PSDSOFT EXPRESS USER MANUAL

Design Software Tool for the PSD Families

CONTENTS

INTRODUCTION

PSDsoft Express is the design software for the PSD (Programmable System Device) families of parts. This new design tool allows you to easily integrate a PSD into your design using a simple point-and-click environment.

This is the user manual for PSDsoft Express. The next section explains how to install PSDsoft Express. Although installation may seem like a trivial process, it is highly recommended that you follow the instructions very carefully, as many problems are caused by incorrect installation. Sections 3 and 4 introduce the PSDsoft Express environment, which is made up of the Design Flow and menus. In Section 5, you are guided step-by-step through a complete design process. The sections that follow Section 5 explain each step of the design process in more detail. The appendices explain certain terms that are used throughout this manual.

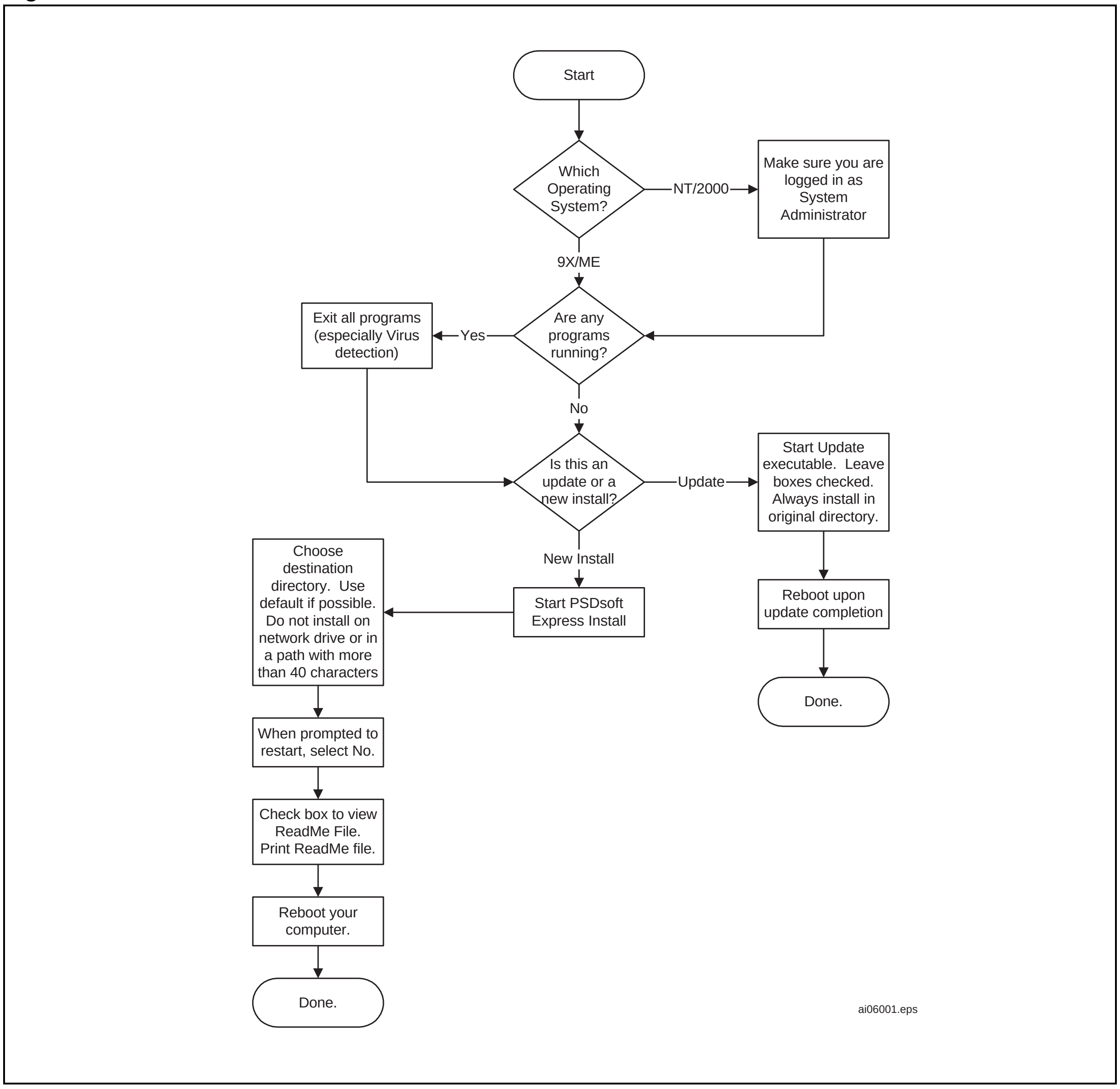
Important note: Section 5 contains information not found in other sections, and therefore, all users should read this section.

The main focus of this user manual is to explain the functionality of the of the software. As such, you are encouraged to download the datasheet associated with device that you will be using because there will sometimes be important information that can only be gleaned from the datasheet.

PSDSOFT EXPRESS INSTALLATION

For a quick reference on how to install or update PSDsoft Express, use the following flowchart:

Figure 1. Installation Flowchart


Start
Which Operating System?
NT/2000
Make sure you are logged in as System Administrator
9X/ME
Are any programs running?
Yes
Exit all programs (especially Virus detection)
No
Is this an update or a new install?
Update
Start Update executable. Leave boxes checked. Always install in original directory.
Reboot upon update completion
Done.
New Install
Start PSDsoft Express Install
Choose destination directory. Use default if possible. Do not install on network drive or in a path with more than 40 characters
When prompted to restart, select No.
Check box to view ReadMe File. Print ReadMe file.
Reboot your computer.
Done.
ai06001.eps

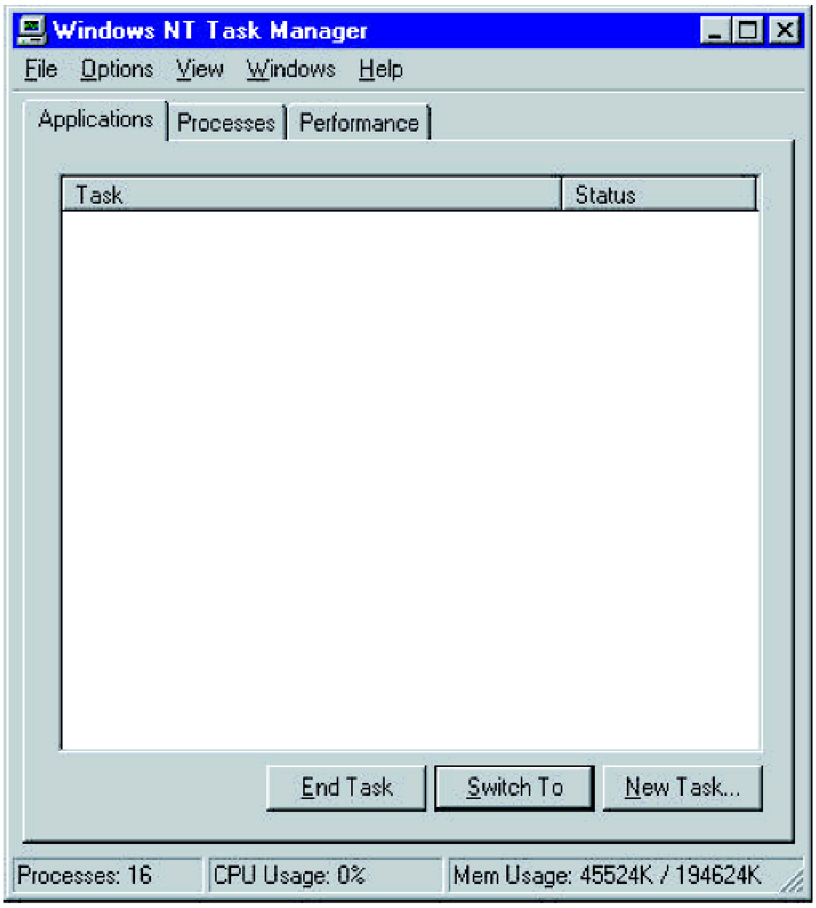
The sub-sections below contain more detailed information.

Prepare your System

Whether you are installing PSDsoft Express or updating to the latest version, you must first prepare your system for the install or update. To do so, follow the steps below carefully. If you do not follow these steps, there is a chance that your installation will not work properly.

Determine Operating System. You first need to determine which operating system you are using. If you are unsure, open the “Explorer” window and go to **Help->About**. If you are using Windows NT or 2000, you must be logged in as the system administrator.

Close All Programs. You must ensure that all programs (especially virus detection) programs are exited before installing the software. There are two places to check for running programs: the “Task Manager” and the “System Tray”. The Task Manager can be brought up by simultaneously pressing the CTRL + ALT + DEL keys. On some operating systems, you must then click the **Task Manager…** button. Ensure that the “Applications” tab is selected. Then, select any running programs and then click **End Task**. You will be prompted in some cases to confirm this action. Your Task Manager should have the following look after you have exited all tasks:


Windows NT Task Manager
File Options View Windows Help
Applications Processes Performance
Task Status
End Task Switch To New Task...
Processes: 16 CPU Usage: 0% Mem Usage: 45524K / 194624K

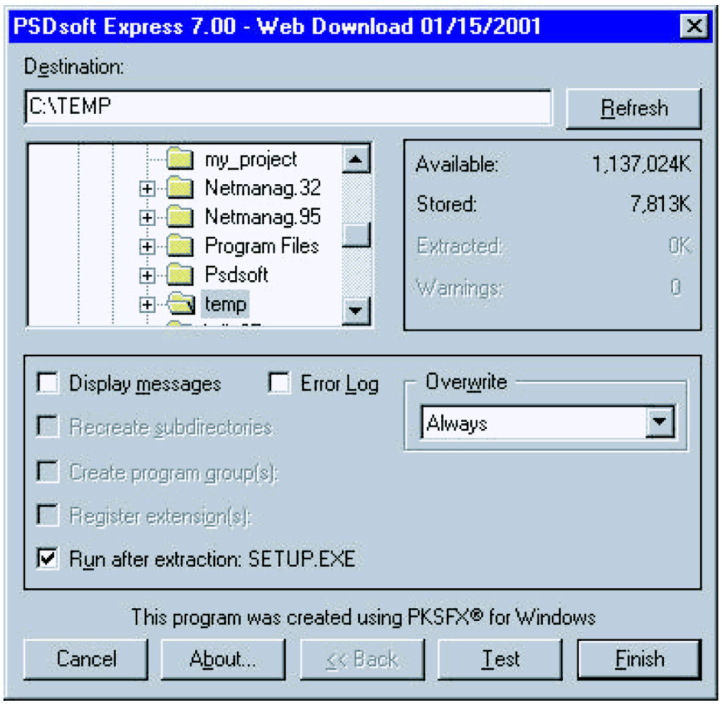
Next and perhaps even more important is to close or disable all items in the system tray, which is usually located in the lower right-hand corner of your Taskbar. This is usually the location of programs that run in the background and don’t always appear in the Task Manager. Things that run in the background can affect software installations. Virus detection programs are especially prone to causing installation problems. To exit or disable programs in the System Tray, right-click on each program and choose the appropriate action, such as “Exit” or “Disable”. Your system tray should basically be empty and have the following look when finished:

Installing PSDsoft Express

If you are installing PSDsoft Express for the first time or you have used the uninstall program and wish to re-install the program, follow these instructions carefully. If you are updating PSDsoft Express, skip to the *Update* section.

New Installation (from downloaded file). If you downloaded from the website, you will first need to un-zip the self-extracting executable.

Next, using your Window’s Explorer, navigate to the location of the executable that you just unzipped and double-click to start the self-extract process. You will see the following window:


PSDsoft Express 7.00 - Web Download 01/15/2001

Destination: C:\TEMP

Available: 1,137,024K
Stored: 7,813K
Extracted: 0K
Warnings: 0

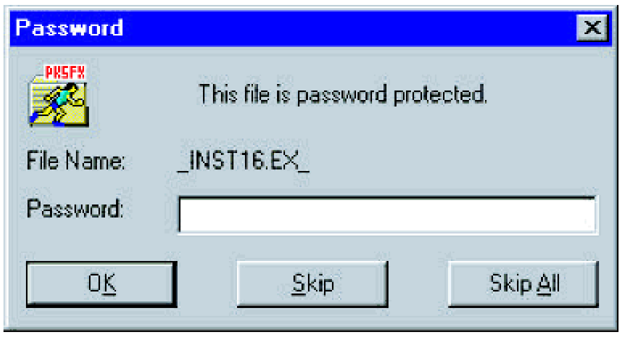
Overwrite: Always

Run after extraction: SETUP.EXE

This program was created using PKSFX® for Windows

Leave the default settings as they are and click **Finish**.

You will be asked for your password, which you should have received via email. If you did not receive a password, contact *ask.psd@st.com*.



This file is password protected.

File Name: _INST16.EX_

Password:

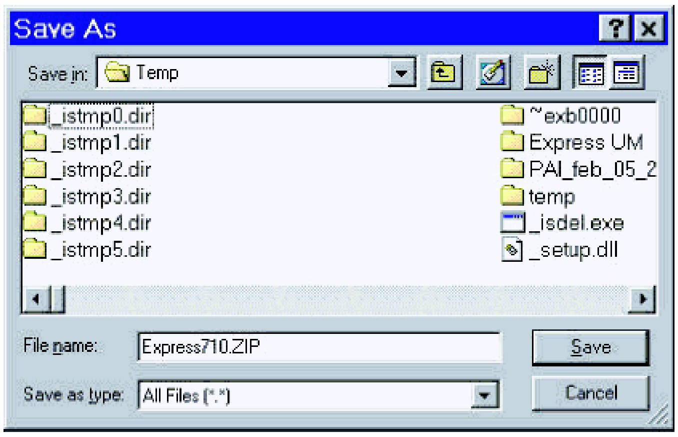
Enter your password and click **OK**. The installation process will begin automatically. Follow the installation instructions on the screen. You are highly encouraged to view the readme.txt file upon completion of the installation process.

New Installation (from CD ROM). If you are installing from a CD, generally the install program will start automatically when the CD is inserted or the tray is closed. If not, open the Window's Explorer, select the CDROM drive, and double-click the **Setup** program and follow the installation instructions as they come up on the screen. You are highly encouraged to view the readme.txt file upon completion of the installation process.

Updating PSDsoft Express

PSDsoft Express is updated periodically to improve performance, add new features, support new parts, and so on. Therefore, you should check *www.st.com/psd* periodically for updates. You can also go directly to the *Update* section: *www.st.com/psd/html/express_download.html*.

When you click the update link, you will get a window similar to the following:



Save in: Temp

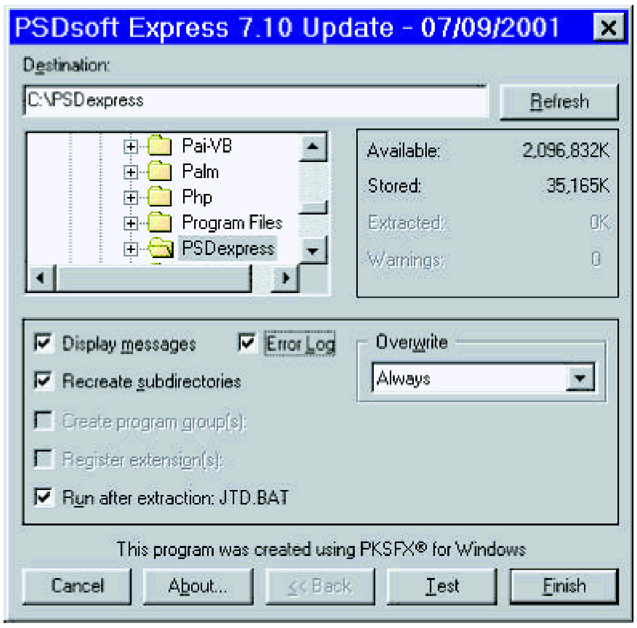
_istmp0.dir
_istmp1.dir
_istmp2.dir
_istmp3.dir
_istmp4.dir
_istmp5.dir
~exb0000
Express UM
PAI_feb_05_2
temp
_isdel.exe
_setup.dll

File name: Express710.ZIP

Save as type: All Files (*.*)

Choose a temporary location for the .ZIP file and click **Save**. The reason there is a self-extracting executable within a .zip file is that some companies have a firewall that prevents downloading executables.

Next, using your Window's Explorer, navigate to the location of the executable that you just unzipped and double-click to start the self-extract process. You will see the following window:


PSDsoft Express 7.10 Update - 07/09/2001

Destination: C:\PSDexpress

Available: 2,096,832K
Stored: 35,165K
Extracted: 0K
Warnings: 0

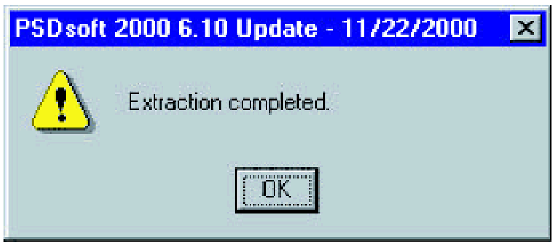
☑ Display messages ☑ Error Log
☑ Recreate subdirectories
☐ Create program group(s):
☐ Register extension(s):
☑ Run after extraction: JTD.BAT

Overwrite: Always

This program was created using PKSFX® for Windows

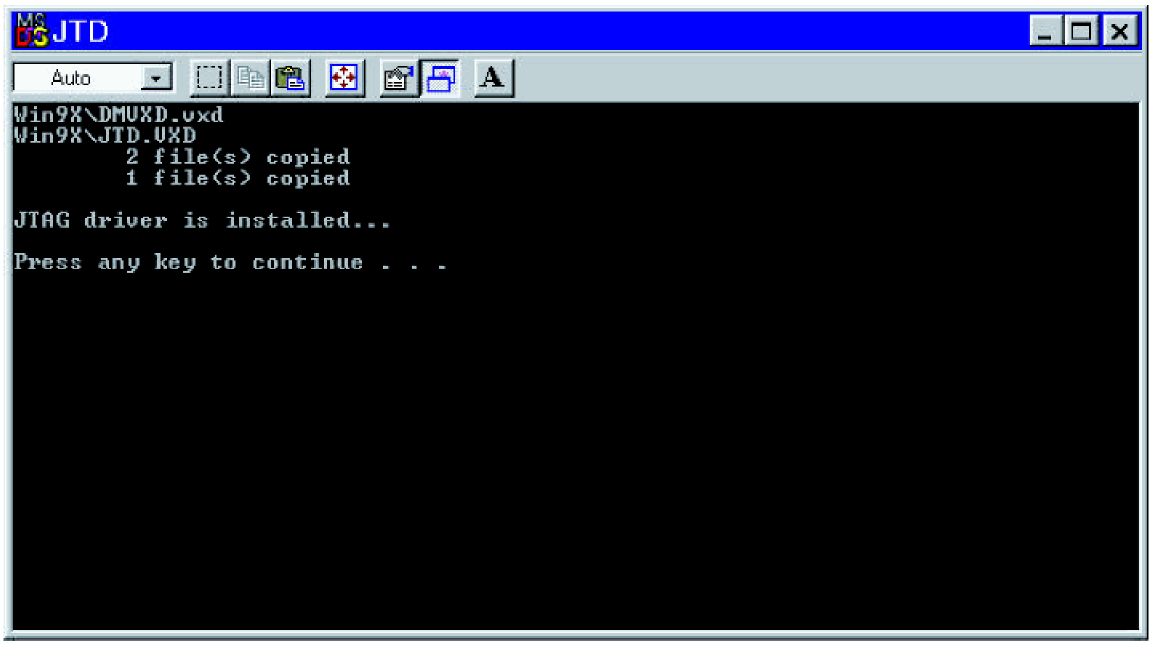
Ensure that you have all the boxes checked, as shown above and make sure that you pick the directory that you had previously installed the software and click **Finish**.

You should get a message stating the extraction completed.


PSDsoft 2000 6.10 Update - 11/22/2000

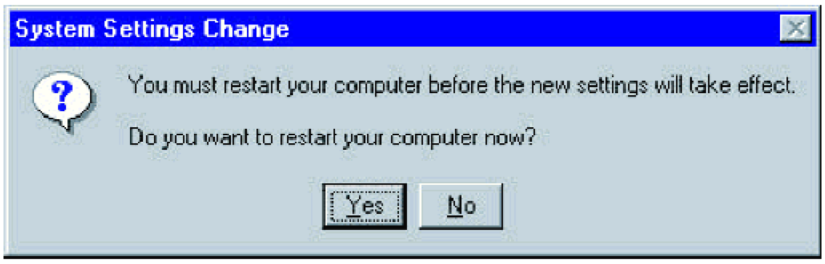
Extraction completed.

You should also see another window appear indicating the JTAG driver is installed.



```
Win9X\DMVXD.vxd
Win9X\JTD.VXD
        2 file(s) copied
        1 file(s) copied

JTAG driver is installed...

Press any key to continue . . .
```

Finally, a window prompting you to restart your computer will appear. Click **Yes** to restart. After your computer has re-started, be sure to go to the directory where you installed PSDsoft Express and view the ReadMe file to ensure that no further actions are necessary.



You must restart your computer before the new settings will take effect.

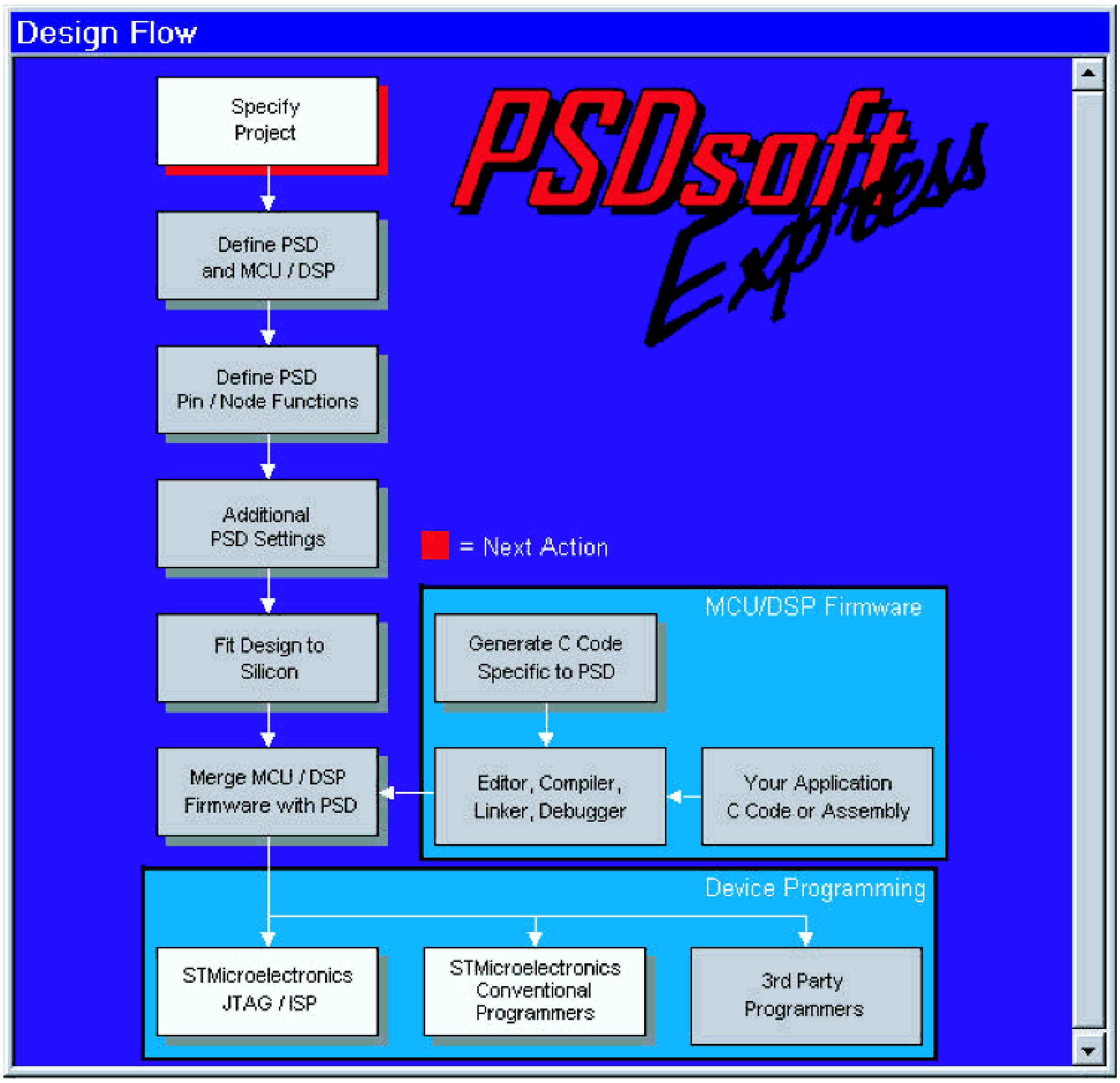
Do you want to restart your computer now?

Note: before you start PSDsoft Express, ensure that your screen's resolution is set to at least 800 x 600 pixels or higher or else it will be difficult to use the software.

PSDSOFT EXPRESS DESIGN FLOW

This section explains how to use the PSDsoft Design Flow. To use the design flow, simply single-click on the box containing the desired action.

Note: depending on the part chosen and the Project Preferences, you may not see all of the choices shown below in the Express Design Flow. Also, on computers with lower screen resolution, you may see a scroll bar on the right-hand side of the window. You may need to scroll down to see all of the choices.


Design Flow
Specify Project
Define PSD and MCU / DSP
Define PSD Pin / Node Functions
Additional PSD Settings
Fit Design to Silicon
Merge MCU / DSP Firmware with PSD
PSDsoft Express
= Next Action
MCU/DSP Firmware
Generate C Code Specific to PSD
Editor, Compiler, Linker, Debugger
Your Application C Code or Assembly
Device Programming
STMicroelectronics JTAG / ISP
STMicroelectronics Conventional Programmers
3rd Party Programmers

The PSDsoft Express Design Flow shown above changes dynamically when you use the software. When a box is gray with a gray shadow, it indicates that the process within the box cannot be invoked until you have completed a previous step. The red shadow indicates the next action to be performed. The two actions that can be invoked at any time are “STMicroelectronics JTAG/ISP” and “STMicroelectronics Conventional Programmers”. “Generate C Code” will be available after you have completed “Define PSD and MCU/DSP” unless you have a EPROM-based PSD, in which case, it is not available. If you go back and change something that invalidates something else, use the red shadow as a guide to see what action needs to be done next.

Non-PSDsoft Express Actions

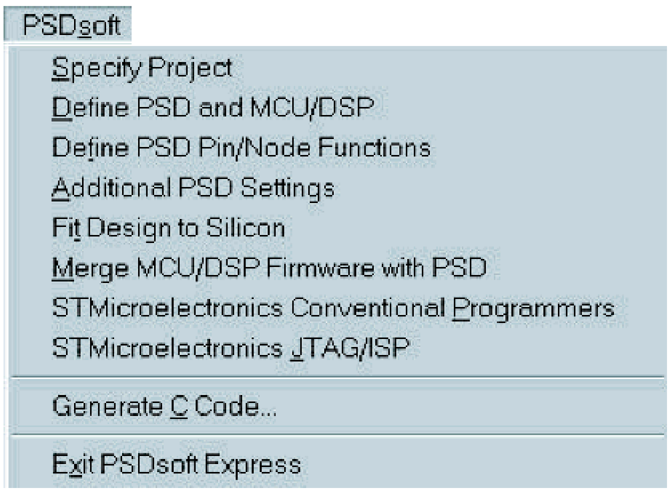
There are some actions that are not supported by PSDsoft Express, yet are part of the design process specific to your MCU/DSP. These boxes are shown in the flow, but are always grayed out and have no shadow. The three boxes that will always be grayed out are:

- "Your Application C Code or Assembly"—this is the source code and libraries specific to your application, generated outside PSDsoft Express.
- "Editor, Compiler, Linker, Debugger"—this is your MCU/DSP firmware development environment, which is also not a part of PSDsoft Express. In this environment, you should tailor the C Code generated by PSDsoft Express to be integrated with your application firmware.
- "3rd Party Programmers"—if you wish to use a 3rd party programmer to program your PSD part, you would need the .obj file created by PSDsoft Express and the software supplied by the 3rd party vendor. For a complete list of ST qualified programmers, check the website; *www.st.com/psd*.

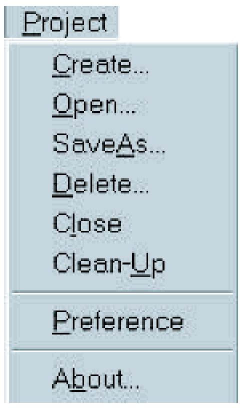
NAVIGATING THE MENUS

Although you can get through a design completely by only using the Design Flow, there are some features in the menus that you should be aware of.

Starting with the "PSDsoft" menu, if we look at the available options, we see that they exactly match what's available in the *Design Flow* (with the exception of "Exit PSDsoft Express").

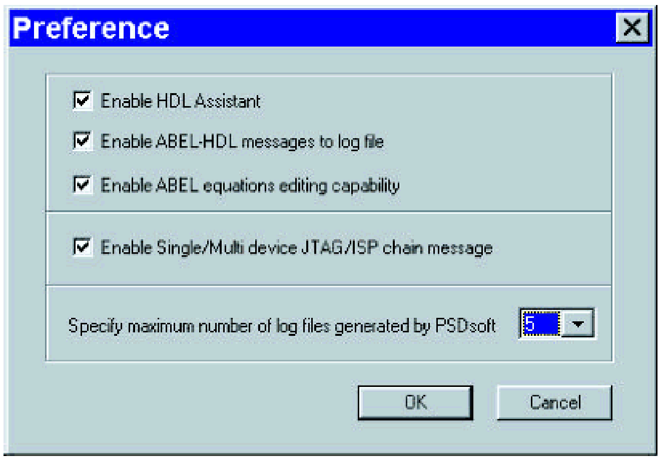

PSDsoft
Specify Project
Define PSD and MCU/DSP
Define PSD Pin/Node Functions
Additional PSD Settings
Fit Design to Silicon
Merge MCU/DSP Firmware with PSD
STMicroelectronics Conventional Programmers
STMicroelectronics JTAG/ISP
Generate C Code...
Exit PSDsoft Express

If we look at the next menu—"Project", we see the following:



While "Create", "Open", "Save As", "Delete", and "Close" are obvious, the other functions are as follows:

- “Clean-Up” gets rid of files generated by PSDsoft Express that are not essential for describing your project. You may wish to run this when you have a working programming data file (.obj) that you are satisfied with.
- “About…” gives the main details about your project and has a few more details than the information that is located in the lower right-hand-side of the PSDsoft Express main window.
- “Preferences” allows you to enable or disable optional features within PSDsoft Express. Depending on the PSD selected, some of the options shown below may not be available.


Preference

☑ Enable HDL Assistant

☑ Enable ABEL-HDL messages to log file

☑ Enable ABEL equations editing capability

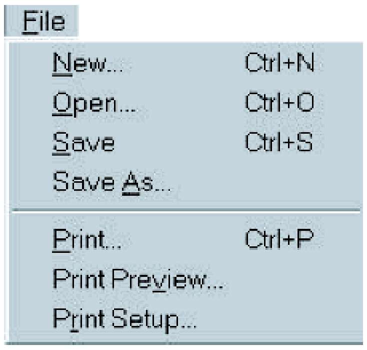
☑ Enable Single/Multi device JTAG/ISP chain message

Specify maximum number of log files generated by PSDsoft: 5

What do the preferences mean?

- “Enable HDL Assistant:” when this box is checked, a window will appear whenever the **Edit/Add Logic Statements** box is clicked on the *Design Flow*. This box allows you to check ABEL keywords and syntax and see some simple logic implementations. See the section “*About the HDL Assistant*”.
- “Enable ABEL-HDL messages to log file:” prints error and warning messages to the log file that is usually open in a window at the bottom of the screen. Most users will want to leave this box checked, with the only exception being if you get persistent warning messages that can safely be ignored
- “Enable ABEL equations editing capability:” checking this box makes the **Edit/Add Logic Statements** box appear in the design flow. See *Edit/Add Logic Statements (CPLD Parts Only)*.
- “Enable Single/Multi device JTAG/ISP chain message:” when this box is checked, PSDsoft Express will always prompt for how many devices are in your JTAG chain when the **STMicroelectronics JTAG/ISP** box is clicked in the Design Flow. If you will always have one or more than one device and had made this selection previously, you can disable the prompt by un-checking this box.
- The “Specify maximum number of log files generated by PSDsoft” selector can be used to set the maximum number of log files (.plg) to be kept in your project directory between one and ten. When the maximum number is reached, the next log file generated will replace the oldest log file.

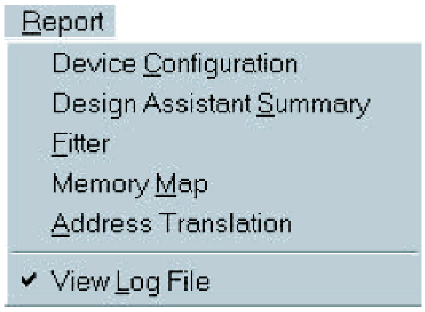
After Project, there is the “File” menu.



The File menu options work as follows:

- New…: creates a new text file for editing.
- Open…: opens an existing text file for editing
- Save, Save As…, Print, Print Preview, and Print Setup are all standard windows options.

The next menu over—“Report” has the following items:



This menu can be used to view and print the following reports:

- Device Configuration—basically shows how your PSD is configured at a glance
- Design Assistant Summary—summarizes how pins and equations were setup in the Design Assistant. This is the same file you would see if you clicked on the **View** button on any of the Design Assistant screens.
- Fitter—shows how your design was mapped to PSD silicon
- Memory Map—provides an overall picture of your memory map, including internal and external chip-select equations
- Address Translation—shows how your MCU/DSP memory map translates to physical PSD addresses and how the MCU/DSP *firmware* files are placed in PSD memory.

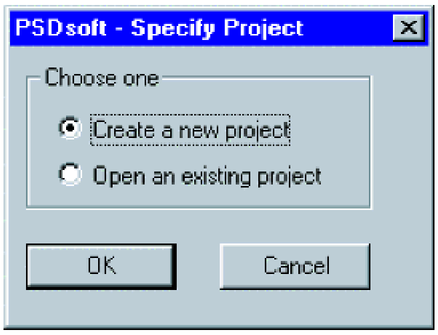
“View Log File” can be checked or unchecked. While checked, each action and its associated error messages (if any) will appear in a log file window normally located at the bottom of your screen. See the *Log File* section for more details.

The “eWindow” menu is straightforward and no explanation is required.

YOUR FIRST PROJECT

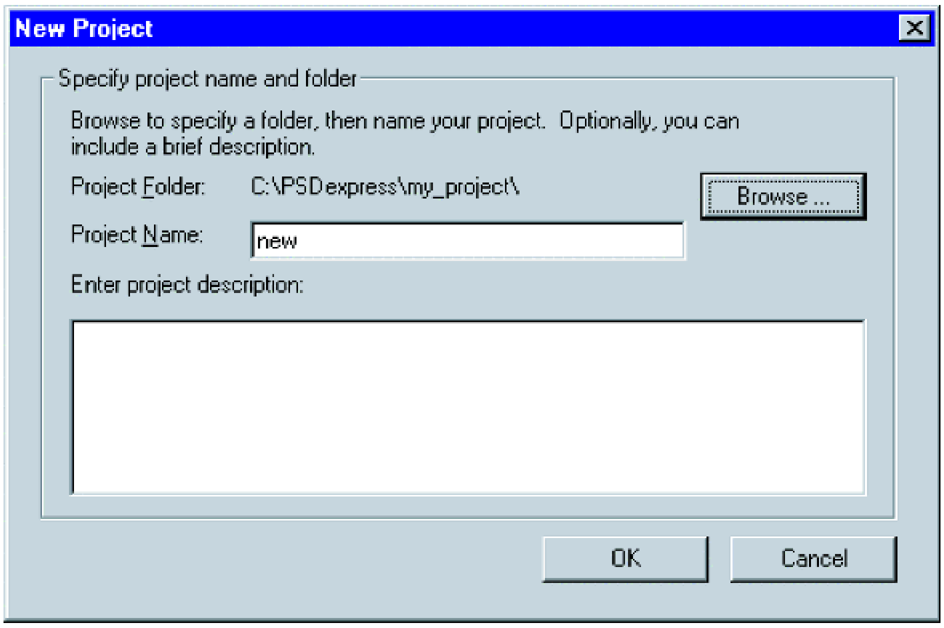
In this section, you will learn the complete PSDsoft Express design process. The steps of the design process are covered in greater detail in the sections following this one. However, there is information in this section not contained in any other sections.

When you double-click on the PSDsoft Express icon for the first time, you should see the following dialog box:



Choose “Create a new project” and click **OK**.

You will be presented with the following screen:



Browse to specify a folder, then name your project. Optionally, you can include a brief description.

Project Folder: C:\PSDexpress\my_project\

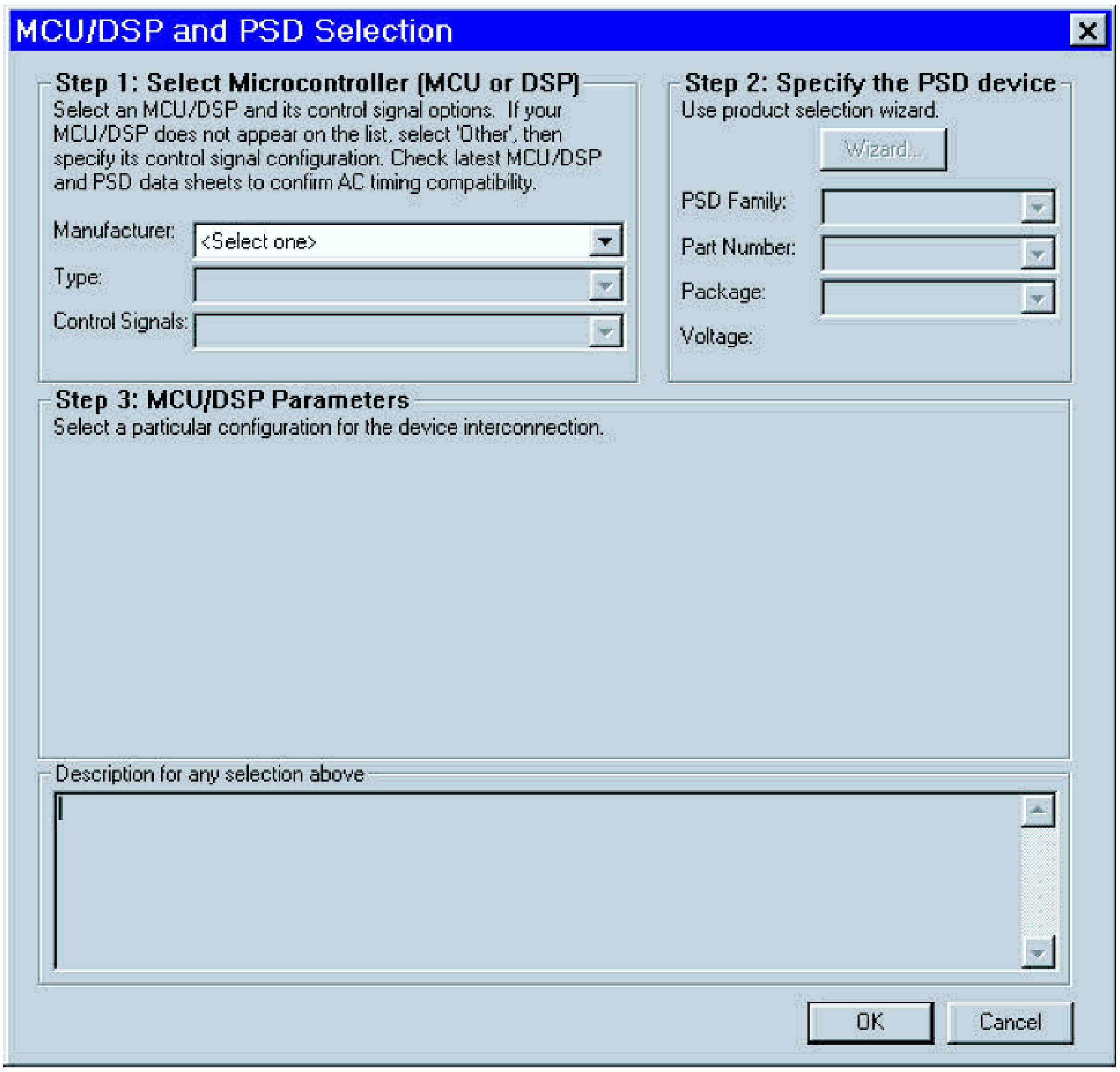
Project Name: new

Enter project description:

Choose an appropriate project name and location via the **Browse…** button. Enter a project description if desired.

Selecting and Configuring your PSD and Microcontroller

Once you click **OK**, you will be presented with the following dialog box:



MCU/DSP and PSD Selection

Step 1: Select Microcontroller (MCU or DSP)

Select an MCU/DSP and its control signal options. If your MCU/DSP does not appear on the list, select 'Other', then specify its control signal configuration. Check latest MCU/DSP and PSD data sheets to confirm AC timing compatibility.

Manufacturer: <Select one>

Type:

Control Signals:

Step 2: Specify the PSD device

Use product selection wizard.

PSD Family:

Part Number:

Package:

Voltage:

Step 3: MCU/DSP Parameters

Select a particular configuration for the device interconnection.

Description for any selection above

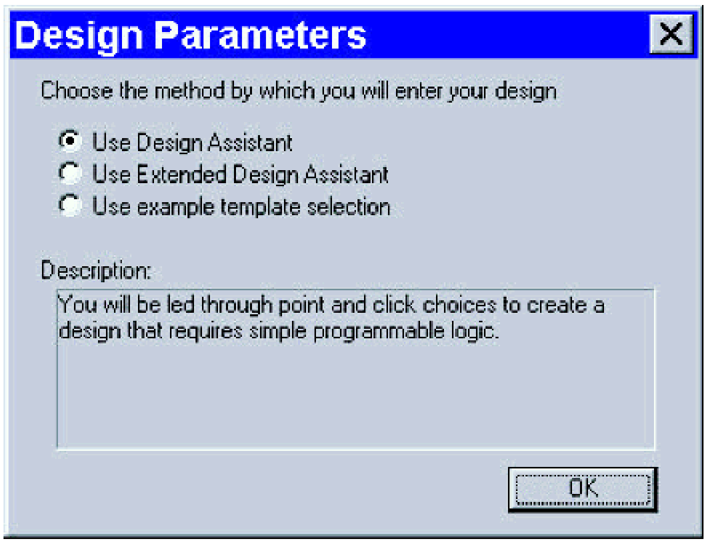
The first time through your project, this dialog box automatically prompts you for the next action. Take the following actions in this dialog box according to the numbered steps:

21. Select your microcontroller (MCU) or Digital Signal Processor (DSP) manufacturer and MCU/DSP type from the list available. If your particular selection is not available, select one that is compatible if possible. If your MCU/DSP is not listed and/or there is no compatible type listed, you must select “<Other>”. If the MCU/DSP manufacturer and type were on the list, some or all of the control signals will automatically be selected for you.
22. Select the PSD family, part, and package you wish to use. If you are unsure of which PSD best fits your application, click on the **Wizard** button and step through the Wizard.
23. In this step, you define the way the MCU/DSP and PSD interact. That is, 8-bit or 16-bit bus mode, multiplexed or non-multiplexed bus, and so on. Based on your MCU/DSP selection, these items may already be selected.

For more information on how to use this screen, go to the *MCU/DSP and PSD Selection* section.

Important note: many of the screen captures in this document will look different based on the MCU/DSP and especially the PSD family you have chosen. In most cases, those differences will be noted.

Once you have finished with this screen, click **OK** to be taken to the next screen:



Design Parameters

Choose the method by which you will enter your design

[✓ Use Design Assistant]
Use Extended Design Assistant
Use example template selection

Description:
You will be led through point and click choices to create a design that requires simple programmable logic.

Note: you will not see the “Use example template selection” option if you have chosen an MCU/DSP that does not have an associated template and you will not see the “Use Extended Design Assistant” selection if you had selected a PSD that has no *CPLD*. Selecting the Design Assistant allows you to do a more generic design in which only the PSD pins that connect to the MCU/DSP are pre-defined and configured.

The Extended Design Assistant provides HDL-specific examples and help for users who have more complicated logic requirements, such as state machines and comparators.

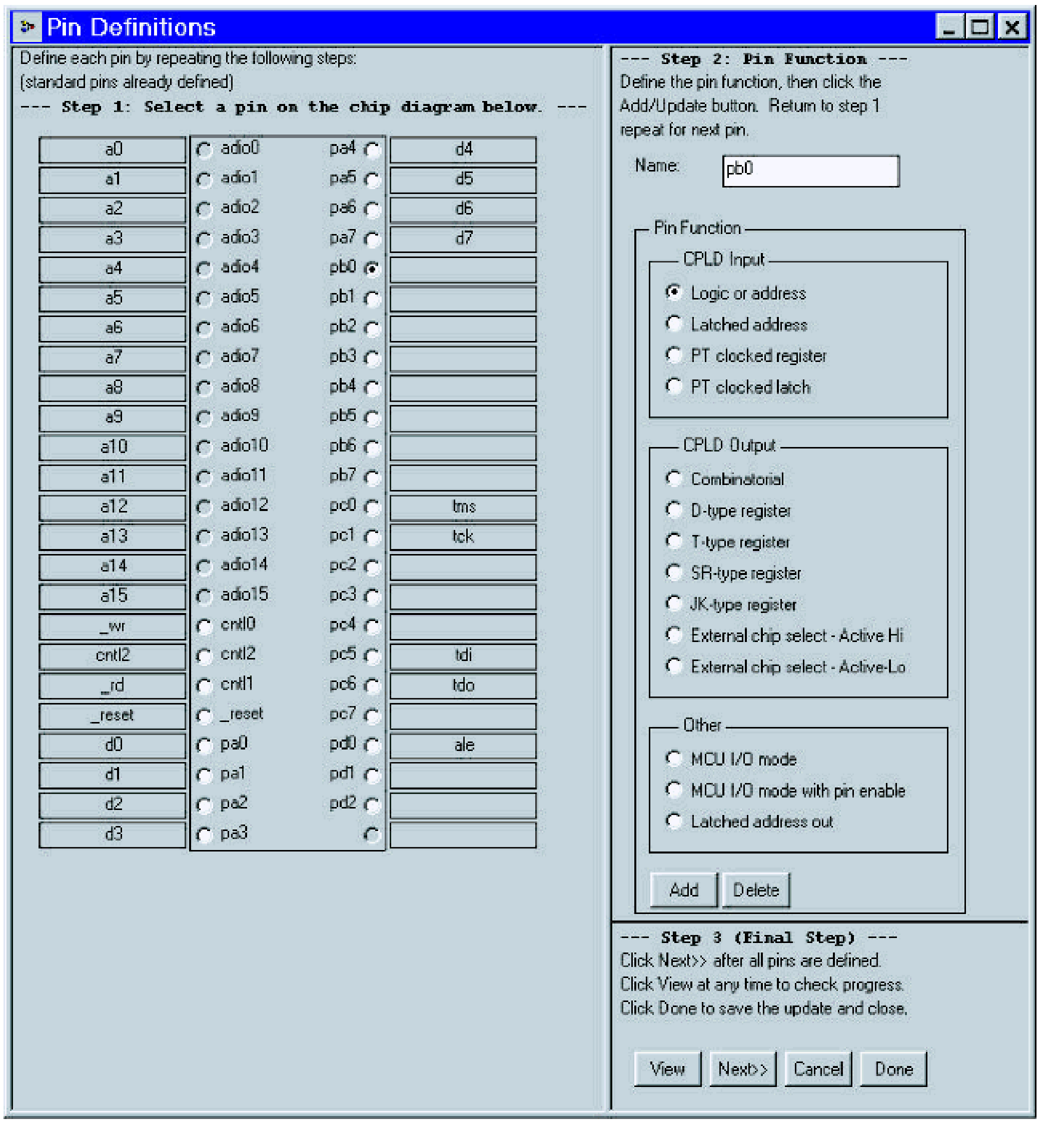
Selecting an example template gives you an example design based on the particular MCU/DSP that you’ve selected. A suggested memory map and I/O examples are included, and only minor additions are needed to suit your particular design in most cases.

If you wish to use the Design Assistant or Extended Design Assistant, continue reading. If you would like to use an example template, go to the section on *Using Example Templates*.

Pin Definitions

The following screen appears automatically on the first pass through your project or if you click on **Define**

PSD Pin/Node Functions in the “Design Flow” window.



Pin Definitions

Define each pin by repeating the following steps:
(standard pins already defined)
--- Step 1: Select a pin on the chip diagram below. ---

a0	adio0	pa4	d4
a1	adio1	pa5	d5
a2	adio2	pa6	d6
a3	adio3	pa7	d7
a4	adio4	[✓ pb0]	
a5	adio5	pb1	
a6	adio6	pb2	
a7	adio7	pb3	
a8	adio8	pb4	
a9	adio9	pb5	
a10	adio10	pb6	
a11	adio11	pb7	
a12	adio12	pc0	tms
a13	adio13	pc1	tck
a14	adio14	pc2	
a15	adio15	pc3	
_wr	cntl0	pc4	
cntl2	cntl2	pc5	tdi
_rd	cntl1	pc6	tdo
_reset	_reset	pc7	
d0	pa0	pd0	ale
d1	pa1	pd1	
d2	pa2	pd2	
d3	pa3		

--- Step 2: Pin Function ---
Define the pin function, then click the Add/Update button. Return to step 1 repeat for next pin.

Name: pb0

Pin Function

CPLD Input
- [✓ Logic or address]
- Latched address
- PT clocked register
- PT clocked latch

CPLD Output
- Combinatorial
- D-type register
- T-type register
- SR-type register
- JK-type register
- External chip select - Active Hi
- External chip select - Active-Lo

Other
- MCU I/O mode
- MCU I/O mode with pin enable
- Latched address out

--- Step 3 (Final Step) ---
Click Next>> after all pins are defined.
Click View at any time to check progress.
Click Done to save the update and close.

In this screen, you’ll be defining the actual pin functions of the PSD. This is an iterative process in which you select a pin and then define the function. The steps are as follows:

1. Select the pin which you want to define
2. The list of available functions may change depending on which pin you select. Select the appropriate function, enter the desired name, and click **Add**. If you had previously defined the pin, “Add” becomes “Update”. To get a brief description of what the pin functions mean, see the *Pin Definitions* section (Section 8).
3. When you have finished defining all the pins, select **Next >>**. If you wish to view your progress, select **View**. If you had only been changing pin functions, and do not need to make any changes to the chip-select equations, select **Done**.

JTAG users note: if you intend to use the PSD’s JTAG port in non-multiplexed mode, ensure the JTAG pin names appear as shown above. See the *Pin Definitions* section for more information.

Notes:

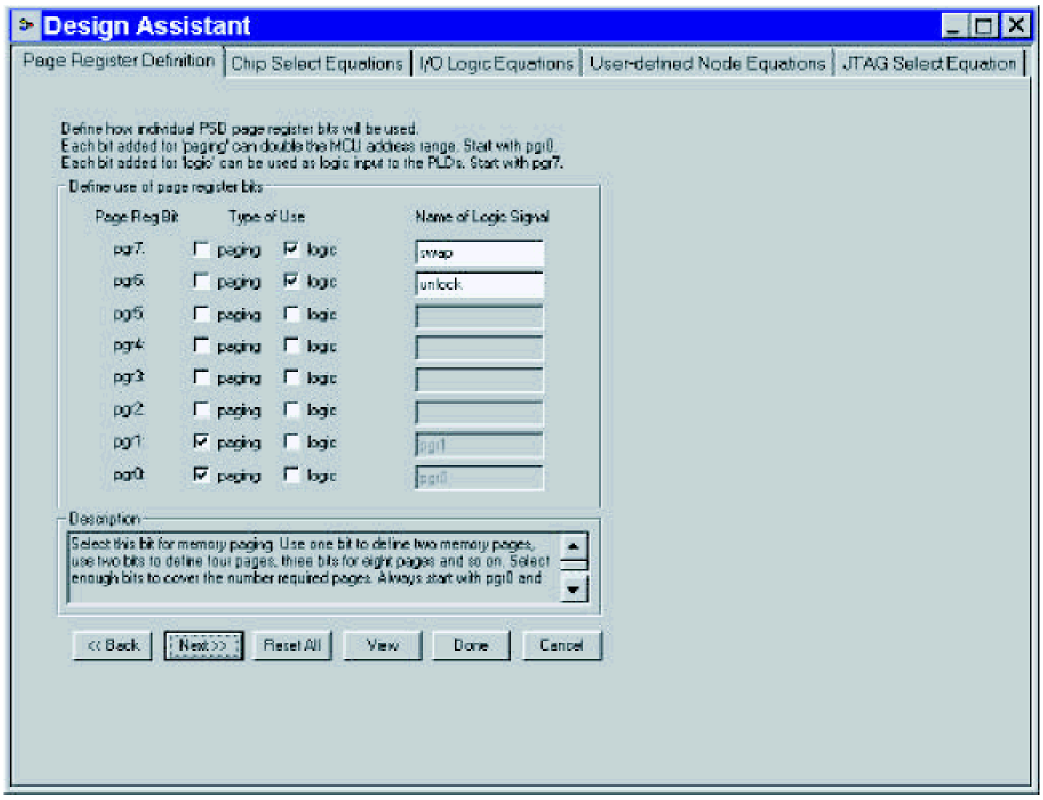
- Unless you have chosen a PSD8XX family part, the available pin functions will be different and thus the screen will look different.
- You can resize the Pin Definitions window and the new size you choose will be remembered on subsequent uses.
- If you want to change the pin association of a previously defined function, take the following steps:

1. Select the old pin assignment and hit the **Delete** button.
2. Select the new pin that the previously defined function will be assigned to.
3. Assign the same type of pin function that was used previously.
4. Name the pin exactly the same name as defined for the old pin.

For example, looking at the screen capture above, "ram_cs1" is defined on pin pa0 as an "External chip select – Active Hi". You can now select pa0, hit the **Delete** button, select a different pin (say pb0), select "External chip select – Active Hi", name the pin "ram_cs1", and click **Add**. Your old equation for "ram_cs1" will still be valid.

Page Register Setup

After clicking **Next >>** from the "Pin Definitions" window, you are presented with the "Page Register Definition" window:



Define how individual PSD page register bits will be used.
Each bit added for 'paging' can double the MCU address range. Start with pgr0.
Each bit added for 'logic' can be used as logic input to the PLDs. Start with pgr7.

Define use of page register bits

Page Reg Bit	Type of Use		Name of Logic Signal
pgr7	paging	✓ logic	swap
pgr6	paging	✓ logic	unlock
pgr5	paging	logic	
pgr4	paging	logic	
pgr3	paging	logic	
pgr2	paging	logic	
pgr1	✓ paging	logic	pgr1
pgr0	✓ paging	logic	pgr0

Description

Select this bit for memory paging. Use one bit to define two memory pages, use two bits to define four pages, three bits for eight pages and so on. Select enough bits to cover the number required pages. Always start with pgr0 and

The page register is ideal for MCUs/DSPs with limited address space. It can be used to extend the physical address space or used for logic inputs to the PLDs. If you plan to use any PSD Page Register bits in your design, they must be defined here before proceeding. Note: clicking **<< Back** returns you to the "Pin Definitions" screen.

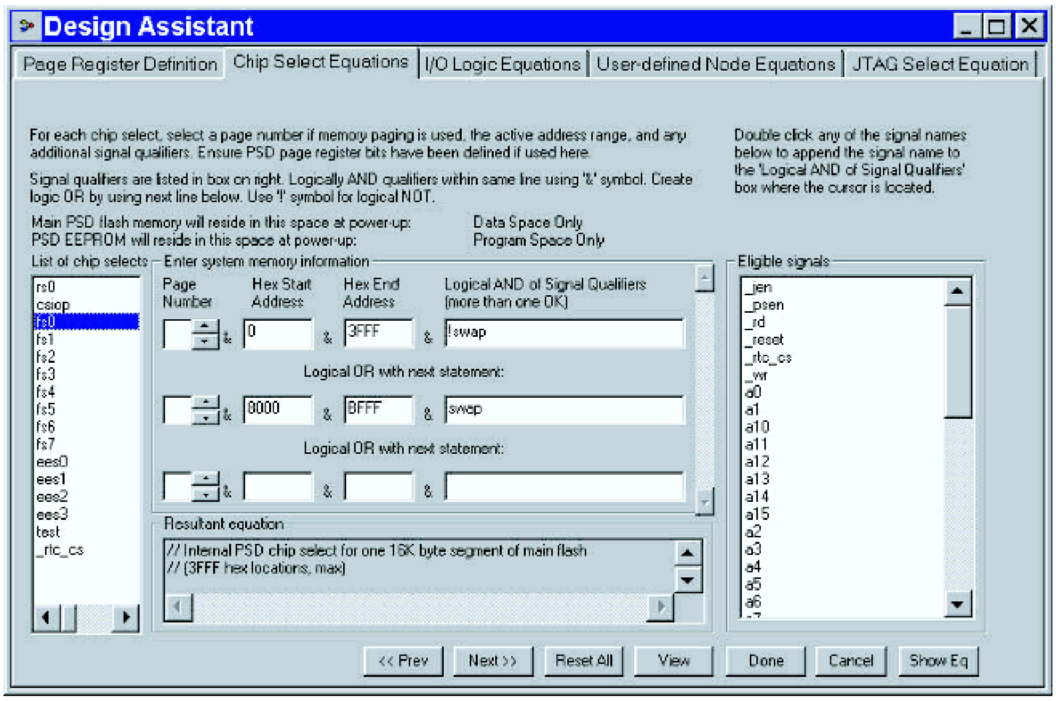
To find out more about the Page Register, what paging is, and if you need paging in your design, see the

Page Register section and the *What is Paging* section.

Important Note: If you selected a PSD with 64 KByte segments (currently only PSD835 and PSD935) and an MCU limited to 64 KByte total address space, you will most likely need to set up a memory page that can be used to control a node called “fa15”. The bit is used to select between 32 KByte halves of a 64 KByte segment. See the subsection on *Assigning two hex files to a Single Memory Segment.*

Chip Select Equations/Memory Map Definition

Upon clicking **Next >>** (or clicking the “Chip Select Equations” tab), you are taken to the “Chip Select Equations” screen:



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

Main PSD flash memory will reside in this space at power-up: Data Space Only
PSD EEPROM will reside in this space at power-up: Program Space Only

List of chip selects: rs0, csiop, fs0, fs1, fs2, fs3, fs4, fs5, fs6, fs7, ees0, ees1, ees2, ees3, test, _rtc_cs

Enter system memory information

Page Number		Hex Start Address		Hex End Address		Logical AND of Signal Qualifiers (more than one OK)
	&	0	&	3FFF	&	!swap
Logical OR with next statement:						
	&	8000	&	BFFF	&	swap
Logical OR with next statement:						
	&		&		&	

Resultant equation

```
// Internal PSD chip select for one 16K byte segment of main flash
// (3FFF hex locations, max)
```

Eligible signals: _jen, _psen, _rd, _reset, _rtc_cs, _wr, a0, a1, a10, a11, a12, a13, a14, a15, a2, a3, a4, a5, a6

In this window, you define the internal and external chip-select equations. To do so, take the following steps:

1. Select the desired chip-select segment in the left-most column.
2. Select the page number for which the chip-select will be valid (if applicable). Then enter the address range in hexadecimal for which the chip-select will be active. You may logically AND signals by placing them in the next column. You may also logically OR signals by entering parameters on consecutive lines.
3. Repeat the above steps until you have defined all your chip-select equations.

For example, if you wanted the internal Flash memory select segment 0 (fs0) to be valid from 0h to 3FFFh AND when “swap” is low OR valid from 8000h to BFFFh AND “swap” is high, you would enter the information as shown above. There is no need to qualify chip-select of internal PSD memory segments with MCU/DSP control signals as this is taken care of at the silicon level. However, you will need to qualify external chip-select signals with the appropriate MCU/DSP control signals (such as _RD and _WR).

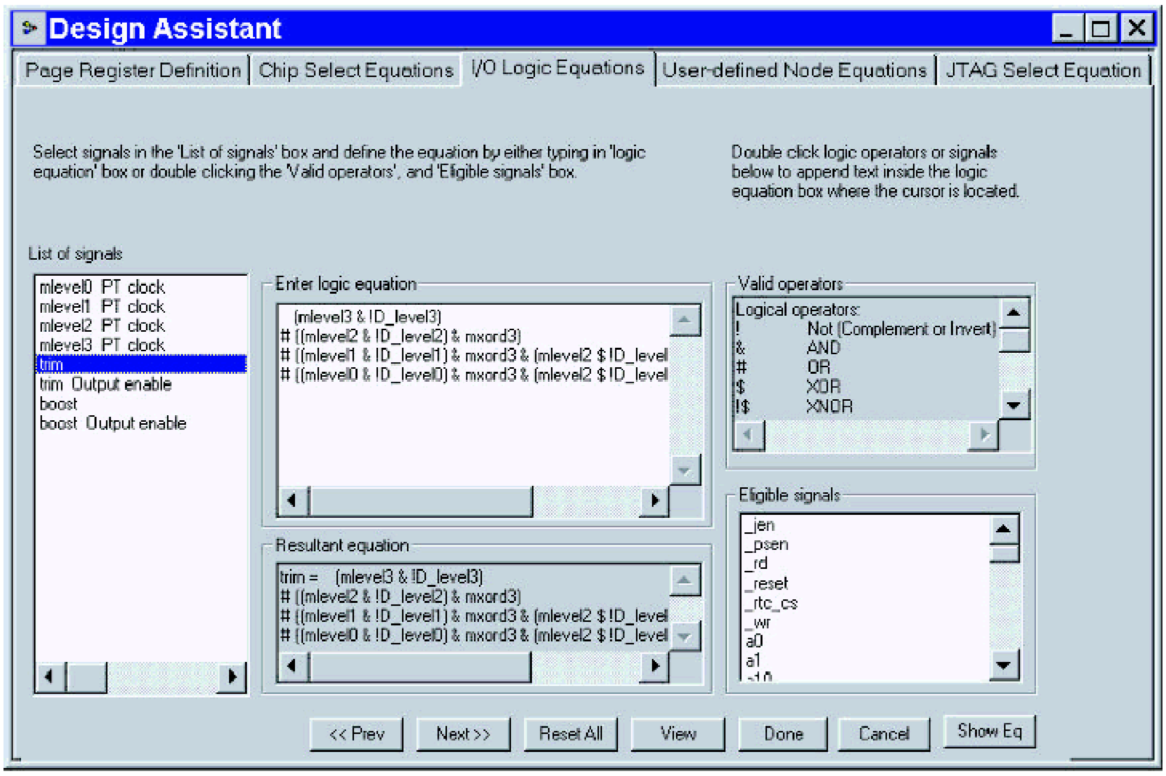
Note: to add a signal from the list of eligible signals, double click on the name you wish to add within the list and it will be added for you. Doing so a second time on the same line will also add the “&” symbol so that the signals are ANDed.

For more information, see the *Chip Select Equations* section (Section 10). If you have a *non-CPLD* part, you will not see the other tabs for defining other types of equations and can click **Done** and skip to *Addi-*

tional PSD Settings. If you are ready to move on, click **Next >>.** If you have a *CPLD* part and are not sure what type of equation you should be using, see more help in "*What's the Difference between Chip Select, I/O Logic, and Node Equations?"* (Section 19).

I/O Logic Equations (CPLD Parts Only)

Upon clicking **Next >>** (or clicking the "I/O Logic Equations" tab), you'll see a screen similar to the following:



Select signals in the 'List of signals' box and define the equation by either typing in 'logic equation' box or double clicking the 'Valid operators', and 'Eligible signals' box.

Double click logic operators or signals below to append text inside the logic equation box where the cursor is located.

```
mlevel0  PT clock
mlevel1  PT clock
mlevel2  PT clock
mlevel3  PT clock
trim
trim  Output enable
boost
boost  Output enable
```

```
  (mlevel3 & !D_level3)
# ((mlevel2 & !D_level2) & mxord3)
# ((mlevel1 & !D_level1) & mxord3 & (mlevel2 $ !D_level
# ((mlevel0 & !D_level0) & mxord3 & (mlevel2 $ !D_level
```

```
trim =    (mlevel3 & !D_level3)
# ((mlevel2 & !D_level2) & mxord3)
# ((mlevel1 & !D_level1) & mxord3 & (mlevel2 $ !D_level
# ((mlevel0 & !D_level0) & mxord3 & (mlevel2 $ !D_level
```

```
Logical operators:
!         Not (Complement or Invert)
&         AND
#         OR
$         XOR
!$        XNOR
```

```
_jen
_psen
_rd
_reset
_rtc_cs
_wr
a0
a1
```

Use this screen to define equations for the I/O within the PSD. If you need to use user-defined nodes in your equations, you will need to go to that screen first by clicking **Next>>** or clicking the "User-defined Node Equations" tab and clicking the **Def Node…** button within that window.

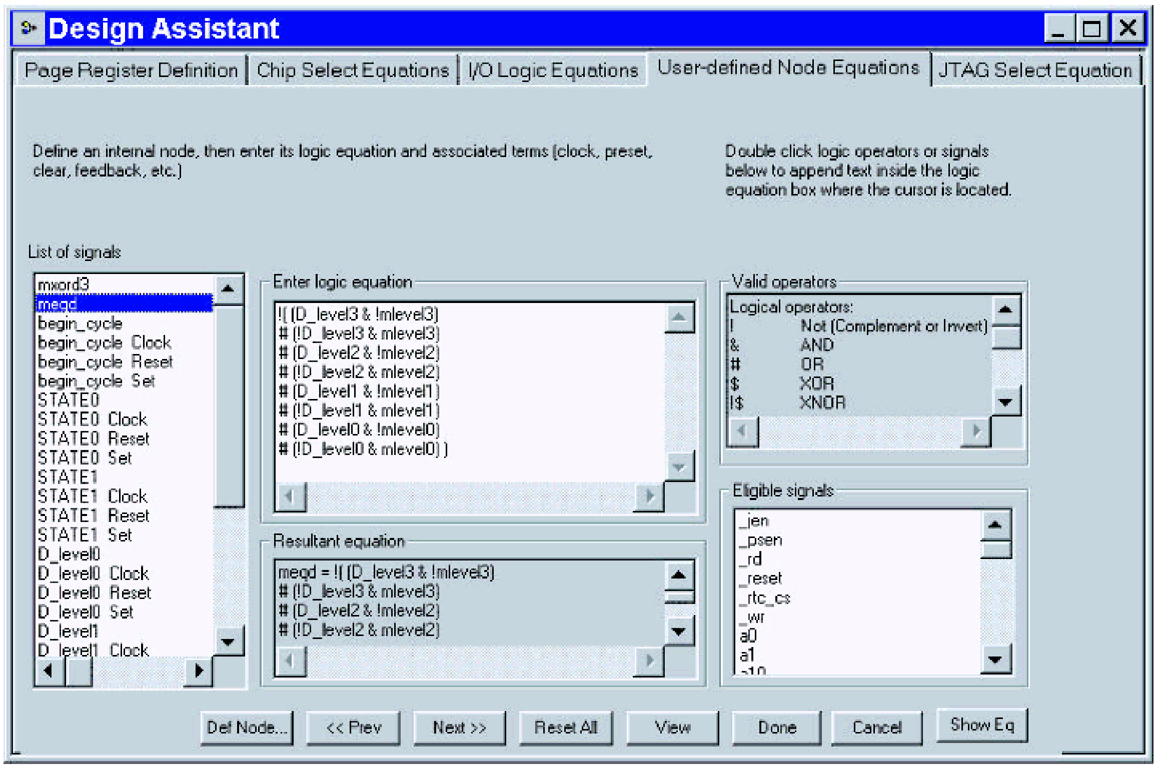
To define an I/O signal, highlight the signal to be defined from the "List of signals" box (on the far left) first by left-clicking on the signal. Then you can start typing in the equation in the logic equation in the "Enter logic equation" box, or the equation can be entered by double-clicking on the list of "Eligible signals". You can also enter logic operators or choose from the list of "Valid Operators" to the right by double-clicking on them and they will be entered for you. Use parenthesis to give priority to expression evaluation. You can separate product terms (ORs) or sum terms (ANDs) by placing them on separate lines for readability if desired. Once you have finished entering the equation you can click the **Show Eq** button and the result will be displayed in the "Resultant equation" box. (You will also see the resultant equation if you go on to define another signal and then come back to the original.) See the Fitter report to view the equations in final form by going to the **Report->Fitter** menu.

Note: the equal sign is implied and there is no need to type it in. In fact, you will get an error if you enter an equal sign.

For example, the signal "trim" had been previously defined as a combinatorial CPLD output pin. To define that signal, it was highlighted on the list as shown, and then the equation was entered as shown. The resultant equation is shown at the bottom.

User-Defined Node Equations (CPLD Parts Only)

Upon clicking **Next >>** (or clicking the “User-defined Node Equations” tab), you’ll see a screen similar to the following:



Define an internal node, then enter its logic equation and associated terms (clock, preset, clear, feedback, etc.)

Double click logic operators or signals below to append text inside the logic equation box where the cursor is located.

List of signals:

```
mxord3
meqd
begin_cycle
begin_cycle  Clock
begin_cycle  Reset
begin_cycle  Set
STATE0
STATE0  Clock
STATE0  Reset
STATE0  Set
STATE1
STATE1  Clock
STATE1  Reset
STATE1  Set
D_level0
D_level0  Clock
D_level0  Reset
D_level0  Set
D_level1
D_level1  Clock
```

Enter logic equation:

```
!( (D_level3 & !mlevel3)
# (!D_level3 & mlevel3)
# (D_level2 & !mlevel2)
# (!D_level2 & mlevel2)
# (D_level1 & !mlevel1 )
# (!D_level1 & mlevel1 )
# (D_level0 & !mlevel0)
# (!D_level0 & mlevel0) )
```

Resultant equation:

```
meqd = !( (D_level3 & !mlevel3)
# (!D_level3 & mlevel3)
# (D_level2 & !mlevel2)
# (!D_level2 & mlevel2)
```

Valid operators:

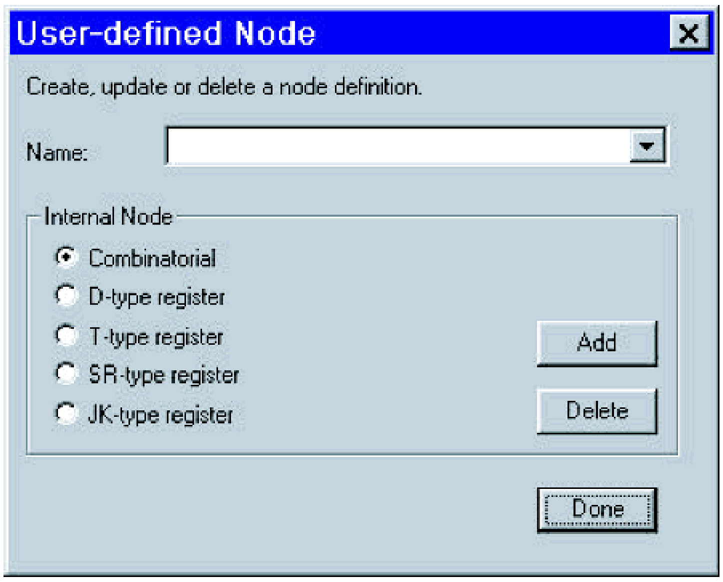
```
Logical operators:
!       Not (Complement or Invert)
&       AND
#       OR
$       XOR
!$      XNOR
```

Eligible signals:

```
_jen
_psen
_rd
_reset
_rtc_cs
_wr
a0
a1
a10
```

This screen is very similar in usage to the “I/O Logic Equation” screen.

To define a new node, or delete or change a previously defined node, click the **Def Node…** button. You will be presented with the following window:



Create, update or delete a node definition.

Name:

Internal Node

[✓ Combinatorial]
D-type register
T-type register
SR-type register
JK-type register

To define a new node, type in the desired name in the “Name:” box and choose the type (Combinatorial or D, T, SR, or JK flip-flop) and click **Add**. To change or delete an existing node, select the node from the pull-down list in the “Name:” box and either select a new function and click **Update** or simply click **Delete** to remove the node. When you have finished defining nodes, click **Done**.

To define an internal node’s functionality, highlight the node to be defined from the “List of signals” box first by left-clicking on the signal. Then you can start typing in the equation in the logic equation in the “Enter logic equation” box, or the equation can be entered by double-clicking on the list of “Eligible signals”. You can also enter logic operators or choose from the list of “Valid operators” to the right by double-clicking on them and they will be entered for you. Use parenthesis to give priority to expression evaluation. You can separate product terms (ORs) or sum terms (ANDs) by placing them on separate lines for readability if desired. Once you have finished entering the equation you can click the **Show Eq** button and the result will be displayed in the “Resultant equation” box. (You will also see the resultant equation if you go on to define another signal and then come back to the original.) See the Fitter report to view the equations in final form by going to the **Report->Fitter** menu.

Note: the equals sign is implied and there is no need to type it in. In fact, you will get an error if you enter an equal sign.

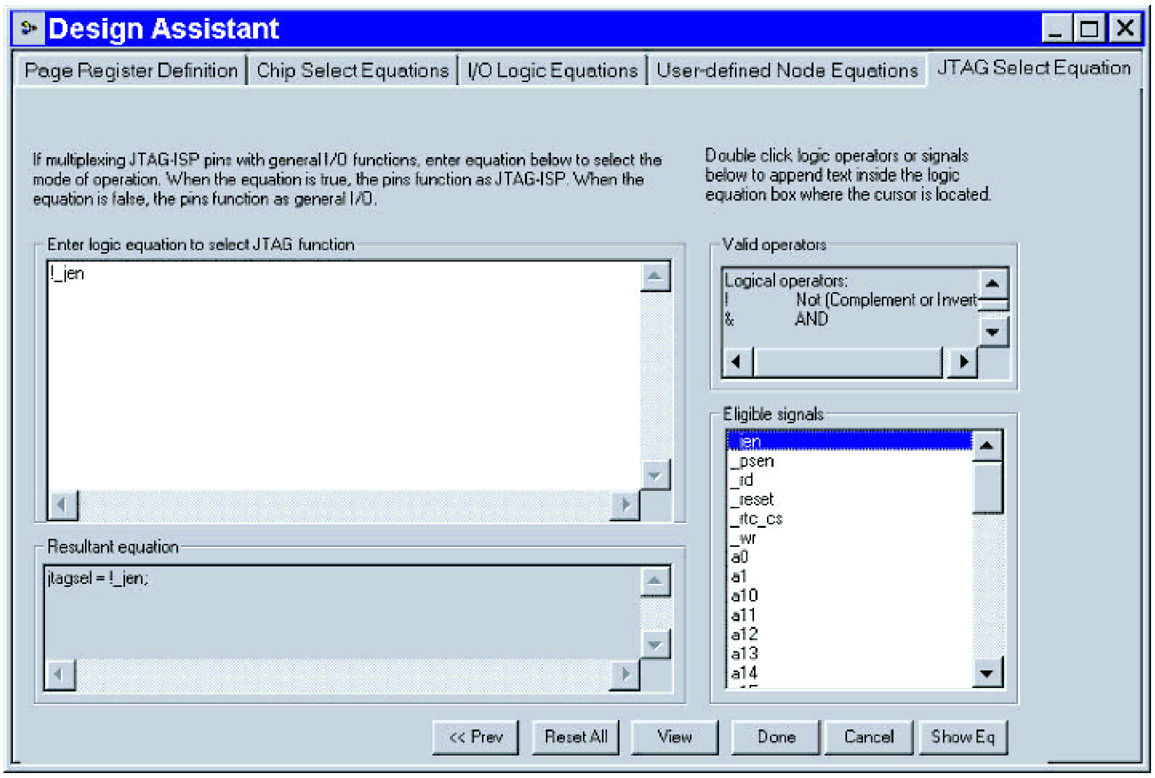
JTAG Select Equation (Parts with Multiplexed JTAG Ports Only)

If you do not see this tab and you have a part with a multiplexed JTAG port, you have dedicated the pins to fulltime JTAG use only (default). In other words, the only way this tab will appear is for parts with a multiplexed JTAG port where the JTAG pins have been defined with additional functionality using the Pin Definitions screen.

This screen is used to enable/disable the JTAG port via an I/O pin. This I/O pin gets connected to a header on your PCB, which in turn is connected to the PSD FlashLINK™ cable for JTAG operations. Then during a JTAG operation, the FlashLINK cable will drive the pin low and enable the JTAG port. The FlashLINK cable uses its /JEN signal to do this. Thus during normal (non-ISP) operation, this pin will remain high and the JTAG port will be disabled and other I/O can be connected to the JTAG port. So, if you intend to share the JTAG port with other I/O, you will need to define a pin in the *Pin Definitions* screen. Once you have your pin defined, you then use this screen to assign the pin you’ve defined to an internal node called “jtagsel”, which actually controls (turns on and off) the JTAG port.

For example, you have defined a port pin to be a “Logic or Address” CPLD input and named the pin _jen. On your circuit board you would connect _jen to the /JEN signal output from the FlashLINK cable. You would then assign _jen to the “jtagsel” node (in the “JTAG Select Equation” screen) by double-clicking on the not symbol (!) and then double-clicking on the _jen signal from the list of eligible signals on the right. Alternatively, you could simply type the equation in. Click the **Show Eq** button to see the resultant equation, which is “jtagsel = !_jen;” This equation basically says that the JTAG port will be enabled when there is a logic-0 on the pin that _jen was assigned to.

Since the JTAG port is a common area of confusion, it is <u>highly recommended</u> that you read both Application Note: “JTAG-ISP Information for Flash PSDs” and the *JTAG* section of the appropriate datasheet.



If multiplexing JTAG-ISP pins with general I/O functions, enter equation below to select the mode of operation. When the equation is true, the pins function as JTAG-ISP. When the equation is false, the pins function as general I/O.

Double click logic operators or signals below to append text inside the logic equation box where the cursor is located.

```
!_jen
```

```
jtagsel = !_jen;
```

```
Logical operators:
!        Not (Complement or Invert
&        AND
```

```
_jen
_psen
_rd
_reset
_rtc_cs
_wr
a0
a1
a10
a11
a12
a13
a14
```

Edit/Add Logic Statements (CPLD Parts Only)

If you have logic requirements that go beyond entering simple combinatorial equations using the Design Assistant, click the **Edit/Add Logic Statements** box in the Design Flow. Note: if you do not see this box in your Design Flow, go the **Project->Properties** menu and check the “Enable ABEL equations editing capability” box. If you intend to edit or add HDL logic statements as suggested in this section, it is recommended that you download Application Note: “Flash PSD, CPLD Primer”. It discusses how to use the PLDs within the PSDs for various purposes. If you have no intention of using this option, you can skip to the next section.

Very Important: if you intend to add any declarations or equations in the ABEL design file, they must be placed in the designated areas. Any declarations or equations placed outside the designated areas **will be lost**! Declarations are pin and node definitions, as well as signal groups that you intend to write equations for. Any item that appears before the *Equations* section is considered a declaration. Equations are the I/Os and nodes that get defined based on the behavior of other signals. For a complete guide to the ABEL language, see the *PSDabel-HDL Reference* downloadable from the Software Center on the *www.st.com/psd* or you can see examples using the *HDL Assistant*.

The two designated areas have the following look:

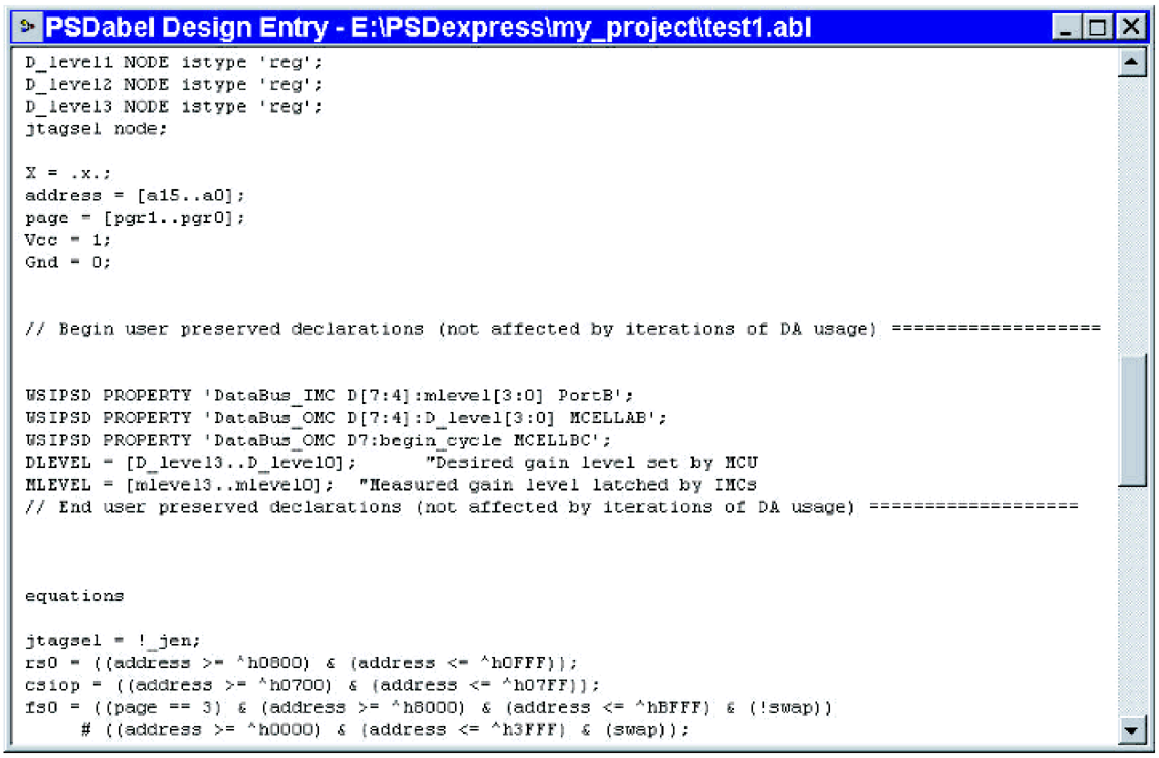
For declarations:

```
// Begin user preserved declarations (not affected by iterations of DA usage)
// End user preserved declarations (not affected by iterations of DA usage)
```

For equations:

```
// Begin user preserved equations (not affected by iterations of DA usage)
// End user preserved equations (not affected by iterations of DA usage)
```

An example ABEL file is shown below with some sample declarations. Note that the menus change when the ABEL file is opened.



```
D_level1 NODE istype 'reg';
D_level2 NODE istype 'reg';
D_level3 NODE istype 'reg';
jtagsel node;

X = .x.;
address = [a15..a0];
page = [pgr1..pgr0];
Vcc = 1;
Gnd = 0;


// Begin user preserved declarations (not affected by iterations of DA usage) ===================


WSIPSD PROPERTY 'DataBus_IMC D[7:4]:mlevel[3:0] PortB';
WSIPSD PROPERTY 'DataBus_OMC D[7:4]:D_level[3:0] MCELLAB';
WSIPSD PROPERTY 'DataBus_OMC D7:begin_cycle MCELLBC';
DLEVEL = [D_level3..D_level0];      "Desired gain level set by MCU
MLEVEL = [mlevel3..mlevel0];  "Measured gain level latched by IMCs
// End user preserved declarations (not affected by iterations of DA usage) ===================



equations

jtagsel = !_jen;
rs0 = ((address >= ^h0800) & (address <= ^h0FFF));
csiop = ((address >= ^h0700) & (address <= ^h07FF));
fs0 = ((page == 3) & (address >= ^h8000) & (address <= ^hBFFF) & (!swap))
     # ((address >= ^h0000) & (address <= ^h3FFF) & (swap));
```

For more help on ABEL syntax, additional available menus, and when to use ABEL versus when to use the Design Assistant, see the *Edit/Add ABEL Equations* section (Section 11).

About the Simulator. There is a simulator that comes with PSDsoft Express that will simulate the PLD equations only. In order to invoke the PSDsoft Express Simulator, you must have valid test vectors within the designated section for equations. To find out what the test vector syntax is, see the section on *ABEL Equations*. To run the simulator, go to **Report->Simulate Results** with the ABEL file open.

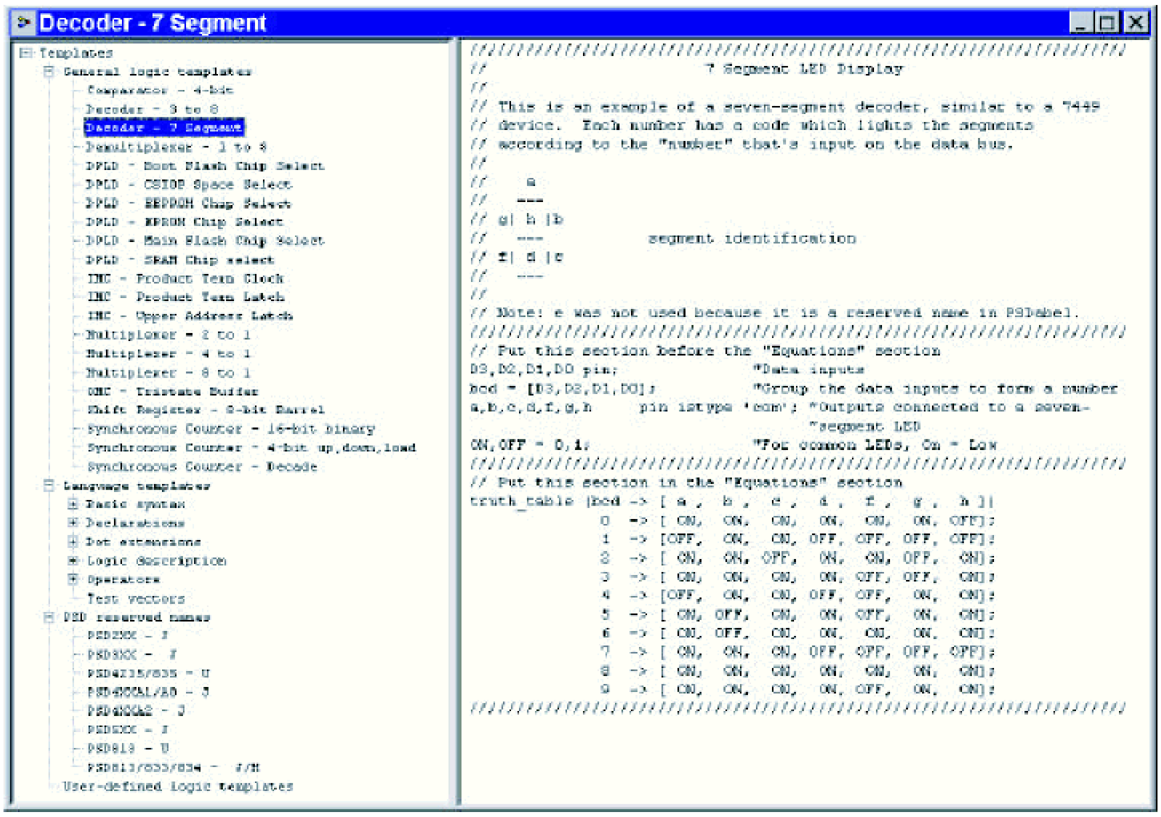
About the HDL Assistant. The HDL Assistant should open automatically whenever you click on the **Edit/Add Logic Statements** box in the Design Flow provided you checked the “Enable HDL Assistant” in the **Project->Preferences** menu. The HDL Assistant has three basic purposes:

- Lists the reserved names for each PSD device. Reserved names are for pins, nodes, and registers within the PSD that have a specific function within the PSD.
- ABEL language usage templates: shows all the basic elements of the ABEL language.
- General logic templates: shows how to implement common logic functions and PSD-specific declarations and functions.

There are two ways to use code within the HDL Assistant:

1. Select the desired template or reserved name from the list on the left. Then highlight the code on the right and paste it within the correct designated areas. For most of the General Logic Templates, it would be best to do two copy and paste actions. In other words, you will paste one portion into the *Declarations* section and cut the equations from that and paste them in the *Equations* section.
2. Place the cursor in one of the designated areas in your ABEL file. Then, select the desired template or reserved name from the list on the left and right-click and select **Use** from the pull-down menu that ap-

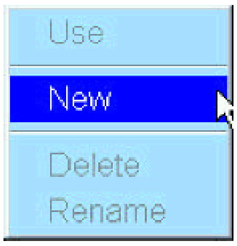
pears. Again, you may need to cut part of this code that appears in one of your designated areas and paste it into the other.



```
Templates
  General logic templates
    Comparator - 4-bit
    Decoder - 3 to 8
    Decoder - 7 Segment
    Demultiplexer - 1 to 8
    DPLD - Boot Flash Chip Select
    DPLD - CSIOP Space Select
    DPLD - EEPROM Chip Select
    DPLD - EPROM Chip Select
    DPLD - Main Flash Chip Select
    DPLD - SRAM Chip select
    IMC - Product Term Clock
    IMC - Product Term Latch
    IMC - Upper Address Latch
    Multiplexer - 2 to 1
    Multiplexer - 4 to 1
    Multiplexer - 8 to 1
    OMC - Tristate Buffer
    Shift Register - 8-bit Barrel
    Synchronous Counter - 16-bit binary
    Synchronous Counter - 4-bit up,down,load
    Synchronous Counter - Decade
  Language templates
    Basic syntax
    Declarations
    Dot extensions
    Logic description
    Operators
    Test vectors
  PSD reserved names
    PSD2XX - J
    PSD3XX - J
    PSD4235/835 - U
    PSD4XXAL/A8 - J
    PSD4XXA2 - J
    PSD5XX - J
    PSD813 - U
    PSD813/833/834 - J/M
  User-defined logic templates
```

```
////////////////////////////////////////////////////////////////////////////
//                         7 Segment LED Display
//
// This is an example of a seven-segment decoder, similar to a 7449
// device.  Each number has a code which lights the segments
// according to the "number" that's input on the data bus.
//
//     a
//    ---
// g| h |b
//    ---            segment identification
// f| d |c
//    ---
//
// Note: e was not used because it is a reserved name in PSDabel.
////////////////////////////////////////////////////////////////////////////
// Put this section before the "Equations" section
D3,D2,D1,D0 pin;                 "Data inputs
bcd = [D3,D2,D1,D0];             "Group the data inputs to form a number
a,b,c,d,f,g,h      pin istype 'com'; "Outputs connected to a seven-
                                        "segment LED
ON,OFF = 0,1;                    "For common LEDs, On = Low
////////////////////////////////////////////////////////////////////////////
// Put this section in the "Equations" section
truth_table (bcd -> [ a ,  b ,  c ,  d ,  f ,  g ,  h ])
              0  -> [ ON,  ON,  ON,  ON,  ON,  ON, OFF];
              1  -> [OFF,  ON,  ON, OFF, OFF, OFF, OFF];
              2  -> [ ON,  ON, OFF,  ON,  ON, OFF,  ON];
              3  -> [ ON,  ON,  ON,  ON, OFF, OFF,  ON];
              4  -> [OFF,  ON,  ON, OFF, OFF,  ON,  ON];
              5  -> [ ON, OFF,  ON,  ON, OFF,  ON,  ON];
              6  -> [ ON, OFF,  ON,  ON,  ON,  ON,  ON];
              7  -> [ ON,  ON,  ON, OFF, OFF, OFF, OFF];
              8  -> [ ON,  ON,  ON,  ON,  ON,  ON,  ON];
              9  -> [ ON,  ON,  ON,  ON, OFF,  ON,  ON];
////////////////////////////////////////////////////////////////////////////
```

If you want to define your own template, right-click on “User-defined logic templates” and select **New** from the pull-down menu that appears:



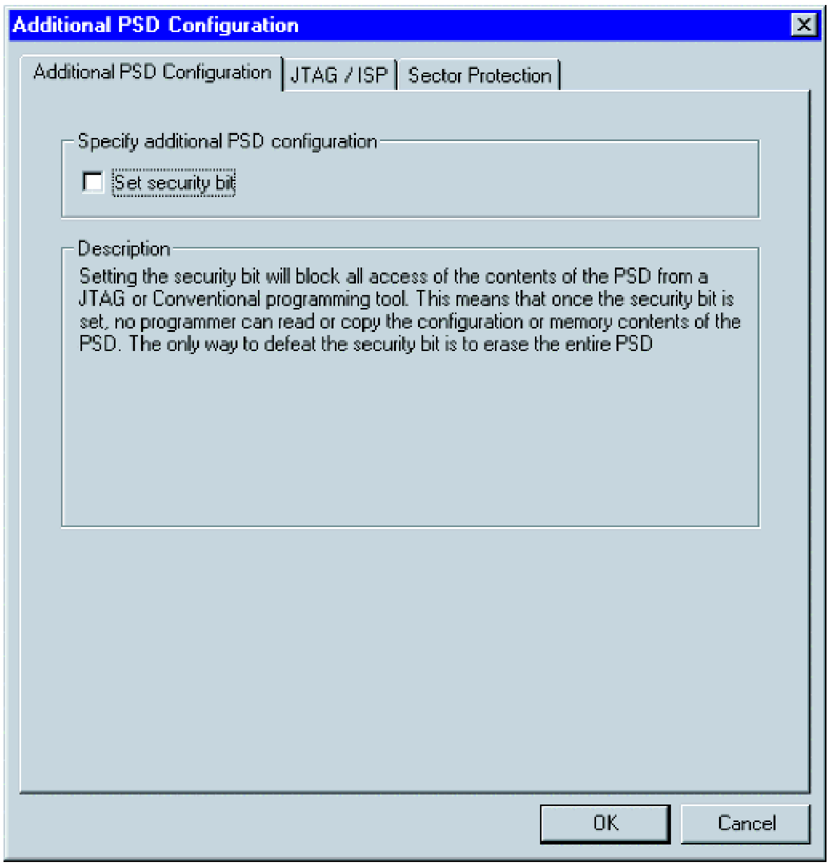
Delete and **Rename** can be used on a User-defined logic template only.

Additional PSD Settings

The next step in the design flow is **Additional PSD Settings**. This screen varies among PSD families.

By clicking on the **Additional PSD Settings** button in the Design Flow, you are presented with the following screen (unless you have a PSD part without Flash memory, in which case, see the *Additional PSD*

Settings section (Section 12) for more information):



Specify additional PSD configuration

☐ Set security bit

Description

Setting the security bit will block all access of the contents of the PSD from a JTAG or Conventional programming tool. This means that once the security bit is set, no programmer can read or copy the configuration or memory contents of the PSD. The only way to defeat the security bit is to erase the entire PSD

There are three functions that can be accomplished in this dialog box:

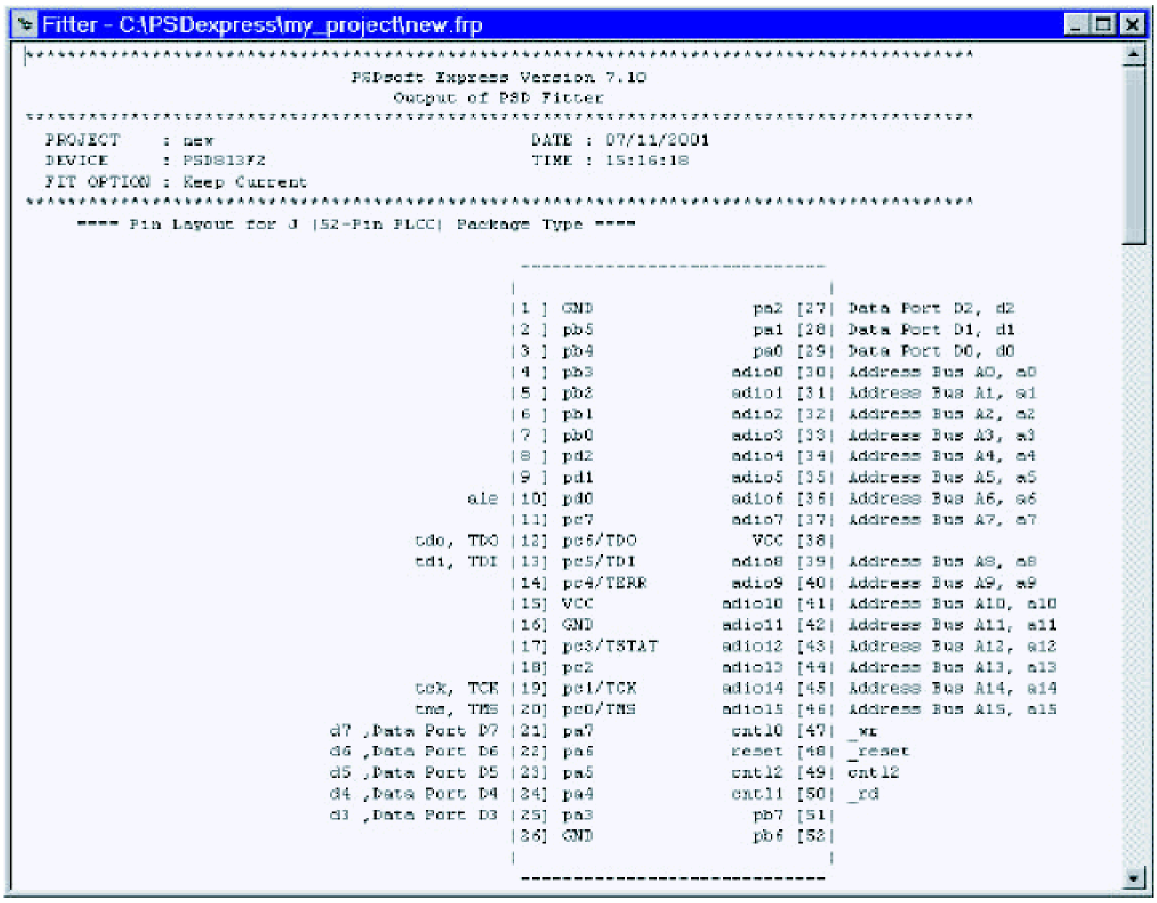
1. Setting the security bit—blocks all access to the contents of the PSD's memories by means of JTAG or a conventional programmer. That is, once the security bit is set, no programmer can read or copy the configuration or memory contents of the PSD. The only way to erase the security bit is to completely erase the PSD.
2. Specify the user code—allows you to enter a 32-bit code which can be used for various functions. Click on the “JTAG/ISP” tab and read the description box for more details.
3. Set the internal memories' sector protections—allows the individual memory sectors within the PSD to be write protected to prevent accidental data loss. The MCU/DSP cannot change these settings at run-time; only a device programmer can alter these settings.

Click **OK** and you will see the Design Flow again. For more information, see the *Additional PSD Settings* section (Section 12). Next, we need to attempt to fit the design to silicon.

Fitting the Design to Silicon

The PSDsoft Express Fitter maps your PLD design to PSD silicon. The Fitter automatically compiles any ABEL code you have placed in the designated sections and reports back any errors. If there are no errors, a partial Programming data file (.obj) will be generated. (To complete the .obj file, the *“Merge MCU/DSP Firmware with PSD”* process must be completed.)

To invoke the Fitter click the **Fit Design to Silicon** box in the Design Flow. You will notice that PSDsoft Express performs a fit of your design to silicon. If you get any errors in this step, you will need to go back and edit your design based on the given error. If you can't seem to figure out what's causing the error, contact *apps.psd@st.com*. If your design fits properly, you should see a screen similar to the one below (note: you may have to go to the **Report->Fitter** menu to see this file):



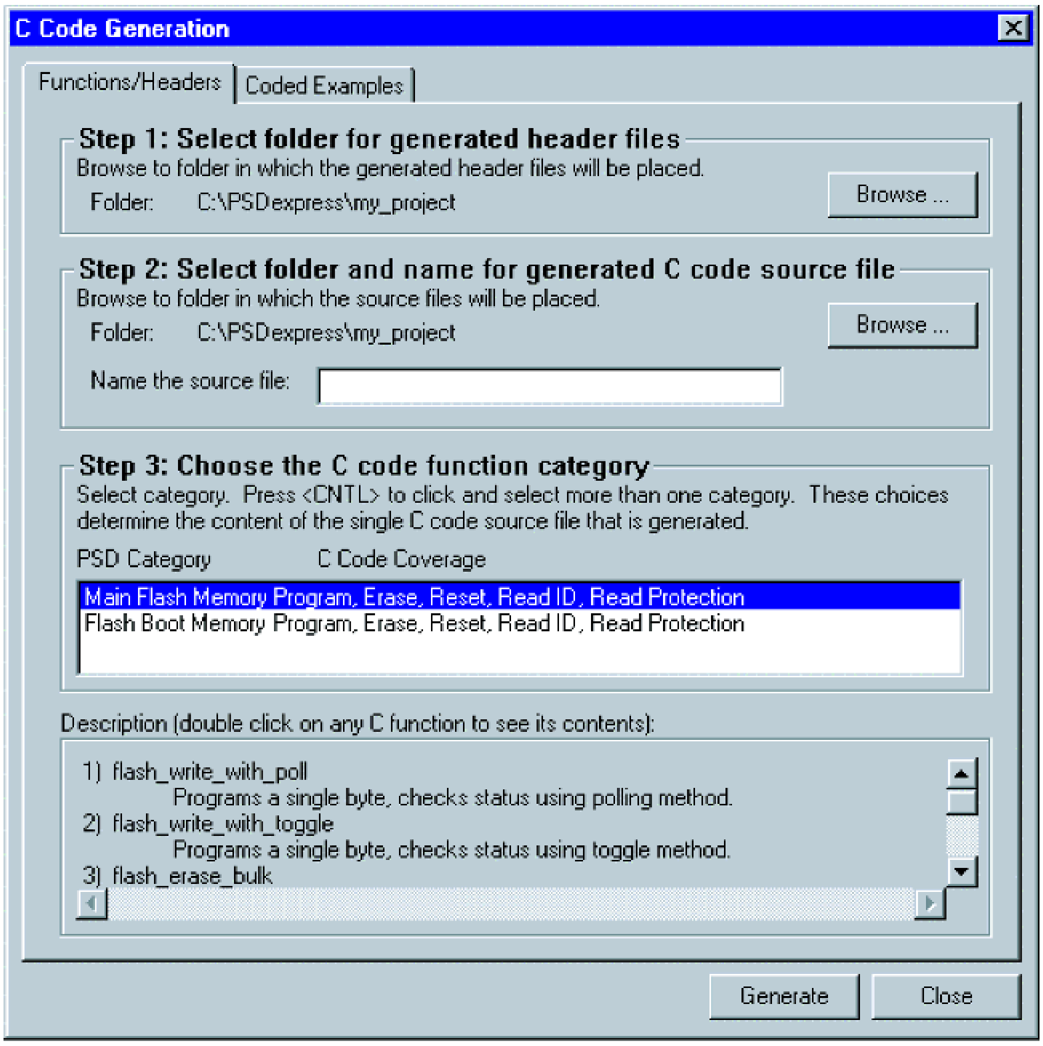
```
*****************************************************************************
                            PSDsoft Express Version 7.10
                               Output of PSD Fitter
*****************************************************************************
  PROJECT     : new                              DATE : 07/11/2001
  DEVICE      : PSD813F2                         TIME : 15:16:18
  FIT OPTION  : Keep Current
*****************************************************************************
    ==== Pin Layout for J |52-Pin PLCC| Package Type ====

                                        ------------------------
                                       |                        |
                                       |1 ] GND            pa2 [27| Data Port D2, d2
                                       |2 ] pb5            pa1 [28| Data Port D1, d1
                                       |3 ] pb4            pa0 [29| Data Port D0, d0
                                       |4 ] pb3          adio0 [30| Address Bus A0, a0
                                       |5 ] pb2          adio1 [31| Address Bus A1, a1
                                       |6 ] pb1          adio2 [32| Address Bus A2, a2
                                       |7 ] pb0          adio3 [33| Address Bus A3, a3
                                       |8 ] pd2          adio4 [34| Address Bus A4, a4
                                       |9 ] pd1          adio5 [35| Address Bus A5, a5
                                   ale |10] pd0          adio6 [36| Address Bus A6, a6
                                       |11] pc7          adio7 [37| Address Bus A7, a7
                              tdo, TDO |12] pc6/TDO        VCC [38|
                              tdi, TDI |13] pc5/TDI      adio8 [39| Address Bus A8, a8
                                       |14] pc4/TERR     adio9 [40| Address Bus A9, a9
                                       |15] VCC         adio10 [41| Address Bus A10, a10
                                       |16] GND         adio11 [42| Address Bus A11, a11
                                       |17] pc3/TSTAT   adio12 [43| Address Bus A12, a12
                                       |18] pc2         adio13 [44| Address Bus A13, a13
                              tck, TCK |19] pc1/TCK     adio14 [45| Address Bus A14, a14
                              tms, TMS |20] pc0/TMS     adio15 [46| Address Bus A15, a15
                     d7 ,Data Port D7 |21] pa7          cntl0 [47| _wr
                     d6 ,Data Port D6 |22] pa6          reset [48| _reset
                     d5 ,Data Port D5 |23] pa5          cntl2 [49| cntl2
                     d4 ,Data Port D4 |24] pa4          cntl1 [50| _rd
                     d3 ,Data Port D3 |25] pa3            pb7 [51|
                                       |26] GND            pb6 [52|
                                       |                        |
                                        ------------------------
```

This screen can be quite useful for seeing resources allocated and remaining and how the software interpreted your equations and design entry. For more information, see the *Fit Design to Silicon* section (Section 13).

Before you finish creating the .obj file that will be used to program the PSD, you may want to see how PSDsoft Express can help you generate some useful C code. If you have a non Flash memory based PSD or do not wish to use PSDsoft Express to generate C code, skip the sub-section on *Creating the File used to Program the PSD—the Programming data file* (Section 5.12).

Generating C Code (Flash/EEPROM PSDs Only)

Click on the **Generate C Code** in the Design Flow to get the following screen:



C Code Generation

Step 1: Select folder for generated header files

Browse to folder in which the generated header files will be placed.

Folder: C:\PSDexpress\my_project

Step 2: Select folder and name for generated C code source file

Browse to folder in which the source files will be placed.

Folder: C:\PSDexpress\my_project

Name the source file:

Step 3: Choose the C code function category

Select category. Press <CNTL> to click and select more than one category. These choices determine the content of the single C code source file that is generated.

PSD Category	C Code Coverage
Main Flash Memory	Program, Erase, Reset, Read ID, Read Protection
Flash Boot Memory	Program, Erase, Reset, Read ID, Read Protection

Description (double click on any C function to see its contents):

1) flash_write_with_poll
    Programs a single byte, checks status using polling method.
2) flash_write_with_toggle
    Programs a single byte, checks status using toggle method.
3) flash_erase_bulk

Using this utility within PSDsoft Express you can generate coded examples written in the C language to accomplish the following:

– Program and erase the PSDs non-volatile memories (NVMs)

– Reset the PSDs NVM(s) to “Read Array Mode”

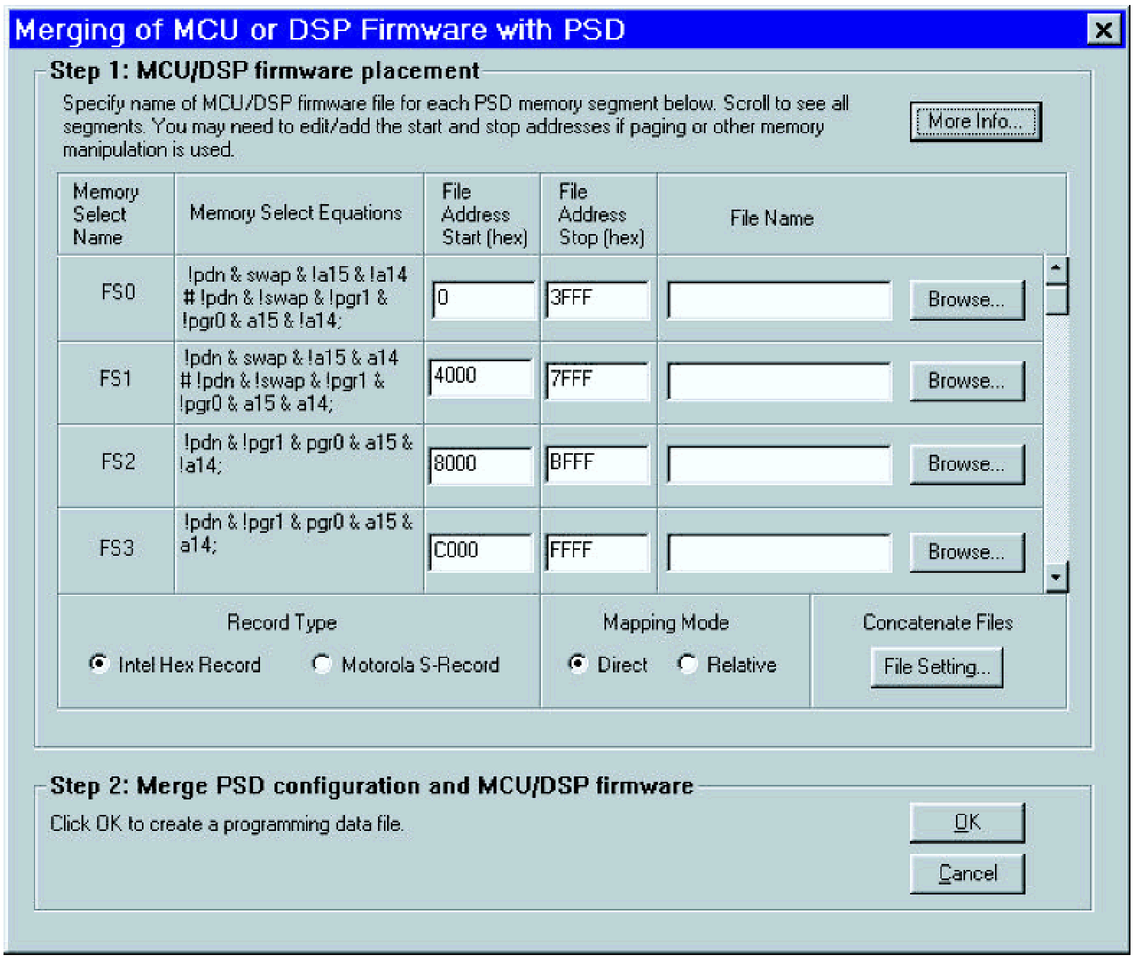
– Read the Flash memory identifier byte and read protection status.

When you are ready to have PSDsoft Express create the desired files, click the **Generate** button. Once the **Generate** button is clicked, the *log file (.plg)* will list the files generated. Note: for the “Functions/Headers” tab, you can double-click on any of the functions in the “Description” box to see a listing of the C code that is contained within the selected function. You can also see complete examples for use with PSD development kits by clicking on the “Coded Examples” tab. For more information and to see a list of generated files, see the *Generating C Code* section (Section 14).

Ok, we are now ready to generate the rest of programming data file (.obj) that will be used to program the PSD.

Creating the File used to Program the PSD—the Programming data file

Click on the **Merge MCU/DSP Firmware with PSD** button in the design flow. You should see a window similar to the following:


Merging of MCU or DSP Firmware with PSD

Step 1: MCU/DSP firmware placement

Specify name of MCU/DSP firmware file for each PSD memory segment below. Scroll to see all segments. You may need to edit/add the start and stop addresses if paging or other memory manipulation is used.

Memory Select Name	Memory Select Equations	File Address Start (hex)	File Address Stop (hex)	File Name
FS0	!pdn & swap & !a15 & !a14 # !pdn & !swap & !pgr1 & !pgr0 & a15 & !a14;	0	3FFF	
FS1	!pdn & swap & !a15 & a14 # !pdn & !swap & !pgr1 & !pgr0 & a15 & a14;	4000	7FFF	
FS2	!pdn & !pgr1 & pgr0 & a15 & !a14;	8000	BFFF	
FS3	!pdn & !pgr1 & pgr0 & a15 & a14;	C000	FFFF	

Record Type: [✓ Intel Hex Record] Motorola S-Record

Mapping Mode: [✓ Direct] Relative

Concatenate Files

Step 2: Merge PSD configuration and MCU/DSP firmware

Click OK to create a programming data file.

Note: you may have received a warning if you used the page register in your design. Basically, the warning is telling you that PSDsoft Express detected chip-select equations that do not contain natural MCU/DSP address signals. The warning is to remind you to ensure your MCU/DSP compiler/linker accounts for this. You may have to manually fill in the File start/stop addresses if they did not appear automatically. Also, for paged designs, check the Start/Stop addresses for validity because the internal checks performed by the software do not include the page bits.

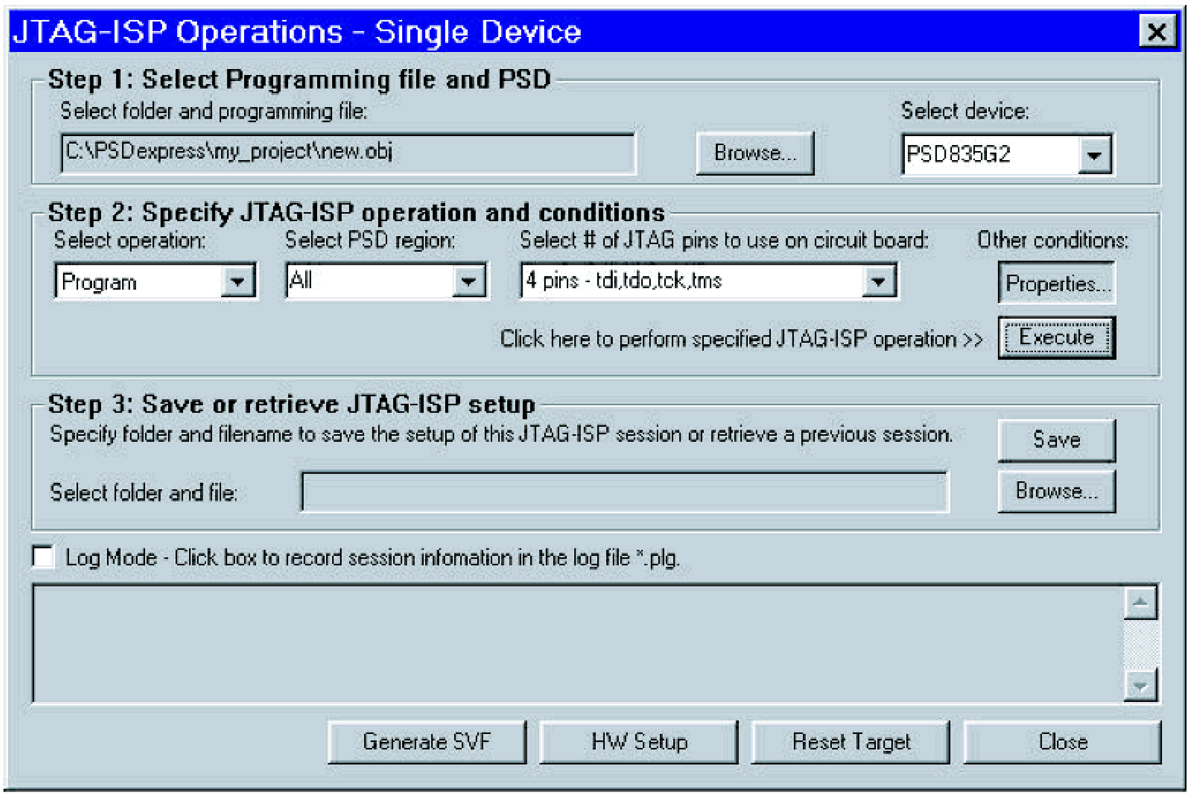
You should have Intel Hex Record or Motorola S-Record files ready to go at this point. For each memory segment within the PSD, simply click the appropriate **Browse** button and locate the file that you want programmed in that segment. Note that one file may span multiple segments. Remember to scroll down to fill in all the file names as necessary. Select the appropriate record type, based on the output of your compiler/linker to either “Intel Hex Record” or “Motorola S-Record”. Next, select the desired mapping mode. “Direct” mapping implies there is a one-to-one correspondence with the Hex files and PSD memory segment locations. “Relative” mapping enables you to specify different physical addresses (output by your MCU/DSP based on your firmware file) than the specified equations that select the memory segments within the PSD. That is, the Hex file will be placed at the bottom of the specified PSD segment. Most users will select “Direct”. When you have finished, click **OK**, and the programming data file (.obj) will be generated. The programming data file contains both your PSD design fit to silicon and your MCU/DSP *firmware*. See the

Merge MCU/DSP Firmware with PSD section for more details.

You are now ready to program a PSD using either JTAG or a conventional programmer. Both methods are covered in the next two sub-sections.

Programming the PSD using JTAG (Flash memory-based PSDs only)

Click on the “STMicroelectronics JTAG/ISP” box at the bottom of the design flow and you will be asked “How many devices are in the JTAG chain on your circuit board?” If you have only one PSD device and no other JTAG-compliant devices in your design, select “Only One”. If you have more than one PSD and/ or JTAG compliant device, click the see the *JTAG* section (Section 16). Assuming you had chosen “Only One” the “JTAG-ISP Operations—Single Device” dialog box will open:


JTAG-ISP Operations - Single Device

Step 1: Select Programming file and PSD
Select folder and programming file: C:\PSDexpress\my_project\new.obj
Select device: PSD835G2

Step 2: Specify JTAG-ISP operation and conditions
Select operation: Program
Select PSD region: All
Select # of JTAG pins to use on circuit board: 4 pins - tdi,tdo,tck,tms
Other conditions:
Click here to perform specified JTAG-ISP operation >>

Step 3: Save or retrieve JTAG-ISP setup
Specify folder and filename to save the setup of this JTAG-ISP session or retrieve a previous session.
Select folder and file:

Log Mode - Click box to record session infomation in the log file *.plg.

To program your PSD, take the following steps:

1. Ensure that the desired programming data file (.obj) appears in the box for “Step 1”. If you wish to use a different file, click the “Browse” button and locate the desired file. Then ensure that the selected device shown matches the device you wish to program.
2. Specify the JTAG operation, which PSD region it applies to, and the number of pins you will be using (4 or 6) to program the PSD. Click on the “Properties” button to set up how PSD pins should behave during a JTAG operation. Check the JTAG attributes for the PSD selected and to enter a user code to compare to one entered previously (in the *Additional Settings* screen). Click the **Execute** button to begin the JTAG operation selected.
3. Save your JTAG configuration for future use. If you have previously saved a file, you may open it as the first step and all your previous setting will be restored.

If you experience any problems during the JTAG operation, you may wish to see the *JTAG FAQ* section of the PSM website. You may also wish to download the JTAG application note: “JTAG-ISP Information for Flash PSDs. You can also save the log file by clicking on the “Log Mode” box and then send the file to *apps.psd@st.com* for analysis.

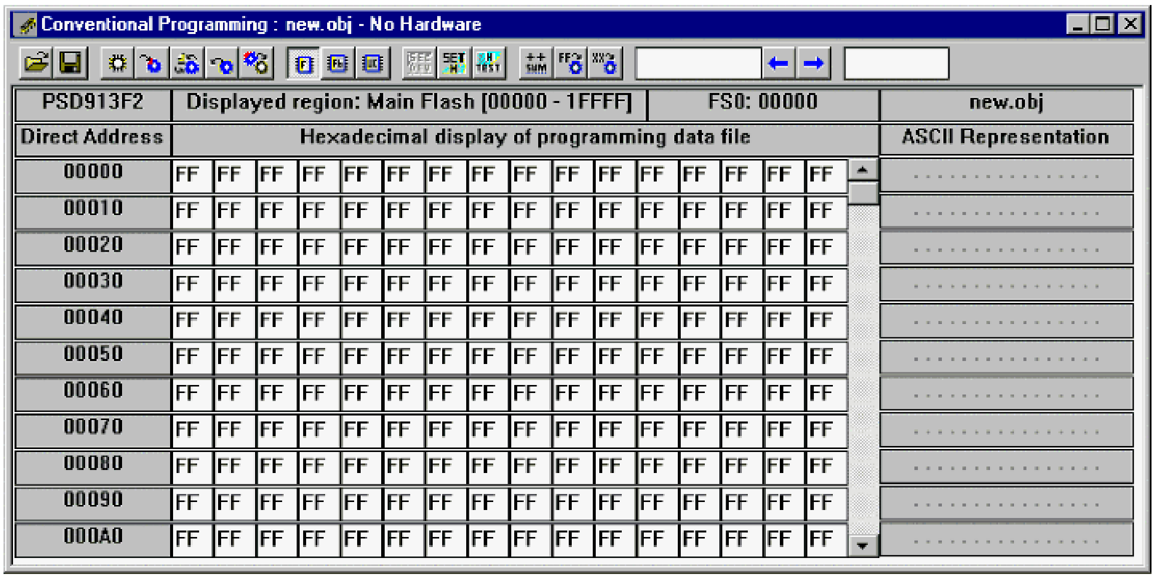
See the *JTAG Programming* section for more details.

Note: the .obj file can be viewed in the “Conventional Programming” window, regardless of whether or not you have a conventional programmer. See the *PSD Conventional Programmers* section (Section 17) for more details.

Note that a new button will become available below the Menu if you have completed the Merge MCU/DSP Firmware (with a valid Hex or S-Record file) and the JTAG-ISP step and saved your setup under Step 3 above to a valid .jcf file. This button will executes the Merge and JTAG operations in one step.

Programming the PSD using the PSD Conventional Programmer

To get to the conventional programming window, click the **STMicroelectronics Conventional Programmers** button.



Conventional Programming : new.obj - No Hardware

PSD913F2	Displayed region: Main Flash [00000 - 1FFFF]								FS0: 00000								new.obj
Direct Address	Hexadecimal display of programming data file																ASCII Representation
00000	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
00010	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
00020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
00030	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
00040	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
00050	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
00060	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
00070	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
00080	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
00090	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....
000A0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	....

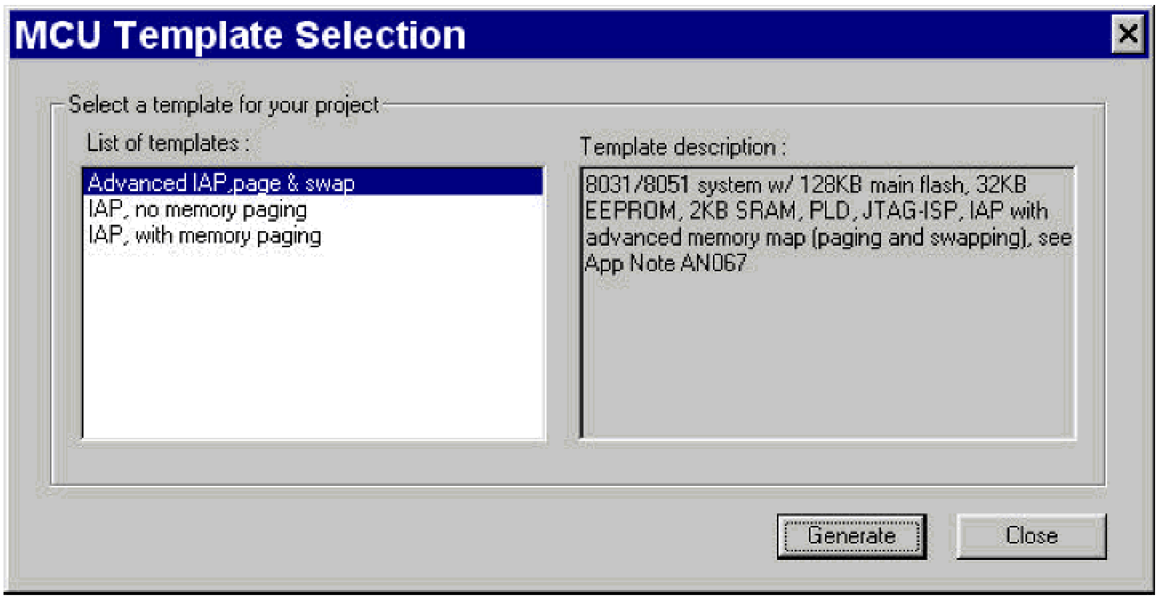
Use the various toolbar buttons to perform the desired functions. If you are unsure as to what a button does, allow the mouse pointer to hover over the button and a “tool tip” will appear.

Note: using the “Conventional Programming” window, you can view and edit your programming data file (.obj).

See the *PSD Conventional Programmers* section (Section 17) for more details.

Using Example Templates

If you had chosen to use an example template in the “Design Parameters” window (shown previously), you would be presented with a screen similar to the one shown below, depending on the MCU/DSP you had selected. Each of the available templates has a brief description associated with it. Select the template that most closely matches your end system.

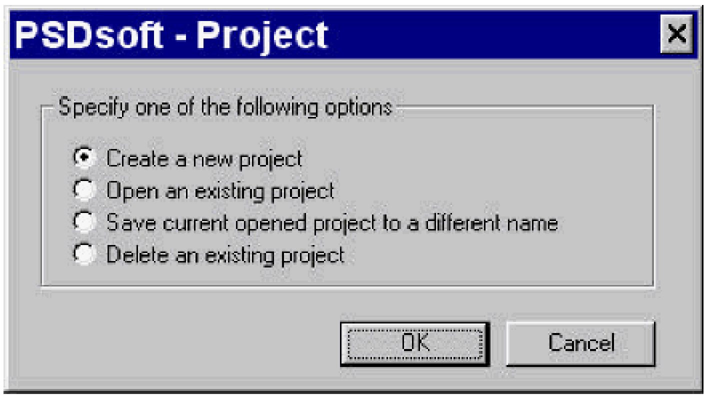


8031/8051 system w/ 128KB main flash, 32KB EEPROM, 2KB SRAM, PLD, JTAG-ISP, IAP with advanced memory map (paging and swapping), see App Note AN067

If you do not wish to use a template, click **Close** and you will be taken back to the “Design Parameters” screen. Otherwise, once you have made your selection, click **Generate** and you will be taken to the *Pin Definitions* screen. Note that some information will be filled in on the Pin Definitions, *Page Register Definition* (if you selected a template with paging), and *Chip Select Equations* screens. Take a look around each of the above-mentioned screens to see how the template generated information and make any desired modifications before clicking **Done**.

SPECIFY PROJECT

By clicking on “Specify Project” in the Design Flow, the following window appears:

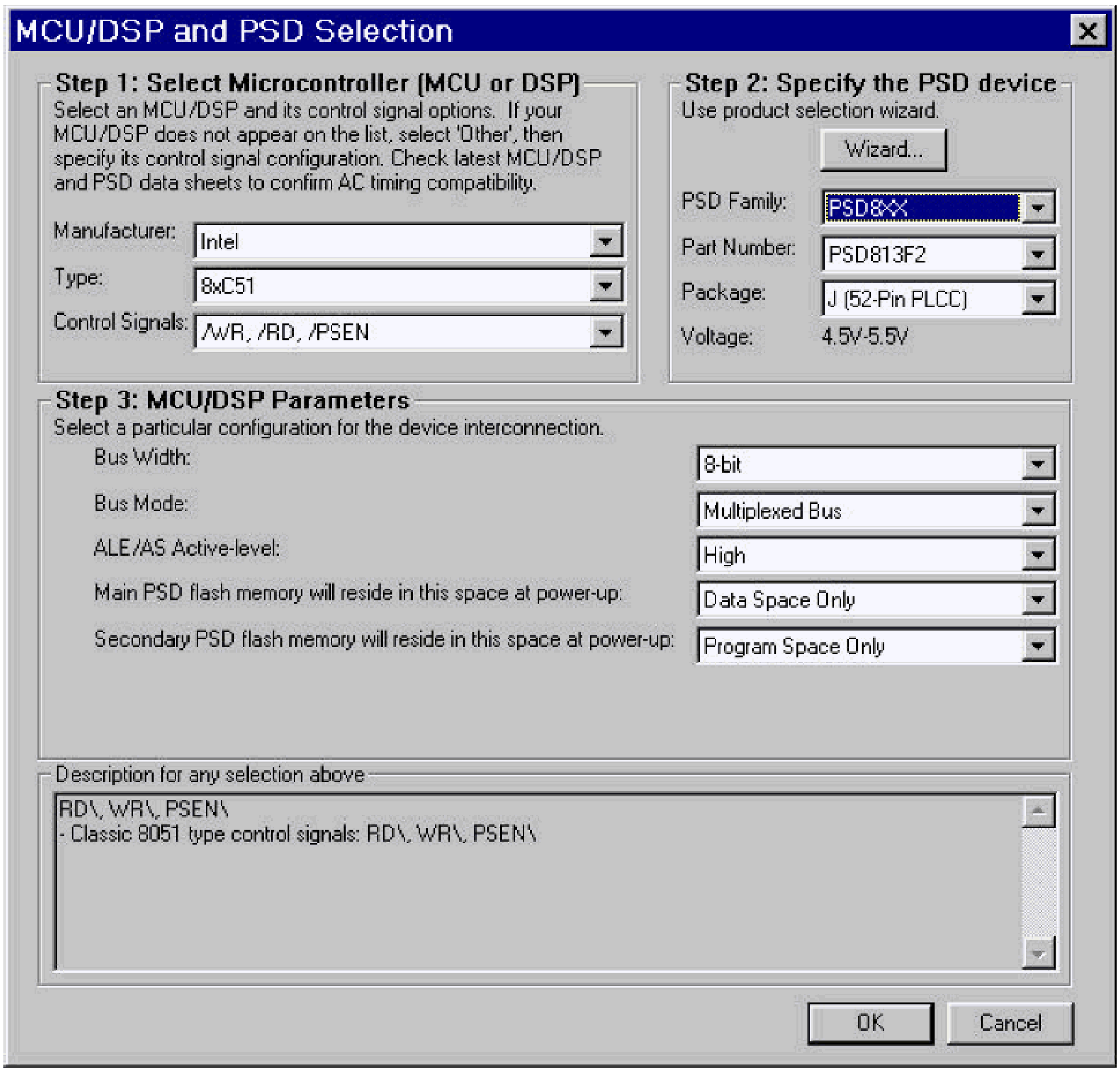

PSDsoft - Project
Specify one of the following options
Create a new project
Open an existing project
Save current opened project to a different name
Delete an existing project
OK
Cancel

In it, you are presented with four options:

- Create a new project—use this option to create different projects for each design that uses a PSD.
- Open an existing project
- Save current opened project to a different name—Useful when you wish to back-annotate projects. Then, if you need to check how something was done previously, you’ll have a reference.

- Delete an existing project—use this only if you wish to entirely purge a project and all of its associated files. Note: if you wish to erase only the files not necessary to re-create the project, you can select Project->Clean Up from the menu instead.

MCU/DSP AND PSD SELECTION


MCU/DSP and PSD Selection

Step 1: Select Microcontroller (MCU or DSP)
Select an MCU/DSP and its control signal options. If your MCU/DSP does not appear on the list, select 'Other', then specify its control signal configuration. Check latest MCU/DSP and PSD data sheets to confirm AC timing compatibility.

Manufacturer: Intel
Type: 8xC51
Control Signals: /WR, /RD, /PSEN

Step 2: Specify the PSD device
Use product selection wizard.

PSD Family: PSD8XX
Part Number: PSD813F2
Package: J (52-Pin PLCC)
Voltage: 4.5V-5.5V

Step 3: MCU/DSP Parameters
Select a particular configuration for the device interconnection.

Bus Width: 8-bit
Bus Mode: Multiplexed Bus
ALE/AS Active-level: High
Main PSD flash memory will reside in this space at power-up: Data Space Only
Secondary PSD flash memory will reside in this space at power-up: Program Space Only

Description for any selection above
RD\, WR\, PSEN\
- Classic 8051 type control signals: RD\, WR\, PSEN\

Step 1: Select Microcontroller (MCU/DSP)

In this step, you are selecting the microcontroller (MCU) or Digital Signal Processor (DSP) that you will be using with the PSD. First, select the manufacturer from the list. If the manufacturer you will use is not on the list, you can look for a similar MCU/DSP under a different manufacturer. If you cannot locate the desired MCU/DSP, set the manufacturer to “<Other>” and you will have to select the control signals yourself. Based on the MCU/DSP selected, the control signals will automatically be selected for you unless there is more than one possible combination. If there is more than one possibility, select the appropriate one from the drop-down list.

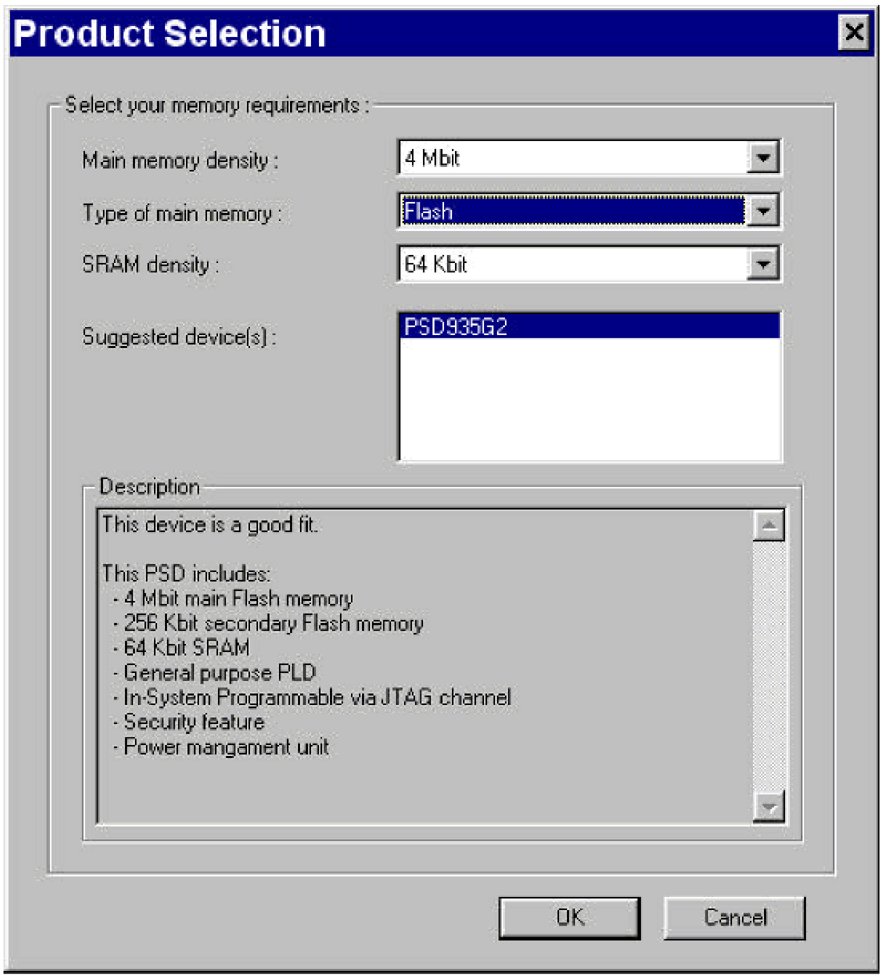
What is an MCU/DSP? If you are unsure what a microprocessor is or what it does, see Microprocessor Systems Notes: *http://www2.tcd.ie/Engineering/Courses/BAI/JS_Subjects/3D1/Notes/index.html*

If your MCU/DSP is not listed. If your Microcontroller (MCU) or Digital Signal Processor (DSP) is not in the list of available MCUs/DSPs and you would like to see it in a future revision of PSDsoft Express, email: *ask.psd@st.com*. Also use this email to provide feedback on additional features you would like to see in future PSDs.

Step 2: Specify the PSD device

If you are unsure what a PSD is, visit the PSM website at *www.st.com/psd*. If you know exactly which PSD you want to use, select the family, part number, and package. If you just need a little help in deciding which PSD to use, simply click on the **Wizard…** button in "Step 2".

You will be presented with the following screen:



Main memory density : 4 Mbit

Type of main memory : Flash

SRAM density : 64 Kbit

Suggested device(s) : PSD935G2

This device is a good fit.

This PSD includes:
- 4 Mbit main Flash memory
- 256 Kbit secondary Flash memory
- 64 Kbit SRAM
- General purpose PLD
- In-System Programmable via JTAG channel
- Security feature
- Power mangament unit

In this screen, you select the amount of "Main" memory that your system requires. The term "Main memory" is used because most PSDs have an optional secondary memory for *In-Application Programming (IAP)*. Make your selections for main memory and SRAM, and you will see a list of devices that match your selection.

Step 3: MCU/DSP Parameters

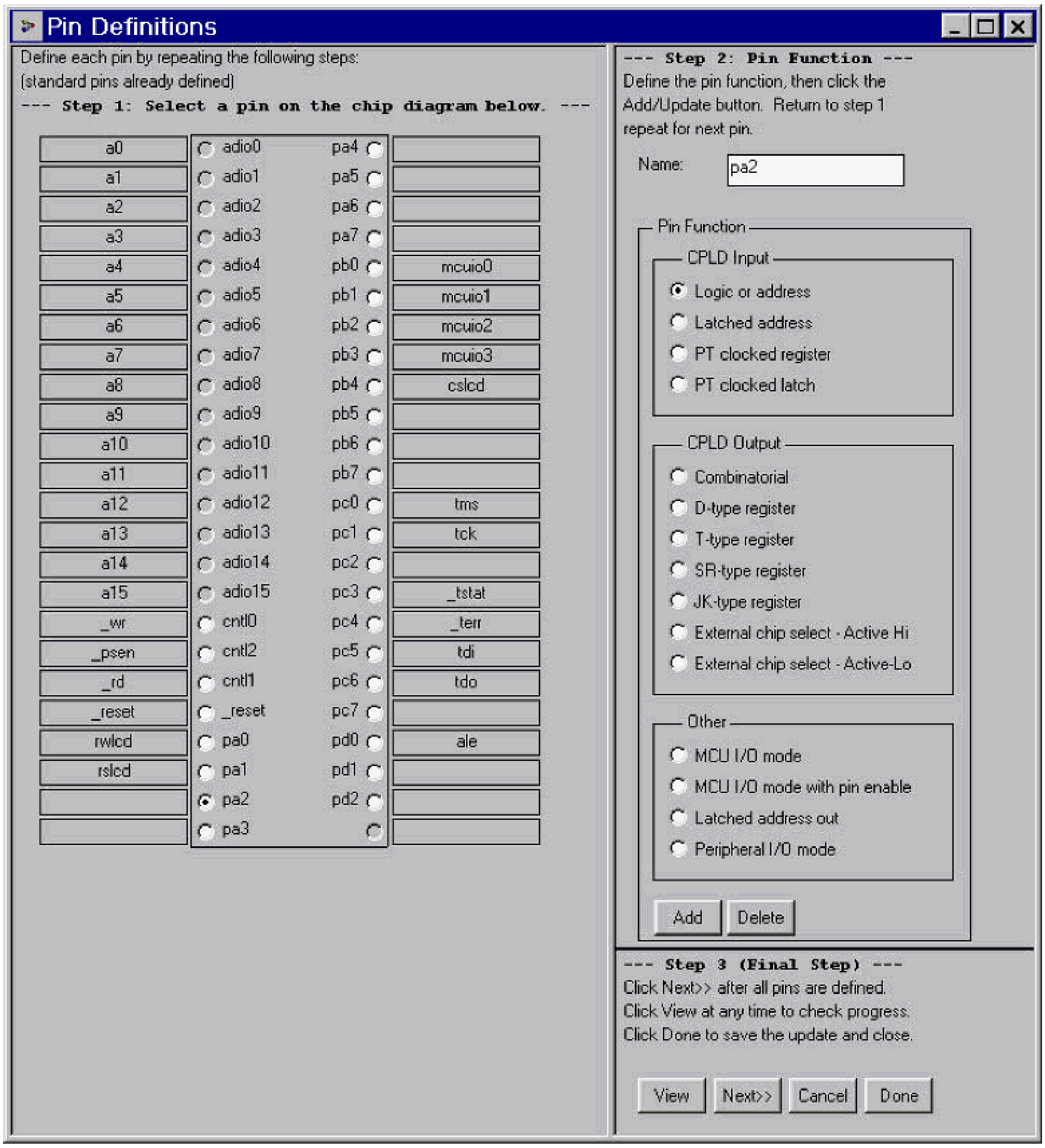
In the last step, you define the way the MCU/DSP and PSD interact. That is, 8-bit or 16-bit bus mode, *multiplexed* or *non-multiplexed* bus, and so on. Based on your MCU/DSP selection, these items may already be selected for you. Otherwise, make the appropriate selections as necessary.

Separate Program and Data Spaces

Some MCUs, such as the 80C51, P51XA, and 80C251, support separate program and data spaces by controlling a signal called Program Space Enable (PSEN) or a similar signal. This is perhaps the most confusing part of the “MCU/DSP and PSD selection” screen because you are asked where to place the main and (optional) secondary memories. By placing a memory in “Program Space Only”, it means that the MCU will only be able to read the memory with PSEN. That is, it is only considered code memory. The opposite is true for “Data Space Only”. In this case, the MCU will only be able to read the memory using the RD signal. However, if you select “Both Program and Data Space” (combined space), either the PSEN or RD signal can access the memory. In Combined space mode, you will have to explicitly add the control signals into your internal PSD *Chip-select Equations* for any overlapping memories. Keep in mind that this selection will specify how the PSD behaves at power-up and at reset. The MCU can override these selections at run-time by writing to the PSD’s VM Register (part of the PSD’s *CSIOP* register).

Commonly, the secondary memory is placed in “Program Space Only” and the main Flash memory is placed in “Data Space Only” at power-up and reset. This allows the MCU to boot from secondary memory and then program and/or validate the contents of main Flash memory. After this, the MCU can write to the VM register to place the main Flash memory into “Program Space Only”. See the *Application Notes* section of the PSM website. Also, the various instruction manuals for the development boards, such as the DK900, contain more examples.

PIN DEFINITIONS



Pin Definitions

Define each pin by repeating the following steps:
(standard pins already defined)

```
--- Step 1: Select a pin on the chip diagram below. ---
```

a0	adio0	pa4	
a1	adio1	pa5	
a2	adio2	pa6	
a3	adio3	pa7	
a4	adio4	pb0	mcuio0
a5	adio5	pb1	mcuio1
a6	adio6	pb2	mcuio2
a7	adio7	pb3	mcuio3
a8	adio8	pb4	cslcd
a9	adio9	pb5	
a10	adio10	pb6	
a11	adio11	pb7	
a12	adio12	pc0	tms
a13	adio13	pc1	tck
a14	adio14	pc2	
a15	adio15	pc3	_tstat
_wr	cntl0	pc4	_terr
_psen	cntl2	pc5	tdi
_rd	cntl1	pc6	tdo
_reset	_reset	pc7	
rwlcd	pa0	pd0	ale
rslcd	pa1	pd1	
	[✓ pa2]	pd2	
	pa3		

```
--- Step 2: Pin Function ---
```

Define the pin function, then click the Add/Update button. Return to step 1 repeat for next pin.

Name: pa2

Pin Function

CPLD Input
- [✓ Logic or address]
- Latched address
- PT clocked register
- PT clocked latch

CPLD Output
- Combinatorial
- D-type register
- T-type register
- SR-type register
- JK-type register
- External chip select - Active Hi
- External chip select - Active-Lo

Other
- MCU I/O mode
- MCU I/O mode with pin enable
- Latched address out
- Peripheral I/O mode

```
--- Step 3 (Final Step) ---
```

Click Next>> after all pins are defined.
Click View at any time to check progress.
Click Done to save the update and close.

Step 1

ADIO0 to ADIO15 are always reserved for address only (*non-multiplexed bus*) or address and data (*multiplexed bus*). No other type of I/O may be connected to these pins and the pin names cannot be changed, and thus the selections are grayed out. If you are using an MCU/DSP with a non-multiplexed bus, various ports will automatically be reserved for the data bus. Also, the CNTL0 and CNTL1 will be reserved for MCU/DSP control signals, such as /RD and /WR. Depending on your MCU/DSP selection CNTL2 may also be used for control signals, such as /PSEN. However, if your MCU/DSP only uses two control signals for memory accesses, CNTL2 can be used as an input to the PLD. _RESET is also a reserved pin and this pin must meet certain requirements outlined in the data sheets. The control and reset pins only have one possible function, but the pin names can be changed if desired. Other port pins can be reserved for special MCU/DSP control signals. These will generally be selected for you based on your MCU/DSP selection. For example, as you can see from the screen capture above, PD0 was reserved for the ALE signal.

For pins with multiple possible functions, if you want to see the potential functions of a pin, simply click on the individual pin in "Step 1" and the available functions will be listed according to the pin chosen. In general, the port pins do not have to be used as a group for particular functions. One exception to this rule is using the JTAG port for dedicated JTAG, but if you do set one of the JTAG signals to be dedicated, the other pin functions will be automatically set for you. For more information on the PSD JTAG port, see the "JTAG-ISP Information for Flash PSDs" application note. The other exception is Peripheral I/O mode in which the entire port will be required.

Brief Pin Function Descriptions

The following are brief descriptions of the pin functions. Note: functionality varies with PSD family and so some of these choices may not be available. See the data sheets and application notes for more details.

(C)PLD Input

Logic or Address is the most general type of input, which can be used within the PLD for decoding and other logic functions. Additional address inputs from the MCU/DSP can be used here.

Latched Address—the signal input on this pin will be latched with the ALE or AS signal. This is useful for additional address inputs from the MCU/DSP that are multiplexed.

PT Clocked Register—the pin will be attached to an IMC D flip-flop that can be clocked using another signal within the PLD.

PT Clocked Latch—the pin will be attached to a IMC transparent latch that can be clocked using another signal within the PLD.

(C)PLD Output

Combinatorial—output based on a combinatorial logic equation.

D-type register—output from an OMC D-type register.

T-type register—output from an OMC T-type register.

SR-type register—output from an OMC SR-type register.

JK-type register—output from an OMC JK-type register.

External chip select-Active Hi—active-high output based on decoded address and control signals.

External chip select-Active Lo—active-low output based on decoded address and control signals.

Other

MCU I/O mode—allows direct access to port registers within the PSD using the MCU/DSP data bus at runtime.

MCU I/O mode with pin enable—same as above except an output enable signal derived from the CPLD can be used to control the output.

Latched address out—multiplexed address bits that were latched using ALE or AS can be output on some port pins. Check the data sheet to see which address bits will be output on which pins.

Peripheral I/O mode—an entire port is used in this mode as a tri-stateable, bi-directional data buffer for your MCU/DSP.

Dedicated JTAG—setting this option will set this pin and other dedicated JTAG pins as well. These pins become full-time JTAG pins and can't be multiplexed with general I/O functions.

SRAM standby voltage input—selecting this option requires a battery or alternate power source be connected to this pin to protect the SRAM contents when the system power is off.

Rdy/Bsy output—indicates the write/erase status of the non-volatile memories within the PSD.

Standby on indicator—indicates when the SRAM is drawing power from the power source connected to the SRAM standby voltage input pin described above.

Common clock input—clock input to the macrocells, automatic power-down unit counter, and PLD AND array.

PSD chip-select input—a signal can be used to place the PSD memories in standby (very low power) mode. This is an active-low signal.

Note: The PSD211R, (Z)PSD3XX , (Z)PSD4XX, and (Z)PSD5XX pins may have some functions not listed here. In this case, please check the data sheet for details.

Special Note for PSD Parts with JTAG ISP Capability

Most JTAG users will use the port with JTAG capability as a dedicated JTAG port. That is, they will not multiplex the JTAG-capable pins with other I/O. If this applies to you, you can skip this paragraph. If, however, you do intend to multiplex the JTAG-capable pins with other I/O, you should take the following actions:

1. Since the default for the JTAG-capable pins is JTAG, you will need to first change the functions on those pins to the desired I/O. For example, if you wish to use a pin for both the TMS function during JTAG operations and as a chip-select called “CS0” during non-JTAG (normal) operation, you will want to select the pin called “tms” (pin pc0 in the Pin Definitions screen) and re-name it “CS0”, select “External Chip Select” as the function and click the **Update** button.
2. Continue defining other I/O for the remaining JTAG-capable pins in a similar manner.
3. Define a pin that will enable the JTAG function for the JTAG-capable pins during JTAG operations (_jen for example, as shown in the *JTAG Select Equation (Parts with Multiplexed JTAG Ports Only)* section (Section 5.7). Any PLD input pin will suffice. To be consistent with our documentation, the suggested name is “_jen” for “JTAG Enable”. The pin function should be set to “Logic or Address” within the CPLD inputs. This pin will be used later to access a special register within the PSD for turning on and off the JTAG port. See the *JTAG Select Equation* section of the PSDsoft Express User Manual.
4. Remember to read the data sheet and the “JTAG-ISP Information for Flash PSDs” application note for more details on JTAG:

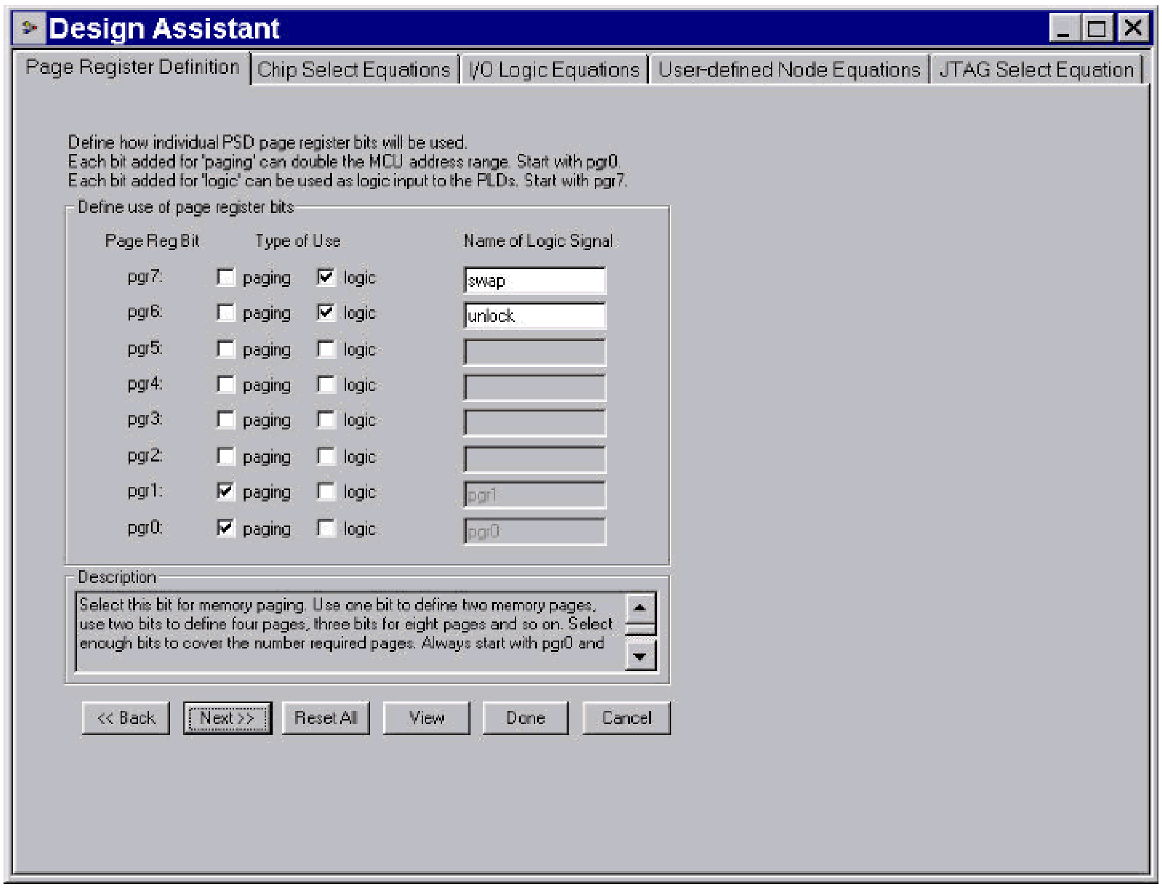
Step 2

When you are defining a Port pin for the first time, type in the name that you want to be assigned to that pin in the “Name:” box in “Step 2” (up to 31 characters) and then assign the pin function from the available list and click **Add**. If you wish to change the function or name of a pin, enter the new name or function and click the **Update** button (which was the **Add** button). If you wish to delete the function of a pin and not assign a new function, simply click the **Delete** button. Note that if you delete a pin function or do not define it in the first place, the default value is “PLD input”. Thus any unused pins should be pulled to Vcc (100K pull-up resistor should suffice) or tied to ground to avoid excess current draw by the CMOS PSD.

Step 3

If at any time, you wish to check the progress of your design, you can click on the **View** button. Clicking **Next>>** takes you to “*Page Register Definitions*” which is the next step in the Design Assistant (except for PSDs which do not have a page register). If you do not wish to save any changes you may have made and you do not want to continue on using the Design Assistant, click **Cancel**. If you only needed to make changes to the pin definitions, click **Done**.

PAGE REGISTER SETUP



Design Assistant

Page Register Definition | Chip Select Equations | I/O Logic Equations | User-defined Node Equations | JTAG Select Equation

Define how individual PSD page register bits will be used.
Each bit added for 'paging' can double the MCU address range. Start with pgr0.
Each bit added for 'logic' can be used as logic input to the PLDs. Start with pgr7.

Define use of page register bits

Page Reg Bit	Type of Use		Name of Logic Signal
pgr7:	paging	[✓ logic]	swap
pgr6:	paging	[✓ logic]	unlock
pgr5:	paging	logic	
pgr4:	paging	logic	
pgr3:	paging	logic	
pgr2:	paging	logic	
pgr1:	[✓ paging]	logic	pgr1
pgr0:	[✓ paging]	logic	pgr0

Description

Select this bit for memory paging. Use one bit to define two memory pages, use two bits to define four pages, three bits for eight pages and so on. Select enough bits to cover the number required pages. Always start with pgr0 and

If you are unsure what paging is or if your design requires paging, see "*What is Paging*".

The Page Register is one of many registers that are accessed through *CSIOP* space. The Page Register can be read or written by the MCU/DSP directly at the offset indicated in the datasheet. The outputs of the Page Register are connected to the PLDs within the PSD.

Page Register Uses

There are three uses for the Page Register:

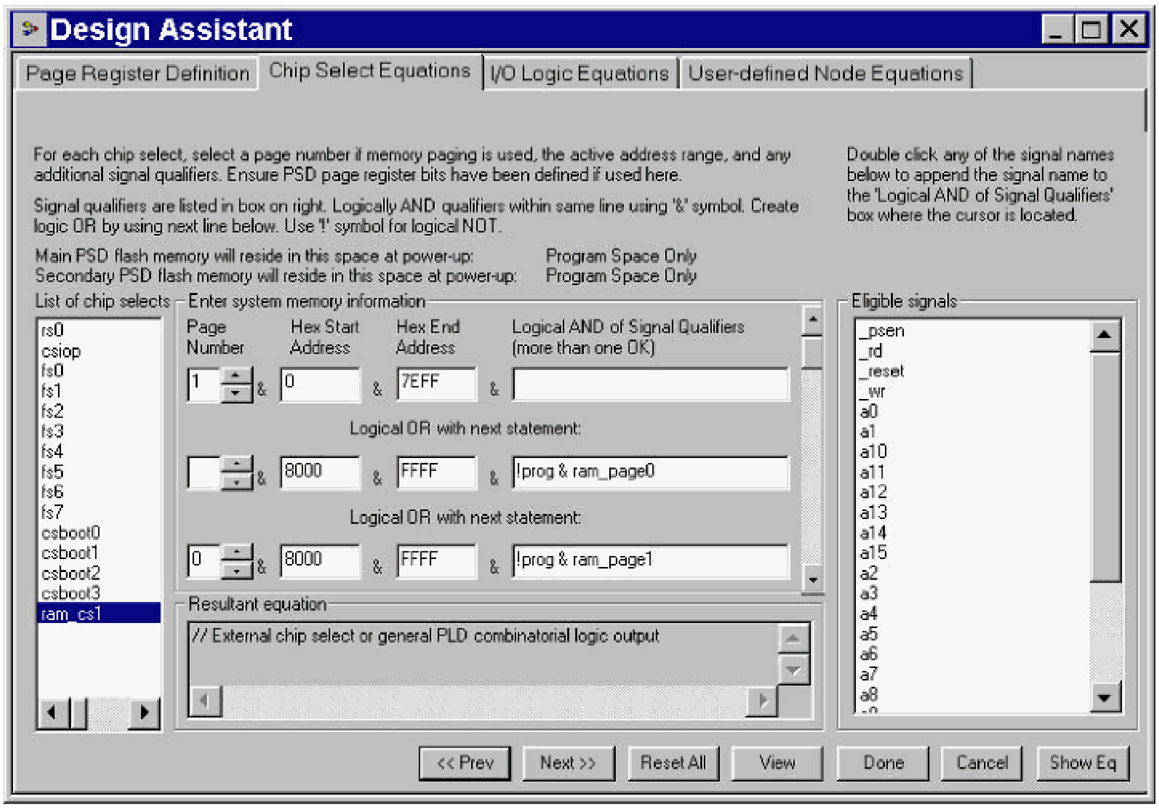
- For paging in your address decode logic.
- For general logic inputs to the PLDs. Remember, the page register is written using the MCU/DSP.
- As a general-purpose storage register accessible by the MCU/DSP. Only a good idea if you are not using it for the previous two purposes.

If you wish to use the Page Register for both paging and logic, the paging bits should be used starting with pgr0 and going up sequentially from there and the logic bits should start at pgr7 and go down from there. The above screen capture shows that four of the eight Page Register bits will be used in the design: pgr0 and pgr1 are used for paging (thus 2^2 = 4 pages will be available). Pgr7 is used for a logic input called "swap" and pgr6 is a logic input called "unlock".

Moving On

When you are satisfied with your Page Register setup, you can click **Next>>** or the "*Chip Select Equations*" tab and you will be taken to the window where you can define your internal and external chip-select equations. Clicking **Reset All** clears all the entries in the screen (after a confirmation). Click **View** to check your design's progress so far. Click **<< Back** to return to the “Pin Definitions” screen. If you will not be making any changes to the chip-select equations, but want to save your progress so far and return to the “Design Flow” window, click **Done**. If you do not want to save any changes you have made in this session, click **Cancel**.

CHIP SELECT EQUATIONS/MEMORY MAP DEFINITION



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

Main PSD flash memory will reside in this space at power-up: Program Space Only
Secondary PSD flash memory will reside in this space at power-up: Program Space Only

List of chip selects: rs0, csiop, fs0, fs1, fs2, fs3, fs4, fs5, fs6, fs7, csboot0, csboot1, csboot2, csboot3, ram_cs1

Enter system memory information

Page Number		Hex Start Address		Hex End Address		Logical AND of Signal Qualifiers (more than one OK)
1	&	0	&	7EFF	&	

Logical OR with next statement:

Page Number		Hex Start Address		Hex End Address		Logical AND of Signal Qualifiers (more than one OK)
	&	8000	&	FFFF	&	!prog & ram_page0

Logical OR with next statement:

Page Number		Hex Start Address		Hex End Address		Logical AND of Signal Qualifiers (more than one OK)
0	&	8000	&	FFFF	&	!prog & ram_page1

Resultant equation

// External chip select or general PLD combinatorial logic output

Eligible signals: _psen, _rd, _reset, _wr, a0, a1, a10, a11, a12, a13, a14, a15, a2, a3, a4, a5, a6, a7, a8

This screen is basically used to define your system memory map with respect to the PSD and any external chip-selects. If you used an example template, some of these chip-selects will already be filled in, in which case you simply need to make the necessary adjustments to suit your needs.

This is a recursive process with the following steps:

1. Highlight a chip-select equation from the column on the left that you wish to define.
2. If you had defined some pages in the *Page Register Definition* screen and you wish to use paging for the chip-select highlighted, type in a page number in its column or select it using the arrows. If no paging is being used for a particular select signal, no action is needed.
3. Enter the start and stop addresses in hexadecimal format in the next two columns for which the chip-select highlighted should be valid.
4. In the “logical AND of Signal Qualifiers” column, you can enter any signal from the list of eligible signals in the far right column that you wish to be included in the end equation.
5. If you have additional signals to logically AND, enter “&” followed by the signal name. Use the ! symbol to negate the signal.

6. If the chip-select signal also requires more product terms (logical ORs), repeat steps 2 through 5 for the next row. Click the **Show Eq** button to view the results in the “Resultant equation” box. Repeat as necessary.
7. Repeat steps 1 through 6 until all the chip-selects in the left-hand column have been defined.

Hint: sometimes it can be very beneficial to draw out a graphical system-level memory map before you start this process. Also, see the appropriate data sheet to find out how big each of the PSDs internal segments should be and the rules (priority scheme) for overlapping memory segments.

Important Note: internal chip-select equations normally do not need to be ANDed with the MCU/DSP control signals as this is automatically taken care of at the silicon level, and sometimes doing so may cause timing problems. The only exception to this is when the internal memories have been placed in Combined space mode and overlap. (See *Select PSD and MCU/DSP*.) **However**, external chip-selects that require qualification of the MCU/DSP control signals must be added by the user.

Looking at the screen shot above as an example, we can see the following:

– The signal being defined is “ram_cs1”

– The signal is valid on Page 1 from 0h to 7EFFh OR

– The signal is valid from 8000h to FFFFh AND !prog AND ram_page0 OR

– The signal is valid on Page 0 form 8000h to FFFFh AND !prog AND ram_page1.

This is a bit confusing since there are two forms of paging being used: the PSD’s internal paging scheme and logic inputs called “ram_page0” and “ram_page1”. For more information on paging, see *Appendix B.4*.

The final address decode would read:

```
ram_cs1 = ((address >= 0h) AND (address <= 7EFFh) AND (Page == 0)) OR (((address >= 8000h)
AND (address <= FFFFh)) AND ((!prog AND ram_page0) OR (!prog AND ram_page1)))
```

When you have finished entering your chip-select equations, click **View** to check the results. If you wish to go back to the *Page Register Definition* screen, click the **<< Prev** button. If you would like to re-enter the definition for a signal, highlight the desired signal and click **Reset All**. If you are satisfied with the results, click **Done**, which performs a preliminary resource check and syntax check to ensure that you have not entered any invalid equations.

EDIT/ADD ABEL EQUATIONS

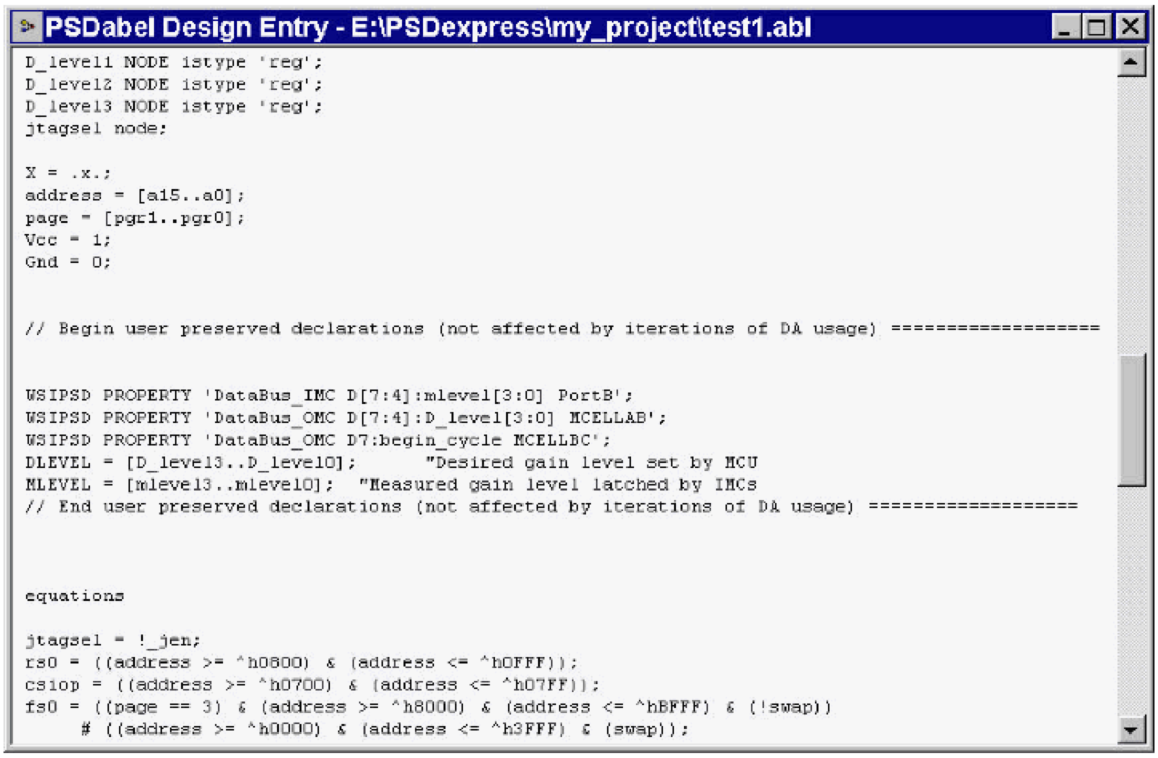
This file contains the following information:

– Menu Items available with the ABEL file open

– PSDabel-HDL Keywords and Syntax Specific to the PSD

– Test Vector Syntax

– When to use the Design Assistant and when to use ABEL



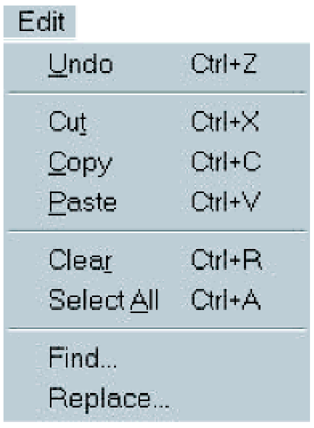
PSDabel Design Entry - E:\PSDexpress\my_project\test1.abl

```
D_level1 NODE istype 'reg';
D_level2 NODE istype 'reg';
D_level3 NODE istype 'reg';
jtagsel node;

X = .x.;
address = [a15..a0];
page = [pgr1..pgr0];
Vcc = 1;
Gnd = 0;


// Begin user preserved declarations (not affected by iterations of DA usage) ===================


WSIPSD PROPERTY 'DataBus_IMC D[7:4]:mlevel[3:0] PortB';
WSIPSD PROPERTY 'DataBus_OMC D[7:4]:D_level[3:0] MCELLAB';
WSIPSD PROPERTY 'DataBus_OMC D7:begin_cycle MCELLBC';
DLEVEL = [D_level3..D_level0];      "Desired gain level set by MCU
MLEVEL = [mlevel3..mlevel0];  "Measured gain level latched by IMCs
// End user preserved declarations (not affected by iterations of DA usage) ===================



equations

jtagsel = !_jen;
rs0 = ((address >= ^h0800) & (address <= ^h0FFF));
csiop = ((address >= ^h0700) & (address <= ^h07FF));
fs0 = ((page == 3) & (address >= ^h8000) & (address <= ^hBFFF) & (!swap))
     # ((address >= ^h0000) & (address <= ^h3FFF) & (swap));
```

When editing the ABEL file manually, all declarations and equations (including test vectors) **must** be placed in the designated sections to be preserved.

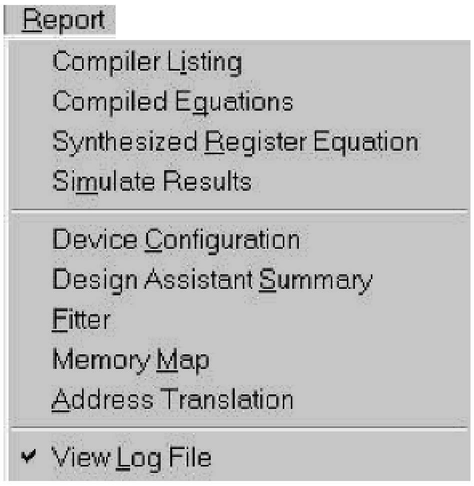
Menu Items available with the ABEL file open

The Edit Menu



The Edit menu only appears when the ABEL file is open, and the functions do the same thing they would in a typical text editor.

The Report Menu



There are additional menu items available in the Report menu when the ABEL file is open:

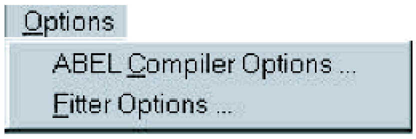
■ Compiler Listing compiles the latest version of your design and displays the compilation list file with line numbers.

■ Compiled Equations compiles and optimizes the latest version of your design file and then displays the output of the optimized compiled equations.

■ Synthesized Register Equation compiles, optimizes, and synthesizes SR, JK, T, and D registers defined within the ABEL design file to a mixture of D- and T-type flip-flops where the smallest possible implementation will be obtained. You should see the results upon completion.

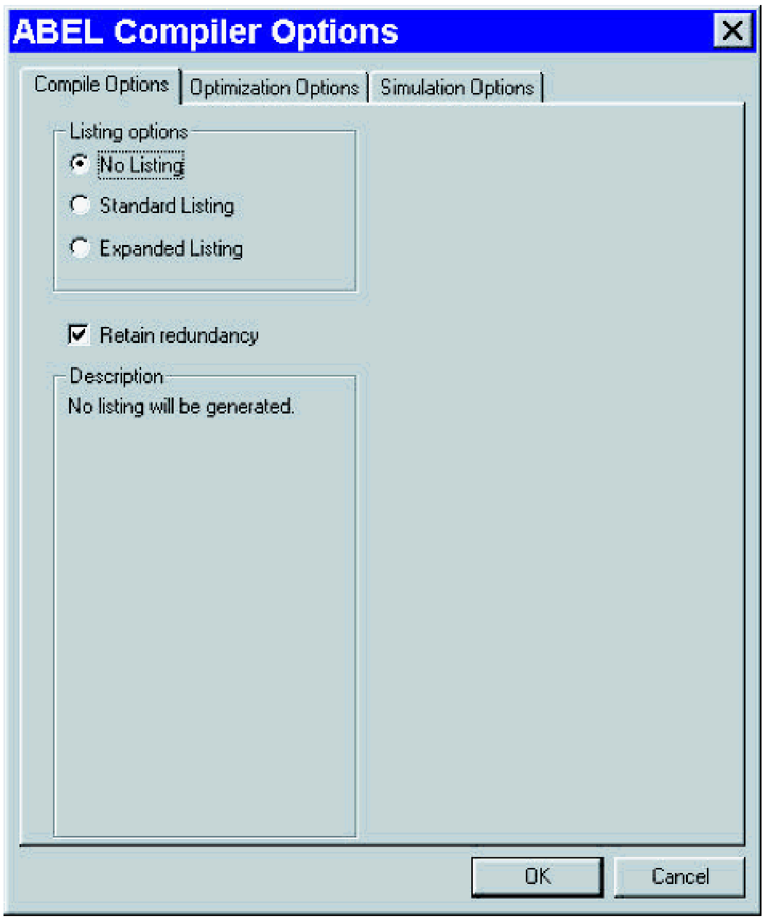
■ Simulate Results performs a simulation based on the *Test Vectors* written in your ABEL file and displays the results based on your Simulation Options.

The Options Menu



The Options menu only appears when the ABEL file is open. The fitter options are covered in the *Fitter* section.

When the **ABEL Compiler Options** is selected, the “ABEL Compiler Options” screen appears:



Listing options
(● No Listing) ○ Standard Listing ○ Expanded Listing

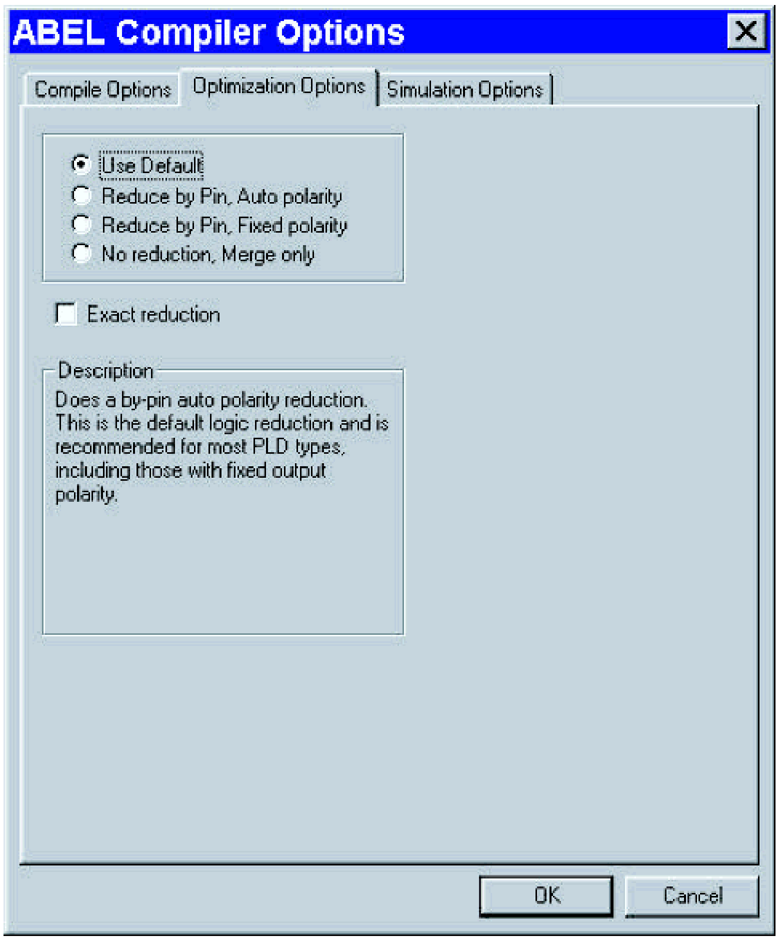
☑ Retain redundancy

Description
No listing will be generated.

Compile Options

- No Listing means that no permanent file will be created from the compile process. This is the default setting and most users will not need to change this.
- Standard Listing generates a listing containing numbered source file lines and error messages (if any).
- Expanded Listing generates a listing containing numbered source file lines, expanded macros, directives, and error messages (if any).
- Retain redundancy: checking this box disables compile-time trivial reductions such as A # !A = 1. Check this box if you wish to preserve redundant product terms (for hazard protection). Note that these reductions are also automatically done in Reduce unless the “No reduction, Merge only” option is chosen for optimization on the next screen.

Click on the “Optimization Options” tab at the top of the dialog box and you will see the following:



(●) Use Default
( ) Reduce by Pin, Auto polarity
( ) Reduce by Pin, Fixed polarity
( ) No reduction, Merge only

☐ Exact reduction

Description

Does a by-pin auto polarity reduction. This is the default logic reduction and is recommended for most PLD types, including those with fixed output polarity.

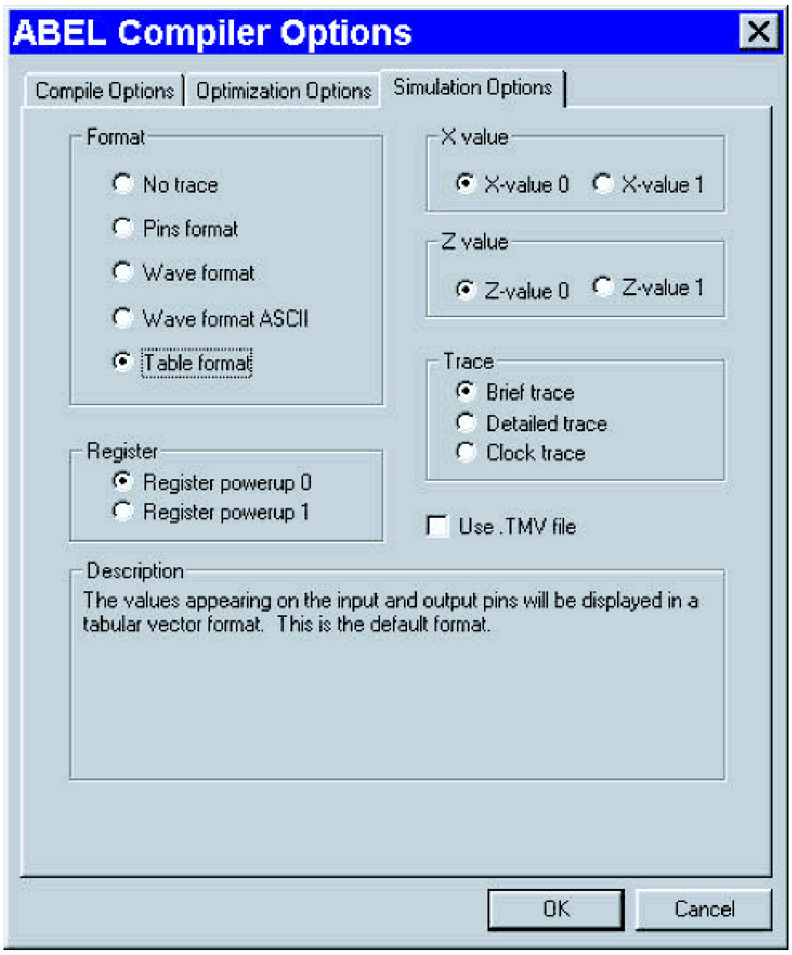
Optimization Options

- Use Default does a by-pin auto polarity reduction. This is the default logic reduction and is recommended for most users.
- Reduce by Pin, Auto polarity is the same as Use Default.
- Reduce by Pin, Fixed polarity reduces the logic such that each signal will have the minimum number of product terms possible. Can be useful in parts with programmable polarity outputs.

No reduction, Merge only reduces the logic so that each signal will have the minimum number of product terms, maintaining the polarity of the signals. This should be used when you want to force output signals to a specified polarity.

Exact reduction invokes Espresso’s exact minimization option. This option will sometimes produce more complete logic minimization than by standard Espresso.

Click on the “Simulation Options” tab to see the following screen:



Format
- No trace
- Pins format
- Wave format
- Wave format ASCII
- [✓ Table format]

X value: [✓ X-value 0] X-value 1

Z value: [✓ Z-value 0] Z-value 1

Trace
- [✓ Brief trace]
- Detailed trace
- Clock trace

Register
- [✓ Register powerup 0]
- Register powerup 1

☐ Use .TMV file

Description

The values appearing on the input and output pins will be displayed in a tabular vector format. This is the default format.

Simulation Options

The format options affect how your simulation file will be displayed:

- No Trace shows only errors (if any). No simulation output will be generated.
- Pins Format shows test vectors for each pin for the device. The values appearing on the input and output pins will be displayed for each test vector.
- Wave format shows the output level that appears on each specified device pin during the simulation process. The output pin voltages are shown as a waveform in the output file that contains a trace for each pin. Each trace represents the logic HI and LO output levels for each test vector. This wave format can generally be viewed but not printed.
- Wave format ASCII is the same as above, except that it can be printed.
- Table format is similar to the Wave format except that the waveform is replaced by H, L, and Z for logic HI, LO, and high-impedance, respectively. This is the default format.

The register options are:

- Register powerup 0 sets all registers to zero (LO) before simulation begins.
- Register powerup 1 sets all registers to one (HI) before simulation begins.

The X and Z value Options are used where “don’t care” and high impedance values are encountered in test vectors and must be given some value during simulation. Thus, setting “X-value 0” for the X value will substitute a LO any time an X is encountered.

The trace options are as follows:

- Brief trace generates a report of the simulation results for each clock cycle for registered designs, or for the stabilized output values for combinational designs.
- Detailed Trace generates a report of the simulation results for each level in the sum-of-products logic circuit being simulated. This option is useful for debugging complex logic circuits.
- Clock trace generates a simulation report that shows register values when the clock is LO, HI, and LO again for each vector. This option is useful with the macrocell trace for debugging asynchronous circuits.

Use .TMV file checkbox forces the simulator to use the vectors in a .tmv file.

PSDabel-HDL Keywords and Syntax Specific to the PSD

This section expands on keywords and syntax specific to the PSDs. See the “Flash PSD CPLD Primer” application note for more details.

The keyword that allows access to PSD-specific functional blocks is the `WSIPSD PROPERTY` statement. This is used in conjunction to the `Databus_IMC` and `Databus_OMC` keywords to access the PSD’s Input Macrocells (IMCs) and Output Macrocells (OMCs).

Syntax—IMCs

```
WSIPSD PROPERTY 'DataBus_IMC D[s1:s2]: signal[x1:x2] portname'
```
or
```
WSIPSD PROPERTY 'DataBus_IMC D s1 ,D s2 ,D s3 ...: x1,x2,x3... portname'
```

Syntax—OMCs

```
WSIPSD PROPERTY 'DataBus_OMC D[s1: s2]: signal[x1:x2] portname'
```
or
```
WSIPSD PROPERTY 'DataBus_OMC D s1 ,D s2 ,D s3 ...: x1,x2,x3... portname'
```

Where

- signal[x1:x2] is a user-defined signal name or signal with an inclusive subscript range of x1:x2. x1 and x2 are integer values. The signal defined in PSDabel would be signalx1 and so forth.
- D [s1:s2] is an inclusive data bus subscript range. s1 and s2 are integer values representing data bus bits. The data bus name **must** be “D”.
- s1:s2 for 8-bit microcontrollers, the range is less than or equal to 8. For 16-bit microcontrollers, the range is less than or equal to 16.
- Valid portnames for DataBus_IMC are PortA/B/C/D
- Valid portnames for DataBus_OMC are PortA/B/C/D or PortAB/BC/C, depending on the architecture. See the datasheets for which declaration should be used.

Examples

```
WSIPSD PROPERTY 'DataBus_IMC D[7:4]:mybus[3:0] PortC'
WSIPSD PROPERTY 'DataBus_OMC D[7:4]:mybus[3:0] PortC'
```

Purpose

The DataBus_IMC/DataBus_OMC property tells the fitter to assign the user-defined signals to the corresponding port/micro-cell, when specified, and associate to the specified data bus bits. It is a one-to-one mapping to the data bus bit as specified on the DataBus_IMC/DataBus_OMC property. When the port name is not specified, the fitter will assign user-defined signals to the resources available on the device where the specified data bus bits can be accessed. This is useful to force a certain data bit order when creating registered logic for counters and shift registers, for example.

Errors to watch out for

- When the specified data bus bit is not accessible (8-Bit or 16-Bit mode), the following error message is generated:
- “In mode, D[m1:m2] bit(s) is/are not accessible”.
- When the specified port name conflicts with an existing assignment, the following error message is generated:
- “Conflicting port name assignment in DataBus_IMC/DataBus_OMC property”.
- When the total number of data bus bits does not equate to the total number of signal bits, the following error message is generated:
- “Inconsistency in number of bits in DataBus_IMC/ DataBus_OMC property”.

More Examples

Example 1: assign signals to IMCs on Port B

```
WSIPSD PROPERTY 'DataBus_IMC D[7:0]:key[7:0] PortB'
```

Notes:

- With no prior pin or node definitions, key7 to key0 are assigned to Port B as pins and thus pin assignments are redundant and could cause errors.
- These signals can be assigned to nodes only and the pins could be used for other purposes.
- By default, IMC is configured as a transparent input.

Example 2: assign signals to OMCs

```
WSIPSD PROPERTY 'DataBus_OMC D[7:4]:sreg[3:0] MCELLAB';
```

- No signal name definition for “sreg” is required in the PSDabel file prior to the PROPERTY statement.
- sreg3 to sreg0 are reserved to MCELLAB as nodes.

If you want OMCs to be assigned to specific pins, use this syntax:

```
sreg3..sreg0 pin; or
sreg3..sreg0 pin 24, 23, 22, 21;
```

Note: if you do assign OMCs to specific pins, ensure that the pins match the port that the OMCs belong to.

To see how IMCs and OMCs can be used for various functions, see The “Flash PSD CPLD Primer” application note on the web.

Test Vector Syntax

Test vectors are a means to test the PLD code. The following rules apply to Test Vector syntax:

- The signals and groups to be tested must appear in the *Declarations* section of the ABEL file
- The start of the Test Vectors is indicated by the keyword `Test_Vectors`.
- The entire list of input and output signals used in the vectors should be enclosed in parenthesis and the input group should be separated from the output group by the “->” symbols. See the example below.
- The inputs to be used should be placed in a comma separated list enclosed by brackets.
- Each of the test vectors should be a comma separated list, enclosed in brackets, where the input and output groups are separated by the -> symbols and the vector should end with a semi-colon.
- There is no limit to the number of test vectors or number of test vector sets.
- The test vectors should be placed within the *Designated Preserved* sections.
- Inputs on the left include logic-0s and logic-1s, “don’t cares”, high-impedance, and clock values. Outputs are the expected behavior and can be logic-0s, logic-1s, or “don’t cares”. If you choose don’t cares, the value will be filled shown in the output file when the simulation is run.
- Inputs and outputs can be entered as binary, octal, hexadecimal, and decimal. The example below shows binary and hexadecimal inputs and binary outputs.

Example:

```
// Begin user preserved equations (not affected by iterations of DA usage)
Test_Vectors
// Test the state machine, trim, and boost signals
([clkin, reset, begin_cycle, MLEVEL, DLEVEL] ->
 [Start_Conv, Intrn, STATE1, STATE0, Trim, Boost])
 [ X, 0, X, ^h3, ^h4 ] -> [ X, X, X, X, 0, 1 ];"system in reset
 [ 0, 1, X, ^h4, ^h4 ] -> [ 0, 1, 0, 0, 0, 0 ];"system out of reset
 [ C, 1, 1, ^h5, ^h4 ] -> [ 1, 1, 0, 1, 1, 0 ];
 [ C, 1, 1, ^h5, ^h4 ] -> [ 0, 1, 1, 0, 1, 0 ];
 [ C, 1, 1, ^h5, ^h4 ] -> [ 0, 0, 1, 1, 1, 0 ];
 [ C, 1, 1, ^h4, ^h4 ] -> [ 0, 1, 0, 0, 0, 0 ];
 [ C, 1, 0, ^h4, ^h4 ] -> [ 0, 1, 0, 0, 0, 0 ];
// End user preserved equations (not affected by iterations of DA usage)
```

Note that the X and C that appear above in the test vectors would have to be defined in the *Declarations* section of the ABEL file as follows:

```
   C = .C.;
   X = .X.;
   Z = .Z.;
```

The simulator is invoked by going to the **Report** menu and choosing **Simulate Results**. The simulation results and the output format will be based on the *Simulation Option* settings, discussed earlier.

When to use the Design Assistant and when to use ABEL

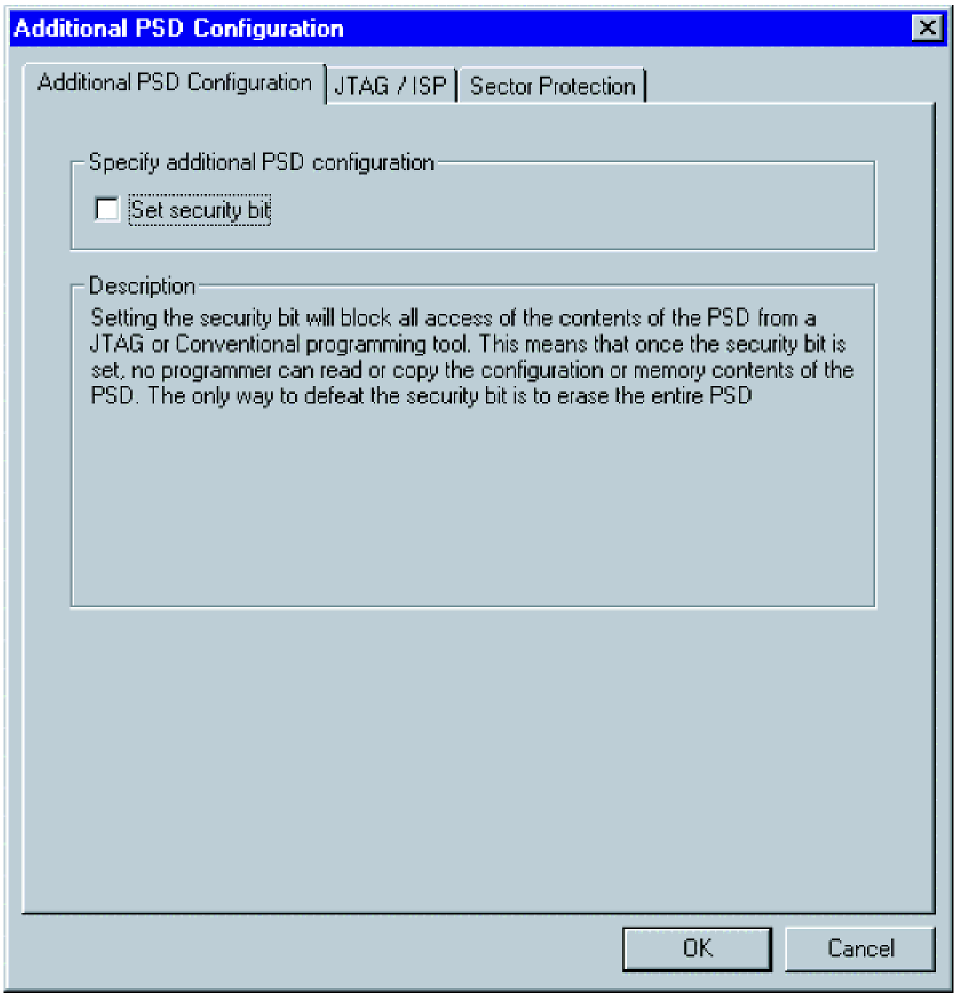
Hopefully, the Design Assistant will be useful enough that you will not have to use ABEL at all. However, there are certain times where using ABEL will be unavoidable, including the following:

- Defining groups of signals. For example: `Group = [signalA, signalB, SignalC];` or `Group = [Signal3..Signal0];`
- Using groups in a high-level language context. For example, if you had previously defined A and B to be two 4-bit signal input groups and C to be a 4-bit output group, then you must use ABEL to write a statement like: `C = A + B;`
- Aligning Input or Output macrocells to the MCU/DSP databus using the `WSIPSD PROPERTY` statement as discussed previously.
- Any complex logic requiring feedback terms from other logic elements, such as state machines, shift registers, counters, and so on are best handled using the ABEL language, although they can be done using the Design Assistant if you are willing to write out the equations.

ADDITIONAL PSD SETTINGS

By clicking on the tabs at the top, you can switch between the windows shown below.

Additional PSD Configuration



Specify additional PSD configuration

☐ Set security bit

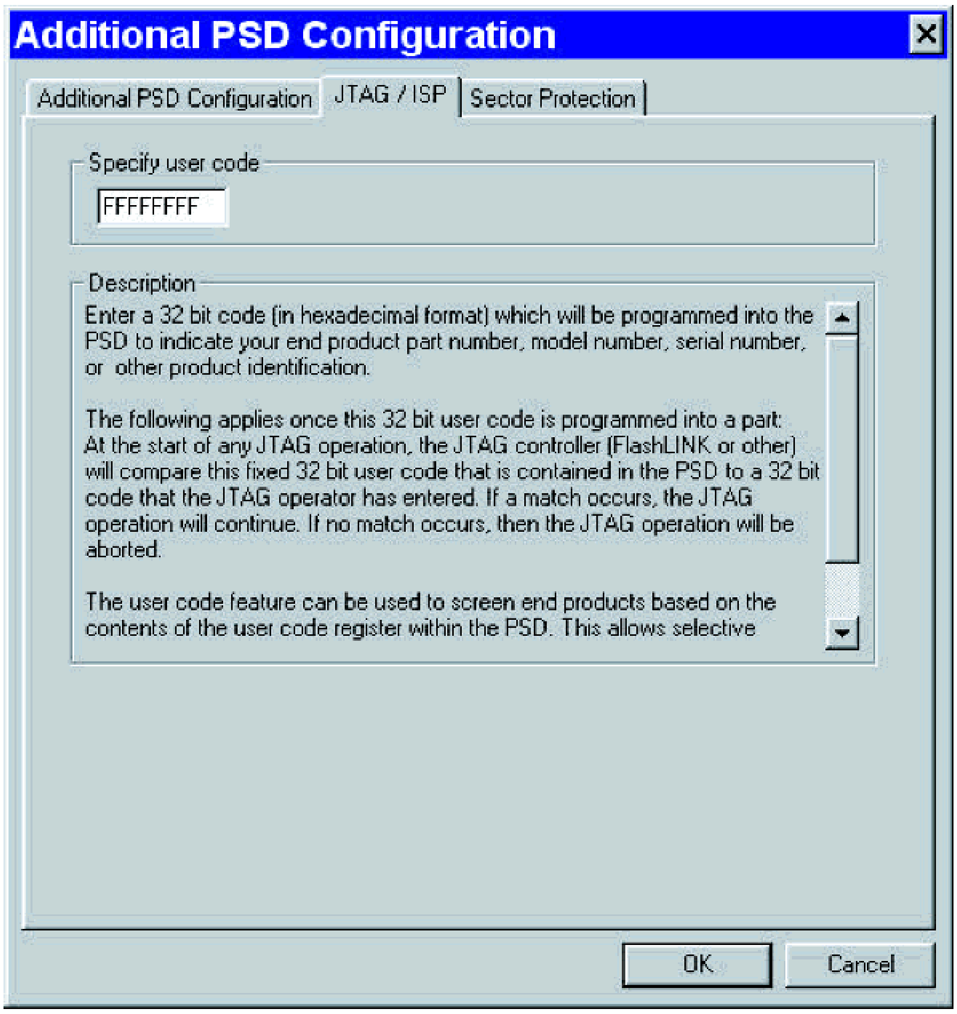
Description

Setting the security bit will block all access of the contents of the PSD from a JTAG or Conventional programming tool. This means that once the security bit is set, no programmer can read or copy the configuration or memory contents of the PSD. The only way to defeat the security bit is to erase the entire PSD

Setting the Security Bit. With this tab selected, you can opt to set the security bit of the PSD. Basically, by setting the security bit, access is blocked to the contents of the PSD from any type of programming device. That is, any attempt at uploading the contents (memory or configuration) will result in garbage with the security bit set. Only a full-chip erase function will erase the security bit. Note that any checksums computed in the *Conventional Programming* window will not be valid.

Enable EPROM/SRAM Low Power Mode (PSD3XX/211R family only). Checking this box enables the EPROM and optional SRAM to enter a low power mode when not being accessed.

JTAG/ISP (Flash-Based PSDs Only)



Specify user code

FFFFFFFF

Description

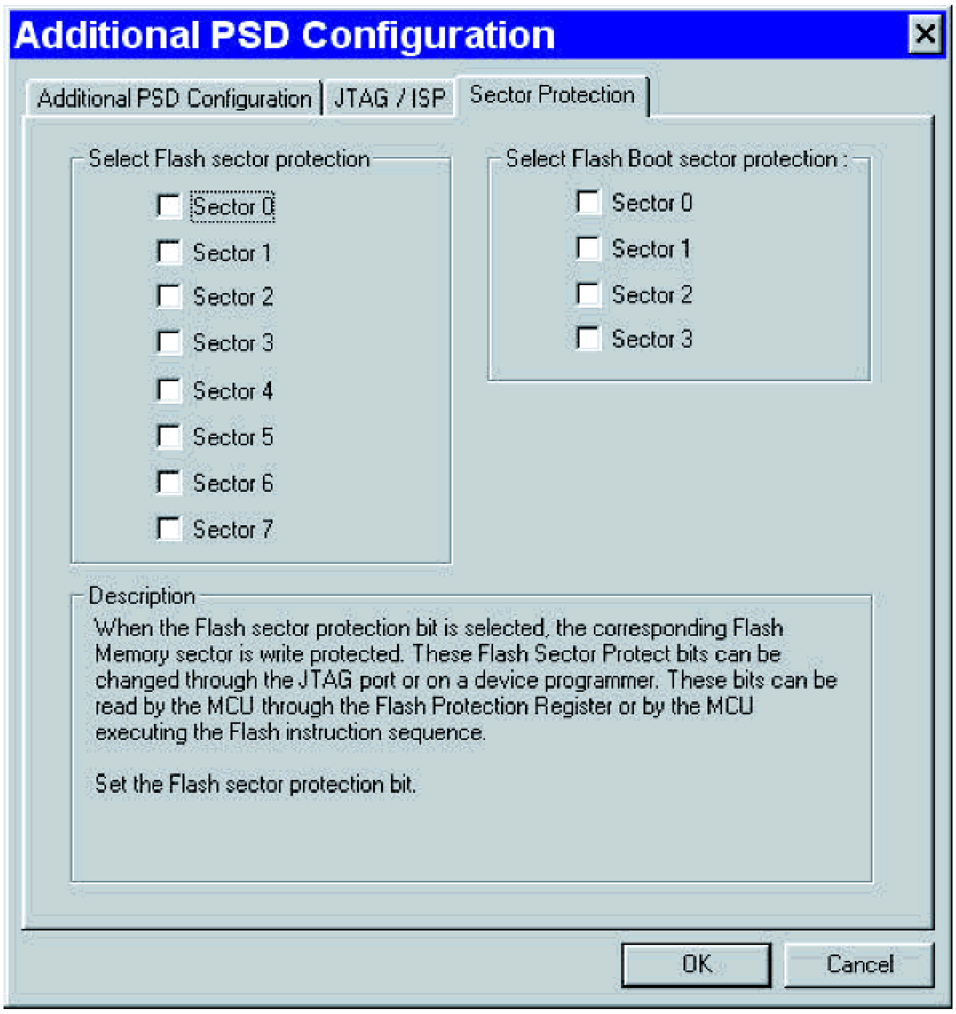
Enter a 32 bit code (in hexadecimal format) which will be programmed into the PSD to indicate your end product part number, model number, serial number, or other product identification.

The following applies once this 32 bit user code is programmed into a part:
At the start of any JTAG operation, the JTAG controller (FlashLINK or other) will compare this fixed 32 bit user code that is contained in the PSD to a 32 bit code that the JTAG operator has entered. If a match occurs, the JTAG operation will continue. If no match occurs, then the JTAG operation will be aborted.

The user code feature can be used to screen end products based on the contents of the user code register within the PSD. This allows selective

Follow the instructions in the *Description* section on how to use this window. This window is used in conjunction with the “JTAG ISP Operations” window.

Sector Protection (Flash-Based PSDs Only)



When the Flash sector protection bit is selected, the corresponding Flash Memory sector is write protected. These Flash Sector Protect bits can be changed through the JTAG port or on a device programmer. These bits can be read by the MCU through the Flash Protection Register or by the MCU executing the Flash instruction sequence.

Set the Flash sector protection bit.

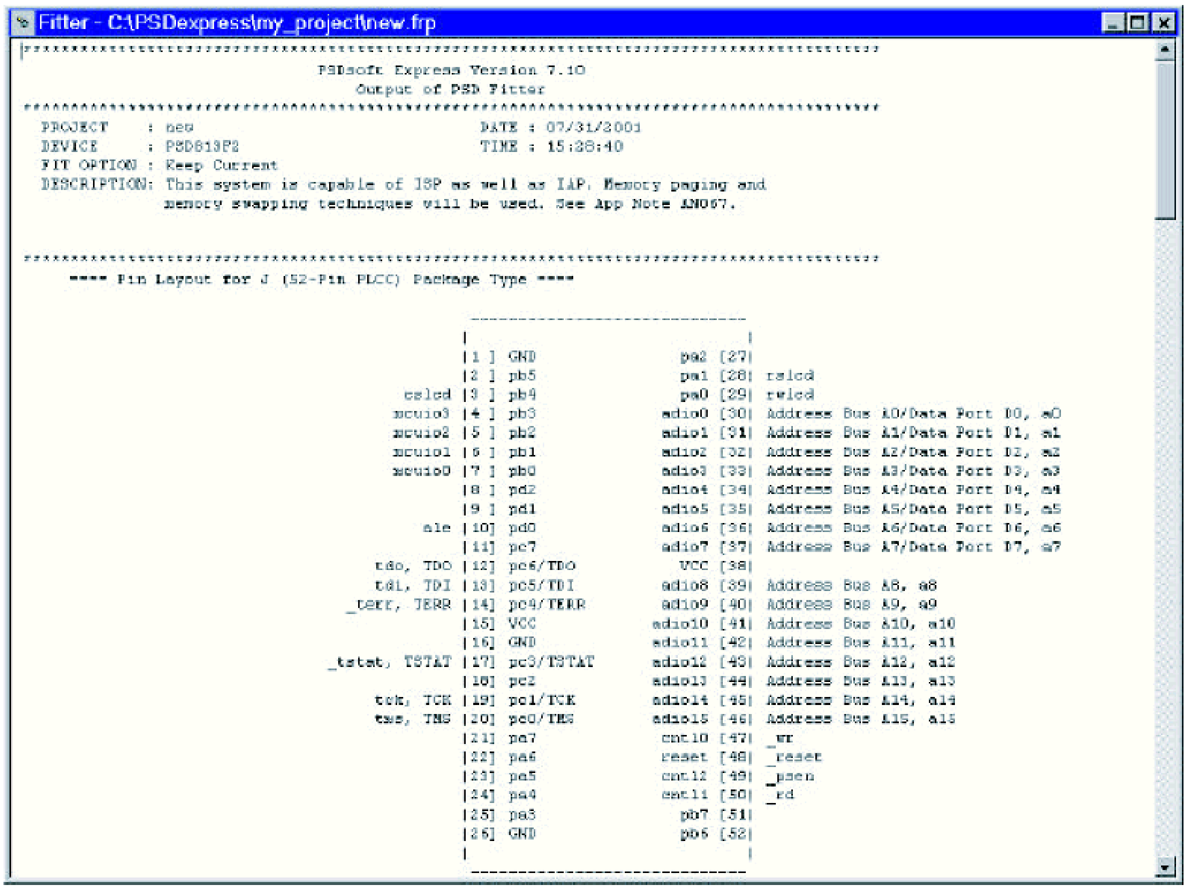
Use this window to write-protect the desired sectors of the non-volatile memories within the PSD. This is one way to protect from accidental data loss statically. To enable write-protection dynamically will require use of the *Page Register*. There are application notes examples on the website.

FITTING THE DESIGN TO SILICON

This section covers the fitter in more detail, including:

- The Fitter Report
- Changing the Fitter Options
- What to do when the Fitter keeps Running
- What to do if you get an error after running the Fitter

The Fitter Report



```
|*************************************************************************************
                                PSDsoft Express Version 7.10
                                   Output of PSD Fitter
*************************************************************************************
  PROJECT     : new                                    DATE : 07/31/2001
  DEVICE      : PSD813F2                               TIME : 15:38:40
  FIT OPTION  : Keep Current
  DESCRIPTION: This system is capable of ISP as well as IAP. Memory paging and
               memory swapping techniques will be used. See App Note AN067.


*************************************************************************************
    **** Pin Layout for J (52-Pin PLCC) Package Type ****

                              -----------------------------
                             |                             |
                             |1 ]  GND              pa2  [27|
                             |2 ]  pb5              pa1  [28|  rslcd
                      cslcd  |3 ]  pb4              pa0  [29|  rwlcd
                    mcuio3   |4 ]  pb3            adio0  [30|  Address Bus A0/Data Port D0, a0
                    mcuio2   |5 ]  pb2            adio1  [31|  Address Bus A1/Data Port D1, a1
                    mcuio1   |6 ]  pb1            adio2  [32|  Address Bus A2/Data Port D2, a2
                    mcuio0   |7 ]  pb0            adio3  [33|  Address Bus A3/Data Port D3, a3
                             |8 ]  pd2            adio4  [34|  Address Bus A4/Data Port D4, a4
                             |9 ]  pd1            adio5  [35|  Address Bus A5/Data Port D5, a5
                       ale   |10]  pd0            adio6  [36|  Address Bus A6/Data Port D6, a6
                             |11]  pc7            adio7  [37|  Address Bus A7/Data Port D7, a7
                  tdo, TDO   |12]  pc6/TDO          VCC  [38|
                  tdi, TDI   |13]  pc5/TDI        adio8  [39|  Address Bus A8, a8
               _terr, TERR   |14]  pc4/TERR       adio9  [40|  Address Bus A9, a9
                             |15]  VCC           adio10  [41|  Address Bus A10, a10
                             |16]  GND           adio11  [42|  Address Bus A11, a11
             _tstat, TSTAT   |17]  pc3/TSTAT     adio12  [43|  Address Bus A12, a12
                             |18]  pc2           adio13  [44|  Address Bus A13, a13
                  tck, TCK   |19]  pc1/TCK       adio14  [45|  Address Bus A14, a14
                  tms, TMS   |20]  pc0/TMS       adio15  [46|  Address Bus A15, a15
                             |21]  pa7            cntl0  [47|  _wr
                             |22]  pa6            reset  [48|  _reset
                             |23]  pa5            cntl2  [49|  _psen
                             |24]  pa4            cntl1  [50|  _rd
                             |25]  pa3              pb7  [51|
                             |26]  GND              pb6  [52|
                             |                             |
                              -----------------------------
```

The fitter report file (.frp) contains an abundance of useful information, including:

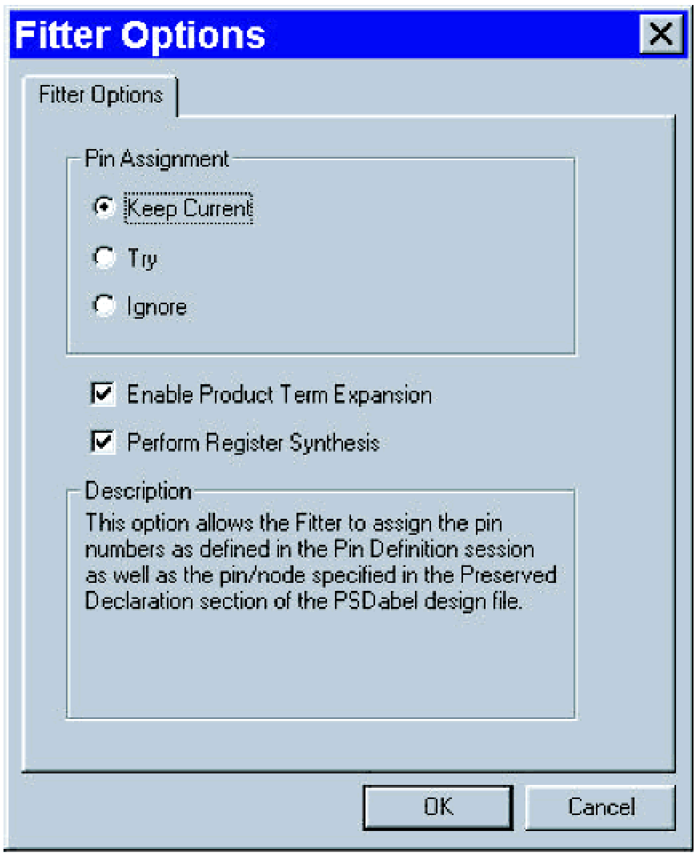
- The signals defined in the “*Pin Definitions*” screen and their associated pin numbers
- Information on how the PSD will be configured
- The address and data bus assignments
- Input and Output Macrocell assignments (where applicable)
- Product term, I/O, and node usage by port
- All the reduced logic equations in their final form

This information can be useful in determining the resources used. Used resources can, in turn, be used to determine potential power usage and yet-available resources.

If you clicked **Fit Design to Silicon**, the fitter report will open automatically upon a successful fit. If this report did not pop up or is not available from the menu, you may have gotten an error during the fit process. Check your log file (.plg) to see if there were any errors.

Changing the Fitter Options

The only way to change the fitter options is to have the ABEL file open, in which case you should see the **Options** menu. If you are unsure how to open the ABEL file, see *Edit/Add Logic Statements*. To change the Fitter options, go to **Options->Fitter Options** and the following window should appear:



Pin Assignment

- (● Keep Current) ○ Try ○ Ignore

[✓] Enable Product Term Expansion

[✓] Perform Register Synthesis

Description

This option allows the Fitter to assign the pin numbers as defined in the Pin Definition session as well as the pin/node specified in the Preserved Declaration section of the PSDabel design file.

The Pin Assignment options have the following meaning:

- Keep Current: allows the Fitter to assign the pins as they are defined in the Pin Definitions screen as well as the pins/nodes declarations specified in the *Preserved Declaration* section of the PSDabel design file.
- Try: allows the Fitter to attempt to preserve the pins as they are defined in the Pin Definitions screen as well as the pins/nodes declarations specified in the *Preserved Declaration* section of the PSDabel design file.
- Ignore: allows the Fitter to ignore all pin assignments as they are defined in the Pin Definitions screen as well as the pins/nodes declarations specified in the *Preserved Declaration* section of the PSDabel design file.

If you are experiencing problems fitting due to an apparent lack of PSD resources, it is best to run the Fitter with the “Ignore” option set so that if there is a possible fit, it will be found. A user can then go back and re-assign pins based on the Fitter report and then use the “Keep Current” option.

Checking the “Enable Product Term Expansion” box enables the Fitter to perform product term expansion on logic that requires more resources than the native resources allow. Most users will want to leave this box checked.

Checking the “Perform Register Synthesis” box allows the Fitter to synthesize SR, JK, T, and D registers defined within the ABEL design file to a mixture of D- and T-type flip-flops where the smallest possible implementation will be obtained.

What to do when the Fitter keeps Running

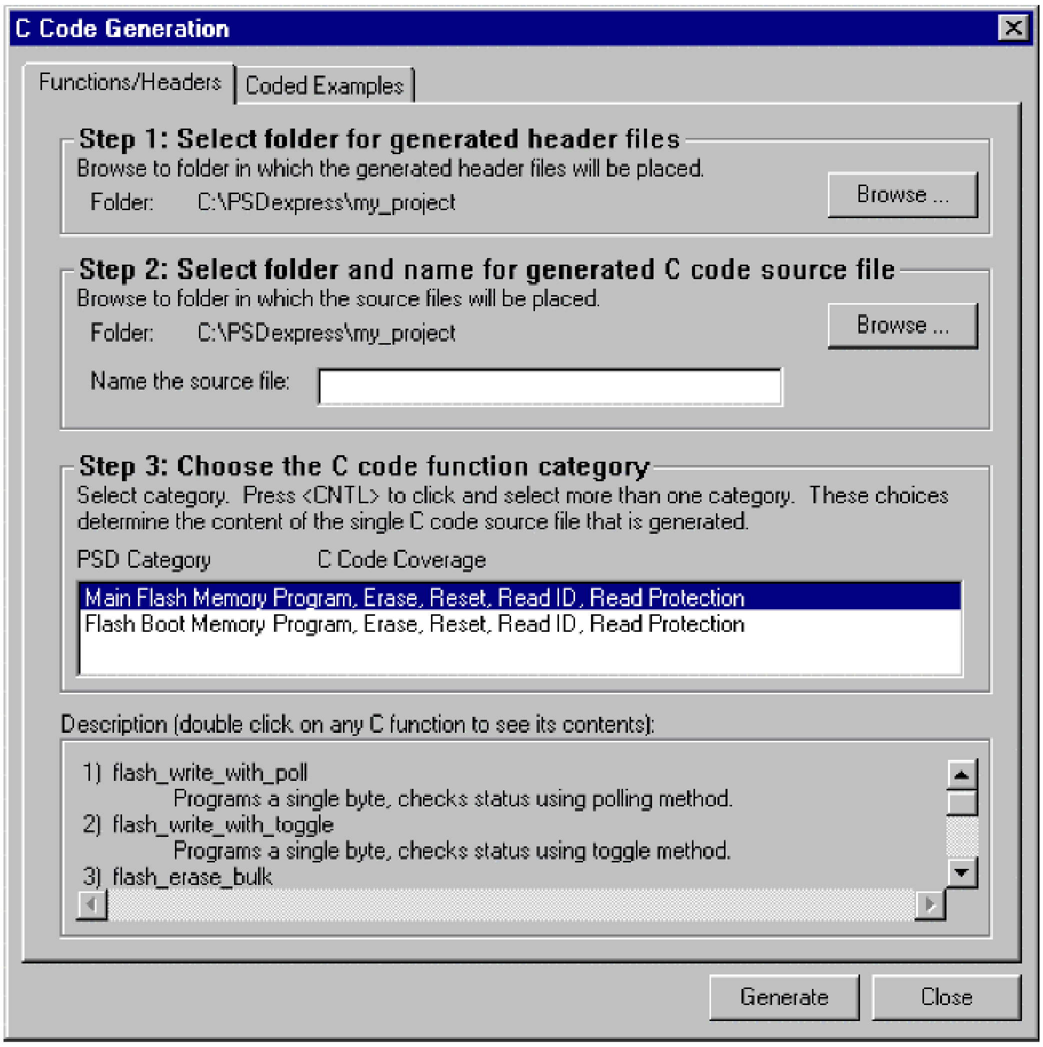
If the fitter keeps running for more than a few minutes, your project will not likely fit to silicon with your current design. Sometimes, it is a simple matter to fix this by changing how the Fitter goes about fitting your design. The best thing to try first is to go to the **Options->Fitter Options** and select “Ignore” in the “Pin Assignments” box. See the above sub-section on how to do this. Also, it is recommended to use the Design Assistant as much as possible and stay away from the ABEL file where possible.

What to do if you get an error after running the Fitter

If you stick to using the Design Assistant and don’t enter any equations manually in the ABEL file, your chances of getting an error during Fitting are less. However, if you need to add statements in the ABEL file because there is no other way and you do receive an error, usually the source of the error will be apparent based on the message you receive. If you should receive an error message that is not obvious or you can’t seem to fix your problem, contact *apps.psd@st.com*. In most cases, it will be helpful to zip up your entire project directory and attach it to the email.

GENERATE C CODE (FLASH/EEPROM PSDS ONLY)

Functions/Headers


C Code Generation

Functions/Headers | Coded Examples

Step 1: Select folder for generated header files
Browse to folder in which the generated header files will be placed.
Folder: C:\PSDexpress\my_project

Step 2: Select folder and name for generated C code source file
Browse to folder in which the source files will be placed.
Folder: C:\PSDexpress\my_project
Name the source file:

Step 3: Choose the C code function category
Select category. Press <CNTL> to click and select more than one category. These choices determine the content of the single C code source file that is generated.

PSD Category	C Code Coverage
Main Flash Memory	Program, Erase, Reset, Read ID, Read Protection
Flash Boot Memory	Program, Erase, Reset, Read ID, Read Protection

Description (double click on any C function to see its contents):

```
1) flash_write_with_poll
        Programs a single byte, checks status using polling method.
2) flash_write_with_toggle
        Programs a single byte, checks status using toggle method.
3) flash_erase_bulk
```

This screen is used to generate functions and headers in the ANSI-C language. These functions can be used for basic actions within the PSD (such as writing or erasing) and are provided to help you integrate the code required to interact with the PSD into your overall system design.

Step 1: Select Folder for Generated Header Files. The header files generated by PSDsoft Express are referenced by the C files. Since you will be integrating and referencing these files in your own C files, you will probably want to store all the files in one place. Select the folder where the files should be stored by clicking the **Browse…** button, then navigate to the desired folder and click **Open**.

Step 2: Select Folder and Name for Generated C Code Source File. First, assign a name to the C source file that will be generated and click the **Browse…** button, then navigate to the desired folder and click **Open**.

Step 3: Choose the C code function category. Select the category of C functions to be generated. If you wish to select more than one category, hold the CTRL key down while making your selections. If you would like to get an idea of the code that will be generated, double-click on the name of the function within the “Description” box.

When you are finished with the steps, click the **Generate** button. You are given a message in the Log Window about the successful generation of the source files and a list of the files generated. Three files will be written to your specified folder(s). For example, if you were using a PSD4135 device, the following files would be generated:

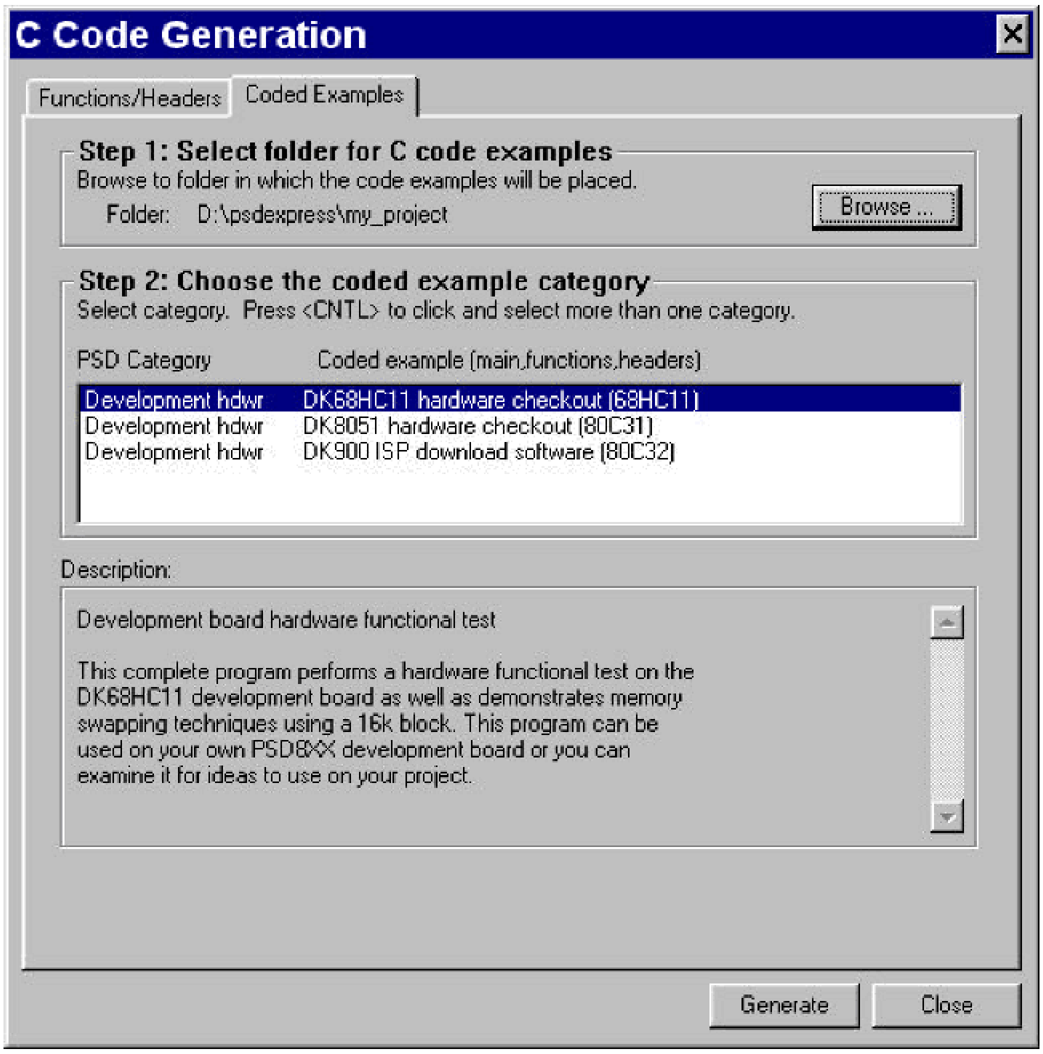
> <your_specified_name>.c—ANSI-C source for all of the selected functions
>
> psd4135g2.h—ANSI-C—header file to define particular PSD registers
>
> map4135g2.h—ANSI-C—header file to define locations of system memory elements (Main/Secondary Flash memory and PSD registers).

Notice that you do not have a choice to rename the two generated header files. This is because those header files are specified by name within the generated C function source file. If you edit the names of the generated header files, be sure to edit the generated C function source file to match the new header file names.

The three generated files may now be tailored and integrated into your compiler environment. The file psd4135g2.h contains a #define statement for each individual C function within the <your_specified_name>.c file. Edit psd4135g2.h and simply remove the comment delimiters (//) from the #define statement for each generated C function that you would like to be compiled with the rest of your C source code.

You may now wish to check out the “Coded Examples” screen (shown below). If not, simply click the **Close** button.

Coded Examples



Step 1: Select folder for C code examples

Browse to folder in which the code examples will be placed.

Folder: D:\psdexpress\my_project

Step 2: Choose the coded example category

Select category. Press <CNTL> to click and select more than one category.

PSD Category	Coded example (main,functions,headers)
Development hdwr	DK68HC11 hardware checkout (68HC11)
Development hdwr	DK8051 hardware checkout (80C31)
Development hdwr	DK900 ISP download software (80C32)

Description:

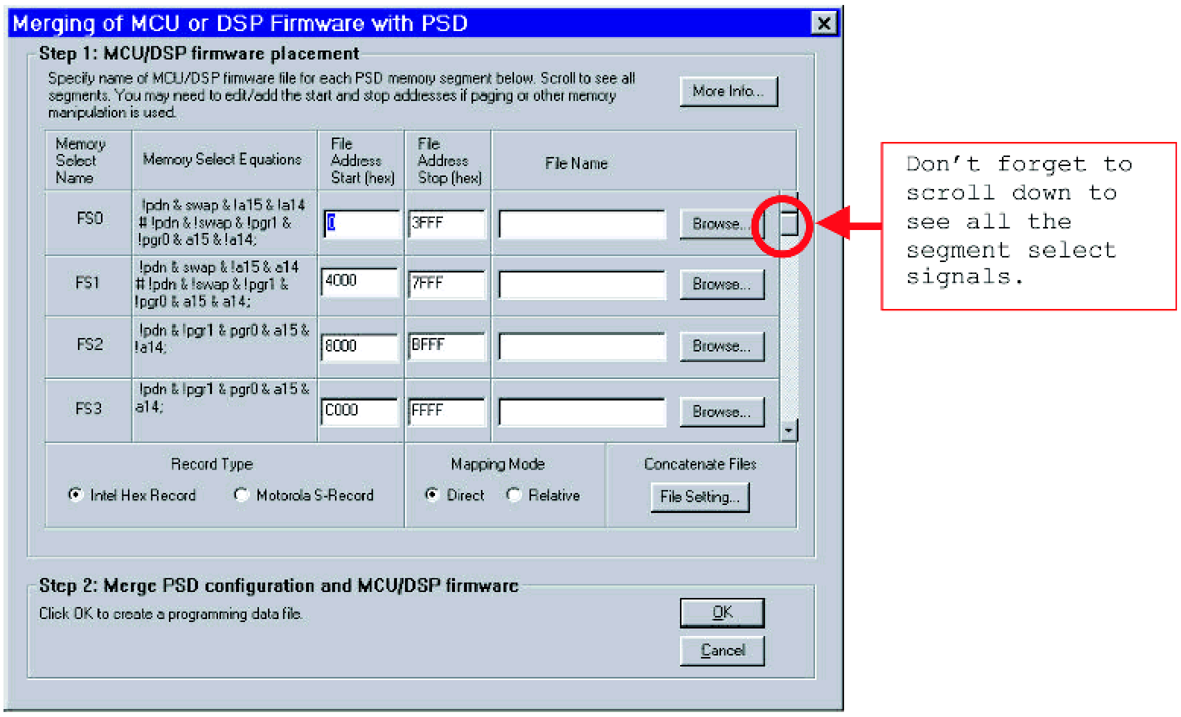
Development board hardware functional test

This complete program performs a hardware functional test on the DK68HC11 development board as well as demonstrates memory swapping techniques using a 16k block. This program can be used on your own PSD8XX development board or you can examine it for ideas to use on your project.

This screen is used to generate coded examples for the PSD development boards. Note that even if you do not intend to purchase a development board, examining the files generated may give you insights as to how pieces of code fit together, example UART code, and so on.

First, select the folder in which you wish to store the example files by clicking the **Browse…** button in Step 1. Then, highlight the development kit that most closely matches your end product and click **Generate**. Again, a message will appear in the Log Window indicating success. When finished, click the **Close** button.

MERGE MCU/DSP FIRMWARE WITH PSD


Don't forget to scroll down to see all the segment select signals.

Merging of MCU or DSP Firmware with PSD

Step 1: MCU/DSP firmware placement

Specify name of MCU/DSP firmware file for each PSD memory segment below. Scroll to see all segments. You may need to edit/add the start and stop addresses if paging or other memory manipulation is used.

More Info...

Memory Select Name	Memory Select Equations	File Address Start (hex)	File Address Stop (hex)	File Name
FS0	!pdn & swap & !a15 & !a14 # !pdn & !swap & !pgr1 & !pgr0 & a15 & !a14;	0	3FFF	
FS1	!pdn & swap & !a15 & a14 # !pdn & !swap & !pgr1 & !pgr0 & a15 & a14;	4000	7FFF	
FS2	!pdn & !pgr1 & pgr0 & a15 & !a14;	8000	BFFF	
FS3	!pdn & !pgr1 & pgr0 & a15 & a14;	C000	FFFF	

Record Type: [✓ Intel Hex Record] Motorola S-Record

Mapping Mode: [✓ Direct] Relative

Concatenate Files: File Setting...

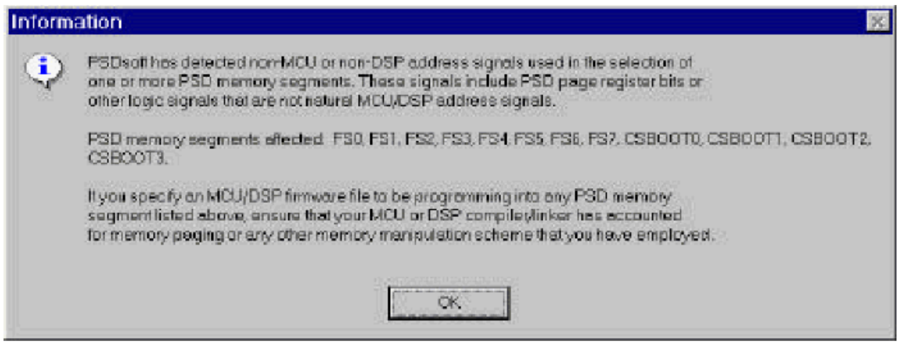
Step 2: Merge PSD configuration and MCU/DSP firmware

Click OK to create a programming data file.

OK Cancel

This screen, although innocent looking, is the main workhorse of PSDsoft Express. For it is here that the Programming data file (.obj) needed to program the PSD is finalized based on your design and firmware files. Read on to see what each step does, what happens behind the scenes and learn what various warning and error messages mean.

Note: you may have gotten the following warning when clicking “Merge MCU/DSP Firmware”:



Information

PSDsoft has detected non-MCU or non-DSP address signals used in the selection of one or more PSD memory segments. These signals include PSD page register bits or other logic signals that are not natural MCU/DSP address signals.

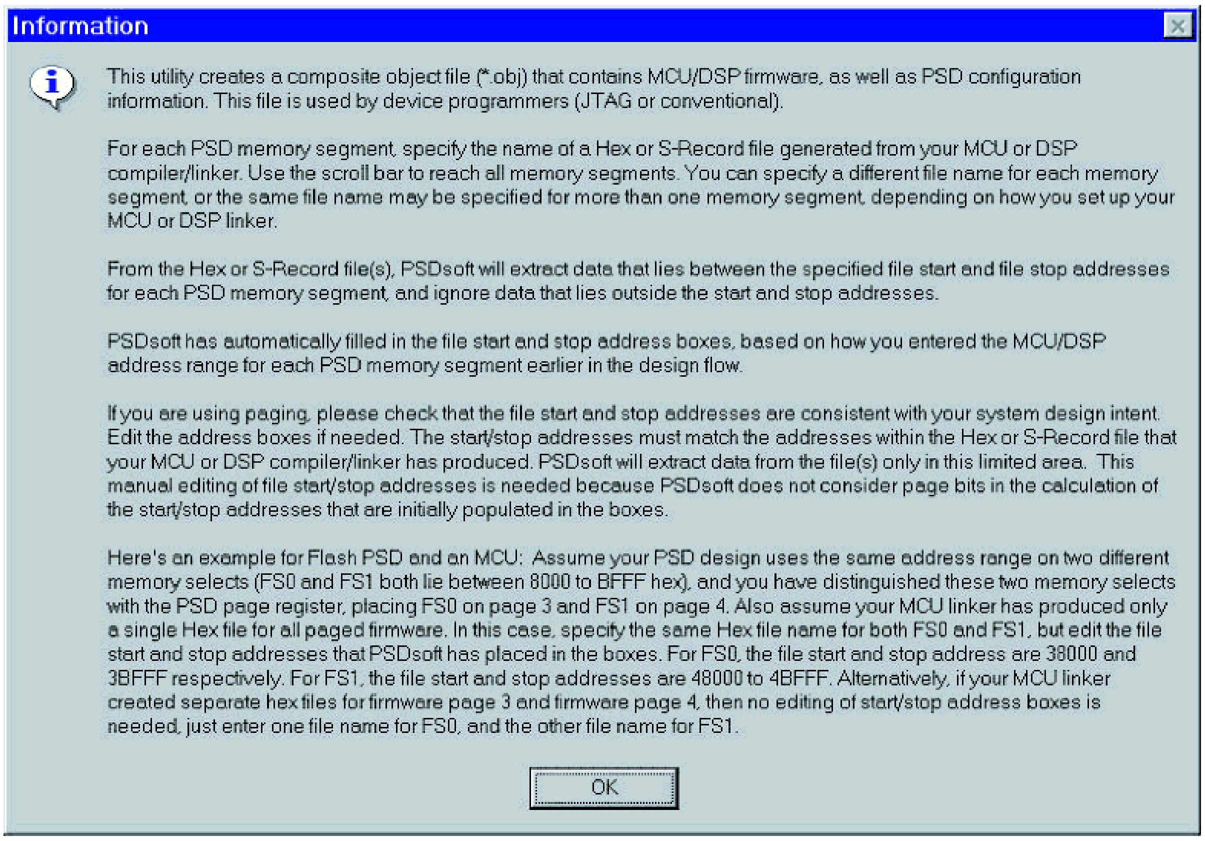
PSD memory segments affected: FS0, FS1, FS2, FS3, FS4, FS5, FS6, FS7, CSBOOT0, CSBOOT1, CSBOOT2, CSBOOT3.

If you specify an MCU/DSP firmware file to be programming into any PSD memory segment listed above, ensure that your MCU or DSP compiler/linker has accounted for memory paging or any other memory manipulation scheme that you have employed.

OK

Basically, the warning is telling you that PSDsoft Express detected chip-select equations that contain signals in addition to the natural MCU/DSP address signals. The warning is to remind you to ensure your MCU/DSP compiler/linker accounts for this. You may have to manually fill in the File start/stop addresses if they did not appear automatically. Also, for paged designs, check the Start/Stop addresses for validity because the internal checks performed by the software do not include the page bits.

If you would like more information on how the Merge utility operates, click the More Info… button at the top of the screen. You will be presented with the following screen:



This utility creates a composite object file (*.obj) that contains MCU/DSP firmware, as well as PSD configuration information. This file is used by device programmers (JTAG or conventional).

For each PSD memory segment, specify the name of a Hex or S-Record file generated from your MCU or DSP compiler/linker. Use the scroll bar to reach all memory segments. You can specify a different file name for each memory segment, or the same file name may be specified for more than one memory segment, depending on how you set up your MCU or DSP linker.

From the Hex or S-Record file(s), PSDsoft will extract data that lies between the specified file start and file stop addresses for each PSD memory segment, and ignore data that lies outside the start and stop addresses.

PSDsoft has automatically filled in the file start and stop address boxes, based on how you entered the MCU/DSP address range for each PSD memory segment earlier in the design flow.

If you are using paging, please check that the file start and stop addresses are consistent with your system design intent. Edit the address boxes if needed. The start/stop addresses must match the addresses within the Hex or S-Record file that your MCU or DSP compiler/linker has produced. PSDsoft will extract data from the file(s) only in this limited area. This manual editing of file start/stop addresses is needed because PSDsoft does not consider page bits in the calculation of the start/stop addresses that are initially populated in the boxes.

Here's an example for Flash PSD and an MCU: Assume your PSD design uses the same address range on two different memory selects (FS0 and FS1 both lie between 8000 to BFFF hex), and you have distinguished these two memory selects with the PSD page register, placing FS0 on page 3 and FS1 on page 4. Also assume your MCU linker has produced only a single Hex file for all paged firmware. In this case, specify the same Hex file name for both FS0 and FS1, but edit the file start and stop addresses that PSDsoft has placed in the boxes. For FS0, the file start and stop address are 38000 and 3BFFF respectively. For FS1, the file start and stop addresses are 48000 to 4BFFF. Alternatively, if your MCU linker created separate hex files for firmware page 3 and firmware page 4, then no editing of start/stop address boxes is needed, just enter one file name for FS0, and the other file name for FS1.

Step 1

Before you begin this step, you must have a valid Intel Hex or Motorola S-Record file or files created in your compiler/linker environment based on your source code. That is, using your MCUs/DSPs *firmware* file(s) to create part of the .obj file that will be later loaded into the PSDs non-volatile memories. There are three basic ways to generate these record files:

Generate an individual record file for each of the PSD’s internal memory segments that you defined earlier in the *Chip Select Equations*.

Generate one record file that spans more than one memory segment.

Any combination of the above.

Now let’s take a look at the “Merging of MCU/DSP Firmware with PSD” screen shown above. The first column shows the memory segment-select name (FS stands for main Flash memory Select, CSBOOT for secondary Flash memory select, EES for EEPROM select, and ES for EPROM select). Be sure to scroll down to get at all the internal PSD memory segments. Recall that you defined these segment-selects earlier using the *Chip Select Equations* screen. The next column shows the logical equivalent of the equations you have entered for the segment selects. The two middle columns show the hexadecimal equivalent of the file start and file stop address for which the segment selects will be valid. Note that if you used any non-address logic (such as paging) in your equations, PSDsoft Express will issue a warning (see below) and may or may not attempt to fill in the start and stop addresses. In the last column, click the **Browse...** button and locate your firmware files to be loaded into the PSD’s non-volatile memories.

For example, if you look at the above screen capture, four Flash memory segments and three Hex files are shown. Thus, FS0 is valid from 0h to 3FFFh, FS1 is valid from 4000h to 7FFFh, and the file “my-hex.hex" contains code for both segments. Notice that FS2 and FS3 are both valid C000h to FFFFh (but on different pages) and therefore they have different Hex files. Different Hex files are generally used whenever code will be located at the same physical MCU/DSP address, but on different memory pages. Thus, you would have had to set up the *Page Register* prior to this point.

Once you have filled in all the file names, choose the appropriate record type (Intel Hex or Motorola S) that matches the type output by your linker.

Next, select the desired mapping mode. "Direct" mapping implies there is a one-to-one correspondence with the address output by the MCU/DSP and the address that selects the internal memory segments of the PSDs. "Relative" mapping enables you to specify different physical addresses (output by your MCU/DSP based on your firmware file) than the specified equations that select the memory segments within the PSD. Most users will select "Direct".

Step 2

When you have completed Step 1 and are ready for PSDsoft Express to complete the .obj file creation, click **OK**. If you do not wish to create the .obj file, click **Cancel** instead.

Behind the Scenes

The following actions take place behind the scenes:

- The !pdn (the active-low internal power-down signal) is automatically included in the segment-select equations for PSDs that have the Automatic Power Down feature.
- If you chose to use Port D, pin 2 (pd2) as the chip-select input (_csi), it will also be automatically be included internal segment-select equations.
- If you opt to use Motorola S-Record as the record type, PSDsoft converts the file to Hex record format before creating the .obj file. It will save a file with the same name as your S-Record, but with a .hex extension. It is for this reason that you must <u>not</u> give your Motorola S-Record file a .hex extension.
- Once you click **OK**, an Address Translation process is invoked. During this process, the record file (firmware) will be loaded into the appropriate PSD non-volatile memory segment. These memory segments have an address within the PSD that are most likely different from the ones defined in the "Chip Select Equations" window. The Address Translator creates the .obj file based on the PSDs own internal addresses. To see how the addresses are translated, select **Report->Address Translation**. Check the *Hints and Tips* section below for more useful information.

Common Warnings and Error Messages

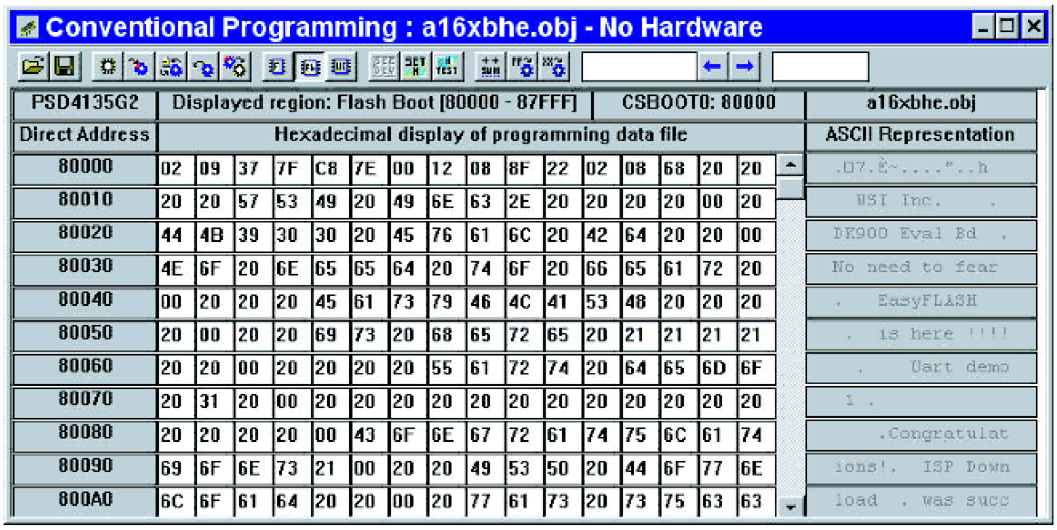
This section will be updated frequently to cover any questions you may have about error and warning messages you may have encountered. If anything is unclear about the message you received and this section either doesn't address the message or does not answer your question, email *apps.psd@st.com*.

Hints ant Tips

It's always a good idea to see if PSDsoft Express interpreted your firmware file the way you anticipated and see where it has placed your code within the PSDs internal memory segments. To do so, click on *STMicroelectronics Conventional Programmers* in the design flow. Ensure that your .obj file has loaded by checking the name at the top of the window. Now select the memory within the PSD that you would like to view by clicking on the appropriate icon in the toolbar. Remember that the addresses you are seeing in the "Direct Address" column are the addresses that only the PSD knows about.

The screen capture below shows that there is code in the selected memory. No data or code would show up as all FFs (blank). For more information on this "Conventional Programming" window, see the *Conven-*

tional Programming section.



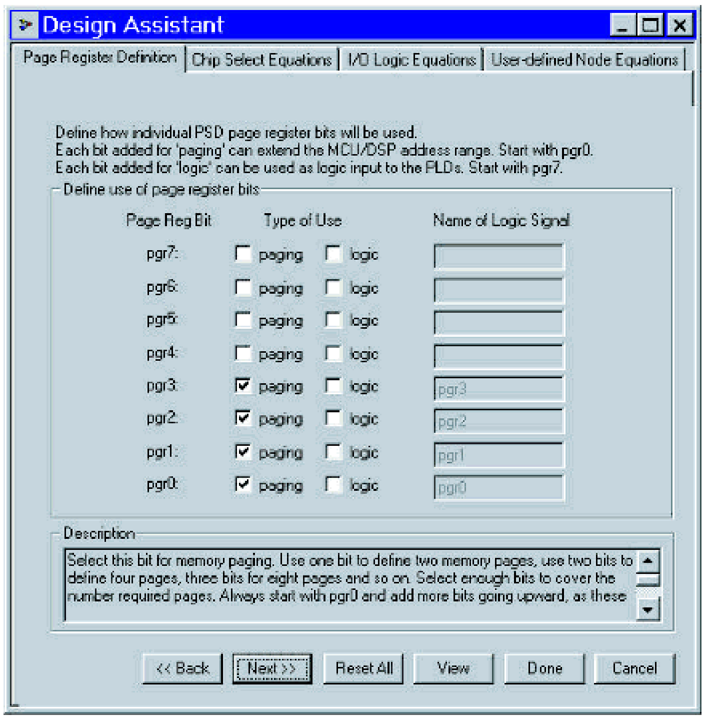
Conventional Programming : a16xbhe.obj - No Hardware

PSD4135G2	Displayed region: Flash Boot [80000 - 87FFF]								CSBOOT0: 80000								a16xbhe.obj
Direct Address	Hexadecimal display of programming data file																ASCII Representation
80000	02	09	37	7F	C8	7E	00	12	08	8F	22	02	08	68	20	20	.07.È~....”..h
80010	20	20	57	53	49	20	49	6E	63	2E	20	20	20	20	00	20	WSI Inc. .
80020	44	4B	39	30	30	20	45	76	61	6C	20	42	64	20	20	00	DK900 Eval Bd .
80030	4E	6F	20	6E	65	65	64	20	74	6F	20	66	65	61	72	20	No need to fear
80040	00	20	20	20	45	61	73	79	46	4C	41	53	48	20	20	20	. EasyFLASH
80050	20	00	20	20	69	73	20	68	65	72	65	20	21	21	21	21	. is here !!!!
80060	20	20	00	20	20	20	20	55	61	72	74	20	64	65	6D	6F	. Uart demo
80070	20	31	20	00	20	20	20	20	20	20	20	20	20	20	20	20	1 .
80080	20	20	20	20	00	43	6F	6E	67	72	61	74	75	6C	61	74	.Congratulat
80090	69	6F	6E	73	21	00	20	20	49	53	50	20	44	6F	77	6E	ions!. ISP Down
800A0	6C	6F	61	64	20	20	00	20	77	61	73	20	73	75	63	63	load . was succ

Assigning Two Hex Files to a Single Memory Segment

With some MCU/PSD combinations, it is necessary to assign two Hex files to a single memory segment. Currently, the only combination that may require this is a PSD835 or PSD935 (each having 64 KByte Flash memory segments) and a MCU limited to 64 KByte total address space. Essentially, each Flash memory segment will need to be broken up into two 32 KByte pages. The way PSDsoft handles breaking up a physical 64 KByte segment into two logical 32 KByte segments is through the use of a keyword (node) called “fa15”. To keep things simple, the example shows how to set up fa15 such that it will be set to be true whenever the first bit of the Page Register is set to 1 or High (pgr0). Thus, no extra overhead nor knowledge of the fa15 on the part of the MCU is required.

Setting up the Page Register. If you intend to use all of the memory within the PSD835/935 and you have an MCU that only supports a 64 KByte address space, you will need sixteen 32 KByte pages, which implies four page bits ($2^4 = 16$). The screen capture below shows how this is set up in the “Design Assistant”.



Define how individual PSD page register bits will be used.
Each bit added for 'paging' can extend the MCU/DSP address range. Start with pgr0.
Each bit added for 'logic' can be used as logic input to the PLDs. Start with pgr7.

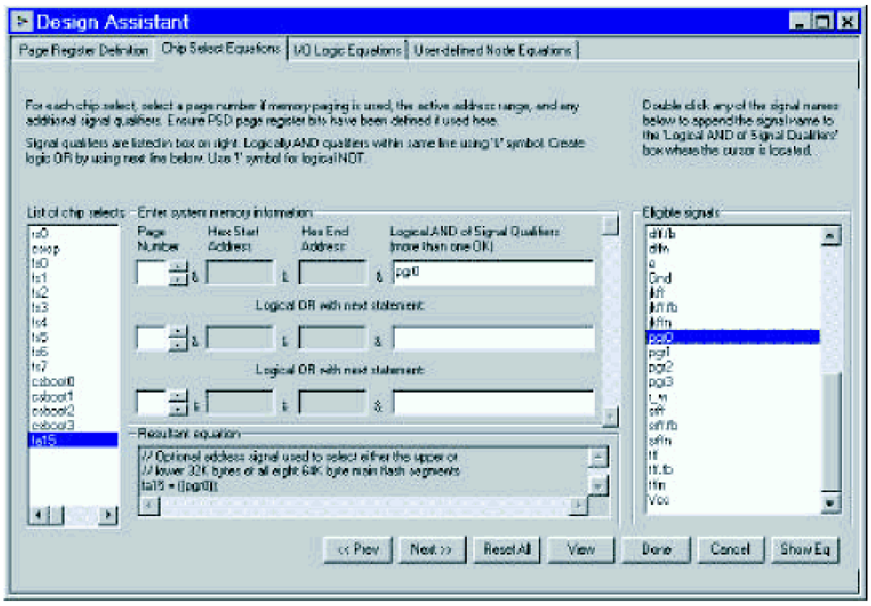
Define use of page register bits

Page Reg Bit	Type of Use		Name of Logic Signal
pgr7:	☐ paging	☐ logic	
pgr6:	☐ paging	☐ logic	
pgr5:	☐ paging	☐ logic	
pgr4:	☐ paging	☐ logic	
pgr3:	☑ paging	☐ logic	pgr3
pgr2:	☑ paging	☐ logic	pgr2
pgr1:	☑ paging	☐ logic	pgr1
pgr0:	☑ paging	☐ logic	pgr0

Description

Select this bit for memory paging. Use one bit to define two memory pages, use two bits to define four pages, three bits for eight pages and so on. Select enough bits to cover the number required pages. Always start with pgr0 and add more bits going upward, as these

Setting up the fa15 bit. The screen capture below shows how the fa15 bit is set to be true when the first bit of the Page Register (pgr0) is true.



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

List of chip selects: rs0, csiop, fs0, fs1, fs2, fs3, fs4, fs5, fs6, fs7, csboot0, csboot1, csboot2, csboot3, fa15

Enter system memory information

Page Number		Hex Start Address		Hex End Address		Logical AND of Signal Qualifiers (more than one OK)
	&		&		&	pgr0

Logical OR with next statement:

Logical OR with next statement:

Resultant equation

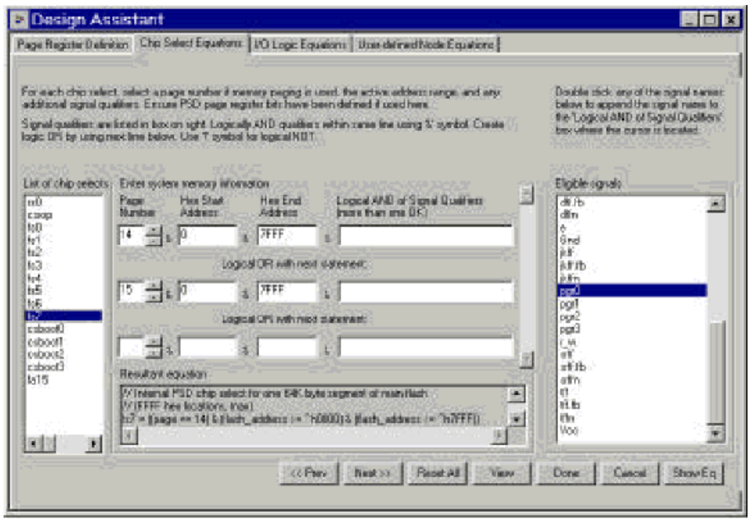
```
// Optional address signal used to select either the upper or
// lower 32K bytes of all eight 64K byte main flash segments
fa15 = (pgr0);
```

Eligible signals: dif/fb, difn, e, Gnd, jkff, jkff.fb, jkffn, pgr0, pgr1, pgr2, pgr3, t_m, sff, sff.fb, sffn, tff, tff.fb, tffn, Vcc

Setting up the Flash memory equations. Recall that each Flash memory sector needs to be broken up into two 32 KByte logical segments. Again, when the MCU code manipulates the Page Register, fa15 will automatically be set and the correct half of the Flash memory segment will be accessed. For reference, fs7 is shown below:

Setting up the Merge MCU/DSP Firmware with PSD. The last step is correct setup of the “Merging of

MCU or DSP Firmware with PSD” screen. Note that your linker should break up the code into 32 KByte pages. For your reference, fs0 and fs1 are shown below:



For each chip select, select a page number if memory paging is used, the active address range, and any additional signal qualifiers. Ensure PSD page register bits have been defined if used here.

Signal qualifiers are listed in box on right. Logically AND qualifiers within same line using '&' symbol. Create logic OR by using next line below. Use '!' symbol for logical NOT.

Double click any of the signal names below to append the signal name to the 'Logical AND of Signal Qualifiers' box where the cursor is located.

Enter system memory information

Page Number	Hex Start Address	Hex End Address	Logical AND of Signal Qualifiers (more than one OK)
14	0	7FFF	
Logical OR with next statement:			
15	0	7FFF	
Logical OR with next statement:			

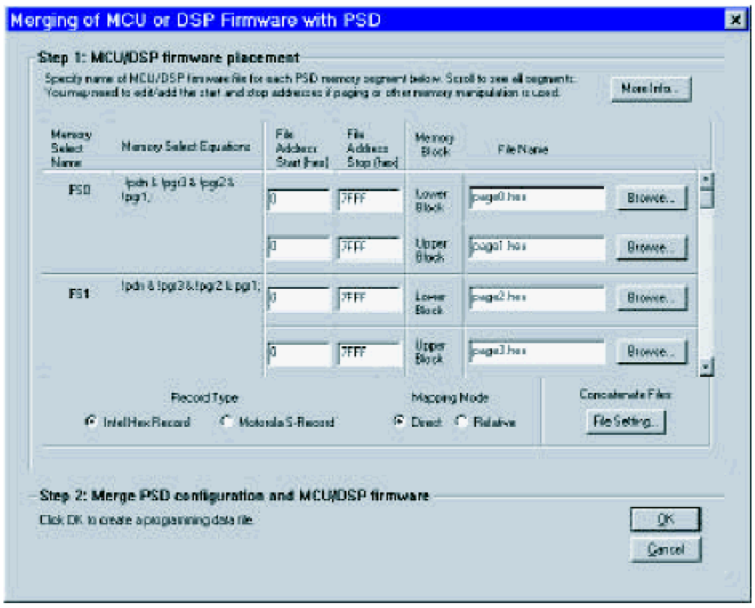
Resultant equation

```
// Internal PSD chip select for one 64K byte segment of main flash
// (FFFF hex locations, max)
fs7 = ((page == 14) & (flash_address >= ^h0000) & (flash_address <= ^h7FFF))
```

Note that the Hex file may contain data outside the range noted (0 – 7FFFh in this case) because the merge utility only extracts information within the 0 – 7FFFh range.

Combining Hex or S-Record Files

PSDsoft provides the ability to combine (concatenate) Hex or S-Record files. Doing so is a simple three step process, as can be seen in the “MCU/DSP Firmware File Concatenation” screen below.



Step 1: MCU/DSP firmware placement

Specify name of MCU/DSP firmware file for each PSD memory segment below. Scroll to see all segments. You may need to edit/add the start and stop addresses if paging or other memory manipulation is used.

Memory Select Name	Memory Select Equations	File Address Start (hex)	File Address Stop (hex)	Memory Block	File Name
FS0	!pdn & !pgr3 & !pgr2 & !pgr1;	0	7FFF	Lower Block	page0.hex
		0	7FFF	Upper Block	page1.hex
FS1	!pdn & !pgr3 & !pgr2 & pgr1;	0	7FFF	Lower Block	page2.hex
		0	7FFF	Upper Block	page3.hex

Record Type: [✓ Intel Hex Record] Motorola S-Record

Mapping Mode: [✓ Direct] Relative

Concatenate Files

Step 2: Merge PSD configuration and MCU/DSP firmware

Click OK to create a programming data file.

The steps to concatenate two or more files are as follows:

1. Select the file type to be concatenated Hex or S-Record.
2. Click the **Browse** button in Step 2 and search for your first file. This will open a “Open” window
3. Locate your file and select **Open**.
4. The file will now be listed in Step 2 on the “Select Folder and File:” line.

5. Now Click Add and the first file will appear in the list.
6. Repeat Steps 2 through 5 until you have all the files you wish to concatenate in the file list.

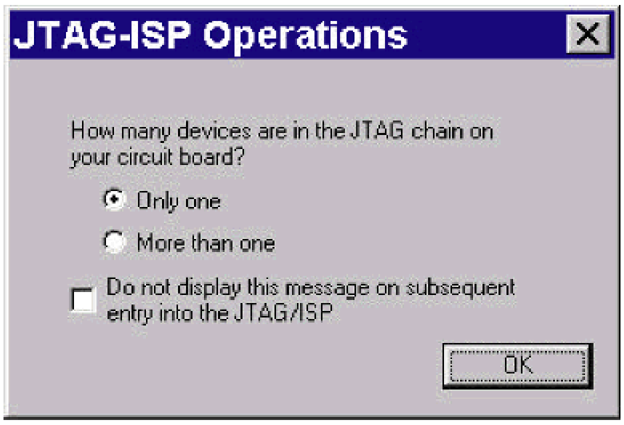
Note: the files will be combined in the order listed in the *File Name* section. If you need to change the order of the files, use the Move up and down arrows. If you wish to delete a file, first left-click on it in the window to select it and then click **Delete**.

7. Click the lower **Browse** button in Step 3. Again, a “Open” window will appear.
8. Type the desired file name in the “File Name” box and click **Open**. Your file name and location now appears in the “Select Folder and File” line of Step 3.
9. Click the **Concatenate** button to combine the files. You will get status messages at the bottom of the screen.

Note: **Save Setup** can be used to save the current state of the screen and **Retrieve Setup** can be used to retrieve the last saved session.

PSD JTAG/ISP (FLASH-BASED PSDS ONLY)

When you click on the “STMicroelectronics JTAG/ISP” box in the *Design Flow*, you are presented with the following question (unless you had previously checked the checkbox):


JTAG-ISP Operations

How many devices are in the JTAG chain on your circuit board?

[✓ Only one]

More than one

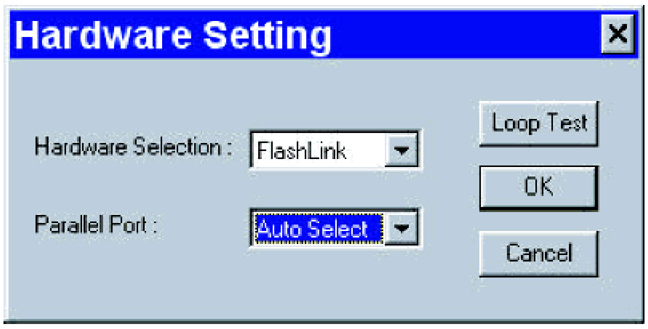
Do not display this message on subsequent entry into the JTAG/ISP

Select the number of devices in your JTAG Chain. If you will always have the same number of devices, you can choose to check the box “Do not display this message on subsequent entry into the JTAG/ISP”. Note: the option to see this message can also be set in the **Project->Preferences** menu. If you are unsure what a JTAG chain is, see the *JTAG Chain Definition and Rules*. Also, every PSD user that is using the JTAG port is encouraged to read The “JTAG-ISP Information for Flash PSDs” application note.

If you have used this window previously and wish to retrieve a JTAG-ISP setup, skip to *Save or Retrieve JTAG-ISP Setup*.

Hardware Setup

If this is your first time using the FlashLINK cable or you have switched computers or hardware, it’s a good idea to set up your hardware first. To do so, click the **HW Setup** button at the bottom of the window. When you do so, the following window pops up:



Hardware Selection : FlashLink

Parallel Port : Auto Select

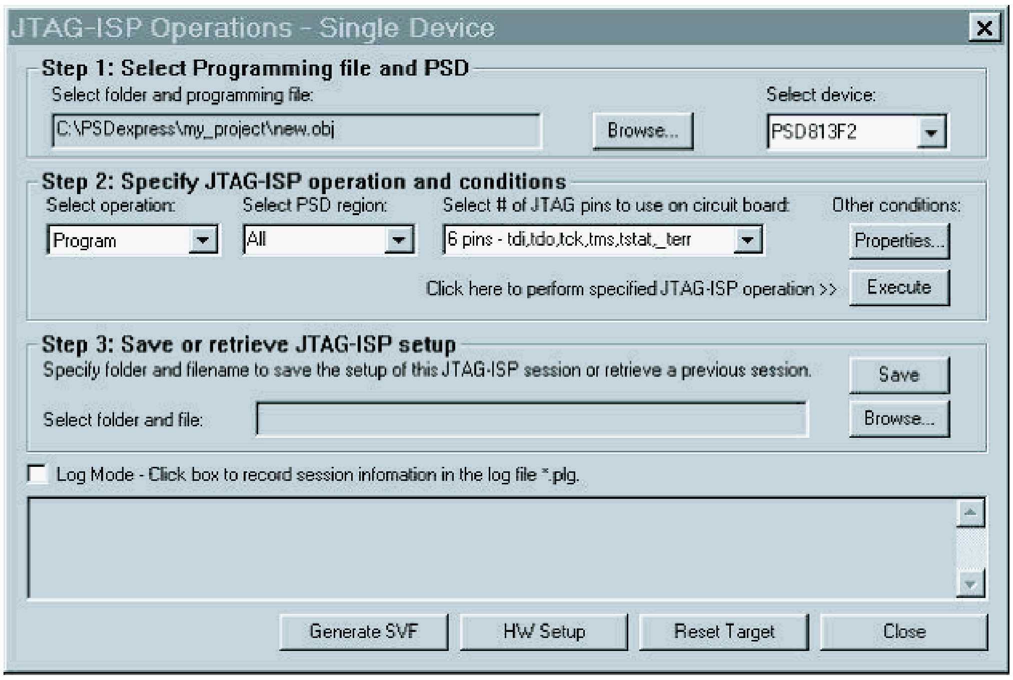
FlashLINK is the only “choice” available for “Hardware Selection”. Depending on your computer, you may or may not be able to change the setting in Parallel Port. Generally, “Auto Select” works in most cases unless your port is not set up correctly.

It’s a good idea to run the Loop Test before attempting any JTAG operations on the FlashLINK cable. To do so, first ensure that you have the special loopback connector attached to the FlashLINK and that the power and ground wires are connected to a power source and the power is on. Then, simply click on the **Loop Test** button and run the test. If you get an error message, see *Error and Warning Messages*. If you pass the test, you are ready to attempt a JTAG operation, but passing the test does not mean you will have success.

If only have one device in your JTAG chain, read the next section. Otherwise, skip to the *JTAG Chaining* section.

JTAG-ISP Operations for a Single PSD Device

If you selected “Only one” when ask how many devices are in the JTAG chain on your circuit board, you should see a window similar to the following:



Step 1: Select Programming file and PSD

Select folder and programming file: C:\PSDexpress\my_project\new.obj

Select device: PSD813F2

Step 2: Specify JTAG-ISP operation and conditions

Select operation: Program

Select PSD region: All

Select # of JTAG pins to use on circuit board: 6 pins - tdi,tdo,tck,tms,tstat,_terr

Other conditions:

Click here to perform specified JTAG-ISP operation >>

Step 3: Save or retrieve JTAG-ISP setup

Specify folder and filename to save the setup of this JTAG-ISP session or retrieve a previous session.

Select folder and file:

☐ Log Mode - Click box to record session infomation in the log file *.plg.

Step 1: Select Programming File and PSD

If you have a valid project open in which you have already created the Programming data file (.obj) then this window will open up with the current project's .obj file already loaded and the device should be selected for you. If you do not have a project open or wish to open a different .obj file, click the **Browse…** button in Step 1. Any time you load an .obj file, the correct device should automatically be loaded. If, for some reason, you wish to program an .obj file that was created for one type of device into another type, you can change the device in this step. This, however, is not recommended and may work if you are selecting a similar device within the same family of devices.

Step 2: Specify JTAG-ISP Operation and Conditions

First, select the desired operation from the pull-down menu. The available options are:

- Blank Test—check to see if any region of the PSD has been programmed.
- Erase—erase one or more regions of the PSD.
- Program—write code, configuration, or data to one or more parts of the PSD based on your .obj file.
- Verify—compare the actual contents of the .obj programmed within the PSD to the expected contents of the .obj file that is currently loaded.
- Upload—dump the contents of the PSD to an .obj file. Note: upload will produce garbage if the *security bit* had been previously set and the only way to erase the security bit is by performing a full-chip erase.
- Bypass—place the device in bypass mode. Since you are only using a single device, this instruction does not have much use other than to quickly check to see if your JTAG interface is working properly.

Next, select the PSD region(s) that the operation selected previously will affect. Note that the software will look at the .obj file before a Program operation if the "All" option is chosen and only program the required areas of the PSD. For example, if you only specified a firmware file for fs0 (main Flash memory, segment 0) during the *Merge MCU/DSP Firmware* process, only fs0 and the PLD portion of the PSD would be programmed.

Select the number of JTAG pins to use based on your selections in the *Pin Definitions* screen. The default should reflect the number of pins you specified in the Pin Definitions screen.

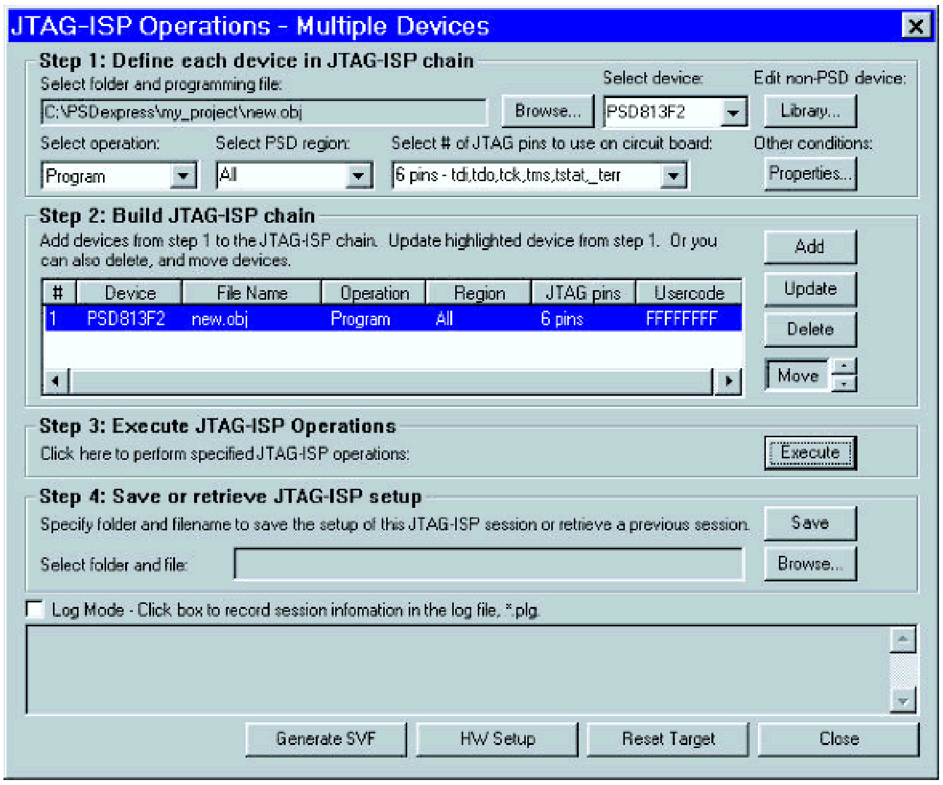
Lastly, click on the **Properties…** button, which is used to set up the PSD's I/O during JTAG operations, view the JTAG Properties, and check the JTAG User Code. See the *Properties* section.

When you are satisfied that everything is in order and ready to go, click **Execute**. If you get an error message during execution, see *Error and Warning Messages*.

When you are done with Step 2, skip to *Save or Retrieve JTAG-ISP Setup*.

JTAG-ISP Operations for Multiple Devices

If you opted to use “More than one” devices in a JTAG chain, you should see a window similar to the following one:


JTAG-ISP Operations - Multiple Devices

Step 1: Define each device in JTAG-ISP chain

Select folder and programming file: C:\PSDexpress\my_project\new.obj

Select device: PSD813F2

Edit non-PSD device:

Select operation: Program

Select PSD region: All

Select # of JTAG pins to use on circuit board: 6 pins - tdi,tdo,tck,tms,tstat,_terr

Other conditions:

Step 2: Build JTAG-ISP chain

Add devices from step 1 to the JTAG-ISP chain. Update highlighted device from step 1. Or you can also delete, and move devices.

#	Device	File Name	Operation	Region	JTAG pins	Usercode
1	PSD813F2	new.obj	Program	All	6 pins	FFFFFFFF

Step 3: Execute JTAG-ISP Operations

Click here to perform specified JTAG-ISP operations:

Step 4: Save or retrieve JTAG-ISP setup

Specify folder and filename to save the setup of this JTAG-ISP session or retrieve a previous session.

Select folder and file:

Log Mode - Click box to record session information in the log file, *.plg

Step 1: Define each device in the JTAG-ISP chain

Adding PSD Devices

If you have a valid project open in which you have already created the Programming data file (.obj) then this window will open up with the current project’s .obj file already loaded and the PSD device should be selected for you. If you do not have a project open or wish to open a different .obj file, click the **Browse…** button in Step 1. Any time you load an .obj file, the correct PSD device should automatically be loaded. To add the device to the chain, click the **Add** button.

Select the desired operation from the pull-down menu. The available options are:

- Blank Test—check to see if any part of the PSD has been programmed.
- Erase—erase one or more parts of the PSD.
- Program—write code, configuration, or data to one or more parts of the PSD based on your .obj file.
- Verify—compare the actual contents of the .obj programmed within the PSD to the expected contents of the .obj file that is currently loaded.
- Upload—dump the contents of the PSD to an .obj file. Note: upload will produce garbage if the *security bit* had been previously set and the only way to erase the security bit is by performing a full-chip erase.
- Bypass—place the part in bypass mode.

Next, select the PSD region(s) that the operation selected previously will affect. Note that the software will look at the .obj file before a Program operation if the “All” option is chosen and only program the required areas of the PSD. For example, if you only specified a firmware file for fs0 (main Flash memory, segment

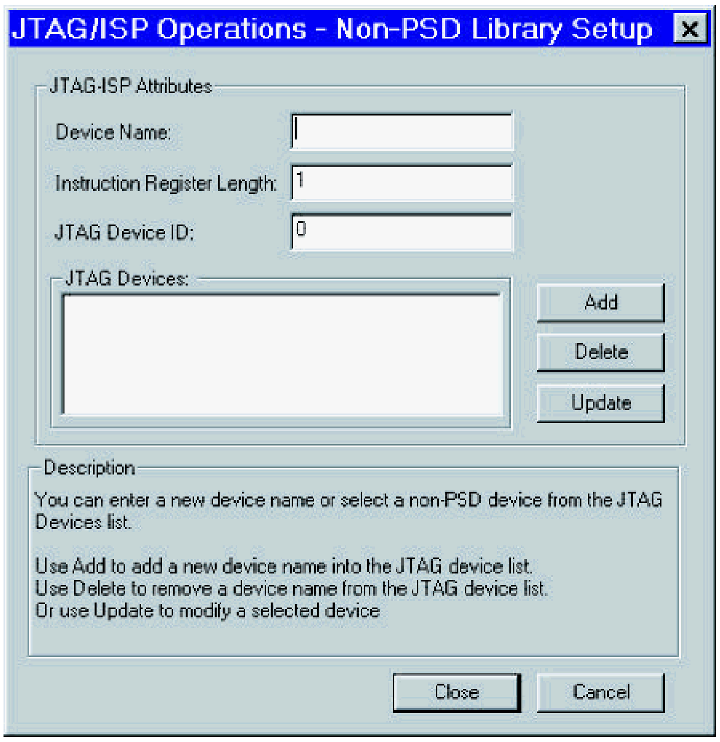
0) during the *Merge MCU/DSP Firmware* process, only fs0 and the PLD portion of the PSD would be programmed.

Select the number of JTAG pins to use based on your selections in the *Pin Definitions* screen. The default value should reflect what you have chosen in the Pin Definitions screen.

Lastly, click on the **Properties…** button, which is used to set up the PSD's I/O during JTAG operations, view the JTAG Properties, and check the JTAG User Code. See the *Properties* section.

Adding Non-PSD Devices

Non-PSD devices can be added to the chain only if they were defined previously by clicking on the **Library** button. When you click this button, the following window appears:



Device Name:
Instruction Register Length: 1
JTAG Device ID: 0

You can enter a new device name or select a non-PSD device from the JTAG Devices list.

Use Add to add a new device name into the JTAG device list.
Use Delete to remove a device name from the JTAG device list.
Or use Update to modify a selected device

Enter the desired device name. This name will appear in the pull-down list in the main screen. Next enter the Instruction Register Length and JTAG Device ID in hexadecimal. You will need to obtain these numbers from the device manufacturer. Once you have entered all the device information, click the **Add** button. You should see the device appear in the "JTAG Devices" box. You can continue adding more devices in this manner. If you wish to delete a device from the list, highlight it and click **Delete**. If you wish to change any information for a device previously defined, highlight the device, change the information, and click the **Update** button. After you are finished adding or updating devices, click **Close** to return to the main JTAG window.

Once you have defined your non-PSD device select it using the "Select Device" pull-down bar and click the **Add** button. Note: for non-PSD devices, the only operation available is Bypass and the number of JTAG pins can only be set to four. If you want PSDsoft Express to check the JTAG Device ID entered previously, click on the **Properties** button and click the checkbox next to the JTAG Device ID.

Step 2: Build the JTAG-ISP Chain

Each time you click the **Add** button, a device is added to the JTAG chain in the order in which you selected them. This order must match how the devices are physically ordered in the chain. If you are unsure of the physical order, see *JTAG Chain Definition and Rules*. If your devices are not in the correct order, highlight the device that needs to be moved by left clicking on it in the chain and then click the Move Up or Move Down arrows. If you need to delete a device, highlight it and click the **Delete** button. If you need to change the operation, the number of pins, or the properties for a device, highlight it and then choose the appropriate operation, number of pins or click the **Properties…** button and click **Update** when finished. Note that Device #1 should be the first device on the board that has its TDI pin connected to the FlashLINK cable. See the *JTAG Chain Definition and Rules*.

Important note: if you wish to change anything, such as the operation, PSD region, and so on, you must highlight the line to be changed in Step 2 before you make your changes in Step 1. Once the desired changes have been made, click the **Update** button in Step 2. Alternatively, for changing the operation or properties only, you can right-click on the line to be changed.

Step 3: Execute JTAG-ISP Operations

When you have completed the first two steps, simply click the **Execute** button. If you get an error message during execution, see *Error and Warning Messages*. If you have more than one operation or more than one PSD device, go back to the previous steps, and update and continue execution until you are finished.

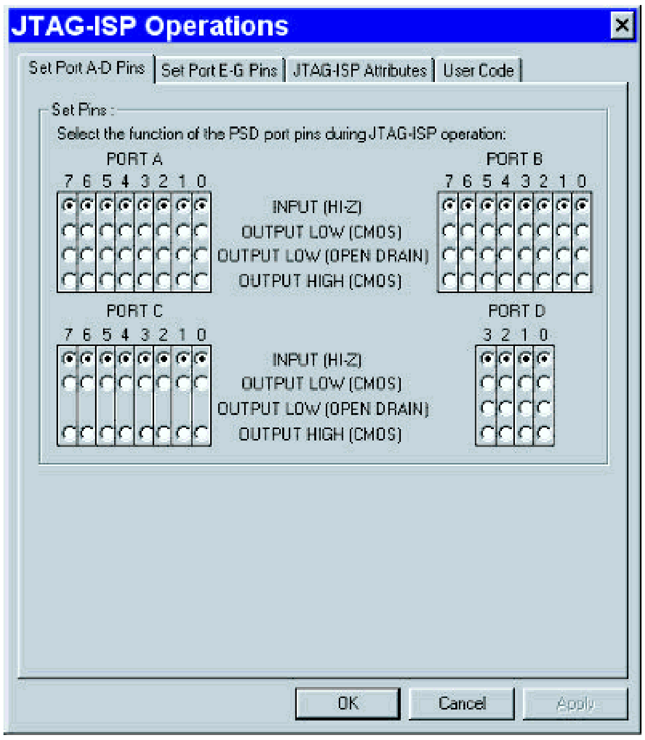
Save or Retrieve JTAG-ISP Setup

Save. If this is your first use of the “JTAG-ISP Operations” window for your current project, click the **Save** button. A “Save As” window will appear prompting for you to enter the file name. Click **Save** in this window when done or **Cancel** to exit this without saving.

Retrieve. If you had previously saved the JTAG settings for your project, click the **Browse…** button (make sure you are clicking the correct Browse button) and select the appropriate file in the ”Open” window and click **Open**.

Properties

Clicking on the **Properties…** button when a PSD device is selected will display a window similar to the one below. If you have clicked the button with a non-PSD device selected, see the *JTAG-ISP Attributes*.



Set Pins :
Select the function of the PSD port pins during JTAG-ISP operation:

PORT A 7 6 5 4 3 2 1 0		PORT B 7 6 5 4 3 2 1 0
● ● ● ● ● ● ● ●	INPUT (HI-Z)	● ● ● ● ● ● ● ●
○ ○ ○ ○ ○ ○ ○ ○	OUTPUT LOW (CMOS)	○ ○ ○ ○ ○ ○ ○ ○
○ ○ ○ ○ ○ ○ ○ ○	OUTPUT LOW (OPEN DRAIN)	○ ○ ○ ○ ○ ○ ○ ○
○ ○ ○ ○ ○ ○ ○ ○	OUTPUT HIGH (CMOS)	○ ○ ○ ○ ○ ○ ○ ○

PORT C 7 6 5 4 3 2 1 0		PORT D 3 2 1 0
● ● ● ● ● ● ● ●	INPUT (HI-Z)	● ● ● ●
○ ○ ○ ○ ○ ○ ○ ○	OUTPUT LOW (CMOS)	○ ○ ○ ○
	OUTPUT LOW (OPEN DRAIN)	○ ○ ○ ○
○ ○ ○ ○ ○ ○ ○ ○	OUTPUT HIGH (CMOS)	○ ○ ○ ○

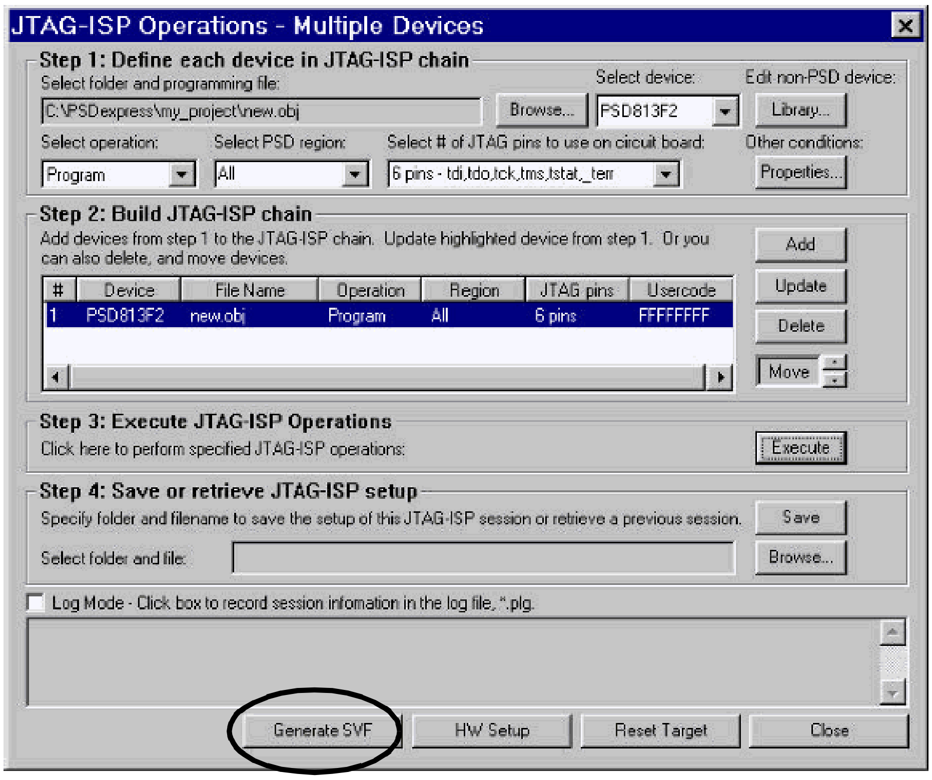
Set Port Pins. Depending on your PSD, the first one or two tabs are dedicated to setting up the PSD's I/O pins during JTAG operations. The default (except for the JTAG pins) is Input, which is usually fine for most pins. (Note that the PSD will not respond to any non-JTAG I/O.) However, sometimes it may be desirable to set a pin or pins to output during JTAG. For example, if you have chip-select signal being generated from the PSD that selects a device that potentially could drive signals on the JTAG lines (if you are multiplexing the pins), you would want that chip-select to be inactive during the JTAG operation.

JTAG-ISP Attributes. If you click on the "JTAG-ISP Attributes" tab for a PSD device, you are shown the device name and Instruction Register length. This information may be useful to other design programs (that don't have an "auto-detect" function).

User Code. Basically, by clicking on the "User Code" tab, you are provided with a space to enter an IEEE 1149.1 User Code that will be compared to the value previously entered in the *Additional PSD Settings*. See the Description box for more details.

Generating a SVF File

PSDsoft can be used to generate a Serial Vector Format file (.svf) by clicking the **Generate SVF** button.


JTAG-ISP Operations - Multiple Devices

Step 1: Define each device in JTAG-ISP chain

Select folder and programming file: C:\PSDexpress\my_project\new.obj

Select device: PSD813F2

Edit non-PSD device:

Select operation: Program

Select PSD region: All

Select # of JTAG pins to use on circuit board: 6 pins - tdi,tdo,tck,tms,tstat,_terr

Other conditions:

Step 2: Build JTAG-ISP chain

Add devices from step 1 to the JTAG-ISP chain. Update highlighted device from step 1. Or you can also delete, and move devices.

#	Device	File Name	Operation	Region	JTAG pins	Usercode
1	PSD813F2	new.obj	Program	All	6 pins	FFFFFFFF

Step 3: Execute JTAG-ISP Operations

Click here to perform specified JTAG-ISP operations:

Step 4: Save or retrieve JTAG-ISP setup

Specify folder and filename to save the setup of this JTAG-ISP session or retrieve a previous session.

Select folder and file:

Log Mode - Click box to record session information in the log file, *.plg.

The main purpose of the .svf file is to provide a method to program a PSD during manufacturing or at runtime using your MCU or DSP via the JTAG pins. When the **Generate SVF** button is clicked, two sets of files are generated: a process file and data files. The .svf file is a process file that shows the steps necessary to program the PSD via the JTAG port. If you wish to use this with your MCU or DSP, you will need interpret the low-level commands in the .svf file and write equivalent code. The .dat files that are generated are the equivalent of the code update for the PSD. Thus the software routines written for your MCU/DSP would need to control the four or six JTAG pins, depending on the state of the JTAG screen when the .svf file is saved. The idea is to write code that mimics the actions in the .svf file. This will reprogram the PSD via the JTAG lines. This could be done by a PC (test equipment) using the .dat files or even from another MCU on the board.

Behind the Scenes

Whenever you click the **Execute** button, quite a few actions take place before the desired operation is executed, including:

- The parallel port is taken over by the driver and the driver checks to see if there is communication between your PC's parallel port and the FlashLINK cable.
- The User Code, and Device ID(s) are checked
- Some signals will be sent and a response must be received within a certain number of TCK clock cycles.

Each of these steps can generate error messages before your desired operation gets under way. If you get an error message, see the next section.

Error and Warning Messages

If you get an error message, you can take the following steps:

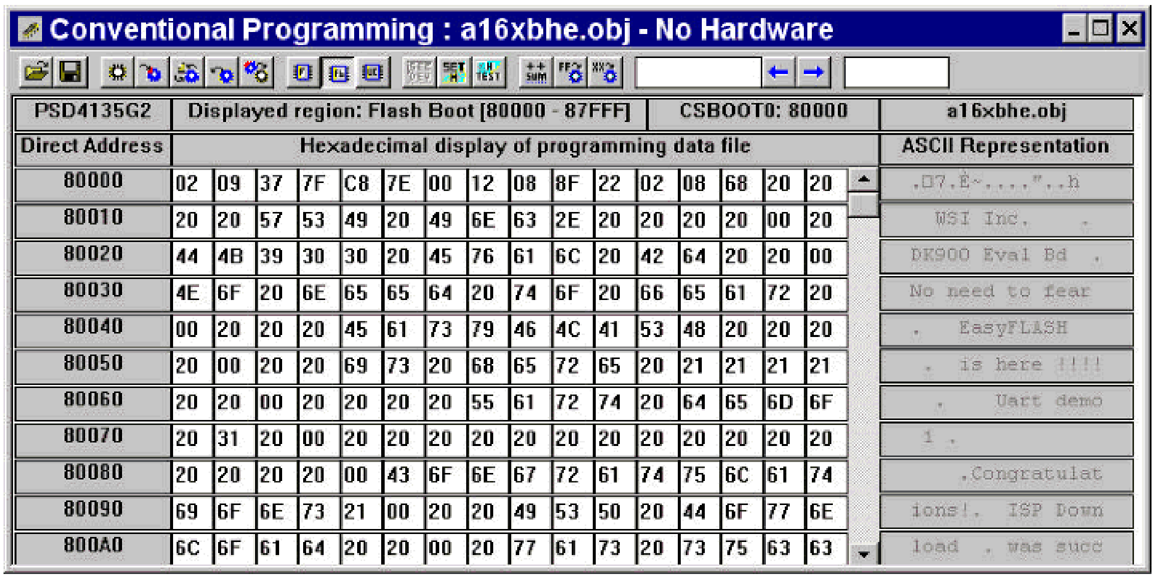
1. Check to see if your error is among the ones listed in the PSD *JTAG FAQ* on the website
2. If your error message is not listed in the FAQ or you have additional questions, contact *apps.psd@st.com*.

Hints and Tips

Batch JTAG Download Program

You may be thinking that this software is great to get your system up-and-running, but what if you have many boards to program and would like to run a batch program to do this. There is a program called FLINK that was designed for just this purpose and is available for free as a download from the Software Center on the PSM website.

PSD CONVENTIONAL PROGRAMMERS



Conventional Programming : a16xbhe.obj - No Hardware

PSD4135G2	Displayed region: Flash Boot [80000 - 87FFF]										CSBOOT0: 80000						a16xbhe.obj
Direct Address	Hexadecimal display of programming data file																ASCII Representation
80000	02	09	37	7F	C8	7E	00	12	08	8F	22	02	08	68	20	20	.□7.È~....."..h
80010	20	20	57	53	49	20	49	6E	63	2E	20	20	20	20	00	20	WSI Inc. .
80020	44	4B	39	30	30	20	45	76	61	6C	20	42	64	20	20	00	DK900 Eval Bd .
80030	4E	6F	20	6E	65	65	64	20	74	6F	20	66	65	61	72	20	No need to fear
80040	00	20	20	20	45	61	73	79	46	4C	41	53	48	20	20	20	. EasyFLASH
80050	20	00	20	20	69	73	20	68	65	72	65	20	21	21	21	21	. is here !!!!
80060	20	20	00	20	20	20	20	55	61	72	74	20	64	65	6D	6F	. Uart demo
80070	20	31	20	00	20	20	20	20	20	20	20	20	20	20	20	20	1 .
80080	20	20	20	20	00	43	6F	6E	67	72	61	74	75	6C	61	74	.Congratulat
80090	69	6F	6E	73	21	00	20	20	49	53	50	20	44	6F	77	6E	ions!. ISP Down
800A0	6C	6F	61	64	20	20	00	20	77	61	73	20	73	75	63	63	load . was succ

The first thing to note about this window when it opens is that if you had a project open and have created a valid programming data file (.obj) in that project, that programming data file will automatically be loaded and displayed. Looking at the screen capture above, we see that the project's name was "a16xbhe" and we are viewing its .obj file.

The Tool Bar

Next, note the available toolbar icons, which perform the following functions (from left to right). Let the mouse “hover” over an icon to display its function.

- Open file—click in case you did not have a project with a valid .obj file open or you wish to view a different .obj file.
- Save file—allows you to save any changes you may have made to the .obj file itself. This implies that the .obj file can be modified directly in this window, which it can. To do so, simply place the cursor at the desired location and type in the new value (you’ll see that the old value is overwritten automatically). Note that changing any given byte within the .obj file will also change the checksum (see below) for that non-volatile memory. As a general rule, it is not recommended to modify the .obj file directly. Any changes should be made at the source level and a new record file (*firmware*) should be generated using your compiler/linker environment.
- Blank Test—check to see if any part of the PSD has been programmed.
- Program—write code or data to one or more parts of the PSD based on your .obj file.
- Verify—compare the actual contents of the .obj file programmed within the PSD to the contents of the .obj file that is currently loaded.
- Upload—dump the contents of the PSD to an .obj file. Note: upload will produce garbage if the *security bit* had been previously set and the only way to erase the security bit is by performing a full-chip erase.
- Erase (Flash memory/EEPROM only)—erase one or more parts of the PSD.
- The next two or three buttons allow you to view the contents of the .obj file for the various non-volatile memories within the PSD. Note that “F” stands for main Flash memory, “Fb” for secondary Flash memory, “UC” for user code, “E” for EPROM, and “EE” for EEPROM.
- Select Device—allows you to select which PSD device is to be used with your new .obj file. This can only be used if a new .obj file is opened (from the “File” menu). This choice will be grayed out once the new file has been saved.
- Hardware Setup—allows selection of programming hardware and parallel port to be used.
- Hardware Test—checks your hardware to ensure that it’s connected and functioning properly. Run this test the first time you plug in new hardware and every time that you switch devices on that parallel port.
- Checksum—compute the checksum for the various non-volatile memories within the PSD based on the currently loaded .obj file. Note that PSDsoft uses a simple binary adding technique. It is therefore possible to get the same checksum value for two .obj files that differ only slightly, but the greater the number of variations, the more unlikely the chances are of this happening.
- Fill blank—the default “un-programmed” value of the non-volatile memories within the PSD is logic-high or “1”. Clicking this button fills all the non-volatile memories within the PSD to “1s” and is thus the equivalent of erasing those memories.
- Fill value—it may be desirable to fill the non-volatile memories within the PSD with a value or values that could be used for debugging, for example. When you click this button, a window pops-up allowing you to select which memories to fill, the start and stop addresses, and the fill value.
- The final two areas on the toolbar can be used for pattern searches. The left arrow is used to search for previous occurrences and the right for next occurrences.

The Display

The next row down from the toolbar displays the following information (from left to right):

- PSD part number associated with the .obj file open.
- The non-volatile memory within the PSD displayed based on your selection in the toolbar.
- The segment within the selected non-volatile memory that is currently being displayed
- The name of the .obj file that is currently open.

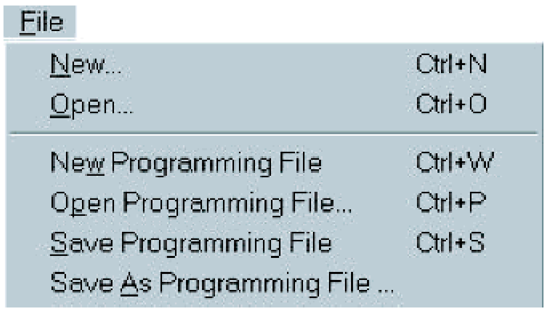
The columns below that display the following information (from left to right):

- The addresses shown in the Direct Address column are the internal PSD addresses that are associated with the non-volatile memory currently being displayed. To see how these direct addresses translate into the system addresses that are output by your MCU/DSP, go to **Report->Address Translation**.
- Hexadecimal display of programming data file shows your firmware file byte-for-byte.
- The last column shows the ASCII representation of the bytes displayed in the previous column.

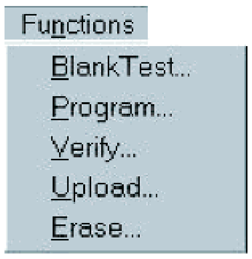
The Menus

Note that there are three new menus available when the “Conventional Programming” window is the active window. Let’s see what these additional menu items offer:

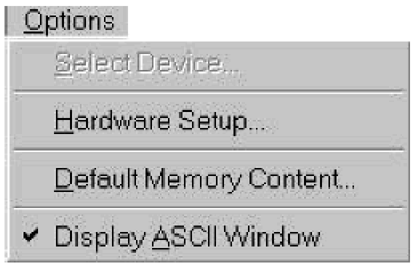
The “File” menu is shown below and its functions allow you to create, open, and save an programming data file (.obj).


File
New... Ctrl+N
Open... Ctrl+O
New Programming File Ctrl+W
Open Programming File... Ctrl+P
Save Programming File Ctrl+S
Save As Programming File ...

The functions in the “Functions” menu are exactly the same as those described above:



The “Options” menu is shown below:

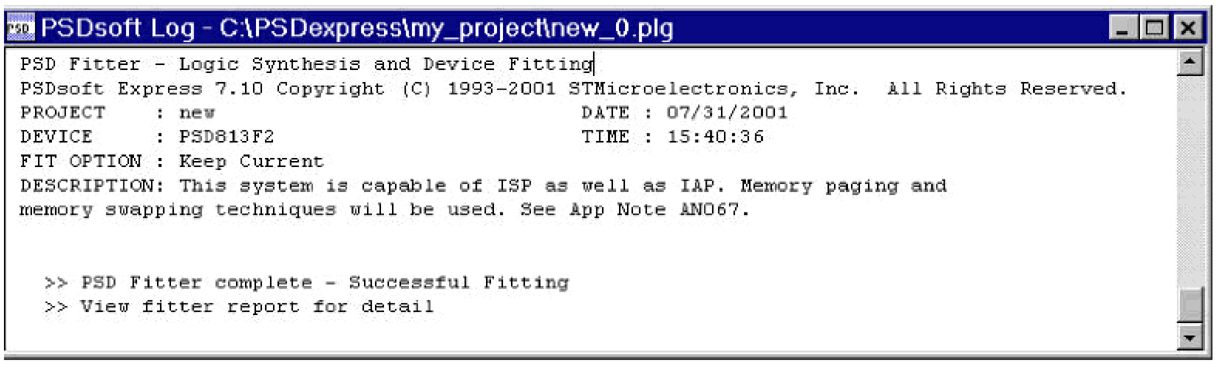


“Select Device…” and “Hardware Setup…” are described above. The “Default Memory Content…” brings up a window that allows you to fill the entire non-volatile memories with the same hexadecimal value. “Display ASCII Window” allows you to turn off and on the ASCII representation of your firmware file.

THE PSDSOFT LOG FILE

The log file is a record kept by PSDsoft Express that shows project information, such as errors and warnings during the design process. You can specify the number of log files to be kept on your system (up to 10) by setting this value in the **Project->Preference** menu. The file format ProjectName_X, where X is an integer between 0 and the number specified (minus 1) in the Preferences. When the maximum number of log files has been written, X will be reset to 0.

As stated above, the log file can be kept in view by going to the “Report” menu and ensuring there is a checkmark next to “View Log file”. You may check this file at any time in the design process to see the status of your design and watch for messages that may appear. If you are getting errors while using PSDsoft Express, it may be helpful to send your log file to *apps.psd@st.com* with an explanation of your problem, along with your design files.



```
PSD Fitter - Logic Synthesis and Device Fitting
PSDsoft Express 7.10 Copyright (C) 1993-2001 STMicroelectronics, Inc.  All Rights Reserved.
PROJECT     : new                        DATE : 07/31/2001
DEVICE      : PSD813F2                   TIME : 15:40:36
FIT OPTION  : Keep Current
DESCRIPTION: This system is capable of ISP as well as IAP. Memory paging and
memory swapping techniques will be used. See App Note AN067.


  >> PSD Fitter complete - Successful Fitting
  >> View fitter report for detail
```

WHAT’S THE DIFFERENCE BETWEEN CHIP SELECT, I/O LOGIC, AND NODE EQUATIONS?

If, at any point, you are unsure just exactly what type of equation you should be using for an I/O or internal node, use the following guidelines to aid you in your selection:

- Chip-select equations will generally decode some address lines and optional control logic or other PLD input. Examples include internal segment selects for memory segments within the PSD and any non-registered outputs that don’t require an output enable signal.
- User-defined nodes are normally used for temporary storage or refer to a specific Input or Output Macrocell within the PSD.
- I/O Logic will be any output that is either registered or requires an output enable signal.

For example, suppose you had the following requirements for your design:

- 8-bit loadable shift register where only one bit will be shifted out.
- A chip-select signal for an LCD module
- An I/O pin accessed directly through the MCU/DSP

To implement this design, seven bits of the 8-bit shift register would be defined as registered internal nodes and equations for these nodes would be written in the “User defined Node Equations” part of the Design Assistant. The bit to be shifted out should be defined in the “*Pin Definitions*” screen as a Registered CPLD output and the equation for it would be written in the *I/O Logic Equations* section of the Design Assistant. The shift register is made loadable by associating (and aligning) the nodes to specific output macrocells within the PSD. Then, the shift register can be directly accessed by the MCU/DSP via the data bus using the PSD’s *CSIOP* registers. The LCD chip select signal must first be defined as an “External chip-select – Active Hi/Lo” within the Pin Definitions screen, then the equation for it can be written using the *Chip Select Equations* section of the Design Assistant. Lastly, the I/O pin directly accessible by the MCU/DSP is simply defined as an MCU I/O mode pin in the Pin Definitions screen and the rest is done at runtime.

Since this is all probably very confusing, luckily there are other resources to view for more specific help, including:

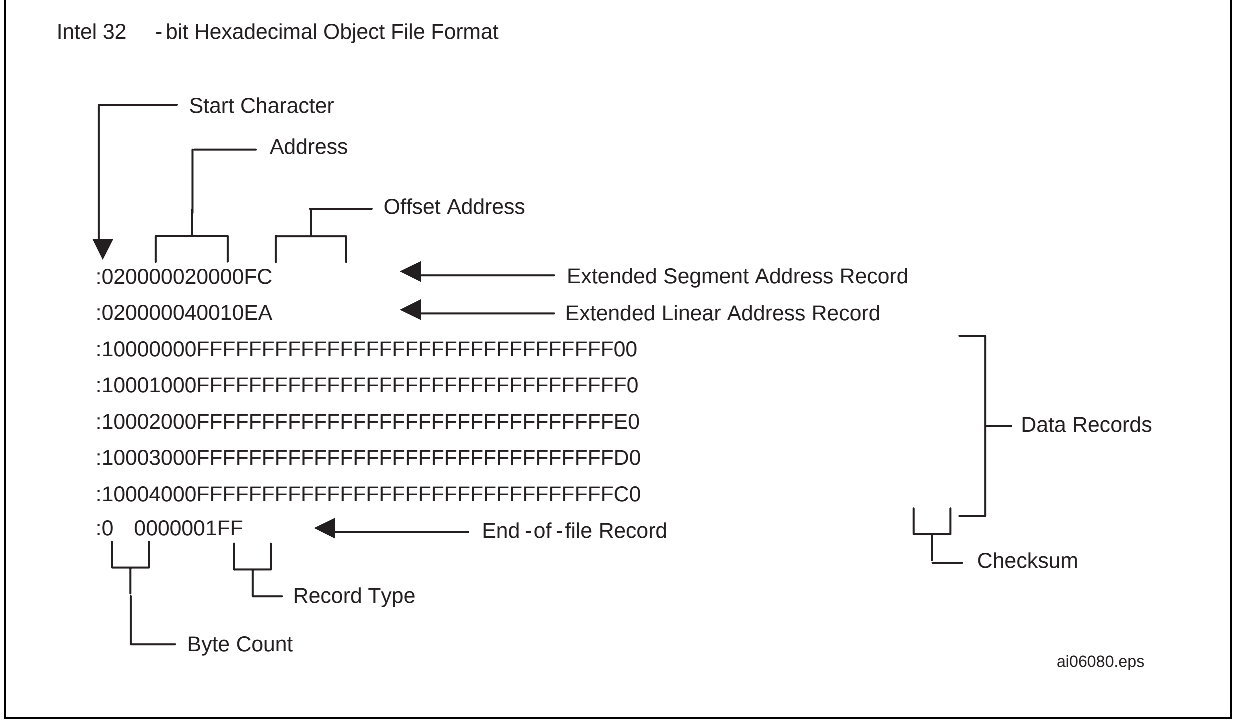
- The “Flash PSD CPLD Primer” application note explains the *PLD* section of the PSD and it’s many uses in detail. It covers what needs to be done in the design and what needs to happen at run-time
- The “Design Guide for PSDsoft 2000 and PSD4235 application note offers detailed examples of a PSD4235G2 part designed with PSDsoft Express. It is a good resource for any *CPLD-based* PSD part.
- Other PSD and MCU/DSP-specific application notes are available on the PSM website.

APPENDIX

Appendix A—Intel Hex-32 Record Format

The Intel 32-bit hexadecimal object file record format has a 9-character (4-field) prefix that defines the start of the record, byte count, load address, record type, and a 2-character checksum suffix. The .hex file below illustrates the sample records of this format:

Figure 2. Intel 32-bit Hexadecimal Object File Format



```
:020000020000FC
:020000040010EA
:10000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF00
:10001000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF0
:10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
:10004000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC0
:0   0000001FF
```

The four record types are as follows:

- **00—Data Record:** This record begins with the colon start character, which is followed by the byte count (in hexadecimal notation), the address of the first data byte, and the record type (equal to “00”). Following these are the data bytes. The checksum follows the data bytes and is the two’s complement (in binary) of the preceding bytes in the record, including the byte count, address, record type, and data bytes.

- **01—End Record:** This end-of-file record also begins with the colon start character and is followed by the byte count (equal to “00”), the address (equal to “0000”), the record type (equal to “01”), and the checksum, “FF”.
- **02—Extended Segment Address Record:** This is added to the offset to determine the absolute destination address. The address field for this record must contain ASCII zeros (hexadecimal 30s). This record type defines bits 4 to 19 of the segment base address; it can appear randomly anywhere within the object file and affects the absolute memory address of subsequent data records in the file. The following example illustrates how the extended segment address is used to determine a byte address.

Problem:

Find the address for the first data byte for the following file:

```
:02 0000 04 0010 EA
:02 0000 02 1230 BA
:10 0045 00 55AA FF ..... BC
```

Solution:

Step l Find the extended linear address offset for the data record (0010 in the example).

Step 2 Find the extended segment address offset for the data record (1230 in the example).

Step 3 Find the address offset for the data from the data record (0045 in the example).

Step 4 Calculate the absolute address for the first byte of the data record as follows:

```
00100000  Linear address offset, shifted left 16 bits
+  12300  Segment address offset, shifted left 4 bits
+   0045  Address offset from data record
________
00112345  32-bit address for first data byte
```

The address for the first data byte is 112345.

Note: always specify the address offset when using this format, even when the offset is zero.

During output translation, the firmware will force the record size to 16 (decimal) if the record size is specified greater than 16. There is no such limitation for record sizes specified less than 16.

- **04—Extended Linear Address Record:** This record specifies bits 16–31 of the destination address for the data records that follow. It is added to the offset to determine the absolute destination address, and can appear randomly anywhere within the object file. The address field for this record must contain ASCII zeros (hexadecimal 30s).

Appendix B—Definitions

CPLD versus Non-CPLD Parts. Throughout the manual, there are many references to CPLD-based PSDs and non-CPLD-based PSDs. The difference is simple: CPLD-based PSDs have registered logic and non-CPLD PSDs only have combinatorial logic. What this means is that CPLD-based PSDs can implement simple registered functions, such as counters, shift registers, and state machines. Basically, the (Z)PSD211R, (Z)PSD3XX, PSD9XX, and PSD41XX are all non-CPLD parts and all other PSD parts have a CPLD.

Multiplexed versus Non-Multiplexed Bus Definition. Microcontrollers (MCUs) and Digital Signal Processors (DSPs) have two types of address and data busses: multiplexed and non-multiplexed. The difference is that with a multiplexed bus, the address is output from the MCU/DSP and the data is read or written using the same physical bus, but at different times (time multiplexed). When the address is output and the data read/written on separate busses, the MCU/DSP is referred to as non-multiplexed. The tradeoff is multiplexed busses require less pins, but non-multiplexed busses are usually a bit faster.

What is Firmware? Firmware is code that has been translated from the source code in which it was written (C, C++, Assembly) into an executable format (machine code) that will be understood by your Microcontroller (MCU)/Digital Signal Processor (DSP). It is called firmware because it is generally stored in a non-volatile memory (EPROM, EEPROM, Flash memory, or battery-backed SRAM). Thus if you intend to boot from the PSD, you would need to put your reset vector, interrupt vectors, and boot code in the firmware. Also, if you intended to be able to update the PSD at runtime (requires a PSD with two non-volatile memories) through a serial port or other medium, you will need to have the code that handles the update stored in one of the memories. See the various application notes on the PSD website: www.st.com/psd for more information.

What is Paging? Paging is a method of expanding the physical address space of a Digital Signal Processor (DSP) or Microcontroller (MCU). MCUs/DSPs have a fixed number of address bits that make up their address bus. For example, most 8-bit MCUs/DSPs have only 16-bits of address for external devices. This means that they can access up to 2^{16} or 64 kilobytes (KBytes) of address space. This is limited by today’s standards, but paging gives you a way to overcome this limitation. Paging allows creation of a virtual memory space that goes beyond the limit of the physical address range. For example, if you had an 8-bit page register, you could potentially have up to 2^8 pages or 256. The number of pages becomes the multiplier to the physical address space size. So, if you have 16 address bits and 8 page bits, your virtual memory space is 2^{16} times 2^8 or 16 Megabytes (MBytes).

Do I need Paging?

To answer this question, you must determine how many address bits your MCU/DSP outputs and see if the size of the physical memory space of the MCU/DSP is greater than or equal to the amount of all the memory external to the MCU/DSP combined. For example, if you aren't using an MCU/DSP that outputs 16 address bits and you have two external memories: a 128 K x 8 Flash memory device and a 32 K x 8 SRAM. You will need paging in this case. Three pages will be required (leaving 32 Kbytes of free space on one of the pages). Since pages only come in factors of two, two page bits will be required ($2^2 = 4$). Beware that some MCUs/DSPs have internal memory that sometimes uses up some of the physical address space, so this must be taken into consideration as well when determining if you need paging.

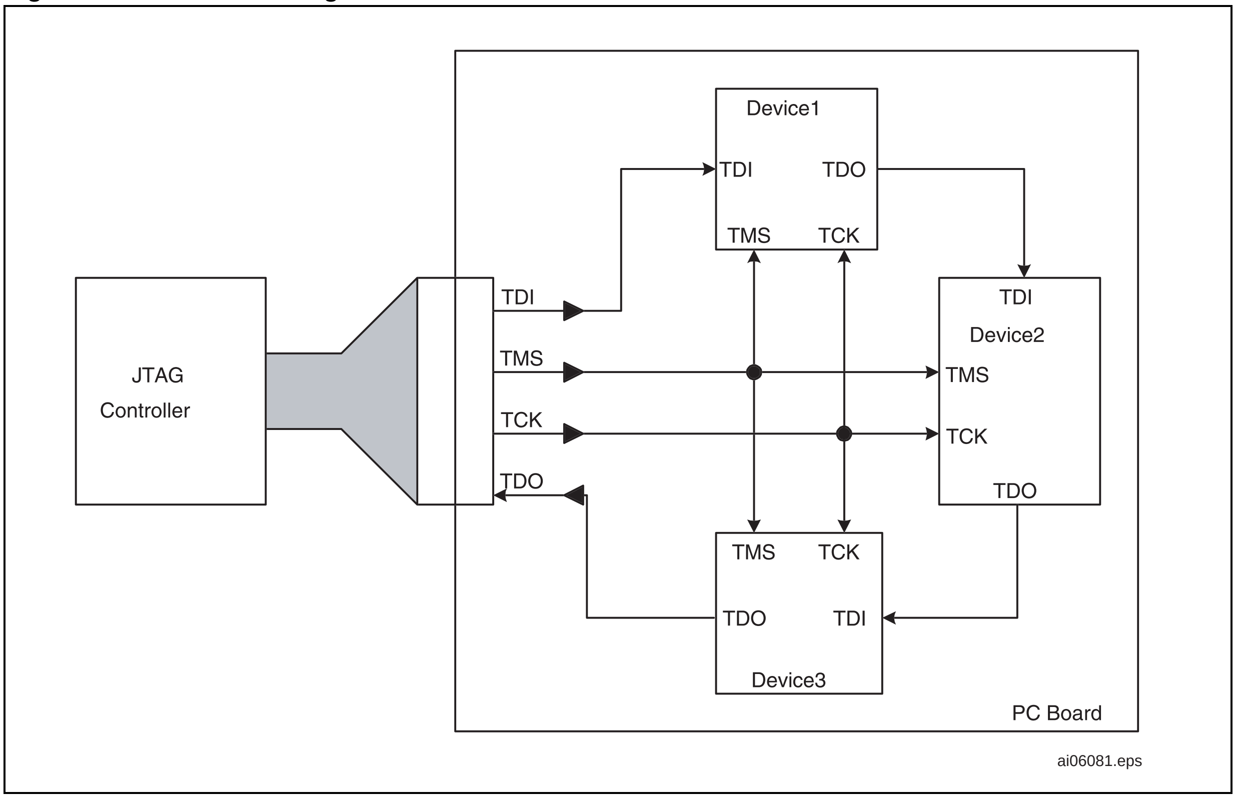
What is IAP? Our industry uses the term In-System Programming (ISP) in a general sense. ISP is applicable to programmable logic, as well as programmable Non-Volatile Memory (NVM). However, an additional term is used in this manual: In-Application Programming (IAP). There are subtle yet significant differences between ISP and IAP when microcontrollers are involved. ISP of memory means that the MCU/DSP is off-line and not involved while memory is being programmed. For IAP, the MCU/DSP participates in programming the memory, which is important for systems that must be online while updating firmware. Often, ISP is well suited for manufacturing, while IAP is appropriate for field updates. Flash PSD devices are capable of both ISP and IAP. Keep in mind that IAP can only program the memory sections of the PSD and not the configuration and programmable logic portions. With ISP, the entire PSD can be erased or programmed.

What is CSIOP? CSIOP (Chip-Select I/O Port) is a collection of internal registers within the PSD. You must allocate an amount of address space indicated in the datasheets to CSIOP. Then, to access one of the registers within CSIOP space, simply add the offset listed in the datasheet to the base address that you defined in the “Chip Select Equations” step.

Appendix C—JTAG Chain Definition and Rules

A JTAG chain is defined to be two or more JTAG compliant (according to IEEE 1149.1 standard) devices connected together in a chain, as shown in the diagram below.

Figure 3. JTAG Chain Diagram



Each device in the chain must support the four basic JTAG signals: TDI, TDO, TCK, and TMS. Note that some devices are In-System Programming (ISP) via means other than a standard JTAG port (for example Boundary Scan). These devices are **not** IEEE 1149.1 compliant devices and will not work in a JTAG chain.

The following are general JTAG chaining rules:

- Only one device in a chain can be programmed at a time and the other devices must be placed in “Bypass Mode”.
- Different brands of chips will require different programming software.
- Before any JTAG operation can begin the chain order must be defined and the instruction register length for each device must be known by the software.

Note that the order of the chain is important when programming the devices. You must set up the chain in your programming software such that it matches physical ordering of the devices on your board. For the diagram above, Device1 is the first device in the chain, Device2 is the second, and Device3 is the last device.

For more information on JTAG and JTAG chaining specific to the PSD, see The “JTAG-ISP Information for Flash PSDs” application note.

Table 1. Document Revision History

Date	Rev.	Description of Revision
	1.0	Initial revision based on PSDsoft 2000 version 6.02
	2.0	Revised for PSDsoft Express version 7.0. Added Installation guide.
Oct-01	2.1	Revised for PSDsoft Express version 7.3. See the software Revision History on the PSD website in the Software Center.

For current information on PSD products, please consult our pages on the world wide web:

www.st.com/psd

If you have any questions or suggestions concerning the matters raised in this document, please send them to the following electronic mail addresses:

apps.psd@st.com	(for application support)
ask.memory@st.com	(for general enquiries)

Please remember to include your name, company, location, telephone number and fax number.