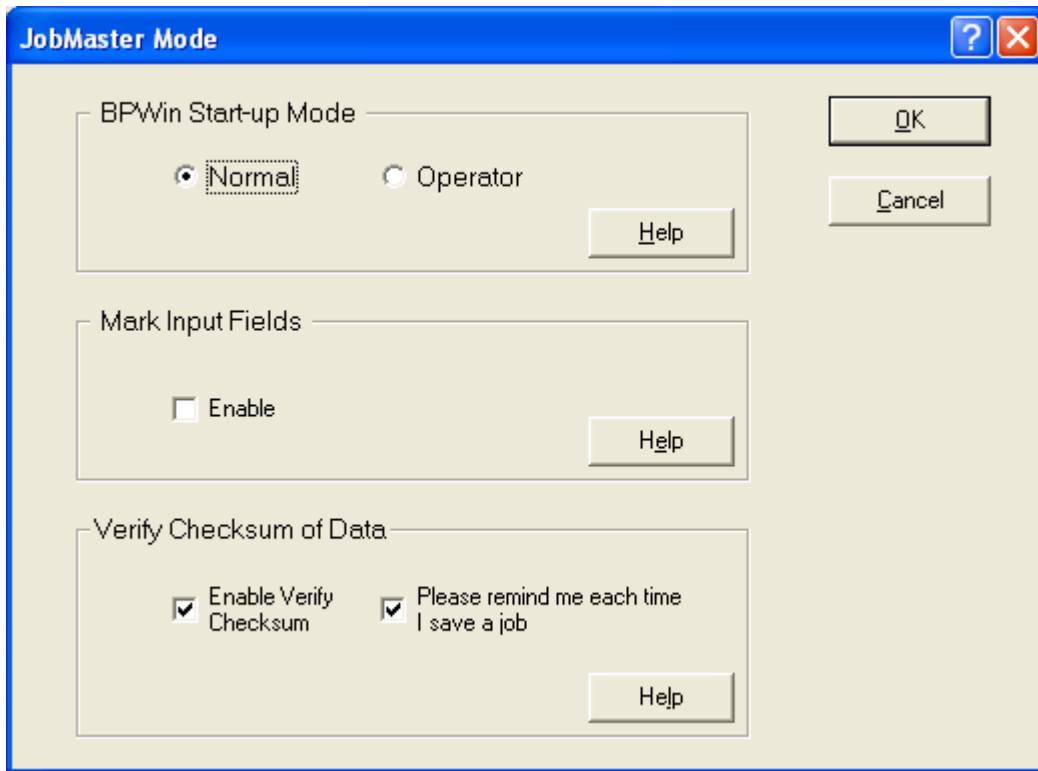


Figure 29 - JobMaster Mode screen

When **Operator** is selected in *BPWin Start-up Mode*, it allows only basic functions such as selecting a job file, testing, verifying, viewing notes and programming. It covers the basic areas that the Operator will need to have access to in order to execute jobs.

 There are three (3) different Note buttons located on this screen. This gives the user, either Supervisor or Operator, more information about the functionality of these areas in the JobMaster Configure window.

Change Password

Change Password allows a Supervisor to change a JobMaster password.

 This function is only allowed to Supervisors with the original password or when setting up a JobMaster password for the first time.



Figure 30 - JobMaster/Change Password screen

Select *JobMaster/Change Password* from the pull-down menu. The BPWin Software will first prompt you for the current password (if presently used), then prompt you to enter a new password into the space provided. Once you have entered a new password and verified the new password, the software will store and announce that the new password has been recorded.

- ✎ **The maximum limit of characters that can be used for a JobMaster password is 15 and IS case-sensitive.**

View Note

This option shows any notes attached to a particular job file.

Choose *JobMaster/View Note* from the pull-down menu. Any notes attached to the loaded job file will appear in the *Note* window, complete with a revision history to keep track of all notes added, date and time as well author.

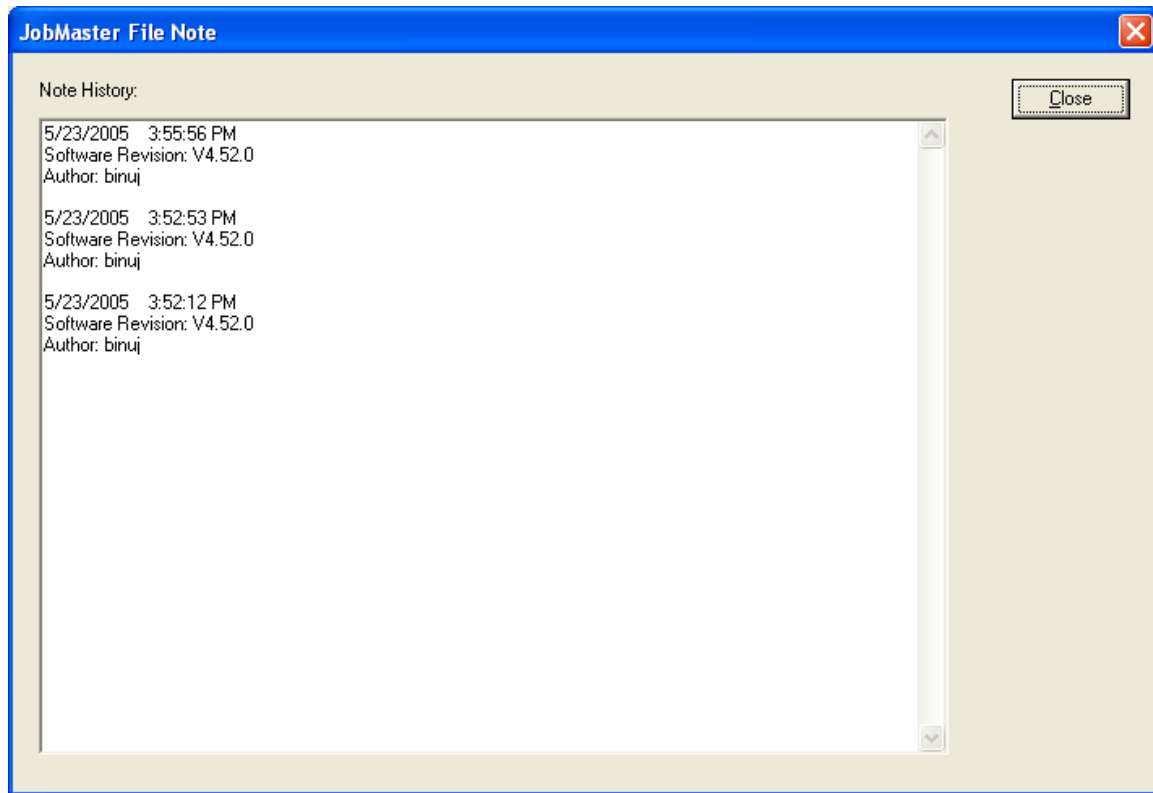


Figure 31 – JobMaster Note window

Help

The Help system is comprised of how to contact Customer Service, registration, and software information.



Figure 32 – Help pull-down menu

Help/Tech Support lists the phone numbers and e-mail address for reaching Customer Service. In the US 800-225-2102. Outside the US 713-688-4600. Tech@bpmicro.com

Help/Registration prompts for information for registering your programmer model and software.

Select Yes to continue

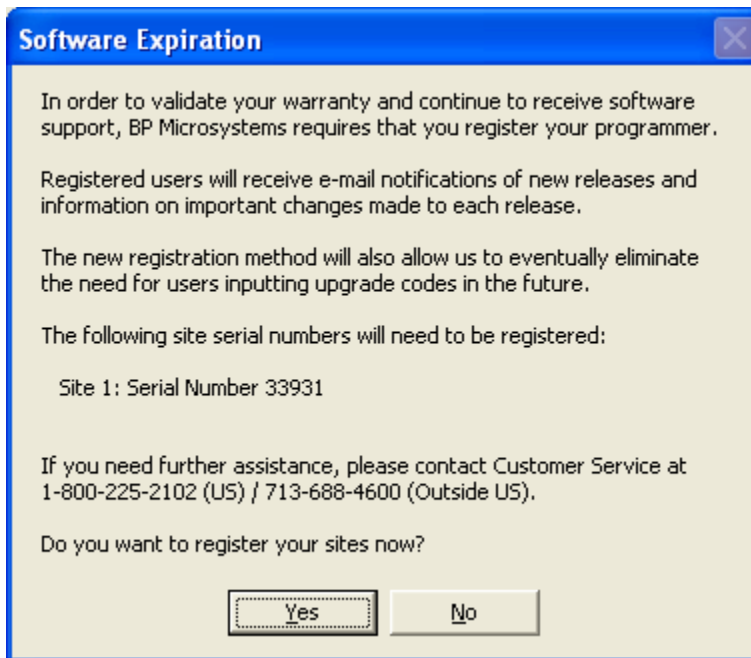


Figure 33 – Registration notice

Provide the information below

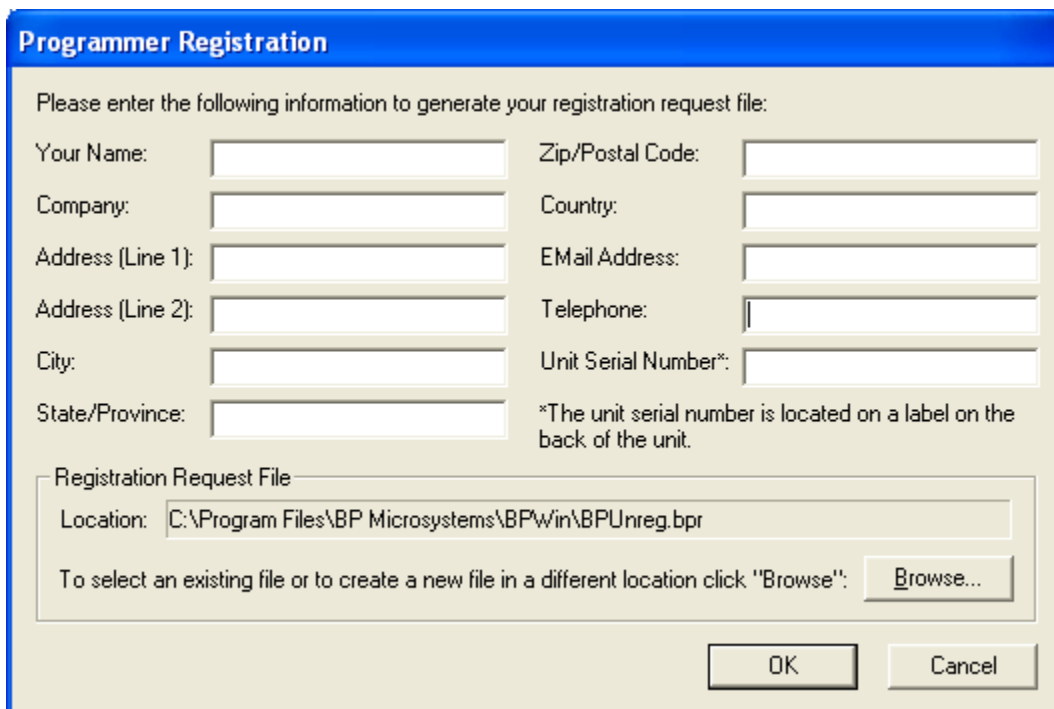
A Windows-style dialog box titled "Programmer Registration" with a blue header bar. The text asks the user to enter information to generate a registration request file. It contains two columns of text input fields: "Your Name:", "Company:", "Address (Line 1):", "Address (Line 2):", "City:", "State/Province:", "Zip/Postal Code:", "Country:", "E-Mail Address:", "Telephone:", and "Unit Serial Number*:". A note at the bottom right states: "*The unit serial number is located on a label on the back of the unit." Below the fields is a section titled "Registration Request File" containing a "Location:" label and a text box showing the path "C:\Program Files\BP Microsystems\BPWin\BPUnreg.bpr". Below this is a "Browse..." button. At the bottom right are "OK" and "Cancel" buttons.

Figure 34 – Registration menu

After completing the registration screen above, please email us the BPUreg.bpr to BP Microsystems

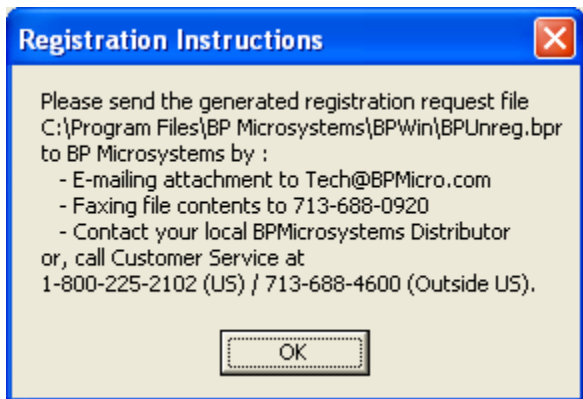


Figure 35 – Registration instructions

Upon receipt of the BPUreg.bpr file, we will validate the information regarding the programmer and email you a BPREG.db file that must be placed in the directory of the BPWin executable.

Help/About BPWin displays the current BPWin version number and specifics about the software.



Figure 36 – About BPWin

ACTION BUTTONS

Quantity Field



The *Quantity* field allows you to enter the number of devices you are going to program. The software will automatically keep track of how many are programmed during the job, how many failed and how many are remaining. When this number is reached successfully, a Job Summary Report will be presented for printing.

Open Button



Pops up the *Open* window and allows a user to choose a JobMaster (.bp) file to open.

Save Button



Opens the *Save* window and allows the user to save the current information as a JobMaster (.bp) file.

Device Marker Button



Opens the *Device Marker* window and allows the user to designate a device package type (plastic or ceramic) and up to three (3) lines of text.

Device Button



The **Device** button gives users access to the list of devices supported by the designated device programmer via the *Select Device* window.

Select Device Window

The *Select Device* Window offers a list of devices supported on the programmer model selected in *Configure*. Selecting a device will configure the programmer with the correct programming algorithm. The algorithm is specified by the semiconductor manufacturer and contains the voltage, timing, and pin-out requirements.

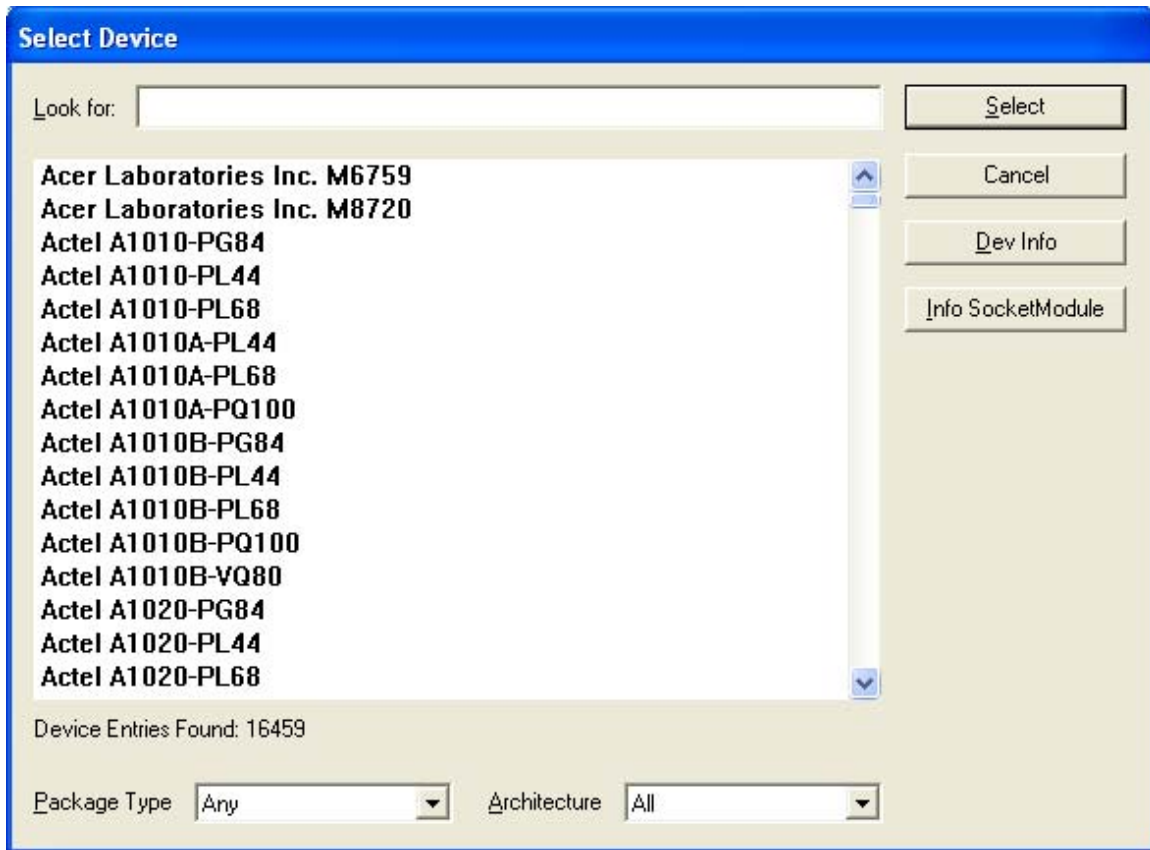


Figure 37 – Select Device window functionality

In the *Look for* box, you can type the device manufacturer and part number. You can also make use of the search engine by typing in the first few letters of the manufacturer's name and part number separated by a comma (,). You will notice that as characters are typed in, the device list narrows to show only those devices containing the characters you have typed, in the order you type them. You can also narrow the list by designating a particular *Package Type* or *Architecture* in the bottom right-hand corner of the screen.

Click on the desired selection from the list of supported devices. It is important to select the correct device for programming. Some character suffixes at the end of the part number pertain to temperature or speed ratings and are not of importance for programming. Choose the device that exactly, or most closely, matches your device part number.

When the correct device is highlighted and the appropriate information entered, click the **Select** button to load the algorithm for that device or **Cancel** to escape without selecting.

The *Package Type* window narrows the search by only listing devices of the package type selected.

The *Architecture* window narrows the search by only listing devices of the type of architecture selected.

☐ Changing the settings in the *Package Type* and *Architecture* fields may cause the changes to become the default settings.

Dev Info Button

The **Dev Info** button lists the current information on the device highlighted along with certain notes and precautions to be aware of if pertinent to the selection of that particular device, such as algorithm settings, incorrect data pattern, erasing procedures, etc.

Once a device is chosen and the *Select Device* window closes, the device information will show up in the field next to the **Device** button. A note or warning may be presented in a *Device Information* window at that time to inform the user of any important information regarding the selection.

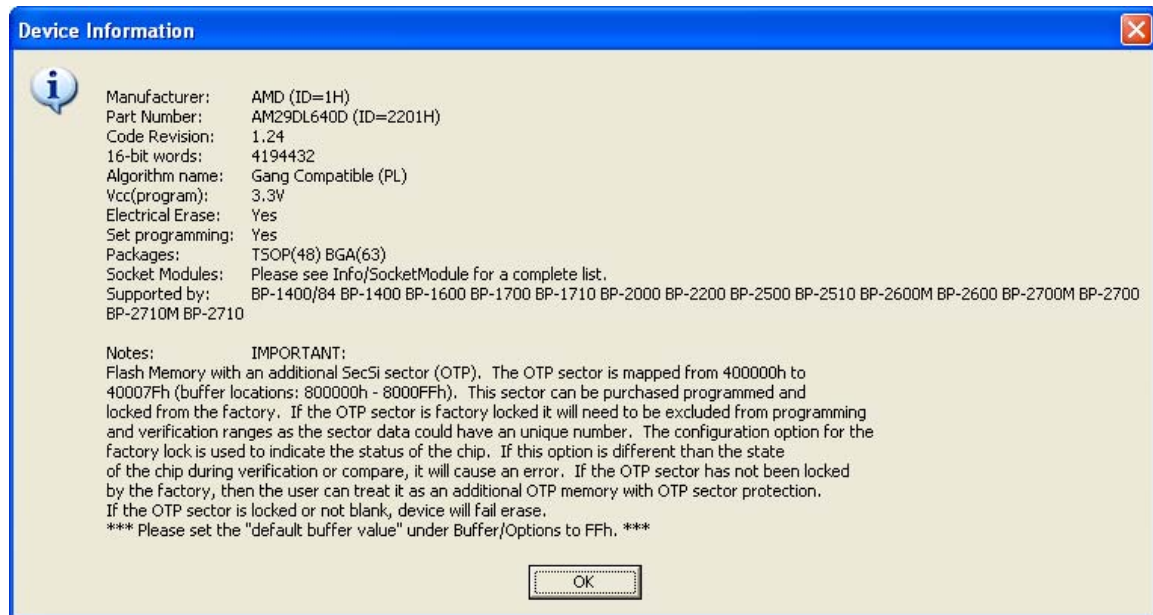


Figure 38 – Dev Info Button

Info Socket Module Button

The **Info Socket Module** button lists the current socket module information on the device highlighted along with certain information pertaining to the socket modules listed, such as package type and device dimensions.

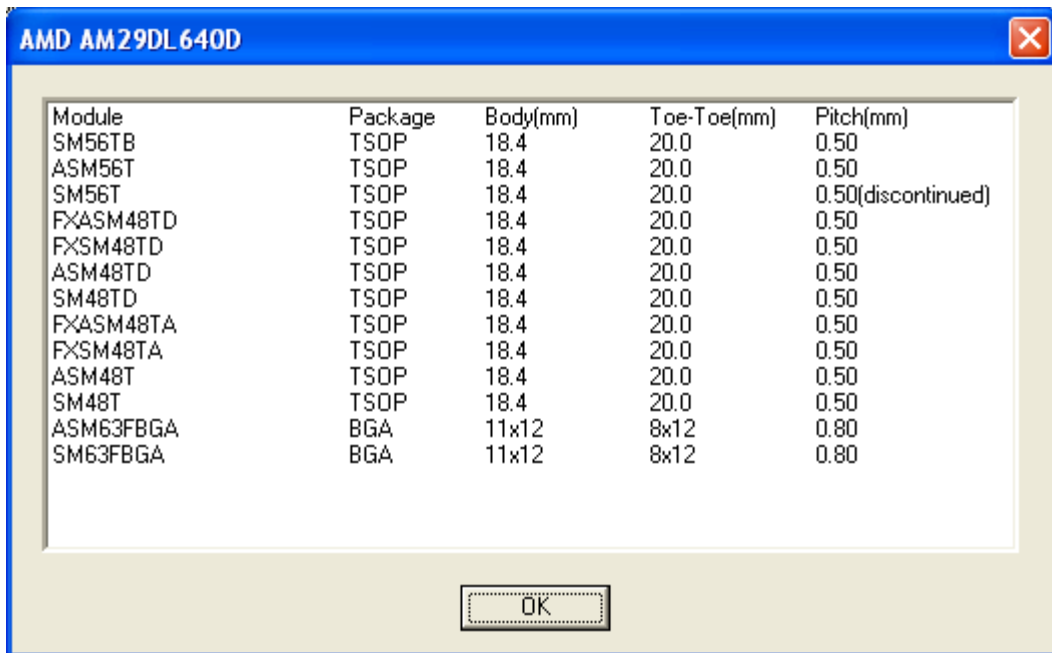


Figure 39 –Info Socket Module Button

Data Pattern Button

Click on the **Data Pattern** button to access the window where a data file can be selected. This window is where data to be programmed can be loaded into the buffer from a file or from another device. It works like most Windows programs for file loading. It also offers the user more fields and options for control such as designating .HEX or .DEC files, and editing the buffer.

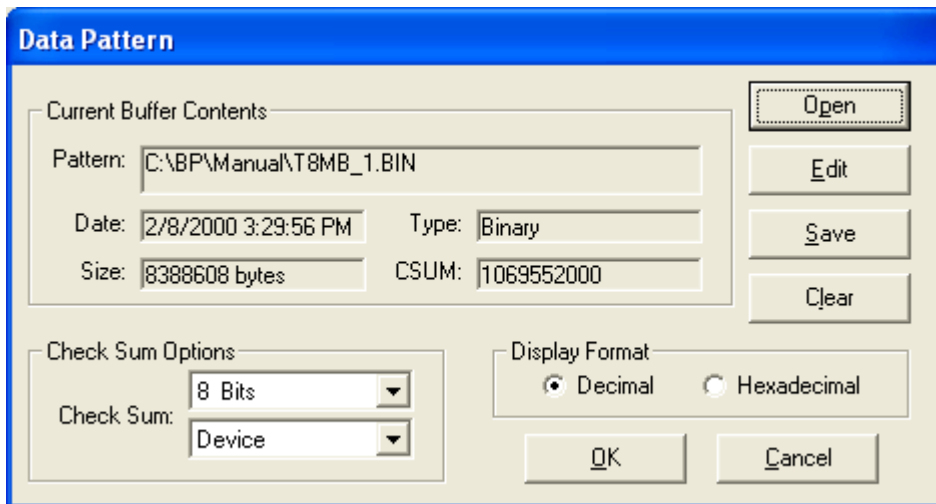


Figure 40 – Data Pattern window functionality

**Check Sum Options* displays the checksum of the data in the file highlighted.

**Size* displays the size of the file selected in bytes.

**Date* displays the creation date of the file selected.

**Type* displays the file format type.

Open Button

The **Open** button brings up the *Load File into Buffer* window. Several options are available before loading the file.

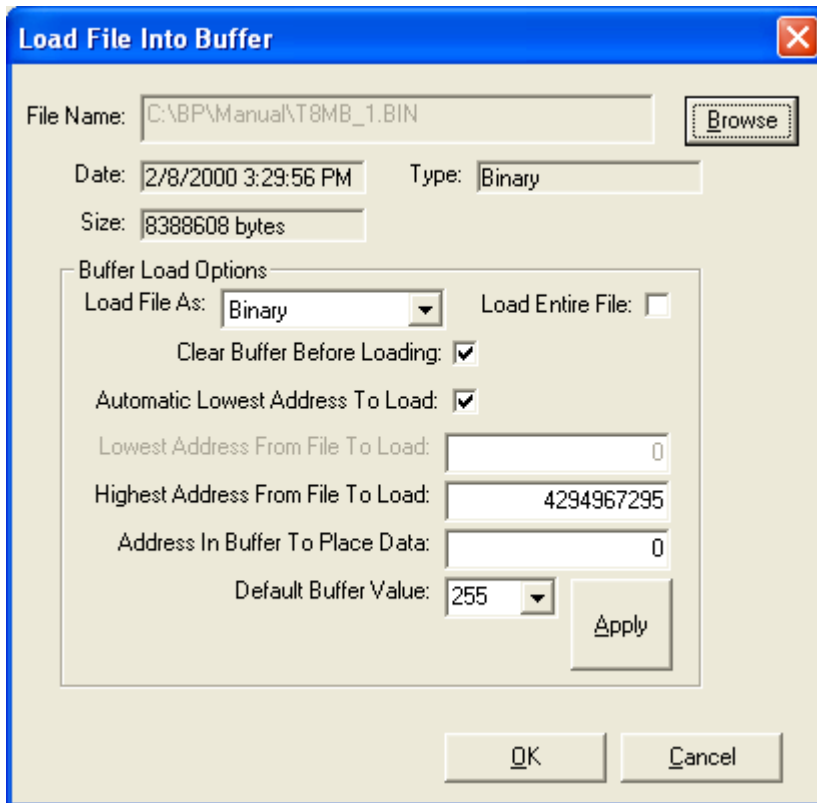


Figure 41 – Load File into Buffer

The *Load File into Buffer* window allows the user flexibility in loading the buffer.

Example: Part of a file can be loaded, multiple files can be loaded, files can be loaded at specific addresses, a default buffer value can be specified, or any combination of these options can be set in the window.

Once a file has been loaded, the options in the *Load File into Buffer* window will become available for the user.

Load File As allows you to specify the file type of the data to be loaded. Even though the software will interrogate the file and automatically choose the file type, it is important that the correct file type is selected so that the correct data is loaded at the correct address. If the file type is not consistent with what you want to load, you may use the pull-down menu option to choose the appropriate type.

Load Entire File is checked as default. Uncheck this box if you intend to load only a portion of the file. Once unchecked, options for *Lowest Address from file to load* and *Highest Address from file to load* become available.

Clear Buffer before Loading is checked as default. To load more than one file into the buffer, begin by loading the first file. Select the second and uncheck this box. You will then need to specify an address to load the second file in the **Address in Buffer to Place Data** field. Repeat the process for any additional files needed.

A common requirement is loading a file generated by an assembler or compiler with a starting address other than zero. Therefore, the program allows you the opportunity to specify another address.

Example: *If your code executes at address F0000hex, you will specify this location in the “Lowest address to load” field.*

For most purposes, this is preferable; however, some compilers generate the code in random order, which will make this feature less desirable.

Automatic Lowest Address from File to Load may be used to discard any data from your file with a lower address than specified here.

Highest Address from file to load may be used to discard any data from your file with a higher address than specified here.

Address in Buffer to Place Data answers another common requirement: loading multiple files into the buffer. In this case, use this field to specify the address where the first byte of the second file should be loaded.

Decimal or **Hexadecimal** allows user preference for viewing address information in any preceding fields.

Once you have determined your specs and are ready to load the file into the buffer, click the **OK** button. If you wish to exit *without* loading the buffer, click **Cancel**.

To load the buffer from a file – select the drive, directory, folder and file. Once you have highlighted the file to load, click on the **Open** button. The *Load File into Buffer* window will appear to prompt you for file type designation. After designating the data type for the file, the data will be loaded into the buffer.

To retrieve data from another device – place the original chip on the socket module and verify the correct data shows up in the *Main Screen* fields. Select the *Read* file tab/**Read** button to load the data from the device into the buffer. The data can then be viewed in the buffer through the *Data Pattern* window by clicking on the **Edit** button or saved to file by clicking on the **Save** button.

Save Button

The **Save** button appears on the *Data Pattern* window once a file has been loaded and brings up the *Save File from Buffer* window and saves the buffer contents to disk. You may choose any local or network drive available to save the file in.

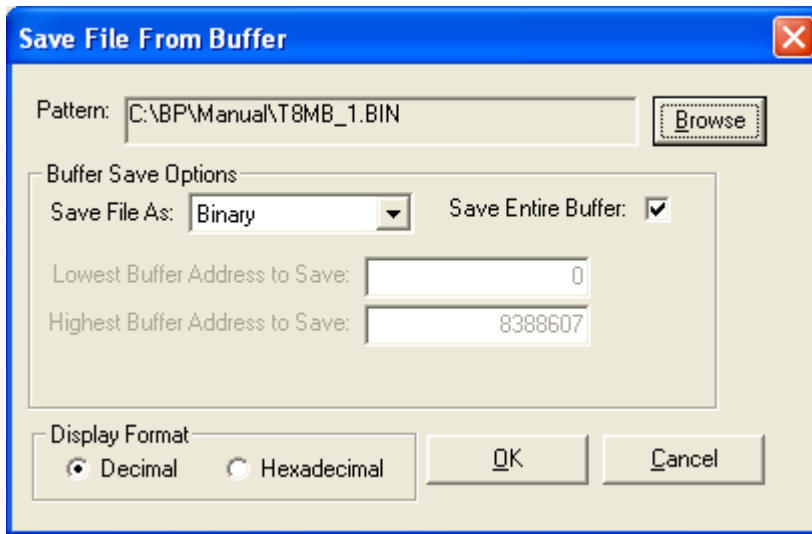


Figure 42 – Save File from Buffer

Pattern field is where the name of the file will appear. To type in a new name for the data file, click the **Browse** button and type the name in the **File Name** field. *Save file as* field allows you to designate the file type to use for formatting the data file.

Save Entire Buffer is checked as default. In order to save a specific address range, uncheck this box. Once unchecked, the *Lowest buffer address to save* and the *Highest buffer address to save* options become available.

Lowest Buffer Address to Save field allows you to specify the lowest buffer address to save.

Highest Buffer Address to Save field allows you to specify the highest buffer address to save.

Decimal or **Hexadecimal** allows user preference for viewing address information in any preceding fields.

To proceed with saving the file, click the **OK** button. To exit without saving the data, click **Cancel**.

Close Button

The **Close** button closes the *Data Pattern* window without opening a file. If you have saved a file or made any checksum selections, these changes will be saved over the original file and be in effect when you retrieve the file at a later time.

Edit Button

You may choose to edit the buffer contents by clicking on the **Edit** button.

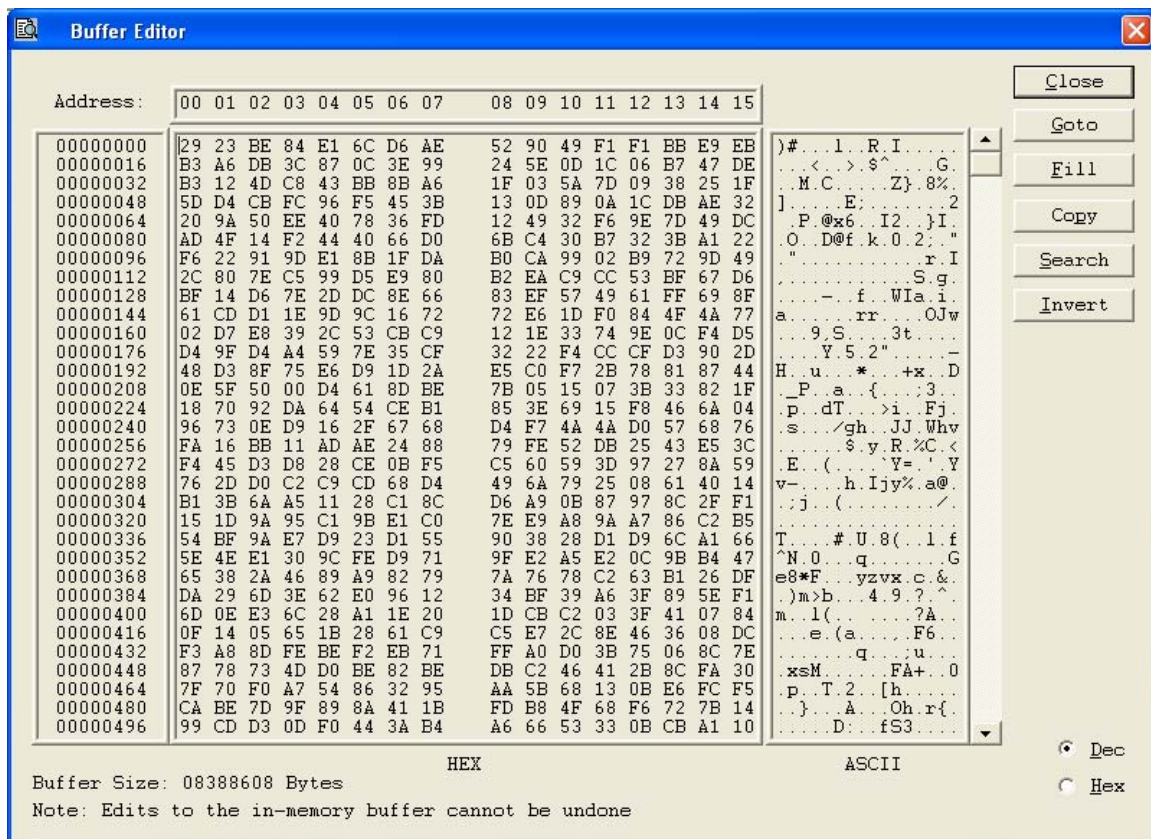


Figure 43 – Buffer Editor window

The *Buffer Editor* window is setup to list the addresses on the left of the screen, the data for each address in the middle with the fuse legend directly above, and the ASCII version of the file on the right.

The *Buffer Editor* window allows you the options to **Goto** a certain memory location within the buffer, **Fill** a range of addresses with a certain value, **Copy** a specified memory range to another location within the buffer, **Search** for a data pattern within the buffer, and specify **Dec** or **Hex** via a radio button.

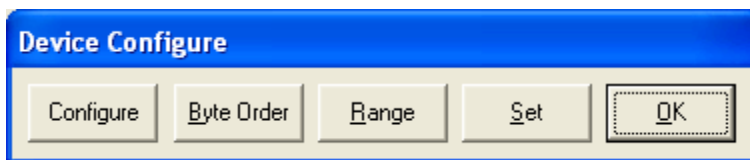
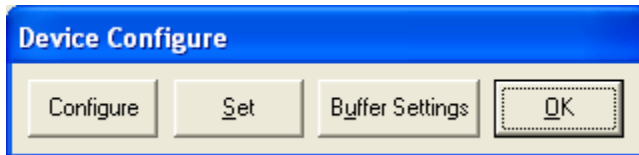
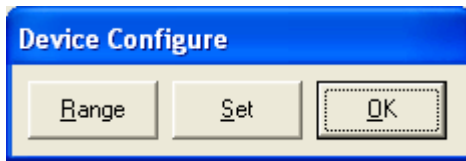
Within the **Search** function, you may choose to start the search from a particular address in the *Start Address* field or search for a particular pattern of data in the *Pattern* field (can be by multiple bytes or ASCII text). The *Pattern* field will allow you to specify whether the pattern you are searching for is in hex or ASCII and whether the search goes forward or backward from the starting address.

Clear Button

The **Clear** button clears the buffer contents to the value specified as the default buffer value. This button only appears when the *Configure* menu option under the *File* pull-down menu is set to *Experienced Mode*.

Device Config Button

You may set any options that are available for a specific device through the **Device Config** button. Listed below are the definitions of common configuration options available. Not all device options found in the software are listed and not all options will pertain to every device.



Figures 44, 45, 46 – Device Configure windows showing some of the different functions available for different devices

Range sets a specific address range in the device to program data or set an address to offset data as it is programmed into the device.

Set divides the data into two or more devices to be read either in parallel by entering values in the *Data path width* field, or bank – programming by dividing the data into two or more devices to be read in series - by increasing the number in the *Number of banks* field. The **Programming Mode** allows you to select either a single device or a set of devices.

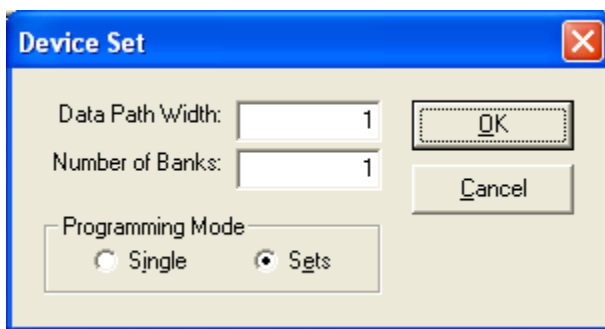


Figure 48 – Device Set window

Vector sets test vector parameters. These options define how the vectors are applied and how the results are returned. This option is only available on appropriate devices, if there are test vectors in the specified file and **Test** has been enabled under the *Test* file tab.

ResetPol sets the Reset Polarity configuration bit on devices that have this feature.

Configure displays any of the variety of special options available to specific devices such as sector protection, boot block lockout, watchdog timers, oscillators, brownout timers, code protection, identifiers, lock blocks, etc.

Byte Order allows you to select most-significant bit (MSB) or least-significant bit (LSB) for programming.

User register allows you to set special options in the device for those devices that have this as an option

Buffer Settings allows you to designate an offset for the buffer.

Use the **Test** button to begin the functional test operation.

Secure performs a stand-alone secure. Options to *ID Check* and *Continuity Check* are available.

Use the **Secure** button to begin the secure operations.

MAIN SCREEN FILE TABS

Depending on the device chosen, any of the following action/function file tabs may appear in the bottom of the viewing screen.

 *Depending on applicability, each particular file tab may include any or all of the following actions pertaining to each function.*

Program – Chksum, Secure, Program, Test, Blank Check, Continuity, Probe, Erase, Verify & Test, Secure & Test, Continuity & ID, ID Check, Verify Once, and Verify Twice.

Verify – Continuity, Compare, Verify, ID Check, Continuity & ID, Verify Once, Verify Twice, Compare, and Sum.

Read – Clear Buffer First, Continuity, Read, ID Check, and Continuity & ID.

Test – Continuity and Test.

Blank – Continuity and Blank Check.

Secure – Continuity, Secure, Secure & Test, Program, and Probe.

 *Check boxes next to certain functions within each file tab may be grayed out, depending on the device selected. These functions are mandatory for the particular device selected and may only give you the choice of choosing a certain type within that function pull-down menu.*

 *Each file tab allows for a start button (named after the function tab, i.e. the Program tab has a Program button) and a stop button to abort the request.*

HANDLER WINDOW

Handler

Auto Handler Settings | Services

Run Quantity

Auto Handler JobMaster File: Glenn's 1st Job.bp

Primary Alternate

Source

.bp File **Browse**

Media Transfer Only. Skip Programming Step If Box is Checked. ☐

Marker: **Marker Configuration**

Destination

Vision: **Vision Configuration**

Package

System Check
☒ Auto
☐ Interactive
☐ Disable

Load **Save** **Save As** **Log Details** ☐

Figure 49 – Handler Window, Auto Handler Settings Tab

Due to the fact that the BP software is Windows-driven, all functions of the Programmer and additional Laser or Labeler units are handled and maintained through the software in the *Handler window*.

Input/Output media is stored within the software and you may choose any combination of input/output available to you.

AUTO HANDLER SETTINGS TAB

Run

When all information has been set and the job is ready to begin, select the **Run** button to start the job.

Quantity field

Enter the number of passed devices required for your job.

- ✎ This is not the total number of parts being programmed nor is it the number of attempts made before failing a part. This number should be the exact number of passed devices needed to complete the job. The Programmer will use this number as its goal no matter how many devices fail, and will not end a job until this number is reached.

Source and Destination fields

The *Source* and *Destination* fields are where to designate primary and alternate locations for input and output of devices. The default settings for these fields will list whatever was last used prior to exiting the software.

.bp File Field

This field is where the designated .bp file is selected. Click the **Browse** button to locate a .bp file. Highlight the specific file desired, and click **Open**. The software will load the selected file and bring up the *Job Setup Verification* window detailing the history of the .bp file loaded, including any notes on the file.

 For more information about .bp and .abp files, see Chapter 5, Using JobMaster, page 5-1.

Marker Option fields

The *Marker* field has been provided for designating an optional marking device, either Laser or Labeler. This pull-down menu field will list either option that may be available to you. You may choose not to have a marking option by selecting **(none)** in this field or highlighting the text in the field and pressing <Delete>, leaving the field blank.

The **Marker Configuration** button allows you to configure the Laser or Labeler to your specifications needed for a particular job. Use this button when setting up the optional marking capabilities. You may designate speed, power, and text in the provided fields. Each time you create a new job, you should configure the marker as specifications will change with each job

Package field

Enter a device package type in this field.

Action Buttons

Load

Once all information has been put into the various fields within the *Handler* window, select **Load** to load the handler file into the software.

Save

Selecting **Save** saves all changes made to an existing .abp file.

Save As

Choose the **Save As** button when creating a new .abp file or copying an existing file.

SERVICES TAB



Figure 50 – Handler Window, Services Tab

Action Buttons

Teach

This button activates the window for teaching all locations, including I/O, Reject tray.

Initialize

Use this button to initialize the programmer for startup.

Cycle Power

Cycles the power and re-initializes the Handler.

PNP Control

Allows the user to temporarily turn the vacuum on or off for testing.

Open Sockets

Releases the pressure plates and opens the sockets on the Programmer.

Close Sockets

Closes the sockets on the Programmer and secure the pressure plates.

Empty Socket Test

Can be used to test a site to see that parts are properly removed.

Diagnostics

This is where all diagnostics on the Handler can be performed.

Win SPTerm

This is used for advanced troubleshooting by authorized field service personnel.

Tape Advance

This will advance the tape loader one pocket and can be used to test communication.



KEYBOARD USAGE AND HOT KEYS

The BPWin program also utilizes keystroke combinations, called hot-keys or quick-keys, to allow the user more freedom in directing the program functions. Listed below are common keys and keystroke combinations used within the software. You may also use the keystroke combination **ALT** + Underlined character wherever applicable.

*Example: **File** pull-down menu. Instead of using your mouse, you may also press the **ALT** key and the **F** letter key at the same time to gain access this menu.*

COMMON USAGE KEYS	
The following keys may be used at any time:	
KEY	FUNCTION
Esc	Close the current window/retain original information. *Does not work in <i>Configure</i> window, must actually click on the Cancel button.
←, →	Move left or right to a selection.
↑, ↓	Move up or down to a selection.
Enter	Execute function command that is selected.

COMMON HOT-KEYS (QUICK KEYS)	
KEY COMBINATION	FUNCTION
Alt-C	<i>Device Configure</i> window.
Alt-D	Brings up <i>Select Device</i> window.
Alt-E	Selects <i>Erase</i> file tab.
Alt-F	Select <i>File</i> drop-down menu.
Alt-H	Selects <i>Help</i> drop-down menu.
Alt-O	Displays Output window.
Alt-P	Selects <i>Program</i> file tab.
Alt-R	Selects <i>Read</i> file tab.
Alt-S	Selects <i>Secure</i> file tab or Stop button if <i>Secure</i> tab is not available.
Alt-T	Selects <i>Tools</i> drop-down menu.
Alt-V	Selects <i>Verify</i> file tab.

DIALOG BOX COMMAND FUNCTIONS

Dialog boxes gather information before executing commands. The dialog box control keys are:

KEY	FUNCTION
Esc	Cancel the active window and return to Main Screen.
Enter	Exit the dialog box and execute command.
Home	Move to first selection of current line.
End	Move to last selection of current line.

EDIT COMMAND FUNCTIONS

KEY	FUNCTION
←, →, ↑, ↓	Move cursor.
Home	Move to front of current line.
End	Move to end of current line.
Ctrl-U	<i>Upgrade</i> window.
Enter	Make selection.

CHAPTER 5

USING JOBMASTER

OVERVIEW

JobMaster is a powerful tool for automating production. It allows a Supervisor to set up a job to precise specifications, test the results and then protect the routine so that it cannot be inadvertently modified. Once a master device is programmed and approved, JobMaster is ready to begin full-scale programming immediately, without further set up.

JobMaster relates to two different types of people within the production facility:

Supervisor(s) – designated to set up jobs within the software, as well as regulate the changes made to any existing jobs. Under *Operator Mode* these functions can only be done by a Supervisor possessing the password to enter JobMaster.

Operator(s) – runs the jobs to program devices.

FEATURES

JobMaster has been designed for maximum simplicity and consistency. It takes the “guesswork” out of many of the steps needed to program devices, such as choosing a device, selecting a data pattern, setting special configuration needs, etc. Once a job has been set up, the Operator’s only tasks are to specify the number of devices to program, and click the **Program, Test, Verify, (or other option)** button. Configuration, loading the buffer, and device selection are taken care of automatically.

.bp FILES AND .abp FILES

JobMaster jobs are created and stored as singular files with a “.bp” extension. The information regarding the Autohandler, such as input/output, additional components (laser/labeler) and locations of placement and pickup are created and stored as singular files with a “.abp” extension. This is a feature that only BP programmers have. The “.bp” extension identifies a file as a BP Microsystems “job file” while the “.abp” extension identifies a file as a BP Microsystems “autohandler file”. The BPWin software uses the .bp files to perform specific jobs set up by a Supervisor. These files are extremely flexible, enabling a user to create and transfer any “.bp” or “.abp” file to and from any location using the BPWin software.

Example: *A design engineer, using a desktop programmer, can create a job to program a part used in design prototype. This job file can then be emailed to a production facility equipped with a BP production programmer using*

BPWin. Utilizing the AutoHandler window, the operator can then link a .abp file to the .bp file created by the design engineer. The software would use the .bp file to set up the programmer, guaranteeing that parts produced would be identical to the part the engineer programmed at his desk.

Both “.bp” and “.abp” files can also be accessed directly through the Windows Explorer program. Simply locate the file and double-click on it. The BPWin software will be launched and the designated “.bp” or “.abp” file will be loaded into the BPWin software through JobMaster.

A .bp file takes the role of setting up the programmer by configuring the programmer, selecting the device, setting the device configuration, storing the data pattern, and choosing the programming options.

A .abp file takes the role of setting up the AutoHandler by configuring the pickup and placement locations for input, output and rejects; z-height and theta rotation; and additional components (laser/labeler).

Some other .bp and .abp file benefits are:

- Files are in binary format.
- Files are error checked automatically.
- Files cannot be used if they have been modified inadvertently.
- Data sharing with .bp and .abp files makes it easier for global companies to share data. Transferring designs from engineering to manufacturing or from customers to programming centers is easy and hassle-free. These files can be simply copied to a floppy disk, positioned on a network account or emailed.

SIMPLE AND SECURE

JobMaster sufficiently takes the place of using macros. Originally, macros were the standard for creating a recorded file of commands and settings to run a programming function. The JobMaster software incorporates the usability of the macro function with enhanced features and supplies a way to monitor, via revision history and logs, and secure all jobs created and stored within the software.

Since JobMaster can be password protected, it also ensures that Operators can only run predefined jobs, which further cuts down on the possibility of Operator errors. When *Operator Mode* is selected for startup, only those functions allowed without a password will be available to the Operator.

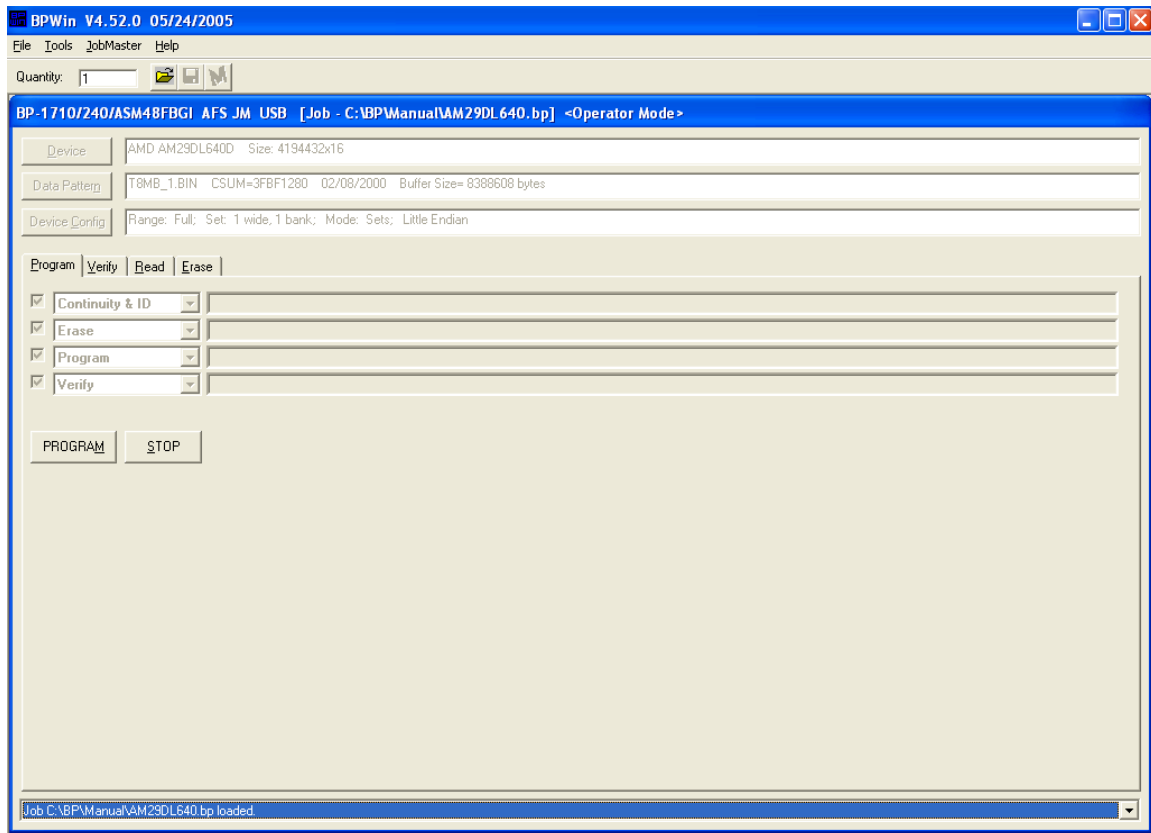


Figure 51 – Operator Mode showing secured, grayed out areas

JobMaster also supplies a user-definable *Note* field, complete with revision history, for additional information that the Operator may need to know before performing a job. When a job file is chosen, a window will appear displaying any information contained in this field. The option to view a note is also given under the *Jobmaster* pull-down menu, so that the note may be reviewed when and as needed.

JobMaster also tracks the original author of each job and all changes to jobs. This information pops up in a window after the *Note* appears when a job file is loaded. This information is helpful in checking the status of jobs that have been edited or updated as well as keeping track of when revisions are made to jobs and by whom.

JobMaster allows you to create a job on any BP Microsystems programmer running the BPWin software, then run it on a BP automated programmer running the BPWin software. This can be a great advantage, since the BP automated programmer can be devoted to actual production runs. The only restriction is that marking information can only be added to the job if the laser marker is attached to the programmer. However, the job can be created on another programmer and the marking information added (in a relatively short amount of time) once the job has been transferred to an automated programmer.

INSTALLATION

JobMaster is supplied within the BPWin Software. To receive JobMaster on an older version of software, you must acquire the JobMaster software upgrade from our Customer Service. Once you have purchased and loaded the upgrade, the *JobMaster* pull-down window will be visible from the command menu at the top of the BPWin software screen. For upgrade information contact:

Customer Service or Sales 800-225-2102
713-688-4600 (outside the U.S.)

CREATING A NEW JOB

The procedure for creating a new job file for JobMaster is very straightforward and user-friendly. This section assumes you understand the basics of programming a device using the BPWin software. The steps for programming a device as well as individual commands and functions are detailed in the programmer's manual. Before creating a new job within JobMaster, make sure you have taken the following steps:

- Select the device that you want to program.
- Load the buffer with the information to be programmed into the device.
- Set any special configuration options.

Once you have chosen the device and have loaded the buffer contents within the **Data Pattern** button and have set any special configurations using the **Device Config** button, all that is needed to create a job file is to click on the **Save Job** button or select *Save* from the *JobMaster* pull-down window. Give the file a name and choose a location for the file to be saved.

When the Operator chooses *Load Job* from the *JobMaster* pull-down menu, a window listing all .bp job files will appear for the Operator to choose from. Once a file is chosen, it will be loaded into the BPWin software.

The *Note* option is another user-definable field that will be displayed whenever an Operator chooses this job. You may use this field to inform the operator of any information that might be useful to the job, or you may leave it blank. If the field contains information, that information will pop up on the screen for the Operator when a job has been selected.

COPYING AN EXISTING JOB

To copy an existing job, simply load the job by selecting the *Load Job* option in the *JobMaster* pull-down menu. Once the file is loaded, select the *Save Job As* option from the *JobMaster* pull-down menu. Type in the new file name in the *File Name* field of the *Save As* window and select a location for the file to be saved in.

☐ *JobMaster keeps track of the date, time and author of each job's creation, and of every subsequent revision. This information is displayed whenever you select the JobMaster/Load option.*

LOCKING THE PROGRAMMER IN OPERATOR MODE

One of the major highlights of JobMaster is the ability to secure the programmer and limit the Operator's access to certain functions through the use of *Operator Mode*. To lock the programmer in *Operator Mode*, select *JobMaster/JobMaster Mode* from the pull-down menu bar and set the default mode to *Operator Mode* by clicking on the appropriate radio button. Click on the **OK** button to accept.

 Before you select the *Operator Mode* default option, you should enter a password. To establish a password, go to *JobMaster/Change Password* and enter the password (must be 15 characters or less) in the *New Password* field and re-type for verification in the *Verify JobMaster Password* field. If you enter a password, it will be necessary to use it to get out of *Operator Mode*. If this protection is not desired, do not enter a password.

PASSWORD PROTECTION

In a typical production environment, it can be useful to limit password access to those people who are authorized or qualified to make changes to the job specifications.

The *Operator Mode* selection is stored both in the programmer itself and in the BP software. As a result, even if a different PC is attached, that programmer will still be locked in *Operator Mode*. Similarly, the software on any PC that is attached to a programmer locked in *Operator Mode* will become locked also. In either case, the password will be necessary to unlock that programmer and exit the *Operator Mode*.

This feature is very useful in preventing inadvertent or unauthorized alteration of the '.bp' job files. Although not mandatory, a password should be utilized, as this is the most significant security feature of JobMaster.



Figure 52 – JobMaster New Password screen

✎ **JobMaster passwords are case-sensitive.**

To change an existing password, go to *JobMaster/Change Password* pull-down menu option, type in the current password in the *JobMaster Password* field, type the new password in the *New Password* field and re-type for verification in the *Verify JobMaster Password* field. Click the **OK** button when finished.

✍ Please make sure to write your password down and put it in a safe place.

RUNNING A JOB (OPERATOR MODE)

JobMaster is designed to make programming as efficient and foolproof as possible.

Assuming the *Operator Mode* default has been selected, the programmer will “wake up” in *Operator Mode*. If not, refer to the *Locking the Programmer in Operator Mode* section above. *Operator mode* offers only four options: *Program*, *Verify*, *Test* and *Stop*. To begin programming, click on the **Load Job** button or go to *JobMaster/Load Job* pull-down menu option and select the file to load. Click the **OK** button when the file has been highlighted. Once the job is loaded, click on the **Program**, **Test**, or **Verify** button.

Once you have selected a job file, JobMaster will display any Notes that are applicable and the revision history for those notes.

Enter the number of devices to be programmed in the *Copies* field above the device information in the Main Screen.

 This refers to the number of devices that should be successfully programmed, not the number of attempts, to allow for any rejects that may normally occur.

Verify all the options, configuration, file loaded, checksum and any notes. If everything is correct, click on the **Program**, **Verify** or **Test** tabs, and subsequently the appropriate button, depending on what you wish to do.

RETURNING TO NORMAL MODE

To return the BPWin Software to normal mode upon startup, simply go to the *JobMaster/Enter Supervisor Mode* pull-down menu option and enter the *Supervisor* password, and then return to the *JobMaster* pull-down menu and select *JobMaster Configure Option*. Click on the *Normal* radio button for the BPWin Start-up Mode and click the **OK** button. When the BP Software is started up again, it will be in normal access mode.

CHAPTER 6

BPWIN SERIALIZATION

This chapter describes the basic procedures for using the optional Serialization feature available for BP Microsystems' programmers. This feature supports simple serialization: a linear sequence of numbers incremented by one. It also supports complex serialization for users who wish to generate a unique sequence of numbers, including non-linear sequences, by writing their own algorithm.

To use the Serialization feature, you must purchase the Serialization option. If you are using one of BP Microsystems' concurrent programmers, you *may* also need to remove and replace the BIOS EPROMs in each programmer site.

- ✦ Although this feature works with 98% of all EPROMs and Microcontrollers, it DOES NOT work with PLDs in concurrent mode.
- ✦ Serialization does not work in FX4/FX2 mode when using FX4/FX2 socket modules. Contact Customer Service for more details.

For information on obtaining this option and updating the software and hardware, contact BP Microsystems' sales department, contact information provided in *Chapter 7, Troubleshooting*, page 7-1.

- ✦ Under NO circumstances should the contents of the serial.dat file be changed during the execution of a job. The contents of this file should be changed only by the external serialization program.

OPERATION

1. From the *Tools* drop-down menu on the *BPWin Main Screen*, select *Serialization* and press <Enter>.
 2. On the screen you will be given three serialization options: *NONE*, *Simple*, and *Complex*. Select the option you prefer and press <Enter> to accept.
 3. Depending on the option you selected, you will be prompted to provide certain essential information.
- ✦ The serialization program creates a .dat file to be used in conjunction with any serialization program of your choosing. When prompted, it is imperative to specify the exact same location where the serialization program is stored for the .dat file.

None

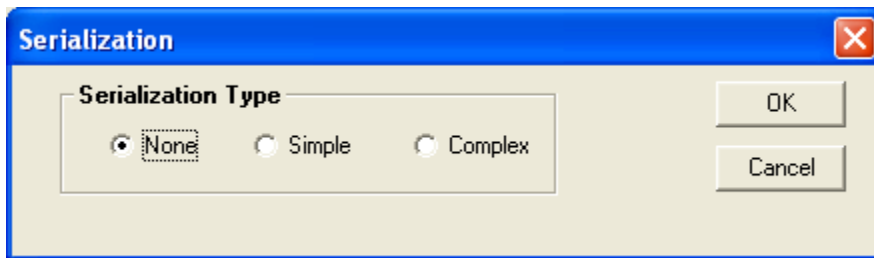


Figure 53 – Serialization type, None

Choosing this option disables serialization completely and requires no additional information.

Simple

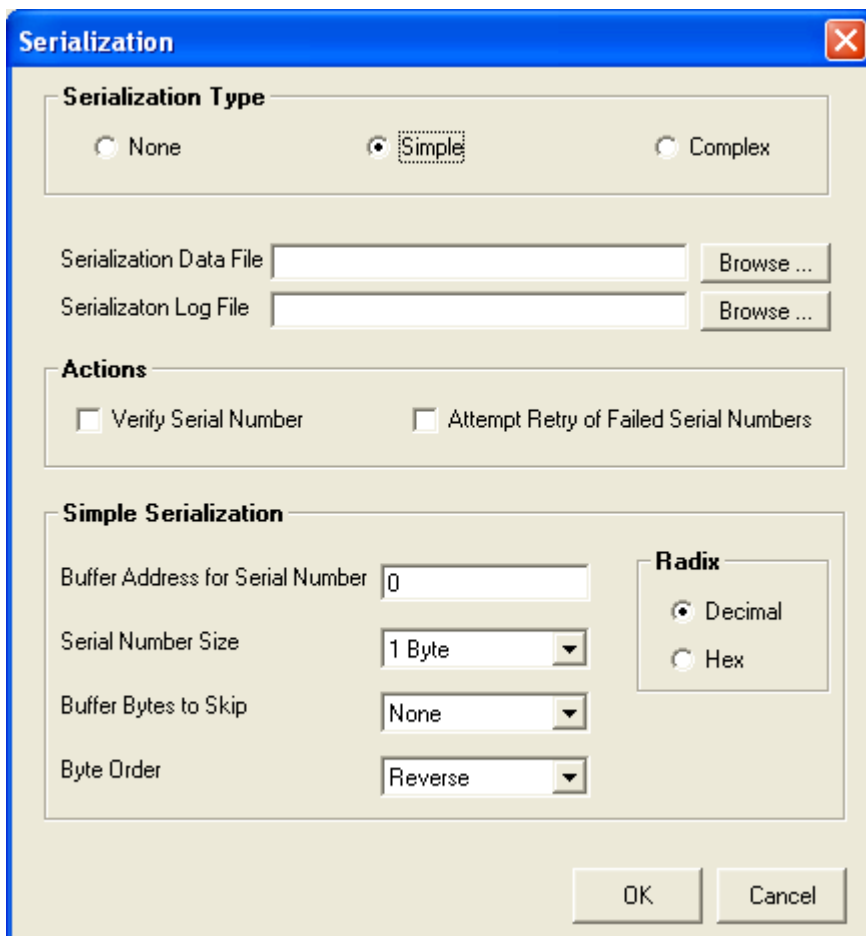


Figure 54 – Serialization type, Simple

This option instructs the BP software to generate a linear sequence of numbers and requires that you input the following information:

- *Serial number data file:*
Enter the path and name for the data file that will contain the current serial number. The BP

software automatically updates this file as each new number is generated. This permits the user to interrupt a job and continue it later with the next number in the series. Now tab to the next selection.

✎ **The *Serial number data file* field should identify the “.dat” file used by the BP software to keep track of numbers for simple serialization. DO NOT enter the name of the “.exe” file used to create numbers in the complex serialization mode! This will cause that file to be over-written and destroyed.**

- *Buffer address for serial number:*
Enter the address at which you wish to place the serial number. The default address is 0 (zero). Now tab to the next selection.
- *Serial number size:*
Select from *NONE*, *1_byte*, *2_bytes* or *4_bytes* and tab to the next selection.
- *Buffer bytes to skip:*
Select from two options, *NONE* and *EVERY_OTHER*. This is useful, for example, when programming paired 8-bit EPROMs that will represent one 16-bit address. The *EVERY_OTHER* option allows all of the serial number to be written into one of the two chips in the 16-bit set. Tab to the next selection.
- *Byte order:*
Select from the options *REVERSE* and *FORWARD*. *Reverse* means the least significant byte (LSB) comes first. *Forward* means the most significant byte (MSB) comes first. Tab to the next selection.
- *Radix:*
This option specifies whether the *Buffer address for serial number:* entered above is to be interpreted as decimal or hex. Select the correct option and tab to the next selection.
- *OK/Cancel*
If you are satisfied with your choices, click **OK**. If you have specified a serial number file that already exists, skip to “Working with an existing file” below.

If a new serial number file is being created, this will bring up a screen, which shows *Serial number file created, starting at Serial Number 1*.

By pressing any key, you will bring up another screen that says *Starting Serial Number: 1*, which is the default starting number for a new file.

1 is the default starting number. Press <Enter> to accept this, or change the starting number and then press <Enter>. Now you can press <Esc>, which will take you back to the main menu, and begin programming.

Working with an existing serial file

If you have specified a serial number file that already exists, neither of the previous screens will appear, and the starting serial number will be whatever was stored in that file.

- **DECIMAL HEX**
Select the format in which the serial number is to be represented, then tab to the next selection.
- **OK/Cancel**
If you are satisfied with the information you have input, click **OK**.

Now you can click **OK** to return to the main menu and begin programming.

Complex

The screenshot shows a Windows-style dialog box titled "Serialization" with a close button (X) in the top right corner. The dialog is divided into several sections. The first section, "Serialization Type", contains three radio buttons: "None", "Simple", and "Complex". The "Complex" radio button is selected. Below this are two text input fields: "Serialization Data File" and "Serializaton Log File" (note the typo), each followed by a "Browse ..." button. The next section, "Actions", contains two checkboxes: "Verify Serial Number" and "Attempt Retry of Failed Serial Numbers", both of which are unchecked. The final section, "Complex Serialization", contains four controls: a "Serialization Program" text field with a "Browse ..." button, a "First Serial Number" text field, a "Last Serial Number" text field, and a "Compatibility Mode" dropdown menu currently set to "OFF". At the bottom right of the dialog are "OK" and "Cancel" buttons.

Figure 55 – Serialization type, Complex

If you choose the *Complex* option, you must first have created a program to generate the unique sequence of numbers desired. This can be a linear sequence such as “increment by five” or a completely non-linear sequence. Details of the protocols involved will be covered in a following section. Assuming this program already exists, here is the procedure for using Complex Serialization:

The screen will prompt you for the following information:

- *Serialization program:*
This field must contain the full path and name of the user’s Serialization program.

- *First Serial Number:*
You must specify the beginning number in the sequence of serial numbers. This information is passed to your Serialization program.
- *Last Serial Number:*
Any number you enter will represent the maximum value desired. If a value is entered in this field, then it is passed as a command-line parameter to the user's serial-number generating program. It is the burden of that program to output the appropriate error condition to the file it generates.
- *Compatibility Mode:*
If ON is selected, the software will read your Serial.dat file in order to determine the starting serial number for the next job. This is useful if you run the same job multiple times and want to continue serializing where you left off. Setting this to OFF will cause the program to start over at the First serial Number as specified on this screen.
- *OK or CANCEL*
If you are satisfied with your choices, select **ACCEPT** and press <Enter>.

ALGORITHM PROTOCOLS

To use the Complex Serialization option, you must first create a program that will generate the unique sequence of serial numbers. The path and name of this program must be specified in the set-up procedure as outlined above. The BPWin software supplies to this program the Command-line Parameters outlined below, and the program supplies the Serial.dat file with data in the form Txx, where "xx" is the appropriate code number, followed by the data in ASCII representation.

 See example serialization program at the end of this chapter.

COMMAND-LINE PARAMETERS

The BPWin Software supplies the user's program with three command-line parameters.

- **-N <Serial Number>**
This is the current serial number, in ASCII representation. This is taken from the Serialization window (Compatibility OFF) or from the Serial.dat file (Compatibility ON).
- **-E <Serial Number>**
This is the ending serial number, in ASCII representation. It is up to the program to specify how this number is to be used. The user may want to stop once a given number is reached. If so, the user's program must specify an Error Message to be generated (see below under "Optional Codes - T06").
- **-F**
This command-line parameter tells the program that this is the first time it's being called. It is only passed to the user's program when it is being called for the first time in a given session. Subsequent calls to the user's program will not pass this parameter.

File Codes

The Serialization program must create a file to pass information to the BP software. This file must be named SERIAL.DAT.

The Serialization program should not specify any path information. This will cause the file to be created in the active directory, where the BPWin software will look for it. The active directory is not necessarily the same directory as the BPWin software.

The SERIAL.DAT file elements each begin with a code. These are in the form of “Txx:” where “xx” represents the appropriate code, followed (with no space) by the data in ASCII representation. These elements inform the BPWin software to perform specific tasks.

Four of these elements are essential, the rest are optional.

Essential Codes

- **T01** Current Serial Number
- **T02** Next Serial Number
- **T03** Translation format number, specifies the file format (See the following chart for supported formats and their codes.)

SUPPORTED FILE TYPES For Serialization	
File Types:	T03-Codes
Absolute Binary	16
ASCII Hex Apostrophe	52
ASCII Hex Comma	53, 58
ASCII Hex Percent	51, 56
ASCII Hex SMS	57
ASCII Hex Space	50, 55
Fairchild Fairbug	80
Formatted Binary	10
Intel Hex-32	99
Intel Intellec 8/MDS	83
Intel MCS-86 Hex Object	88

Intel OMF 286	98
Intel OMF 386	97
JEDEC (full)	91
JEDEC (Kernel)	92
Motorola 32 bit (S3 record)	95
Motorola Exorcisor	82
Motorola Exormax	87
POF	14
Tektronix Hexadecimal	86
Tektronix Hexadecimal Extended	94
Texas Instruments SDSMAC	90
Texas Instruments SDSMAC (320)	04

Supported File Types

- **T04** Specifies the beginning of the actual data that will be outputted to the buffer. The data must be in the format specified by the T03 code. The data will continue from this point until the end of the file. If the data is in a non-addressed format, it will be put at address zero. The user may specify a different beginning address.

Example: Here is an example of a file that the user's program will output as *SERIAL.DAT*, which the BP software will use to manage the serialization process.

Current Serial Number:	T01:D60043C8
Next Serial Number:	T02:D60043C9
Translation Format:	T03:99
Data to Output to Buffer:	T04:
	:020000020000FC
	:020000040000FA
	:0600200000A0D60043C859

OPTIONAL CODES:

- **T05** Message, Fatal error, Abort.
On receiving this code, the BP software will display a message and abort the serialization process. Programming will not continue. The content of the message, in ASCII representation, and the conditions under which this code will be issued must be specified by the user's serialization program.

- **T06** Current Serial Number greater than limit, Message, Abort.
On receiving this code, the BP software will display a message and abort the serialization process. Programming will not continue. The content of the message must be specified in ASCII representation by the user's serialization program. This code should only be issued if the -E command line parameter has been given a value in the serialization set-up *and* that value is exceeded.
 - **T11** Warning Message string.
On receiving this code, the BP software will display a message and will continue the serialization process. The message will be specified by the user after the T11 code, in ASCII representation. The conditions under which this message should be displayed are to be determined by the user.
- ✎ **The serialization program specifies an address range to which the serial number will be written. This must be the same range each time the program is run for a given set. If the program attempts to write to a different address range, an error message will be generated and serialization will be aborted.**

EXAMPLE PROGRAM

- ✎ Under **NO** circumstances should the contents of the serial.dat file be changed during the execution of a job. The contents of this file should be changed **only** by the external serialization program.

The following is a sample serialization number program to be called by the BP software.

```
//*****
//*****
// THIS SOURCE CODE IS PROVIDED AS-IS WITH NO WARRANTY.
//*****
//*****
//Serialization program for Jackson Tan with KETECA
//Generates four 6 digit MAC addresses in a single serialization
file at addresses
//0x23, 0x29, 0x2f, 0x35
//Assigned:      1/26/2000
//Completed: 1/26/2000

#include <stdlib.h>
#include <stdio.h>
#include <string.h>          //For strlen()

// Macro definitions

//The name of the serial file
#define SERIALFILE          "serial.dat"

//The starting address for this specific device
#define DEFAULT_ADDRESS1    (0x0023)
#define DEFAULT_ADDRESS2    (0x0029)
#define DEFAULT_ADDRESS3    (0x002f)
#define DEFAULT_ADDRESS4    (0x0035)

//Amount the serial number will increase every time the program
is run.
#define DEFAULT_INCR        (4)

/*
   PutHexByte()  --  Convert the byte to hex and write it to the
file
*/
static void PutHexByte(FILE *fp, int c)
{
    static char hex[17]="0123456789ABCDEF";

    c=c&255;
    fputc(hex[c>>4], fp);
    fputc(hex[c&15], fp);
}

/*
   putcr()  --  Put a carriage return at the end of the line

```

```

*/
static void /*neare*/ putcr(FILE *fp)
{
    fputc('\r',fp);
    fputc('\n',fp);
}

/*
    ParseCommandLine()  --  Parse the command line arguments
    to get the serial number information
*/

int ParseCommandLine(FILE *fp, int argc, char **argv, long
*plSerialNum,
    long *plEndSerialNum)
{
    int i, nLength, nSerNumberFound=0, lastCall = 0;

    // Initialize values
    *plSerialNum = 0;

    // Loop through command line arguments and
    // process relevant command line options
    for(i=1; i<argc; i++)
    {
        nLength = strlen(argv[i]);

        if ( (*(argv[i]) == '-') && (nLength > 2) )
        {
            switch( *(argv[i]+1) )
            {
                case 'N':
                    nSerNumberFound = 1;
                    sscanf((argv[i] + 2), "%lx", plSerialNum);
                    break;

                case 'E':
                    sscanf(argv[i] + 2, "%lx", plEndSerialNum);
                    break;

                case 'L':
                    lastCall = 1;
                    break;

                default:
                    break;
            }
        }
    }

    // Return an error if there was no -N<serial number> in the
    command line
    if ( !(nSerNumberFound) )
    {
        fprintf(fp, "T05:Serial number not passed to
serialization program\n");
        return 1;
    }
}

```

```
    }

    if ( lastCall == 1 )
    {
        fprintf(fp, "T06:Current serial number greater than
limit\n");
        return 1;
    }
    return 0;
}

/*
EmitLine() -- Write a line of data to the file
*/
static void EmitLine(FILE *fp, int size, unsigned short addr, int
type,
unsigned char * data)
{
    int i;
    unsigned char csum, sumOfData = 0;
    fputc(':',fp);
    PutHexByte(fp, size);
    PutHexByte(fp, addr>>8);
    PutHexByte(fp, addr);
    PutHexByte(fp, type);
    for (i=0;i<size;i++){
        PutHexByte(fp, data[i]);
        sumOfData += data[i];
    }

    csum = -(size + addr + type + sumOfData);
    PutHexByte(fp, csum);
    putcr(fp);
}

/*
EmitDataRecord() - Writes the line containing the serial
number to the file
*/
static void EmitDataRecord(FILE *fp, unsigned short addr, int
type, unsigned char * serialNum)
{
    unsigned char largeCsum = 0; //The checksum of the entire record

    unsigned char sumOfData = 0; //The sum of the contents of serialNum

    int backWards; //For controlling for loop structures and
indexing arrays

    fputc(':',fp); //Print the beginning colon signifying the
start of a record

    PutHexByte(fp, 3); //Print the byte count
```

```
PutHexByte(fp, addr>>8);      //Print the first byte of the address
PutHexByte(fp, addr);         //Print the second byte of the address
PutHexByte(fp, type);         //Print the Record Type

//Loop through serialNum, printing its contents to the file
for (backWards = 2; backWards >= 0; backWards--){

    //Print the contents of serialNum
    PutHexByte(fp, serialNum[backWards]);

    //Calculate the sum of the contents of serialNum
    sumOfData += serialNum[backWards];
}

//Calculate the checksum of the entire record
largeCSum = -((3 + (addr>>8) + (addr&255) + type +
sumOfData)&255);

//Print the checksum of the entire record
PutHexByte(fp, largeCSum);

//Place a carriage return/line feed at the end of the record
putcrl(fp);
}

/*
WriteDataInFormat() -- Write the formatted data into the
file
*/

void WriteDataInFormat(FILE *fp, long szData, int nDataLen)
{
    int i = 0;
    int nIndex = 0;
    int default_data = 0;

    //Print the Data records containing the serial numbers
    EmitDataRecord(fp, (unsigned short)(DEFAULT_ADDRESS1&65535),
0, (unsigned char *)&szData);

    szData += 1;

    EmitDataRecord(fp, (unsigned short)(DEFAULT_ADDRESS2&65535),
0, (unsigned char *)&szData);

    szData += 1;

    EmitDataRecord(fp, (unsigned short)(DEFAULT_ADDRESS3&65535),
0, (unsigned char *)&szData);

    szData += 1;

    EmitDataRecord(fp, (unsigned short)(DEFAULT_ADDRESS4&65535),
0, (unsigned char *)&szData);
```

```
EmitLine(fp, 0, 0, 1, NULL); // end record
}

/*
GenSerialFile() -- Generate the serial number file
*/
void GenSerialFile(FILE *fp, long lSerialNum,
                   long lNextSerialNum, long lEndSerNum)
{
    fprintf(fp, "T01:%lx\n", lSerialNum);
    fprintf(fp, "T02:%lx\n", lNextSerialNum);
    fprintf(fp, "T03:99\n");
    fprintf(fp, "T04:\n");

    // Write the formatted data portion
    WriteDataInFormat(fp, lSerialNum, sizeof lSerialNum);
}

int main(int argc, char **argv)
{
    int rtn;          // Value returned from ParseCommandLine()
    for error checking

    long lSerialNum, lNextSerialNum, lEndSerNum;
    FILE *fpSerial=NULL;

    // Create the serial.dat file
    fpSerial = fopen(SERIALFILE, "w");
    if (fpSerial == NULL)
    {
        printf("Unable to create %s file.\n", SERIALFILE);
        return 1;
    }

    // Parse the command line arguments
    rtn = ParseCommandLine(fpSerial, argc, argv, &lSerialNum,
&lEndSerNum);

    // if there was no error with the command line arguments
    // calculate next serial number and generate serial number
file
    if (rtn == 0)
    {
        lNextSerialNum = lSerialNum + DEFAULT_INCR;

        if(lSerialNum > lEndSerNum)
        {
            fprintf(fpSerial, "T06:Current serial number
greater than limit\n");
        }
        else
        {
            GenSerialFile(fpSerial, lSerialNum,
lNextSerialNum, lEndSerNum);
        }
    }
}
```

```
// Close the serialization file
fclose(fpSerial);

return 0;
}
```


CHAPTER 7

TROUBLESHOOTING

The information in this chapter may help you solve or identify a problem with your software and/or programmer. If you have a problem that you cannot solve, please call us. We are dedicated to making BP Microsystems software and programmers as trouble-free as possible.

HOW TO REACH US

CUSTOMER SERVICE

Inside the US	800-225-2102
Outside the US	713-688-4600
Fax	713-688-0920
Email	tech@bpmicro.com
Web Page	www.bpmicro.com (Internet)

SALES SUPPORT

Inside/Outside US	713-688-4600
Fax	713-688-0920
Email	info@bpmicro.com

CALLING CUSTOMER SERVICE

You can obtain Customer Service from BP Microsystems anytime that you are experiencing a problem that you cannot solve. We request that you have the following information ready when you contact us:

- The model number of the BP Microsystems programmer (title bar of secondary screen).
- Your software version number (from the top of the main screen).
- The exact error message and error number you received.
- The exact algorithm that was selected.
- The exact part number on the chip you were trying to program.
- The command you executed.
- The results of running the self-test command on your programmer (*BP Microsystems Diagnostics*).

It is also useful to have a print-screen of the error. You may be asked to upload your file and/or send in your devices so we can analyze the error at the factory.

If you need to return your programmer to BP Microsystems for any reason, you must call and get a Return Material Authorization (RMA) number before shipping; mark the RMA number clearly on the shipping container. Be sure to include a description of the problem experienced, a return address, contact person and a phone number.

TESTING THE HARDWARE

The programmer can test its hardware quite extensively. The self-test routine can detect problems in the pin-drivers, power supply, microprocessor, data cable, printer port, and several other circuits. The hardware test cannot detect problems resulting from a dirty socket. To execute the test:

- *Remove any chip(s) from the programmer site(s).*
- *Choose **BP Microsystems Diagnostics** from the list of devices found in the Select Device window.*
- *Watch the screen and the programmer LED lights for any error messages.*

 *For more in depth instructions on running the self-test, see page 2-18.*

⚡ If you receive an error during the test, please call Customer Service for assistance.

SOFTWARE UPDATES

The control software for your programmer is updated on a frequent basis (typically every six weeks) to add features and provide you with support for new chips.

Software updates may be obtained from BP Microsystems as a subscription for manual multi-site concurrent and automated systems. Please contact a sales person to purchase a software subscription.

Software upgrades may be obtained through our Internet address, but depending on the type of programmer you have (engineer, production or automated), upgrades and renewals may need to be made by contacting our sales department.

If you decline the software/hardware upgrade and your software support runs out, you will not be able to use the latest software to program devices. The software will run in demo mode, but it will not allow any device operations.

Your programmer is designed to be highly flexible and programmable, allowing it to program a wide variety of chips. Consequently, when a problem does arise, it can usually be fixed with just a software update.

We recommend that you obtain the latest software revision before calling our support line with a software problem. Many of the calls to Customer Service result in the user obtaining the latest version of the software.

FAQ's

How reliable is the USB bus itself?

USB 2.0 is extremely reliable. All data transactions are integrity checked using the very comprehensive CRC32 calculation. Additionally, the USB hardware will retry any packets that fail the integrity check during transfer. This makes the USB bus extremely robust in that it can withstand transient events in the environment (ESD, EMI and other electrical noise, and faulty connection) without breaking the flow in other hardware and software systems. USB also has strict requirements on cabling, PCB routing, and connectors. All of this is done to provide a very controlled electrical environment.

What if my computer does not have a USB 2.0 port?

Then your USB 1.1 port will work just fine. It will perform any transactions slower. You can purchase a USB 2.0 expansion card from your local computer distributor.

In the DOS version, the configuration settings were saved to a file. Where did they go?

With the introduction of the BPWin software, the configuration settings are no longer saved to a file. These settings are now being saved in the Windows 98/Me/2000/XP Registry and are not deleted upon being uninstalled. This allows the user to continue to use previous settings when upgrading to a new version of the software.

Please visit www.bpmicro.com for more information on the latest FAQ's!

ERRORS WHILE PROGRAMMING

If you experience problems while trying to program a chip, try to narrow down the problem. If you receive a **Cannot program** or **Cannot erase** error while programming:

Make sure you have selected the proper programming algorithm

If you are using a PLD or microcontroller, the device may have been previously secured

The device may not be fully erased (this should be verified by confirming the **Blank Check** option has been chosen in the *Program* file tab)

The device may have a newer die than the one supported by the programming algorithm

You may want to run the **Compare** command from the *Verify* file tab to see where the command failed. Look at the first error indicated by the **Compare** command. This is the location that caused the **Program** command to abort and return an error message. Three possibilities exist:

- The byte or bit didn't program at all.
- There are extra programmed bits.
- A combination of unprogrammed bits and extra programmed bits. On memory devices, compare the two hex data values to see what went wrong. On most EPROMs, the erased state is FFh and the fully programmed state is 00h.

If the expected value has more 0 bits than the value read from the chip, the chip may not have been fully erased before programming. If the byte read from the chip is completely unprogrammed (FFh), you are probably either using the wrong algorithm or have a defective chip. If you find that the same bit fails every byte, you should suspect a dirty socket or a bent IC pin.

You can expect a small number of fuse-link programmable PLDs to fail during programming. This is because they cannot be fully tested at the factory without blowing out the fuses! Most semiconductor manufacturers claim you should get a 98% programming yield. Your yield shouldn't drop below the value specified by the semiconductor manufacturer.

Most semiconductor manufacturers fully test EPROM and EEPROM-based PLDs. You should achieve a very high yield on these parts. These parts do, however, have a limited number of erase cycles. Most parts can be erased and programmed up to 100 times without failure.

CLEANING A DIRTY DIP SOCKET

If the DIP socket becomes dirty, it will fail to make contact with all the chip pins. The simple fix is to place your chip in the socket, push the lever down, and slide the chip left and right a few times. Cleaning sockets with a blast of high-pressure air on a regular basis is also recommended.

If this does not resolve the problem, run the hardware self-test described in *Chapter 2 ,Getting Started*, page 2-4. If your hardware has passed the self-test, there may be an error in the programming algorithm you are using or the semiconductor manufacturer may have updated the programming algorithm for your device. In either event, your problem will probably be corrected with a software update.

ERROR MESSAGES

Error messages are usually generated by the device programmer. The first step to take when an error message has been received is to refer to the appropriate device programmer manual's Troubleshooting section. The following list of error messages could originate from the software.

Version Mismatch - BPEng.dll and BPAIg.db versions mismatched.

The engine and Algorithm database versions do not match. Accepting this error message will prompt an exit from the software. Please contact Customer Service for further assistance.

BPPgmr.ocx - BPPgmr.ocx internal error

When the unique coding sequence attached to both the BPPgmr.ocx and the BPEng.dll do not match, this error will be generated. Accepting this error message will prompt an exit from the software. Please contact Customer Service for further assistance.

Algo Database Error – Error loading BPAIg.db

This is a 'catch all' error code. If any thing not listed above proves to cause an error while loading the Algorithm Database, this error will be generated. Accepting this error message will prompt an exit from the software. Please contact Customer Service for further assistance.

BPntIoDII – No parallel port is available.

When using BPWin NT 4.0, an error is generated if the device drivers have not been installed. The software attempts to establish communication with the programmer through the driver. When the driver is not found, it fails the software. In order to remedy this, you will need to install the device drivers for the software. Please refer to section *FAQ*, page 7-3, for procedures.

Error 3: Cannot reset hardware.

The software cannot establish communications with the programmer. Here are some suggestions:

Be sure the programmer has proper power and that the green PASS LED is on. Since the programmer will do an automatic Power On Self Test (POST) upon startup, it could be that the programmer failed the test and has signaled the software into the default DEMO mode. If this should happen contact Customer Service, see page **Error! Bookmark not defined.**

Make sure the cable from the programmer to the computer is properly connected to a parallel printer port. If you are using a ribbon cable, this is probably the problem (ribbon cable connectors are designed for use inside a chassis where the cable is not flexed). You should use a shielded 25-conductor cable (not an RS-232 cable).

Your LPT port could be the problem. If you have multiple parallel ports, you may have the ports configured incorrectly; that is, two at the same address. Some laptops have the ability to disable the port. If you have one, make sure the LPT port is enabled.

Another program may be interfering with the port such as a print cache. When running under Windows, you increase the potential of another program trying to access the same parallel port and changing the expected status at the port.

If you have a hardware lock key between the programmer and the port, try removing it.

Last but not least, the programmer may be damaged. Try another computer and/or parallel port and see if it works there. See *Calling the Customer Service Hotline*, page 7-1.

Error 5: Hardware time-out.

This error message is generated when the software is waiting on a response from the programmer while executing a command and the programmer does not respond within the expected amount of time. This error may result from several causes. You may be experiencing communication errors (see “Error 3: Cannot reset hardware” above). There may be a bug in the software for this particular algorithm (see *Error 10: Error in programming algorithm* below. See also *Self-Test*, page 2-18.

Error 6: Wrong model number.

See “Error 3: Cannot reset hardware” above for possible causes.

Error 8: LPTx: is not a functioning port.

The parallel port LPTx (where x=1, 2, or 3) that is selected with the *Configure* command does not exist in your computer, is not functional, or has a bad cable connected to it.

Error 9: Programmer execution error.

The programmer failed an internal consistency check. See “Error 3: Cannot reset hardware”, page 7-4 and “Error 5: Hardware time-out”, page 7-5 for possible causes.

Error 10: Error in programming algorithm. Please call Customer Service.

The software has detected an internal error. You should contact BP Microsystems to report the error. You may need to obtain a software update. See *Calling the Customer Service Hotline*, page 7-1.

Error 11: There is no data in the buffer. You must load a file or read a chip.

A command tried to read data from the buffer to program or verify a chip, but nothing has been loaded into the buffer yet or the buffer was recently cleared.

Error 14: There is no chip in the programmer site.

Be certain that your chip is inserted correctly. If the chip was inserted correctly, remove it and run the hardware self-test to be sure your programmer is functioning correctly (*BP Microsystems Diagnostics*). A defective chip may cause this error. When using an autohandler, the contactor may not have closed or the connection between the programmer and the contactor may be disconnected.

Error 15: The chip is not inserted in the programmer site correctly.

The continuity test determined that the chip in the programmer site does not have continuity on all the proper pins. You should examine these pins carefully. Possible causes are:

- A bent pin.
- The chip is not in the proper position in the programmer site.
- The chip has a different number of pins than the chip selected.
- The algorithm selected has a ‘*’, indicating it requires an adapter, but you did not use the adapter, or vice-versa.
- The socket is dirty and not making a connection. See *Cleaning a dirty DIP socket*, page 7-4.
- The wrong socket module or adapter is being used for this part.
- The device may be a very low power device that is not properly detected by our continuity methodology. If so, please let us know.

✍ **It’s not easy to get continuity on an LCC device in a PLCC socket. If you are trying to do that, then you may need to add a spacer between the chip and the lid in order to apply the proper force to the device pins. The best solution is to purchase an LCC socket module that does not require any such jury rigging. Note also, LCC devices will not work at all in the autoject sockets designed strictly for PLCC devices.**

Error 16: The chip is inserted backwards.

The chip has passed the continuity tests, but appears to have the GND and Vcc pins improperly placed in the socket. If the DIP, SOIC, or TSOP device is not actually inserted backwards and the LCC, PLCC, or QFP is not accidentally rotated, then the device is probably defective. Try a known good device.

Error 17: Out of base memory. You should have at least 200K free.

Your computer's configuration does not have enough RAM available to run the software. You should have 640K RAM installed with at least 200K available for program execution. Memory resident programs, such as network drivers, may reduce the RAM available to the programmer, so you may need to remove these programs from your CONFIG.SYS and AUTOEXEC.BAT files. If you are using DOS 5.00, you can specify that DOS be loaded into high memory, saving base memory for BP software. See your Microsoft Windows manual for details.

Error 18: Temporary file error.

Our software's virtual memory manager is trying to store data that is currently not needed in RAM to the disk. The program was unable to create a temporary file or the disk is full. You should make sure you have plenty of disk space (the larger the data files, the larger the requirement for temporary disk space). The program does take advantage of EMS memory if you have an expanded memory manager installed. This is much faster than using the disk for temporary swap space.

Error 21: Cannot program.

Not able to program the device in the programmer site. See *Errors while Programming*, page 7-3.

Error 22: Cannot erase.

Not able to erase the device in the programmer site. See *Errors while Programming*, page 7-3.

Error 23: Invalid electronic signature in chip (device ID).

The chip may be damaged or the chip manufacturer may have changed the programming algorithm without notifying us. See "Error 25: Invalid electronic signature in chip (manufacturer ID)" below.

Error 24: Invalid electronic signature in chip (algorithm ID).

The chip may be damaged or the chip manufacturer may have changed the programming algorithm without notifying us. See "Error 25: Invalid electronic signature in chip (manufacturer ID)" below.

Error 25: Invalid electronic signature in chip (manufacturer ID).

Many of the new EEPROM based PLDs (such as all the GALs) have electronic identifiers that specify the manufacturer, the device code, and the proper programming parameters. The most common cause of this error is when you have selected the wrong manufacturer for the particular part you are using (e.g., you may have a National Semiconductor part in the programmer site and a Lattice part selected). It is also possible that your chip has a newer ID than your software revision supports. See *Software updates*, page 7-2.

Error 26: Device is not blank.

The *Device/Blank* command was executed or the “*Blank check before programming*.” option was enabled in the *Device/Options* dialog box and the device in the programmer site is determined to have programmed data. Possible causes are:

The part needs to be erased longer.

The device was previously programmed and cannot be erased (OTP EPROMs and fuse-link PLDs).

The wrong algorithm was used.

Error 27: Device is not secured.

An attempt to secure a device was made, but it failed. See *Errors while Programming*, page 7-3.

Error 28: Data in chip does not match buffer.

A verify operation was performed to see if the chip and the buffer have the same data, but there is a difference in the two patterns. The verify may have been performed via the *Device/Verify* command or the *Device/Program* command when the “*Verify after programming*” was enabled under the *Device/Options* dialog box. The verify will not show where the difference occurred but the *Device/Compare* command will show you all the discrepancies. Here are some reasons why a device may fail to verify:

- The part was secured.
- The part was programmed with a different pattern.
- The part has not yet been programmed.
- The device was incorrectly programmed by a different system.
- The wrong algorithm is selected.
- The chip was not properly erased before programming (this would be caught by the blank check if it was enabled).
- The wrong adapter is being used.
- The device is not inserted correctly and you have the continuity test turned off or you ignored its error messages.
- The device is defective.

Error 31: Database file is invalid. The .EXE file is corrupted.

The .EXE file you are executing has been corrupted. You should get a new copy from BP Microsystems. See *Calling the Customer Service Hotline*, page 7-1.

Error 32: Sorry, algorithm not found. Please call Customer Service.

The .EXE file you are executing has been corrupted. You should get a new copy from BP Microsystems. See *Calling the Customer Service Hotline*, page 7-1.

Error 33: You must reselect the chip you want to program.

The device was selected before establishing communications with the programmers, perhaps prior to turning on the programmer or before switching to a different programmer. Simply reselect the chip and the error should not re-occur.

Error 36: You must properly install the correct socket module.

The software interrogates the socket module before each operation to determine the correct mapping for the algorithm selected. You will get this error if:

- There is no socket module installed.
- The socket module installed does not support the device you have selected (*e.g.*, you have selected a 20 pin device and you have a 28 pin PLCC socket module attached).
- The socket module installed is not supported by the version of the software you are using. Use the latest version.
- The pinout has not yet been defined for this package type. It may be an oversight on our part. If so, please call Customer Service and inform us of this problem.

Error 39: Device already secured.

The device cannot be legitimately programmed, read, etc., because it has been secured. If it is a PLD it may still be functionally tested with the **Test** command under the *Test* file tab.

Error 40: No test vectors present.

The file you loaded did not contain any test vectors. Therefore, the **Test** command will not be executed. You should set the **Test** “Vector test after verify” to **NONE**.

Error 44: Internal error. Please call Customer Service.

The software detected an internal inconsistency. This may be caused by the computer not performing correctly.

Error 46: AFS software required to execute this function.

This is a function that is available to users that have purchased the Advanced Feature Software only. In order to use the chosen function you must buy the AFS upgrade. See *Calling the Customer Service Line*, page 7-1.

Error 47: Self-test failed. This unit may need service. Please call Customer Service.

The self-test (**BP Microsystems Diagnostics**) has detected a hardware problem. The unit may need to be returned for repair. Note the exact error message and see *Calling the Customer Service Line*, page 7-1.

Error 48: Cannot Unprotect.

An attempt was made to unprotect a sector and failed. See *Errors while Programming*, page 7-3.

Error 49: Cannot Protect.

An attempt was made to protect a sector and failed. See *Errors while Programming*, page 7-3.

Error 57: You must purchase support for this device to use it.

The device that you selected is not supported in the default device set for this programmer. Call BP Microsystems Sales line to purchase an upgrade code for your programmer.

Error 70: The buffer data cannot be used to program this device.

You loaded a file type that is not a valid option for the currently selected device. Re-select the device and load the buffer again. If the error persists, call BP Microsystems Customer Service.

WARNING MESSAGES

Specific warning messages will pop up wherever applicable when choosing a device. These messages, pertaining to the type of chip being chosen, are designed to assist you with helpful information such as if the chip needs a third party adapter or requires the use of a special head, when the chip needs to be placed a certain way in the socket module, etc.

The warnings listed below are generalized warning messages to alert you of a problem that should be resolved before continuing.

Warning: Device code is not correct

The programmer automatically checks the electronic identifier of your memory chip. If your chip does not read out the identifier expected by the software, there are two possible actions may be taken by the programmer. If you have “*Check electronic identifiers:*” set to AUTO-SELECT and your chip reads out a known identifier, the software will automatically select the new algorithm for your chip. If you have it set to ENABLE, you will receive an “*Abort, Retry, Ignore or Select?*” query. The **Select** option will try to select the appropriate algorithm. Use it if you feel the wrong algorithm may have been selected in the first place. You can replace the chip and use **Retry** or you can abort the command. Finally, you can ignore the identifier and proceed.

Occasionally a semiconductor manufacturer will change the die on a chip to add the identifier. If this happens, the software may expect the identifier to be present in your chip. If you are using an old chip made before the identifier was added, you will get an error message. It is safe in this case to ignore the warning message. You can use the *Device/Options* command to disable the identifier altogether if you want to.

Warning: Device is not blank

You will get this warning when using the *Device/Program* command with the “*Blank check before programming*” operation enabled in the *Device/Options* dialog box. You are given the option to “*Abort, Retry, or Ignore*”, which allows you to try another chip before proceeding or to go ahead and program data on top of the data that may already be in the chip.

Warning: Device has been secured

You will get this message only on devices that have the ability to read the security bit prior to performing any other operation. Thus, you may get this when trying to read some PLDs and some microcontrollers. You are given the option to “*Abort, Retry, or Ignore*”, which allows you to try another chip before proceeding or to go ahead and read the erroneous data from the chip.

Warning: X fuses in buffer and Y fuses in chip

You will get this message when you try to program a PLD that has a different number of fuses than the number of fuses loaded in the buffer. This is a precautionary measure designed to help you keep from accidentally programming the wrong JEDEC file into a chip.

This message may not always be the best advice. For example, if you have a GAL22V10 (5892 fuses) selected and a PAL22V10 (5828 fuses) JEDEC file loaded. It is okay to program this JEDEC file into the GAL22V10 because they are 100% architecturally compatible; however, the GAL22V10 has 64 additional fuses allocated as the UES (see *Device/UES* command for more details). If you ignore this message on the first operation, you will not receive any more warnings on the subsequent programming operations.

GLOSSARY

This glossary has been included for convenience and to assist in grasping an overall understanding for the terminology and jargon used within the software and hardware engineering fields. To be as inclusive as possible, many of the terms found herein are not located in the manual, but are provided in order to supply definitions to common engineering acronyms and words.

DEFINITIONS

A

Abel	Advanced Boolean Expression Language. An early hardware description language developed for PLD-based designs.
Accelerator	A specialized piece of hardware that speeds up a software-based task. Usually used for speeding up simulation.
Algorithm	A recipe for performing an operation such as computing an average value.
Analog	A continuously varying signal. For example, if an analog signal's range is from 0 to 5 volts, the signal can assume any voltage within that range such as 1.2, 2.4 or 4.7 volts. Microprocessors and microcontrollers usually cannot process analog signals directly and require conversion by an A/D converter before they can process the signal.
Analog Simulation	Modeling or simulation of an electronic circuit using representations of the actual circuit voltages, currents, and component values instead of simplified digital state representations.
Analog-to-Digital Converter	A/D, ADC. An electronic circuit that converts a continuously varying signal (temperature, pressure, voltage, etc.) into digital zeroes and ones that can be processed by a microprocessor or microcontroller.
Analyzer, Logic	An instrument that allows you to observe the behavior of digital signals in an embedded system.
ASIC	Application-Specific Integrated Circuit. A custom integrated circuit designed specifically for one end product or a closely related family of end products.

ASIC Emulation	Also Logic Emulation. The use of programmable circuits usually based on FPGAs, to emulate the design of an ASIC or an IC before it is built. ASIC emulation allows designers to check the operation of a design before committing the time and money required to fabricate the IC. Emulation serves the same purpose as simulation design verification but is much faster because it is based on hardware rather than software.
Autohandler	A machine that removes ICs from their shipping tubes, connects them to the programmer to be programmed or tested, and places them back in tubes. It may also have a marker that will label the devices. The most common autohandlers are manufactured by Exatron, MCT, and Quality Automation.

B

BGA	Ball Grid Array. A surface mount device with solder balls and a high pin count, similar to PGA.
BiCMOS	Bipolar, Complementary-Symmetry Metal Oxide Semiconductor. An integrated circuit fabrication technology that combines the two major IC technologies, bipolar and CMOS, on one IC.
Bidirectional	A signal or port that can act as either an input to or an output from an electronic circuit.
Binary	The base-2 number system almost universally used by modern computers, microprocessors, and microcontrollers.
Bipolar	The original semiconductor manufacturing process technology. Usually characterized by high-speed, high-power operation.
Bipolar PROM	A fuse-link programmable PROM.
Bit	Contraction of Binary digiT. One digit in the base-2 numbering system used by virtually all modern computers, microprocessors, and microcontrollers. A bit can have a value of either zero or one.
Blank Check	A test performed by a device programmer to ascertain whether a device has been programmed (partial or total) or is in a virgin state.
Block Diagram	A graphical representation of a system using a very high level of abstraction.
Board, Circuit	Also PC Board. A thin card, usually made from fiberglass or plastic, which is covered with copper lines and is used to hold the various integrated circuits in an embedded system.
Bond-Out Chip	A special version of a microprocessor or microcontroller which brings critical internal signals from inside the chip out on special package pins so that developers can more easily observe what's happening inside the processor. Usually used to build In-Circuit Emulators (ICEs).

Boolean Algebra	A mathematical system developed in the 1800s to express the philosophical logic of Aristotle, which was coincidentally ideal for the description of digital circuits 100 years later.
Breadboard	A hand-made system prototype built as a proof of concept. In the early days of electronics (even before transistors were invented), engineers actually mounted circuit components to blocks of wood; hence the term “breadboard”.
Buffer	An isolation circuit used to insulate sensitive analog or digital circuits from higher-power or higher-current levels in other portions of an electronic design. Often seen, for example, as an I/O buffer that separates the sensitive circuits inside of an IC from the signals on the circuit board to which the IC is attached. Data storage unit directly stored on CPU.
Bus	A group of two or more signals that carry closely associated signals in an electronic design.
Byte	A binary word consisting of eight bits. When used to store a numeric value, a byte can represent a number from 0 to 255.

C

CAD	Computer Aided Design. The overarching generic term for all software tools that enable or aid in the creation of engineered systems. Sometimes, CAD refers only to the electronic versions of mechanical drafting tools. Sometimes, it refers to all such tools including EDA tools.
CAE	Computer Aided Engineering. The original term for electronic design automation (EDA). Now, often refers to the software tools used to develop the manufacturing tooling for the production of electronic systems such as for the panelization of circuit boards.
CFI	Complex-Instruction-Set Computer. A design approach for microprocessors and microcontrollers that employs relatively complex instructions that execute over multiple clock cycles. A program written using CISC instructions requires fewer such instructions to perform a task as compared to a program written using RISC (Reduced-Instruction-Set Computer) instructions.
Checksum	A number that results by adding up every element of a pattern. Typically either a four or eight digit hex number, it is a quick way to identify a pattern, since it is very unlikely that two patterns will have the same checksum.
Clock	A master timing signal that sets the operating pace for all other components in the embedded system.

Clock Skew	Variation from the ideal clock timing across an entire electronic design (usually in an IC) caused by parasitic elements. Seymour Cray was an early combatant of clock skew and had to design serpentine traces on the Cray I supercomputer's circuit boards to compensate for clock skew.
Clock Tree	A tree-like configuration of circuitry designed to minimize the effects of clock skew.
CMOS	Complementary Symmetry Metal Oxide Semiconductor. An IC process technology developed in the 1960s that typically runs at lower power than bipolar circuitry. Early on, CMOS was much slower than bipolar but has steadily gained in speed over the decades to rival today's bipolar speeds. Most ICs are now made using CMOS technology.
Co-design	See Hardware/Software Co-design.
Compare	Reading a programmable device and displaying any discrepancies from the desired pattern. Each error is displayed on the screen. This comparison is slower to perform than a verify on the programmer.
Compiler	A computer program that translates programs written in a high-level language (HLL) into assembly-language instructions or machine code.
Complex PLD	These devices are the big brothers of PALs. Their architecture grew out of the familiar sum-of-products design that is common to PALs, but they may have as many as 84 pins, or more, buried logic that is not connected to the outputs, and more complex interconnection schemes.
Concurrency	The ability of an electronic circuit to do several (or at least two) different things at the same time. Contrast with computer programs, which usually execute only one instruction at a time unless the program is running on a processor with multiple concurrent execution units.
Concurrent Design	The ability to develop many parts of a complex electronic design in tandem using EDA tools such as simulation to stand in for portions of the system yet to be designed fully.
Concurrent Programmer	A multiple-socket programmer that starts programming each device as soon as it is inserted in a socket, without all sockets having to be filled.
Controller	An electronic system that directs the operation of some larger system.
Core	A predesigned block of logic employed as a building block for ASIC design.
Co-Simulation	Simulation of hardware and software together, simultaneously.
Coverage	A measure of the 'goodness' of a test or test suite. Usually refers to fault coverage and is expressed as a percentage of the circuit covered by the test. Usually, it is too expensive to achieve 100% coverage and test engineers shoot for coverage in the high 90's. Scan-test technology can

improve coverage results, at the expense of additional silicon on the chip and some additional design time.

CPLD	Complex Programmable Logic Device. A programmable IC that is more complex than the original Programmable Logic Devices such as AMD's (originally MMI's) PALs but somewhat less complex than Field Programmable Logic Arrays.
CPU	Central Processing Unit. The core circuitry of a computer including the ALU (arithmetic logic unit), address-control circuitry, and bus-control circuitry. Usually implemented with a microprocessor or microcontroller in an embedded system.
Crosstalk	A condition where signal activity on one wire in an electronic circuit couples to another wire and causes noise through electrostatic (capacitive) or electromagnetic (inductive) coupling.
CUPL	A hardware description language originally developed for PLDs.
Cyberoptic	Computer generated; connection for light-pulse information to be transmitted

D

Debugging	The art of finding and eliminating errors in system designs.
Design Capture	Design Entry. The process of entering an electronic system design into a computer using EDA tools.
Design Error	A flaw designed into an electronic circuit, which is then faithfully reproduced in every manufactured system (as opposed to a manufacturing error that is a flaw created by the manufacturing process itself). Emulation, simulation, and design-rule-checking tools all help to minimize or eliminate design errors.
Design Rule Check	Verification of an IC or PC board layout for conformance to the physical or electrical limitations of the implementation technology in use.
Device	Microchip or Integrated Circuit chip.
DFT	Design for Test. A design methodology that includes special attention to the design of a circuit and the addition of special circuitry that eases the testing of that design.
Dialog Box	The method used by the device programmer's user-interface software to allow the user to select options and specify information. The user can specify any options and fill any blanks in the box then press <Enter> to force the software to process the information.
Die	The silicon chip that is located within an IC package. It is a small rectangular flat piece of silicon that has been fabricated with many transistors to perform a specific function. It is glued into a plastic or

ceramic package and connected to the external metal interconnect pins of the IC with very small bonding wires. It can be seen through the window of erasable EPROMs.

Digital

An approach to circuit design based on the binary number system. Signals in digital circuitry can only assume well-defined levels; intermediate levels are invalid. For example, in a digital system with a signal range of 0 to 5 volts, the digital signal may have the logical value of 0 if the signal voltage is within the range of 0 to 0.5 volts and a logical value of 1 if the signal voltage is within the range of 2 to 5 volts. Signal voltages of between 0.5 and 2 volts are invalid and are not allowed.

Digital Simulation

A computer simulation of an electronic circuit that uses simple Boolean or logic states to represent the instantaneous state of the circuit. Because the representation is simplified from the actual voltage and currents present in the circuit, digital simulation is much faster than analog simulation.

Digital-to-Analog Converter

D/A, DAC. A circuit that translates a signal from a numeric, digital representation used by microprocessors and microcontrollers into an analog signal.

DIP

Dual Inline Package. An IC package with two rows of through-hole pins, usually on 0.1 pitch, 0.3 or 0.6 inches apart.

Documentation

All of the paper and electronic documents supplied with a component or system that are absolutely critical to fully utilizing the product. For embedded systems designers and developers, there is never enough documentation.

DRC

See Design Rule Check.

DSP

Digital Signal Processor or Digital Signal Processing. A specialized microprocessor or electronic system designed to be very fast at processing continuous signals such as sound and video.

E

EDA

Electronic Design Automation. A large collection of software tools that enhance and aid in the development of complex electronic systems.

EDIF

Electronic Design Interchange Format. A standard representation format for describing electronic circuits used to allow the interchange of circuit design information between EDA tools.

EEPROM

Electrically Erasable Programmable Read Only Memory. An integrated circuit that stores programs and data in many embedded systems. EEPROM stores retains information even when the power is off. Early EEPROM was expensive on a cost-per-bit basis and was infrequently used. Newer “Flash” EEPROM is much less expensive and its cost-per-bit approaches that of DRAM making Flash EEPROM a very attractive

	memory device for embedded-systems design.
EMI	Electromagnetic Interference. Noise generated by electronic systems, which can interfere with other electronic systems by traveling through the air, over communication wires, and through power wiring.
Emulator	ICE or In-Circuit Emulator. A complex, expensive, and often balky electronic system that simulates the presence of an embedded system's microprocessor or microcontroller. Used often and extensively in the development and debugging of embedded-system programs.
EPROM	Erasable Programmable Read Only Memory/UVEPROM. An integrated circuit that stores programs and data in many embedded systems. EPROM can only be programmed once. To erase an EPROM's contents, it must be exposed to intense ultraviolet light for an extended length of time.
ESDA	Electronic System Design Automation. High-level EDA tools used to design and describe entire electronic systems.
Event	A point in time where a change occurs in the state of an electronic circuit.
Event-Driven Simulator	A simulator that only calculates circuit conditions when events (changes) such as the start of a new clock cycle occur in the state of the system. In contrast, timing simulation computes the state of a system using elapsed time (usually in nanoseconds or picoseconds).

F

Fast Prototype	A working product model built quickly to try out product concepts. May lack the fit, finish, and complete capabilities of the planned final product while still giving users an idea of how the product will work. Often assembled specifically for a conference or other dog-and-pony show.
Fault	An actual problem in an electronic circuit that disables or degrades the performance of the circuit. Also, for EDA purposes, a point in the circuit where a potential flaw could damage the circuit's operation.
Fault Coverage	The percentage of potential faults identified and tested by a test program or suite of test programs. If the tests can uncover all potential faults, the fault coverage for those tests is 100%.
Fault Simulation	Simulation of the operation of an electronic circuit with the introduction of simulated manufacturing faults to determine the amount of fault coverage provided by a set of test vectors. These test vectors are then used to test the actual manufactured circuit so they must be able to identify a large percentage of the possible manufacturing faults.
Finite Element Modeling	A relatively complicated numerical method (computer algorithm) that can model complex electrical phenomena such as electromagnetic wavefront propagation.

Floorplanning	The task of determining where each major block of circuitry will go within an IC design.
FPGA	Field Programmable Gate Array. A very complex PLD. An integrated circuit containing a large number of logic cells or gates that can be programmably configured after the IC has been manufactured. Some FPGAs use fuses for this programming and others store the configuration in an on-chip EEPROM or RAM memory. Fuse-programmed parts cannot be reprogrammed so they can only be configured once. EEPROM-based FPGAs can be erased and reprogrammed so they can be configured many times. RAM-based FPGAs can be reconfigured quickly, even while the circuit is in operation. The FPGA usually has an architecture that comprises a large number of simple logic blocks, a number of input/output pads, and a method to make random connections between the elements.
Framework	A unifying graphical user interface, database format, and inter-tool communication scheme that allows a user to combine EDA tools from various vendors to create a desired tool suite for the design of electronic systems.
Functional Test	A test that is performed following the programming of a PLD. The test operates the device in its normal operating mode by simulating the inputs and outputs that the part will experience in normal operation. To perform the test, the engineer must supply a set of test vectors that describe normal operation of the device so the device programmer can apply the specified stimulus and verify that the device is operating as designed. It is important to perform a functional test on PLDs because, in many cases, the PLD cannot be fully tested at the factory before programming so a defective PLD may program correctly but fail the functional test. A properly designed functional test will verify that the part meets the design specification, ensuring that the device, the compiler, the programmer, and the engineer have all performed their respective tasks correctly.
Fuse	A metal connection within a PLD or memory that may be melted during programming to break the circuit. These links typically carry input signals to logic gates. Burning all the fuses except those that are required in the desired circuit forms the desired circuit configuration. Since the fuses cannot be tested nondestructively, fuse-like programmable devices cannot be 100% tested at the factory and consequently expected programming yields are usually 98-99%.

G

GaAs	Gallium Arsenide. A high-speed IC process technology that does not use silicon. Instead, GaAs uses the semiconductor element Gallium doped with the impurity Arsenic. GaAs process technology currently produces the fastest ICs possible but advanced CMOS processing has greatly reduced the speed gap in the past few years. GaAs is most often used today for very high frequency radio circuits such as the transmitter
-------------	--

circuits in cellular phones.

GAL	Generic Array Logic. EEPROM based second generation PAL devices.
Gang Programmer	A multiple-socket programmer that requires each device to be placed in a socket before any can be programmed. See <i>Concurrent Programmer</i> .
Gate Array	A type of ASIC in which the transistors, gates and other active circuit elements are fixed on a wafer called a “master slice”. The customization for a particular application is done using the metal interconnection layers on the chip. Thus, the IC vendor can fabricate and stockpile master slices well in advance of a customer order and then finish the fabrication by adding the metal layer or layers when the order is received. Because of this style of fabrication, gate arrays are the easiest ASICs to design and offer the fastest turnaround time between order and shipment of the finished parts. In the extreme, Chip Express offers laser-programmed gate arrays with 24-hour turnaround time or less.
Gbyte	Gigabyte. 1,073,741,824 bytes.
GDS II	A photoplotting file format usually employed for integrated circuit mask plotting files. Originally developed by GE Calma, an early EDA vendor.
Gerber Photoplot	Gerber file. A de-facto file format standard originally developed for Gerber Scientific for its line of photoplotters. Usually used for representing printed-circuit board designs.
Ground Bounce	Noise signals coupled into the grounding network of an electronic system that cause a variety of operating problems in the circuit. A phenomenon that limits the testability of high-speed PLDs on some device programmers. The term refers to the voltage on the ground terminal of the PLD’s die rising and falling when many outputs switch simultaneously. This voltage can induce extraneous clock signals that will make a device fail a functional test or reduce programming yield.

H

Hard Macro	A relatively complex block of logic or “core” such as a multiplier or an entire microprocessor that has been completely pre-designed for use on a particular ASIC or FPGA technology. Generally, a hard macro cannot be edited except by the company that created it. In exchange for this relative lack of flexibility, hard macros usually provide better performance using a smaller amount of silicon when compared to a “soft macro” or “synthesizable core”.
Hardware/Software Co-design	The simultaneous development of product hardware and software. This design approach is more difficult than a serial design, which first develops the hardware and then the software that will run on the hardware but the benefit is a reduced time to market. To develop software before hardware is ready, software developers often create a behavioral model of the hardware that can run the software and thus

prove its function.

HDL Hardware Description Language. A synthetic computer-based language used for the formal description of electronic circuits. An HDL can describe a circuit's operation, its design, and a set of tests to verify circuit operation through simulation. The two most popular digital HDLs are VHDL and Verilog. An analog HDL called AHDL is under development by many vendors. HDLs make it easier to develop very large designs through formal software engineering methods that define ways to divide a large team through formal software engineering methods that define ways to divide a large team project into smaller pieces that can be implemented by individual team members.

Hex file A human-readable ASCII file that represents any binary data. Each byte in the binary pattern is represented by two hex characters (0-9, A-F) so that any of the 256 possible bytes, which include both control and unprintable characters, may be printed. The hex file may also contain address or checksum information. The pattern represented by the hex file may be represented by a binary file or any of the hex file formats – any file format may contain any pattern. The names of the hex file formats (Intel, Motorola, Tektronix, etc.) indicate who standardized its format and does not indicate anything about the pattern or the device the pattern is intended for.

Hierarchy A method for describing and modeling an electronic system using different abstraction levels. At the bottom level of the hierarchy is the actual physical layout of the design (a concrete level, not at all abstract). At the top of the hierarchy is a functional description of the system or a block diagram (a very high level of abstraction). Intermediate levels include the register-transfer level (RTL), the gate level, and the transistor level.

HLL High-Level Language. A relatively complex computer programming language that allows the programmer to work at a mathematically abstract level instead of the low, physical level of the microprocessor or microcontroller. For example, instead of dealing directly with registers and memory locations, the HLL programmer works with variables and arrays. Java, C, Pascal, Fortran, and BASIC are all examples of high-level languages.

I

I/O Input/Output. The wide range of circuits and sensors used to bring information into an embedded-system processor and to transport processed information back out of the processor. Serial and parallel ports, keyboard and keypad controllers, floppy and hard disk drives, and displays are all examples of I/O devices.

IBIS I/O Buffer Information Specification. A standard simulation format used to model the behavior of an integrated circuit's input/output (I/O) pins. Used in designing and simulating the operation of circuit buses.

IC Integrated Circuit. A silicon chip containing hundreds, thousands, or millions of circuit elements such as transistors, resistors, capacitors, and inductors. RAM, ROM, microprocessors, and microcontrollers are all examples of integrated circuits.

J

JEDEC Joint Electron Device Engineering Council (pronounced JED'eck). A group organized by the IEEE (Institute of Electrical and Electronics Engineers) that has defined a standard file format for PLDs.

JEDEC file A file conforming to a standard format that specifies the configuration and testing procedure for a PLD. The file is in a human-readable ASCII format and consists of fields that start with a letter and end with an asterisk. Fields specify the pattern to program into the part, whether to secure the device, a set of test vectors to perform a functional test, and checksums to verify the integrity of the file.

K

Kbyte Kilobyte. 1024 bytes.

L

LCC Leadless Chip Carrier. A square ceramic package that has no leads; instead it has metal areas that are surface-mount soldered to the target circuit. This package is usually used only for military and aerospace applications. Available up to 84 pins.

Linear See Analog.

Logic Digital circuitry, whether in an IC, an ASIC, a microprocessor, or a microcontroller.

Logic Emulation See ASIC Emulation.

Logic Synthesis See Synthesis.

M

Make file A file commonly used by software engineers in conjunction with a make utility program to automate the building of software projects. If the software project required a device to be programmed following a compile and link operation, the make file can start the device programmer and specify a macro file to perform the programming operation.

Master Site The initial location where the first transaction of a job is performed. This is the first site a part will be programmed in before the job is broadcasted to all other sites. Traditionally, the Master Site is Site 1, but it can be any site available. Reasons for changing the Master Site location include a bad site, convenience of location, etc.

Mbyte	Megabyte. 1,048,576 bytes.
MCM	Multi-Chip Module. A hybrid manufacturing technique that places several IC chips into a single package. MCMs are a way of “creating” an integrated circuit using otherwise incompatible IC fabrication technologies (such as CMOS and GaAs). MCMs are also a way of extending the reach of existing ASIC technologies, which may lack the ability to implement an entire system design on one chip.
Memory device	A device that contains an array of storage locations. The device has a set of inputs, called addresses, which specify which location in the array is being accessed. A set of input/output pins produce the stored number (pattern) when the device is read, and accept a new value when the device is written or programmed. Additionally, there are one or more input pins that select the operating mode (read, write, standby, etc.). Memory devices may be classified by whether they are volatile or non-volatile, and whether they may be erased. The memory’s organization refers to its word width and the number of words in the device.
Microcontroller	mC. A real “computer on a chip”, incorporating a microprocessor, memory, and I/O circuits on one integrated circuit. A device that contains a central processing unit (CPU), memory, and I/O ports on a single IC. Microcontrollers that contain any form of non-volatile memory may be programmed on a device programmer. When connected to a power supply and external crystal, many of these devices form a complete microcomputer. In many embedded systems, the microcontroller may well be the only integrated circuit in the design.
Microprocessor	mP. The original “processor on a chip” introduced by Intel in 1971. An integrated circuit that contains all of the processing components of a computer CPU including the ALU, program sequencer, and bus interface. Newer microprocessors also incorporate cache memory for increased processing speed. Comes in 4-, 8-, 16-, 32-, and 64-bit varieties. Usually requires other ICs to make up an embedded system.
Microprocessor Emulator	A piece of equipment substituted in a circuit for the circuit’s microprocessor. The emulator gives more control over the circuit’s operation and eases debugging and troubleshooting efforts.
MIPS	Millions of Instructions Per Second. A performance figure of merit (numeric score or rank) for microprocessors and microcontrollers.
Mixed-Mode	Operation in both the digital and analog domains (usually refers to simulation as in “mixed-mode simulation”).
Mixed-Signal	An electronic circuit that has both analog and digital sections. Because many “real-world” systems have analog interfaces (for example, most temperature, pressure, sound and video sensors are analog), most electronic systems must accommodate analog signals. However, signal processing is now most efficiently performed by digital circuits. Therefore, almost all modern electronic systems are mixed-signal.

systems although individual ICs in such systems need not be mixed-signal chips. Instead, a design can achieve mixed-signal operation by combining separate analog and digital ICs.

Mixed-Signal Simulation	A simulation that combines the abilities of an analog simulation and a digital simulation. Used to verify the operation of mixed-mode circuitry.
Moore's Law	An empirical law developed and later revised by Intel's Gordon Moore, which predicts that the IC industry is capable of doubling the number of transistors on a silicon chip every 18 months (originally every year) resulting in declining IC prices and increasing performance. Most design cycles in the electronics in declining IC prices and embedded-system development firmly rely on Moore's law.
Multitasking	A programming style that splits the overall job to be performed by the embedded system into a number of smaller tasks, which then execute on the system's processor in a time-shared fashion.

N

Nanosecond	One billionth of a second.
Net	For ASICs, an individual signal path including all of its branches and extensions. An abbreviation for the Internet.
Net Extraction	The identification and cataloging of all signal paths in a circuit. The combination of all nets and circuit elements (transistors, resistors, capacitors, ICs, etc.) of an electronic design completely describes an electronic circuit.
Net List	Netlist. A computer file (sometimes a printed listing) containing a list of the signals in an electronic design and all of the circuit elements (transistors, resistors, capacitors, ICs, etc.) connected to that signal in the design.
Network Simulation	Simulation of a communications network to determine if it has the desired communications capacity, noise insensitivity, and fault tolerance.
Node	A single point in an electronic circuit.
Non-volatile	The characteristic of a memory that does not lose its contents when its power is removed. Non-volatile memory is useful in microcomputer circuits because it can provide instructions for a CPU as soon as the power is applied, before secondary devices, such as disk, can be accessed. Non-volatile memory includes ROM, EPROM and EEPROM.

O

Object-Oriented Programming	A programming styles that combines data blocks and the associated software processing algorithms into "encapsulated" modules with narrowly defined entry and exit points. This programming style was
------------------------------------	--

developed as a way of dealing with extremely large and complex software programming projects by breaking the project down into smaller chunks more easily handled by an individual programmer. The narrowly defined entry and exit points of each module prevent one programmer's module from disrupting another's.

Oscillator A device that produces an alternating output current.

OTP One-time programmable. The characteristic of a memory device that can be programmed once but cannot be erased. When an EPROM is described as OTP, this means that its die is erasable when exposed to ultraviolet light, but because of its package, which is not transparent, it cannot be exposed to light and thus it cannot be erased.

P

Package The plastic or ceramic that protects an IC die and connects it to the target circuit.

PAL Programmable Array Logic. The first truly successful family of programmable logic, originally introduced by Monolithic Memories in the early 1980s.

PALASM PAL Assembler. The HDL originally developed by John Birkner of Monolithic Memories for the creation of PAL-based designs.

Parallel Printer Port A standard port on virtually every PC designed for connection to a printer. This port has eight data lines and several control lines. Parallel ports may be either unidirectional or bi-directional. If your computer has a unidirectional port, the programmer will use the status lines to read data back from the programmer. The port allows high-speed communication (many times faster than a serial port). There may be up to three parallel ports in most PCs designated LPT1, LPT2, and LPT3.

Patch A small piece of code used to repair an error in an existing embedded system program.

PCB Printed circuit board, PC board, also PWB or Printed Wiring Board. A laminated board made from alternating layers of copper and plastic (usually impregnated with glass fibers for strength). The PC board serves as the physical carrier for other electronic components in an electronic design and also provides the electrical connection between these electronic components.

PGA Pin Grid Array. A square, through-hole IC package that has pins located on a square grid with 0.1000-inch pitch. It may have up to several hundred pins. Used primarily for military and prototype designs.

Picosecond One trillionth of a second.

Place and Route A layout task that positions major functional blocks or electronic components within an IC or on a PC board (Place) and the subsequent

	routing of appropriate electrical connections to those components (Route).
Platform	Term for a computer, operating system, or framework.
PLCC	Plastic Leaded Chip Carrier. A low-cost square plastic package that has J-shaped leads on four sides. This can be surface-mounted or placed in a socket for through-hole use. PLCCs have interconnection leads on either two (usually only for memory chips) or all four sides (for logic and ASIC chips). Available in 20 to 84 pins.
PLD	Programmable Logic Device. The generic term for all programmable-logic ICs including PLAs (programmable logic arrays), PALs, CPLDs (complex PLDs), and FPGAs (field programmable gate arrays).
PLD Compiler	A software package that allows an engineer to specify the functionality of a PLD through a high-level language or schematic diagram. The software will convert the design into a JEDEC or other file for the PLD programmer. PLD compilers are available from numerous IC manufacturers and from third parties. The packages from IC manufacturers support only one brand of device and may be free, inexpensive or expensive. The most popular compiler is PALASM (priced under \$200, available from AMD sales offices and representatives) which supports most of AMD's line of PMDs with an easy-to-learn high-level language. The compiler that probably offers the highest level of functionality and flexibility is PLDesigner made by MINC. It supports most PLDs and offers a sophisticated input language with full support for state machines and other complex constructs, partitioning designs into several PLDs, and graphical input. Their tools run on PCs and workstations. PLD compilers have simulators that can be used to test the functionality of your design and validate test vectors that you design before programming a device.
Pneumatic	Of or pertaining to air, gases or wind.
Point Tool	Term for an EDA tool that performs only one function.
Power Simulation	A simulation that determines the power consumption of an electronic circuit operating under a variety of normal and abnormal conditions.
PQFP	Plastic Quad Flat Pack. See <i>QFP</i> .
Price Point	Term for price. Adapted from the consumer markets where there really is a difference in sales between products prices at \$9.99 and \$10.
PROM	Programmable Read Only Memory. An integrated circuit that stores programs and data in many embedded systems. PROM stores/retains information even when the power is off but it can only be programmed or initialized once.

Q

QFP Quad Flat Pack. A square IC package that has surface-mount leads coming from four sides. It is used for high-density applications, usually over 100 pins. Lead pitch may be 0.025 inches or smaller.

R

RAM Random Access Memory. A volatile memory device. An integrated circuit that stores programs and data in many embedded systems. RAM does not retain information when the power is off and must therefore be reinitialized every time the embedded system is switched on. There are many varieties of RAM including the two most popular types: Dynamic RAM (DRAM) and Static RAM (SRAM).

RC Extraction The mathematical computation of an electronic circuit's fundamental circuit elements: resistors (abbreviated R), and capacitors (abbreviated C). RC extraction allows a simulator to determine the expected behavior of the electronic circuit through the mathematical modeling of simple circuit elements.

Register A location inside of a microprocessor, microcontroller, or I/O controller chip that stores control or status information.

RISC Reduced-Instruction-Set Computer. A design approach for microprocessors and microcontrollers, originally developed at IBM, which employs relatively simple instructions that usually execute in one clock cycle. This approach results in a faster, simpler processor design that uses fewer transistors. However, a program written using RISC instructions requires more instructions to perform a task as compared to a program written using CISC (Complex-Instruction-Set Computer) instructions.

ROM Read Only Memory. An integrated circuit that stores programs and data in many embedded systems. A non-volatile memory device that cannot be programmed by the user. It is programmed at the factory through the use of a mask pattern in the final fabrication steps of the die. PROM stores/retains information even when the power is off but it can only be programmed or initialized once and only at the semiconductor factory.

ROM Emulator An embedded-system development tool that substitutes RAM for program ROM and aids in the debugging of the program.

RTL Register Transfer level or Register Transfer Logic. A register-level description of a digital electronic circuit (see Hierarchy). Registers store intermediate information between clock cycles in a digital circuit, so an RTL description describes what intermediate information is stored, where it is stored within the design, and how that information moves through the design as it operates.

S

Scan A specialized test approach that places special shift-register circuits inside of an electronic design just for test purposes. The shift register

allows automatic test equipment to introduce test patterns deep into the circuitry and to read out status information that results from the circuit's response to those test patterns.

Schematic

A graphical representation of an electronic circuit. Until the 1980s, schematics were really the only representation system used to describe circuits. However, with the advent of HDLs and an explosion in circuit complexity, schematics are becoming less important as a representation tool.

Schematic Entry

The process of drawing a schematic using EDA tools. When done with paper and pencil, schematic entry is called schematic drafting or schematic drawing.

SCSI

Small Computer System Interface. Pronounced "scuzzy". An 8-bit parallel computer peripheral interface standard used to connect to a wide variety of peripherals devices including hard disk and CD-ROM drives, tape-backup units, and optical scanners.

Sequencer

A device for automatic or regulation of a sequence.

Serial Memory

An EPROM or EEPROM that is accessed by shifting in addresses and shifting out data one bit at a time. Interfaces are available using one, two or three wires for clock, data in, and data out.

Simulation

Modeling of an electronic circuit (or any other physical system) using computer-based algorithms and programming. Simulations can model designs at many levels of abstraction (system, gate, transistor, etc.). Simulation allows engineers to test designs without actually building them and thus can help speed the development of complex electronic systems. However, the simulations are only as good as the mathematical models used to describe the systems; inaccurate models lead to inaccurate simulations. Therefore, accurate component models are essential for accurate simulations.

Simulation Model

A software representation of a system component that describes how that component operates under various electrical and physical (temperature, pressure, light, etc.) stimulus.

Socket module

An interchangeable metal chassis that contains a programming socket.

Soft Macro

A predefined block of logic (such as a multiplier or microprocessor), which can be used as a building block for creating ASIC designs. In contrast to "Hard Macros", soft macros can be decomposed into component-level parts and edited for a particular application.

SOIC

Small Outline Integrated Circuit. A surface-mount IC package that has two rows of leads on opposite sides. Commonly found in 8 to 32 pin sizes. Leads are usually 0.050 pitch.

SPICE

Simulation Program with Integrated Circuit Emphasis. The original analog simulation program developed at the University of California

analog simulation program developed at the University of California Berkeley in the early 1970s.

SRAM

Static Random Access Memory. An integrated circuit that stores programs and data in many embedded systems. SRAM does not retain information when the power is off and must therefore be reinitialized every time the embedded system is switched on. SRAM is more expensive than DRAM on a cost-per-bit basis but is usually easier to connect to a microprocessor or microcontroller.

Standard Cell

A form of ASIC design that employs predefined logic cells and circuit components to create an ASIC. All mask layers of a standard-cell ASIC are custom for that ASIC, in contrast to a “Gate Array” in which only the metal-layer masks are custom. Standard-cell ASICs usually run faster and use less silicon (and are therefore usually cheaper on a per-part basis) than Gate Arrays. However, because the standard-cell ASIC uses predefined circuit components, its usually easier to design (and therefore requires less time to design) than a full-custom ASIC where every resistor, capacitor, and transistor is custom built.

State Diagram

A graphical representation of a state machine’s operation. State-diagram editors are EDA tools specifically designed to aid in the development of state machine designs.

State Editor

A design-entry EDA tool used to create state diagrams.

State Machine

A digital circuit built from registers and gates that controls the operation of other circuitry. For example, microprocessors contain many state machines that sequence the flow of information over the processor’s bus and through its data-manipulation circuits.

Static Timing Analyzer

An EDA tool that exhaustively checks every signal path in a circuit to identify timing-related design problems.

Symbol

A graphic, schematic library element that represents an electronic component such as a resistor, a capacitor, a transistor, or an IC.

Symbol Editor

An EDA tool for maintaining and creating schematic symbols.

Synchronous

A digital circuit where all of the operations occur in lock step to a master clock signal.

Synthesis

Logic Synthesis. A computer process that transforms a circuit description from one level of abstraction to a lower level, usually towards some physical implementation. Synthesis is to hardware design what compilation is to software development. In fact, logic synthesis was originally called hardware compilation.

I

Test Synthesis

The automatic creation of test patterns and a test program for the verification of manufactured ICs.

Test Vector	A stimulus pattern applied to a circuit to verify the circuit's operation. A set of characters that describe the inputs and outputs of a device during a functional test. There is one character in the vector for each pin on the device. Numbers represent inputs to be applied to the device (1 for Vih, 0 for Vil). Letters represent the outputs that must be tested (H for Voh, L for Vol, Z for high-impedance). During the test, the part will be powered up and each input will be applied to the device for the first vector. Then, each output will be applied to the device for the first vector. This process will continue for each vector and any errors will be reported.
Timing Diagram	A graphical representation of the signals in an electronic circuit that shows how the signals change over time in relationship to each other.
Timing Simulation	Simulation of an electronic circuit's operation over time using calculated circuit parameters such as resistance, capacitance, inductance, and timing delays.
Top-Down Design	A design methodology that starts the design of an electronic system at the very highest level of abstraction and then methodically broadens the design through lower abstract layers until finally reaching the concrete, physical design layer which accurately represents the implementation technology for the system.
TQFP	Thin Quad Flat Pack. Similar to QFP but with a lower profile and physically smaller in length and width.
Transmission Line	A conductor or wire that is suited to carrying high-frequency signals.
TSOP	Thin Small Outline Package. A surface-mount package with fine-pitch leads (usually 0.025 inch pitch) on two sides. This package is very low profile and commonly available in a reverse (mirror image) pinout used to simplify circuit board layout. Usually 32 to 44 pins.

U

Unit Delay Simulation	A simplified form of timing simulation where every digital gate is assumed to introduce one unit of delay to a signal. In reality, different gates have different speeds, but unit delay simulation trades off accuracy for simulation speed.
UV Erasable	The characteristic of an EPROM that allows it to be erased with exposure to short-wave ultra-violet light. This high-energy light can discharge the floating-gate transistor cells that store bits in an EPROM. The most common source of such light is a mercury vapor tube much like an ordinary fluorescent tube, but without the phosphor that turns the UV light emitted by the mercury into visible light. The light from ordinary fluorescent lamps or sunlight generally takes years to erase an EPROM. All UV erasable parts have a quartz windowed ceramic package that allows exposure with UV light.

V

Verification	The task of establishing the correctness of a design using EDA tools to automatically check the timing, connections, and rules used to design the circuit.
Verify	Reading a programmable device and comparing its contents to the desired pattern for that device. This is a go/no-go test – it does not report what the discrepancies are. See also: <i>compare</i> .
Verilog	A hardware description language developed by Gateway Design Automation (now part of Cadence) in the 1980s which became very popular with ASIC and IC designers.
VHDL	VHSIC Hardware Description Language. A hardware description language developed in the 1980s by IBM, Texas Instruments, and Intermetrics under US government contract for the Department of Defense’s VHSIC (Very High Speed Integrated Circuit) program. VHDL enjoys a growing popularity with ASIC designers as VHDL development tools mature.
Volatile	The characteristic of a memory device (specifically RAM) that will lose its contents when the power is removed. These parts are not programmable with a device programmer because they cannot be removed from the programming socket without losing their contents.

W

Word	A unit of memory usually consisting of two bytes (16 bits).
Word width	The number of output pins that a memory device has. The most common sizes for EPROMs is byte wide (8 bits) and “word” wide, or 16 bits. It can also refer to the aggregate width of several memory devices used in a set.

ACRONYMS

A

AAAI	American Association for Artificial Intelligence
AAL	ATM Adaptation Layer
ABET	Accreditation Board for Engineering and Technology
ACE	Advanced Computing Environment
ACL	Advanced CMOS Logic
ACM	Association for Computing Machinery
A/D and D/A	Analog-to-Digital and vice versa
ADI	Autodesk Driver Interface (from AutoCad)
ADPCM	Adaptive Differential Pulse-Code Modulation
AEA	American Electronics Association
AEW system	Airborne Early Warning System
AGC	Automatic Gain Control
AI	Artificial Intelligence
AIA	Aerospace Industries Association
AIAA	American Institute of Aeronautics and Astronautics
ALU	Arithmetic-Logic Unit
AMPS	Advanced Mobile Phone Service
ANSI	American National Standard Institute
API	Applications Programming Interfaces
AQL	Accepted Quality Level
Arpa	Advanced Research Projects Agency
ASCII	American Standard Code for Information Interchange
ASEE	American Society for Engineering Education
ASIC	Application-Specific IC
ASP	Average Selling Price (a sales term)
ASSP	Application-Specific Standard Product
ATA	AT Bus Attachment (a PC interface)
ATE	Automatic Test Equipment
ATM	Asynchronous Transfer Mode
ATPG	Automatic Test Pattern Generation
ATTC	The FCC's Advanced TV Test Center
AWACS	Airborne Warning and Control System

B

BCD	Binary-Coded Decimal
BiCMOS	Bipolar CMOS
BiFET	Bidirectional Field-Effect Transistor
BIOS	Basic Input/Output System
B-NTSC	Baseband National Television Standards Committee

C

CAD/CAM/CAE	Computer-Aided-Design, -Manufacture, -Engineering
CAGR	Compound Annual Growth Rate
CAS	Column-Address Strobe
CASE	Computer-Aided Software Engineering

CAU	Control Arithmetic Unit
CBI	Computer-Based Instrument
C ³	Command, Control and Communications Systems
CCD	Charge-Coupled Device
CCIR	Comite Consultatif International des Radio Communications
CCITT	Consultative Committee for International Telephone and Telegraph, former standards body of IEEE and now succeeded by the ITU-TSS, or International Telecommunications Union, Technical Standards Section
CD-I	Compact Disk-Interactive
CDPD	Cellular Digital Packet Data
CD-ROM	Compact Disk, Read-Only Memory
CEMOS	Complementary Enhanced MOS
CEPT	Conference of European Posts and Telecommunications
CERN	Conseil European pour le Recherche Nucleaire aka the European Particle Physics Laboratory
CFC	Chlorofluorocarbon
CFI	CAD Framework Initiative
CFM	Cubic Feet per Minute
CGA	Color Graphics Adapter
Chill	CCITT High-Level Language
CHMOS	Complementary High-Performance Metal-Oxide Semiconductor
CIF	The CCITT's Common Intermediate Format (for video; 352-pixel x 288-line resolution)
CIM	Computer-Integrated Manufacturing
CIO	Counter/Timer Input/Output
CISC	Complex Instruction Set Computer
CMOS	Complementary Metal-Oxide Semiconductor
CMRR	Common Mode Rejection Ratio
COS	The Corporation for Open Systems, an organization of computer and communications equipment suppliers and users
COSE	Common Open System Environment
CPLD	Complex PLD
CP/M	Control Programs/Microcomputer
CPU	Central-Processing Unit
Crada	Cooperative Research and Development Agreement
CRC	Cyclic Redundancy Checking
CRT	Cathode-Ray Tube
CSA	Canadian Standard Association
CSIC	Customer-Specific IC
CSMA/CD	Carrier-Sense Multiple-Access with Collision-Detection (baseband schemes)
CTE	Coefficient of Thermal Expansion
CVD	Chemical Vapor Deposition
cw	Continuous wave

D

DAC	Design Automation Conference
DAT	Digital Audio Tape
DCC	Digital Compact Cassette
DCE	Data Circuit Terminating Equipment; also Distributed Computing Environment

DCFL	Direct-Coupled FET Logic
DCT	Discrete Cosine Transform
DDS	Digital Data Standard
DECT	Digital European Cordless Telephone (standard)
DESC	Defense Electronics Supply Center
DIP	Dual-In-Line Package
DSB	Display-Station Buffer
Divad	Initial cap; Division Air Defense
DMA	Direct Memory Access Controller
DMM	Digital Multimeter
DMSK	Differential Minimum Shift Keying
DOS	Disk Operating System, Operating System Standard created by Microsoft Corp.
DPMI	DOS Protected Mode Interface
dpst	Double-Pole, Single Throw; for switches or relays
DQPSK	Digital Quadrature Phase-Shift Keying
DRAM	Dynamic Random Access Memory
DRC	Design Rule Checking
DSO	Digital Storage Oscilloscope
DSP	Digital Signal Processing
DSU	Dataport Service Unit
DTE	Data Terminal Equipment
DTL	Diode Transistor Logic
DTMF	Dual-Tone Multifrequency
DUT	Device Under Test
DVI	Digital Video Interactive Technology

E

EAROM	Electrically Alterable ROM
ECC	Error-Correction Code
ECL	Emitter-Coupled Logic
ECO	Engineering Change Order
ECP/PPP	Enhanced Capabilities Port/Enhanced Parallel Port
ECU	European Currency Unit
EDA	Electronic Design Automation
EDAC	Electronic Design Interchange Format
EDIF	Electronic Design Interchange Format
EFTA	European Free Trade Area
EGA	Enhanced Graphics Adapter
EIA	Electronic Industries Association
EIAJ	Electronic Industries Association - Japan
EIC	Equivalent IC
EISA	Extended Industry Standard Architecture
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
EPLD	Erasable Programmable Logic Device
EPOS	Electronic Point of Sale
EPROM	Erasable PROM
EEPROM	Electrically Erasable PROM
ERC	Electrical Rule Checking
ESA	European Space Agency

ESD	Electrostatic Discharge
ESDI	Enhanced Small Device Interface
Esprit	European Strategic Program for Research and Development in Information Technology
ESR	Equivalent Series Resistance

F

FAE	Field-Application Engineers
FCC	Federal Communications Commission
FDDI	Fiber Distributed Data Interface
FDM	Frequency-Division Multiplexing
FET	Field-Effect Transistor
FFT	Fast Fourier Transform
FIFO	First In, First Out
FIN	Flexible Interface Network
FOIRL	Fiber-Optic Inter-Repeater Link
FPGA	Field-Programmable Gate Array
FSM	Finite-State Machine
FSR	Full-Scale Reading

G

GaAs	Gallium Arsenide
G&A	General and Administrative (expenses)
GATT	General Agreement on Tariffs and Trade
GDI	Graphical Driver Interface
GPIB	General-Purpose Interface Bus
GPS	Global-Positioning Satellite
GRIN	Graded Index (lens)
GSM	Global System for Mobile Communications (standard)
GTL	Gunning Transceiver Logic
GUI	Graphical User Interface

H

HAL	Hardwired Array Logic Device, from MMI (trademark)
HBTs	Heterostructure Bipolar Transistors
HDL	Hardware Description Language
HEMTs	High Electron Mobility Transistors
HiPPI	High Performance Parallel Interface

I

IC	Integrated Circuit
I-C bus	Inter-IC bus
ICCAD	International Conference on Computer-Aided Design
ICE	Integrated Circuit Engineering
IDE	Integrated Device Electronics (interface)
IEE	Institution of Electrical Engineers
IEEE	The Institute of Electrical and Electronics Engineers
IECQ	International Electrotechnical Commission of Quality Control System
IFPI	International Federation of Phonogram and Videogram Producers
IGBT	Insulated-Gate Bipolar Transistor

IGT	Insulated-Gate Transistor
IofNewt	Integration of Numerical and Experimental Wind Tunnels
IPC	Institute for Interconnecting and Packaging Electronic Circuits
IPI	Intelligent Peripheral Interface
IR LED	Infrared Light Emitting Diode
ISA	Industry Standard Architecture
ISDN	Integrated Services Digital Network
ISHM	International Society for Hybrid Microelectronics
ISO/IEC	International Standards Organization/International Electrotechnical Commission
ISSCC	International Solid-State Circuits Conference
ISV	Independent Software Vendor
ITC	International Test Conference
ITO	Indium Tin Oxide
ITU-T	International Telecommunications Union, Technical Standards Section. In 95% of the cases since 12/93, it has replaced CCITT (see above) as standards-setting body of IEEE. This is true for modern standards, cabling standards, ISDN and ATM standards, etc.

J

JAN	Joint Army-Navy Military Standard
JEDEC	Joint Electronic Device Engineering Council
JEIDA	Japan Electronics Industry Development Association
JEMI	Joint Equipment Manufacturers Initiative
Jessi	Joint European Submicron Silicon Initiative
JETRO	Japan External Trade Relations Organization
JIT	Just-In-Time; A Manufacturing Philosophy
JPEG	Joint Photographic Experts Group
JTAG	Joint Test Action Group

L

LAN	Local-Area Network
LCA	Logic Cell Array, from Xilinx (trademark)
LCC	Leadless Chip Carrier
LCD	Liquid-Crystal Display
LED	Light-Emitting Diode
LPE	Liquid Phase Epitaxy
LSI	Large-Scale Integration
LSB	Least Significant Bit
LSSD	Level-Sensitive Scan Design

M

MAC	Medium Access Control
MAN	Metropolitan-Area Network
MAP	Manufacturing Automation Protocol
MAU	Mathematics Acceleration Unit OR Media Attachment Unit, Depending on Context
MBE	Molecular Beam Epitaxy
MCB	Molded Circuit Boards
MCC	(yes, we know there's a letter missing in this acronym, but it's correct)

	Microelectronics and Computer Technology Corp.
MCM	Multichip Module
MESFETs	Metal-Semiconductor Field-Effect Transistors
MF	Multifrequency Modulation
MIDI	Musical Instrument Digital Interface
Mips	Million Instructions Per Second
MIS	Metal-Insulator Semiconductor
MISFETs	Metal-Insulator Semiconductor Field-Effect Transistors
MITI	Ministry of International Trade and Industry
MLC	Multilayer Ceramic Capacitor
MMIC	Monolithic Microwave Integrated Circuit (generic designation)
MMU	Memory Management Unit
MOCVD	Metal-Organic Chemical Vapor Deposition
Mops	Millions of Operations Per Second
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
MOSIGT	Metal-Oxide Semiconductor Insulated Gate Transistor
MPEG	Moving Picture Experts Group
Mpps	Megapixels Per Second
MPT	Ministry of Posts and Telecommunications
MRP	Manufacturing Resource Planning
MSI	Medium-Scale Integration
MTBF	Mean Time Between Failure
Mtops	Millions of Theoretical Operations per Second
MVS	IBM's Proprietary Operating System
MWh	Megawatt-hour

N

NAPLPS	North American Presentation Level Protocol Syntax, the ANSI-videtex graphics standard
NIC	Network Interface Controller
NII	National Information Infrastructure – the information superhighway
NISO	National Information Standards Organization
NIST	National Institute of Standard and Technology
NRE	Non-recurring Engineering (as in 'charges')
NSPE	National Society of Professional Engineers
NTSC	National Television Standards Committee

Q

OCR	Optical Character Recognition
OECD	Organization for Economic Cooperation and Development
OEM	Original Equipment Manufacturer
OFDM	Orthogonal Frequency Division Modulation
OLTP	On-line Transaction Processing
OOPS	Object-Oriented Programming Software
OOPSLA	Object-Oriented Programming Systems, Languages and Applications conference
OROM	Optical ROM
OS	Operating System
OSF	Open Software Foundation
OSI	Open Systems Interconnection

OS/2	Operating System from Microsoft, IBM
OSTP	Office of Science and Technology Policy
OTP	One Time Programmable

P

PACE	Professional Activities Council Committees for Engineers
PAL	Programmable Array Logic
PA-RISC	Precision Architecture – Reduced Instruction Set Computer
PBX	Public or Private Branch Exchange
PCI	Peripheral Component Interconnect (local bus)
PCMCIA	Personal Computer Memory Card International Association (PCMCIA cards provide a standardized format and interconnection method for the addition of various peripherals, increased memory storage or other functions)
PCN	Personal Communications Network
PDA	Personal Digital Assistant
PIA	Peripheral Interface Adapter
PIN	Positive-Intrinsic Negative (diode)
PIV	Peak Inverse Voltage
PLA	Programmable Logic Array
PLCC	Plastic Leadless Chip Carrier
PLD	Programmable Logic Device
PRML	Partial-Response, Most-Likelihood (encoding technique)
PRO	Precision RISC Organization (HP consortium)
Promis	Project Management Integrated System, from Strategic Software Planning Corp. (trademark)
PSK	Phase Shift Keying
PSMA	Power Sources Manufacturers' Association
PSRR	Power Supply Rejection Ratio
PTT	Post, Telegraph and Telephone Administrations (a generic term for public telegraph and telephones; much like RBOC is for the seven baby bells)
PVM	Poly-Vector Modulation
PWM	Pulse-Width Modulation

Q

QAM	Quadrature Amplitude Modulation
QFP	Quad Flat Pack
QIC	Quarter Inch Cartridge

R

RAID	Redundant Arrays of Independent Disks
RAMDAC	Random Access Memory Digital to Analog Converter
RAM/ROM	Random Access and Read Only Memories
RARP	Reverse Address Resolution Protocol
RAS	Row Address Strobe
RBOC	Regional Bell Operating Companies
RC time constant	Resistance-Capacitance
R-DAT	Rotary Digital Audio Tape
R&D	Research and Development
RF	Radio Frequency

RFI	Radio Frequency Interference
RFP	Request for Proposal
RGB	Red Green Blue
RISC	Reduced Instruction Set Computer
RTP	Rapid Thermal Processing

S

SAR	Segmentation and Re-assembly
SAW	Surface Acoustic Wave
SCI	Scalable Coherent Interface
SCR	Silicon Controlled Rectifier
SCSI	Small Computer Systems Interface
SDLC	Synchronous Data Link Control
Secam	Système Electronique Couleur Avec Memoire. French TV transmission standard.
SEMI	Semiconductor Equipment and Materials International
SIA	Semiconductor Industry Association
SID	Society for Information Display
SIMD	Single Instruction, Multiple Data
SLIC	Subscriber Line Interface Circuit
SLIP	Serial Line Interface Circuit
SMA	Surface Mount Assembly; also Subminiature Type A (connector)
SMD	Surface Mount(ed) Device
SMDS	Switched Multimegabit Data Services
SMI	System Management Interrupt
SMPTE	Society of Motion Picture and Television Engineers
SMT	Surface Mount Technology
SNA	Systems Network Architecture (IBM protocol)
SNMP	Simple Network Management Protocol
SOI	Silicon On Insulator
SOJ	Small-Outline, J-leaded package
SOP	Small-Outline Package
SOS	Silicon-On-Sapphire
SPA	Software Publishers Association
SPACE	The Satellite Television Industry Association (Society for Private and Commercial Earth Stations)
SPEC	Systems Performance Evaluation Cooperative
Spice	Simulation program with integrated-circuit emphasis
SPIE	Society of Photometric and Instrumentation Engineers
SQUID	Superconductive Quantum Interference Device (from IBM)
SQL	Structured Query Language
SRAM	Static RAM
SRQ	Service Request; an IEEE 488 command
SSI	Small Scale Integration
SSR	Solid State Relay
SSOP	Shrink Small-Outline Package
SVID	AT&T's System V Interface Definition, a definitive guide for programmers writing applications for System V.

T

TCE	Thermal Coefficient of Expansion (spell out on first mention)
TCP/IP	Transmission Control Protocol and Internetwork Protocol
TDMA	Time Division Multiple Access
TFT	Thin-Film Transistor
TFTP	Trivial File Transfer Protocol
T/H	Track and Hold
TOP	Technical Office Protocol
TOPFET	Temperature and Overload Protected Field-Effect Transistor
TOW missiles	Tube-launched, Optically tracked, Wire-guided missiles
tpi	Tracks per inch
TQFP	Thin Quad Flat Pack
TRON	The Real-time Operating System Nuclues project
TSOP	Thin Small-Outline Package
TTL	Transistor-Transistor Logic
TVRO	Television Receive Only; a type of satellite dish

U

UART	Universal Asynchronous Receiver/Transistor
UHF	Ultra-High Frequency
UIM	Universal Interconnect Matrix
UL	Spell out on first mention – Underwriters Laboratories
Unix	Computer Operating System
UPC	Universal Peripheral Controller; alsions (controller)
UPS	Uninterruptible Power Supply
USARTs	Universal Synchronous/Asynchronous Receiver/Transmitters
USC	Universal Serial Communicattage-controlled oscillator
UV	Ultraviolet

V

VAR	Value Added Retailer or Reseller
VAN	Value Added Network
VBR	Variable Bit-Rate
VCO	Volo Universal Price Code (no periods after the letters)
VDE	Verband Deutscher Elektro-Techniker. West Germany's Components Safety Agency
VESA	Video Electronics Standards Association
VFC	Voltage to Frequency Converter
VFET	Vertical Field-Effect Transistor
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit (US Government Program)
VITA	VME International Trade Association
Vital	VHDL Initiative Toward ASIC Libraries
VL-bus	VESA Local Bus (standard), developed by the Video Electronics Standards Association
VLDS	Very Large Data Storage
VLSA	Very Large-Scale Arithmetic
VLSI	Very Large-Scale Integration
VMEbus	One word, note caps and lower case
VMS	DEC's proprietary operating system

Glossary

VTR Videotape Recorder; use VCR

W

WAN Wide Area Network

X

XGA Extended Graphics Array

Z

ZIF Zero Insertion Force

ZIP Zigzag In-Line Package

INDEX

Action Buttons	4-17	Hardware Installation	2-10
Advanced Feature Software		hardware test	7-2
Upgrade.....	4-3	Help	4-14
AFS.....	7-9. <i>See</i> Advanced Feature Software	Hot Keys	4-32
algorithm.....	7-4, 7-5, 7-6, 7-8, 7-9,7-10	Introduction.....	1-1
Algorithm.....	6-11	Keyboard Usage	4-32
autohandler	2-8,5-1,7-6	LED.....	7-5
blank check	4-27,7-4,7-8,7-11	LOADING A DATA PATTERN	3-4
<i>Blank check before programming</i>	7-8, 7-11	Memory.....	7-7
BP-1000, BP-2000 Series	2-10	Open Button	4-17
BP-3000, BP-4000 Series	2-11	OTP	7-8
buffer	4-21,4-22,2-23,2-24,4-25	package type.....	4-18,7-9
cable.....	7-5,7-6	parallel port	2-3,2-10,7-5, 7-6
<i>Check electronic identifiers</i>	7-10	parallel printer port.....	2-11,7-5
Cleaning a Dirty DIP Socket.....	7-4	Placing a Chip	3-7
Clear Button.....	4-25	PLCC.....	7-7,7-9
Close Button	4-24	PLD	7-9,7-11
Command Reference.....	4-1	port	2-3,2-10,2-11,7-5
Compatibility	2-3	Programmer	
continuity test.....	7-6,7-7, 7-8	Configure.....	4-1
Customer Service.....	7-1	Model	4-2
Data Pattern	4-15,4-21,4-22,4-23	QFP	7-7
Dev Info.....	4-19	RAM	2-3,7-7
Device.....	4-18	revision.....	4-9,4-13,7-2
insert	7-6, 7-7, 7-8	RMA	7-2
new.....	7-2, 7-8, 7-12	Sales Support.....	7-1
old	7-12	Save Button.....	4-17
select	7-9, 7-10	secure	4-27,5-2,7-4, 7-8, 7-9, 7-12
Device Config.....	4-25	Self-test	2-17,2-18,2-19
<i>Device/Blank</i>	7-8	signature	7-7, 7-8
<i>Device/Compare</i>	7-8	Socket	
<i>Device/Options</i>	7-8,7-11	Cleaning	7-4
<i>Device/Program</i>	7-8,7-11	socket module	2-16,3-6,3-7,7-6, 7-9
DIP.....	7-4	Software installation.....	2-3
DOS	4-1	Software Updates	7-2
Edit Button.....	4-24	Test	
EEPROM	7-4,7-8	continuity.....	4-27,7-6
electronic identifier	7-8, 7-10	hardware.....	7-2
EPROM	6-7,6-9,7-4,7-8	Tools	4-4
ERROR MESSAGES	7-4	Troubleshooting	7-1
Errors while Programming.....	7-3	TSOP.....	3-9,7-7
File		upgrade.....	4-2,5-4,7-2,7-9, 7-10
AUTOEXEC.BAT	7-7	USB.....	1-1,2-3,2-10,2-11,2-16,4-2
CONFIG.SYS	7-7	Vector test after verify.....	7-9
File Tabs	4-27	vectors.....	7-9
fuse	7-4,7-8,7-11	<i>Verify after programming</i>	7-8
Getting Started	2-3	version.....	2-10,4-16,7-2,7-5,7-9
Handler	4-28	WARNING MESSAGES.....	7-10

