AN1423
APPLICATION NOTE

Introducing the FLASH+PSD PSD9XX Family

CONTENTS

The New Flash PSD3XX
Introducing the FLASH PSD9XX Family

By Dan Harris

Contents

1 Introduction

Does a Flash PSD3XX with the following features sound appealing?

- ✓ Fully In-System Programmable (ISP) and In-Application Programmable(IAP)—allows faster time-to-market by reducing design cycle times. With an ISP/IAP part, a socket is no longer required.
- ✓ Flash-based, eliminating the need for ROMed MCU or a programmer for boot code.
- ✓ No need to throw away anymore OTP devices

A few years back, Waferscale recognized the move to Flash in the embedded control marketplace. Waferscale surveyed the customer base and saw a great need for a Flash PSD3XX type device. As various users were interviewed, knowledge was gained about the mistakes made by other Flash vendors, and the final definition of the Flash PSD became a superset of the current PSD3XX family. Many expressed interest in being able to program an entire blank device, including the memory, right on the board, without MCU intervention. This feature eliminates the need for a socket and prevents OTP part waste during the development and upgrade phases. Also, users expressed the need to be able to program one area of a device while executing code out of another area with low MCU overhead. The Flash PSDs address both these needs by providing a JTAG port for ISP and dual Non-Volatile Memories (NVMs) for IAP.

JTAG-ISP was the evolving standard and Waferscale decided to add this feature to make Flash PSDs the World's first JTAG programmable PLD+Memory device. The extra pins required to address this need plus the numerous requests for a few more I/O resulted in a higher pin count package. Taking advantage of our .35 micron technology, it was decided to also add a few other features normally reserved for higher priced devices, including increased PLD capability.

The resulting product, which is exceeding the activity of all previous OTP products combined, (including the PSD3XX family) is called the FLASH PSD9XX Family. After you read this document, review the datasheet, and take a look at the development environment, you will understand why we added these enhancements and why we place the word *Easy* in the product's name.

Additional Features include:

- ✓ Increased memory capacities.
- ✓ Optional secondary Flash or EEPROM memory array, allowing concurrent memory operations, such as executing from one memory while updating the other.
- ✓ Enhanced PLD with more product terms and capabilities.
- ✓ Battery backup capability for the SRAM.
- ✓ More I/O
- ✓ Automatic C Code generation for the NVMs within the PSD
- ✓ Support of additional microcontrollers (Philips 80C51XA, Intel 80C251, plus others).

The following table compares the major features and functions of the PSD3XX family and the PSD9XX family:

Function/Feature	PSD3XX	PSD9XX
Main Memory	256 Kbits to 1 Mbit EPROM	1 – 4 Mbit Flash
Optional Secondary Memory	None	256 Kbits EEPROM or 256 Kbits Flash
SRAM	None or 16 Kbits	16 or 64 Kbits
# Product Terms	40	Up to 182
# PLD inputs	14 or 18	Up to 73
# Pins	44	52 to 80
Page Register Size	0 or 4 bits	8 bits
# I/O pins	19	27 to 52
In-System Programmable?	No	Yes
In-Application Programmable?	Not directly	Yes
Battery Backup capability for SRAM?	No	Yes
Programmable polarity for Reset Input?	Yes on 5 Volt PSD	No
Write Protection for memory sectors	Not applicable	Yes

This document fully explains all the features and benefits of upgrading from a PSD3XX to a PSD9XX. Appendix A provides a brief functional explanation of Flash memory.

2 Why Switch to a PSD9XX?

You may be asking yourself “Why should I switch from my PSD3XX to the PSD9XX family?” There are many possible answers, including:

- You want concurrent non-volatile memories to gain the benefit of IAP
- You want a Flash-based product that is In-System Programmable (ISP) via an IEEE 1149.1 compatible JTAG port
- You want the ability to perform field updates
- You don’t want to throw away any more OTP parts
- You need a more capable PLD
- You need more I/O
- You want battery-backup capability for your SRAM.

3 Benefits of a Flash-based product

The following sections explain what Flash is and why a Flash-based product is better than an EPROM-based or an OTP-ROM-based product.

3.1 What is Flash?

Flash is an electronically erasable and electronically programmable nonvolatile memory. You can program and re-program Flash memory without ever removing it from the circuit board. Flash memory can be programmed via any communication channel supported by your microcontroller (MCU). (Examples include UART, CAN, SPI, and I^2C.) Note: the Flash PSD can also be programmed via the JTAG port with no MCU assistance.

3.2 Why Flash is Better than EPROM or OTP-ROM

Flash memory brings you the following advantages over EPROM and OTP-ROM:

- Flash is electronically erasable and programmable.
- Flash brings In-System Programmability (ISP). See the next section.
- Flash provides rugged data recording.
- Flash erases much quicker than EPROM. OTP-ROM cannot be erased.

These benefits translate into a product that will give you more features-per-dollar and reduce your time-to-market. Since Flash contents are changed electronically, you can program the PSD9XX while in-system.

4 Flash brings In-System Programmability (ISP)

The FLASH PSD9XX family gives you the advantage of In-System Programmability (ISP). Having ISP capability means that you can solder a blank PSD9XX part directly to the circuit board and never touch it again. The following sections explain ISP and discuss the features and benefits that ISP delivers.

4.1 What is In System Programming (ISP)?

If you are not familiar with ISP, you are missing a whole new world. Using a PSD9XX, you have the ability to erase and program the entire chip while it is on the circuit board! With ISP, you have the ability to perform rapid field updates, product customization, and quick prototyping.

All that you need is a ST FlashLINK™ cable attached to your PC on one end and a header on your circuit board on the other. For example, you can take a laptop computer loaded with your most recent design to a customer site, hook up the FlashLINK to your product, and update

anything and everything, all in a matter of seconds. It's just that *Easy*! No MCU firmware needs to be written and debugged for ISP. The FlashLINK does its job with no MCU assistance.

4.2 Features and Benefits of ISP

With ISP, you can program and reprogram a chip as many times as you want[1].

ISP also brings the advantage of being able to perform field updates. For example, imagine the following situation: your company, Simpson Power and Light, makes products that control nuclear reactors. Due to changing government regulations, the control algorithm needs to be updated frequently. With a FLASH PSD9XX product in the design, you no longer need to retrieve the product, decontaminate it, open it, pull the chip off the board (or de-solder it), reprogram the product (assuming non-OTP product), and reverse the process to put the product back in place. Instead, simply connect a FlashLINK cable to a panel in the control room and update the PSD9XX using PSDsoft Express on your laptop computer.

5 Other Features and Benefits of the PSD9XX

The FLASH PSD9XX brings you other features and benefits, including:

- Optional secondary Flash or EEPROM (Concurrent Memory/Boot Code/IAP)
- More I/O
- Larger, more capable PLD
- Optional SRAM with battery backup capability
- Automatic C Code generation using PSDsoft Express software.

5.1 Optional Secondary Flash or EEPROM

The main Flash memory is available in one-, two-, or four-megabit capacities. The PSD9XX1 provides a secondary EEPROM, while all other versions provide a secondary Flash. The size of the secondary EEPROM or Flash is 256 kilobits.

Concurrent Memory and IAP

When the MCU participates in ISP (no JTAG), the optional secondary memory brings a unique advantage. That advantage is the ability to execute from one of the memories while simultaneously updating the other. This process is now commonly called In-Application Programming (IAP). A solution without IAP requires tremendous overhead by the MCU. The PSD9XX silicon architecture implements dynamic memory swapping and the ability to move memory between "program space" and "data space" on-the-fly.

[1] The Flash portion of the PSD9XX can be programmed up to 100,000 times.

5.2 More I/O

With the PSD9XX device have up to 52 individually configurable I/O pins versus the 19 available in the PSD3XX parts. These pins can be used for the following:

- MCU I/Os
- Extra address inputs
- PLD I/Os
- Latched MCU address outputs
- Special function I/Os.

Note: sixteen of the I/O ports can be configured as open-drain outputs.

5.3 Larger, More Capable PLD

The PSD9XX provides two PLD sections, the Decode PLD (DPLD) and General-purpose PLD (GPLD).

Both PLDs decode addresses and other signals. The DPLD can be used to generate internal chip-select signals and the GPLD can be used to generate external chip-select signals and other PLD outputs.

5.4 Optional SRAM with Battery-Backup Capability

The PSD9XX offers, as an option, 16 or 64 Kbits of SRAM. This SRAM will be automatically backed up while a battery is connected, protecting the SRAM contents in the event of a power failure. You can also configure one of the port pins to indicate when power has switched over to the battery.

5.5 Automatic C Code Generation

PSDsoft makes available low-level drivers for memory programming algorithms, and automatically generates the C code for you. Simply copy the C functions into your source code, edit the parameters to fit your design, and compile.

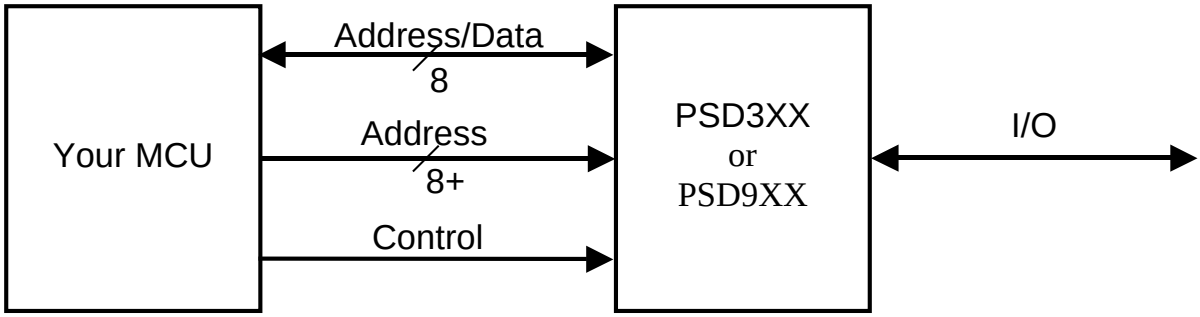
6 Steps Required to Upgrade

Upgrading to the FLASH PSD9XX is relatively simple. The number of steps required will depend on how much of the PSD9XX you plan on utilizing. The following sections will guide you step-by-step through the process required to upgrade to the PSD9XX. The first section covers the steps required for every upgrade. The sections that follow that cover other steps that may be required, depending on what parts of the PSD9XX you plan on using.

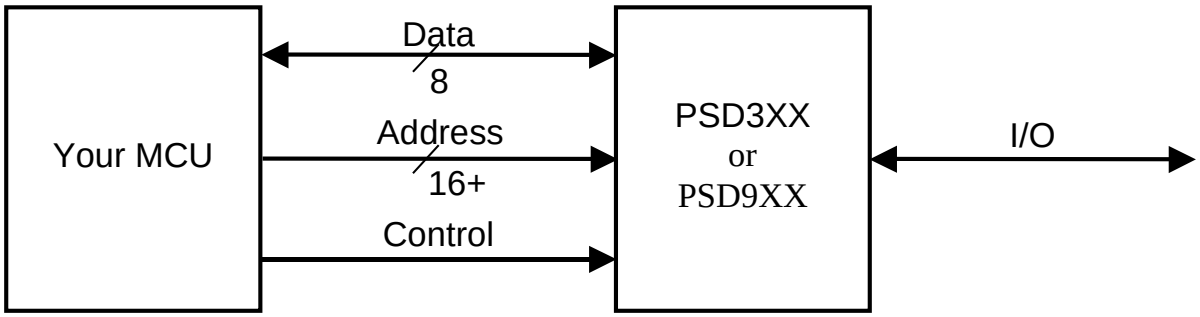
6.1 Steps Required for all Upgrades

For this discussion, pick one of the following two block diagrams that best matches your setup. Note: if you are using a 16-bit MCU, you should consider looking at the PSD4000 family instead. See ST's website for more information. The same basic principles apply to upgrade to the PSD4000 family.

Muxed Bus Setup



Non-muxed Bus Setup



6.1.1 Hardware Changes

Note that in either case (muxed/non-muxed bus), you can still use the same PSD ports when upgrading. However, the pin numbers where the signals are routed will change. Consult the PSD9XX Family Data Sheets for the new pin numbers. Note: no additional changes are required to support the JTAG port if it is **not** used.

6.1.2 Software Changes

Please make sure you are using the latest version of PSDsoft Express. It can be downloaded for free from ST's web site at www.st.com/com . PSDsoft Express greatly simplifies the process of designing and implementing a PSD in your design. For more details, you will want to refer to the *PSDsoft Express User Manual*, available on ST's website.

There are some changes within PSDsoft Express that you need to be aware of. If you had been using PSDsoft previously, some screens and places where information is entered is now different. Also, you no longer need to use or have knowledge of the ABEL-HDL language at all.

If you were using PSDsoft Express previously, the changes will be minor.

Note the following differences between the PSD9XX and the PSD3XX:

- The PSD9XX's PLD has greater address decode resolution. Therefore, you now connect [A0:A10] without sacrificing any inputs. With the PSD3XX, you could only connect up to 3 lower-order address bits to Port C.
- Port A of the PSD9XX can be used for PLD I/O.
- Port B of the PSD9XX is not limited to output only from the PLD. It can also be used for PLD input.
- Depending on your PSD9XX, Port C or Port E now has the JTAG-ISP capability.
- The page register in the PSD9XX is twice as large as the optional one within the PSD3XX.
- Any internal EPROM segment select signal (ESx) for the PSD3XX must be changed to a Flash segment select signal (FSx or CSBOOTx) for the PSD9XX. Note that there is only one product term available in the PSD3XX for the EPROM segment select signals. The PSD9XX provides up to 3 product terms per internal memory select signal.
- Some pin assignments are different.

See the next section for a detailed example of a PSD3XX to PSD9XX conversion.

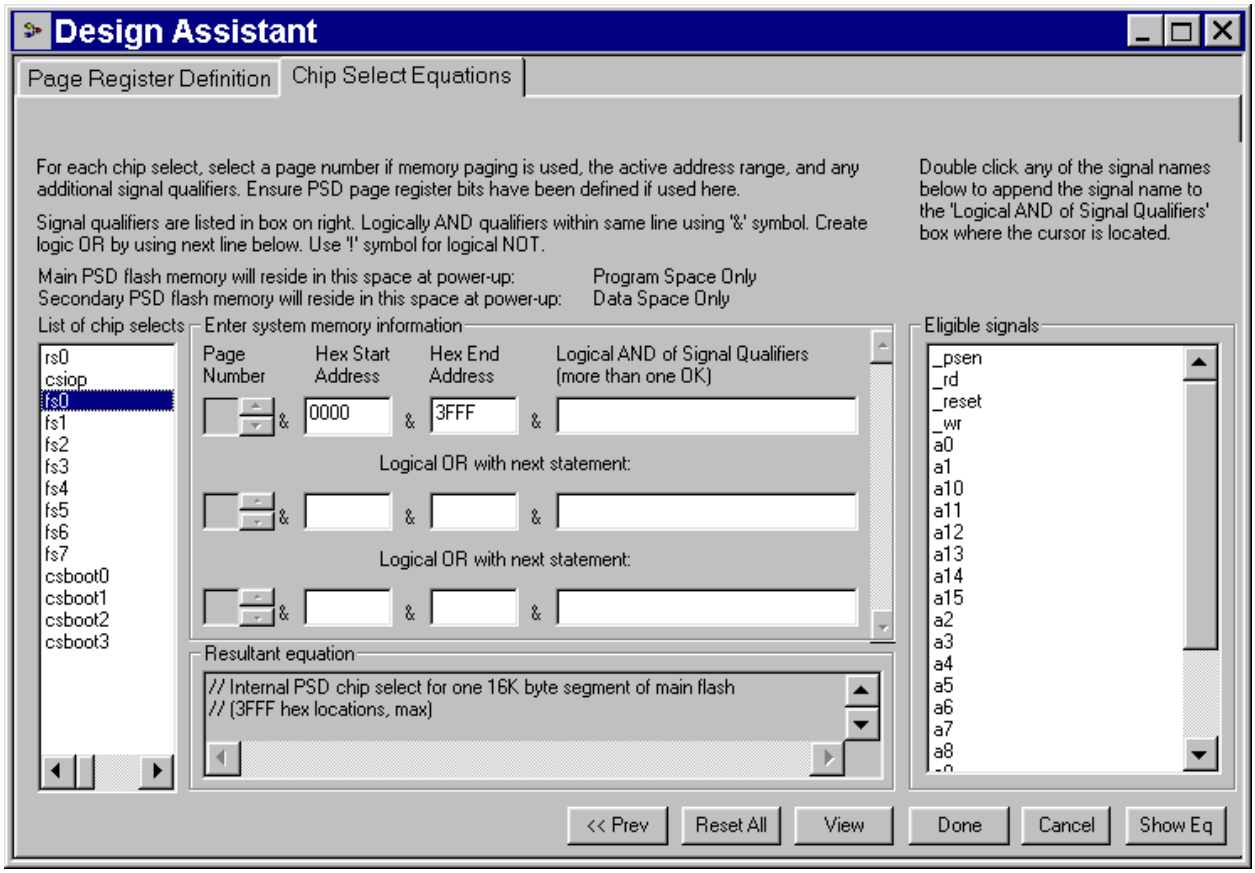
The following example shows sample internal chip select signals for the PSD3XX, and how they are implemented for the PSD9XX using PSDsoft Express:

Example Lines from a PSD3XX ABEL File

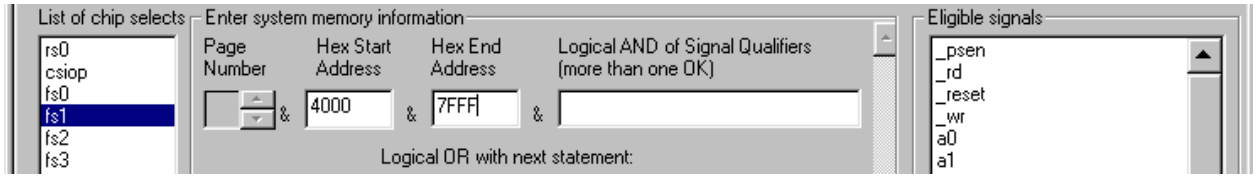
```
es0 = (Address >= ^h0000) & (Address <= ^h3FFF);
es1 = (Address >= ^h4000) & (Address <= ^h7FFF);
csiop = (Address >= ^hA000) & (Address <= ^hA7FF);
```

PSDsoft Express Implementation

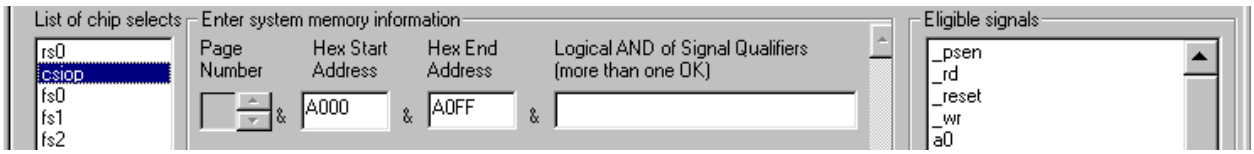
Here is a screen capture from PSDsoft Express showing how fs0 is defined:



Next, fs1 gets defined in a similar manner:



Lastly, we define csiop. Note the reduced address range due to the fact that a0 through a15 are all connected to the PLD now.



The next section shows a detailed conversion example.

7 Example Conversion: PSD313 to PSD913F2

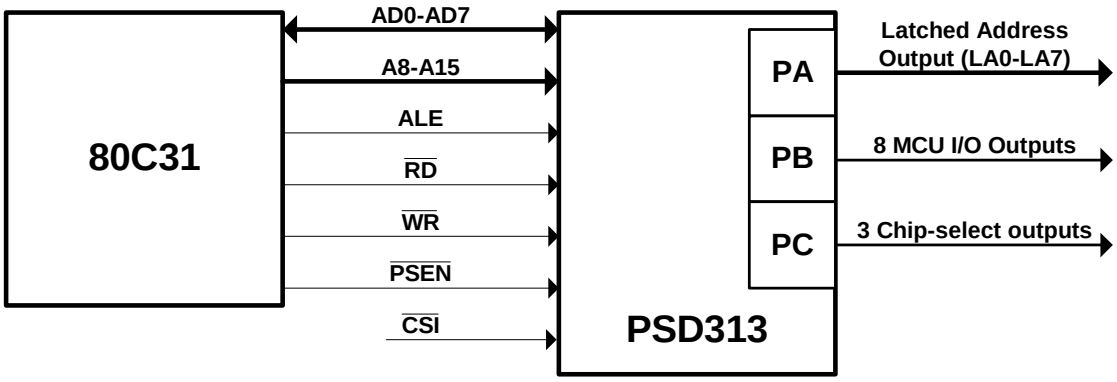
This section contains a specific example of the steps required to upgrade from a PSD313 to a PSD913F2. This section only shows the **required** steps to upgrade, and does **not** cover any of the additional functionality of the PSD9XX. However, the next section does build on this section.

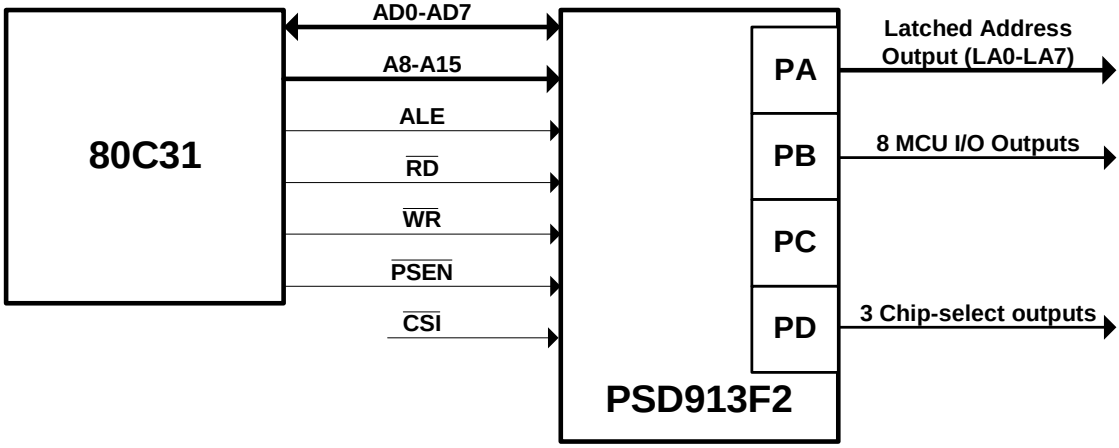
This section assumes you have been using PSDsoft and not PSDsoft Express for your original implementation. Had you used PSDsoft 2000 or PSDsoft Express to implement your PSD3XX design previously, the screens will be mostly the same with only minor changes to implement.

The following table shows the major differences between the PSD313 and the PSD913F2:

Function/Feature	PSD313	PSD913F2
Main Memory	1 Mbit EPROM	1 Mbit Flash
# Product Terms	40	182
# PLD inputs	18	73
# Pins (PLCC package)	44	52
Page Register Size	4 bits	8 bits
# I/O pins	19	27
In-System Programmable?	No	Yes
In-Application Programmable?	Not Directly	Yes
Battery Backup capability for SRAM?	No	Yes
Programmable polarity for Reset Input?	Yes	No
Memory write protection at the sector level	No	Yes

Let's compare two block diagrams: one shows a PSD313 connected to an 80C31 and the other shows an FLASH PSD913F2 connected to an 80C31. For the example, the exact same I/O is assumed.





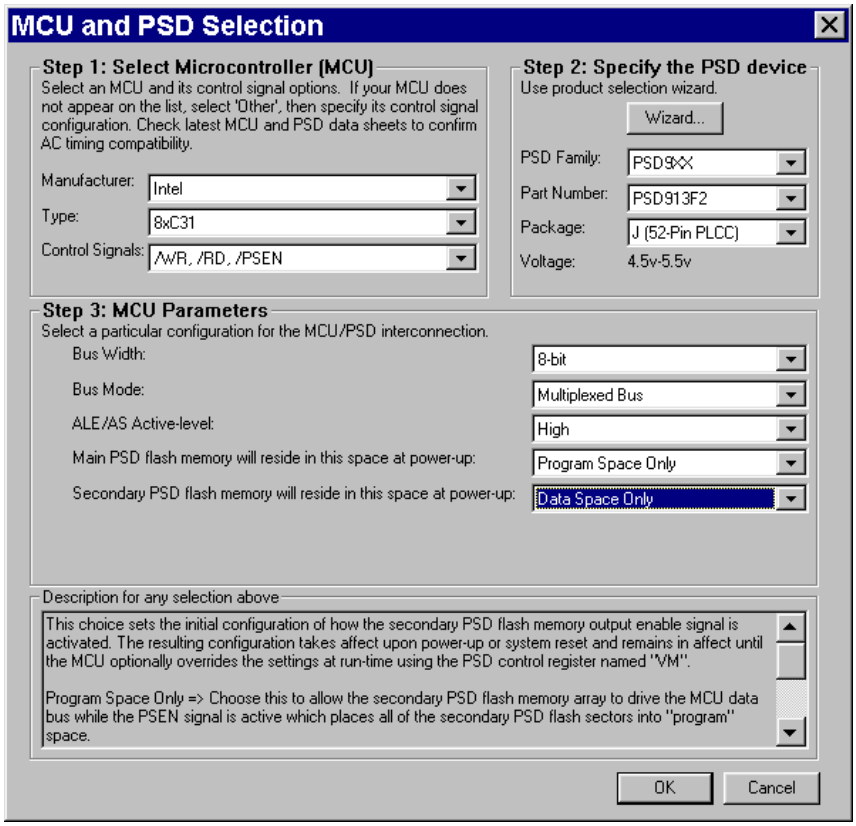
You can see that the block diagrams are virtually identical except the PSD913F2 has 8 more I/O than the PSD313. Check the next section to see what can be done with Port C.

Next we show the differences in the design environment for the programmable logic portion of the PSDs. As such, the following discussion only applies to the programmable logic within the PSDs. Let's look at the changes and additions that need to be made to the PSDsoft design file (.abl) when implementing the same functions using PSDsoft Express:

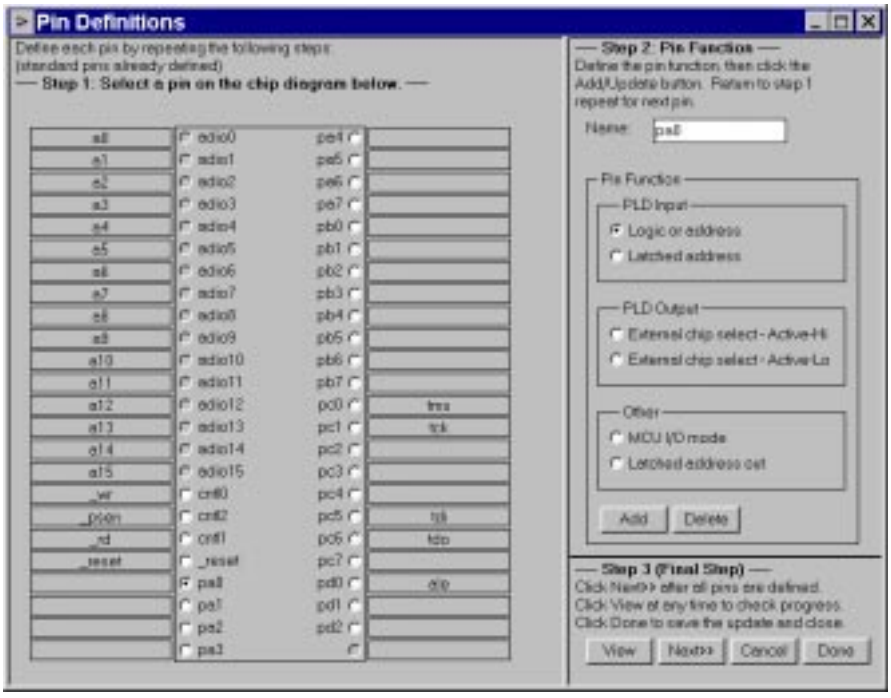
1) Before, the address pins were declared in the PSDsoft design file address inputs:

```
a15..a11  pin;
```

Now, simply select your MCU and PSD and the required address lines are automatically included in the design. See the screen captures below:

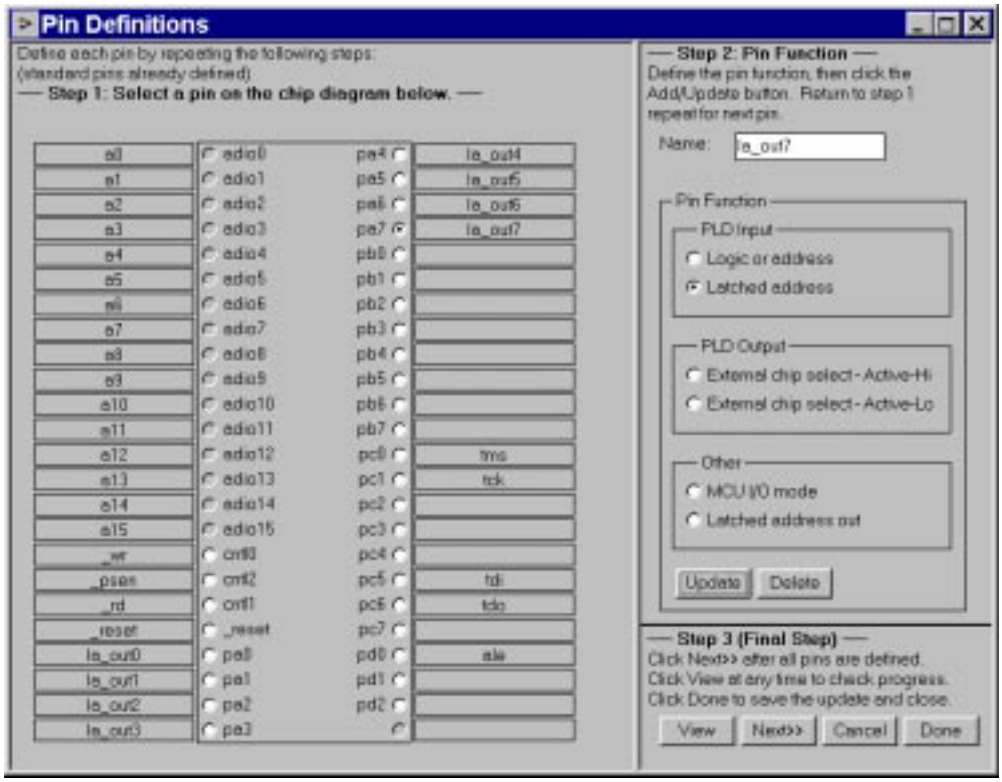


If you had made the above selections in the "MCU and PSD Selection" screen, you would see the following when you clicked on the **Define PSD Pin/Node Functions** button:



Note how the addresses a0 to a15 and some control signals are pre-defined for you.

2) We want Port A to output latched addresses. You now use the "Pin Definitions" screen to accomplish this:



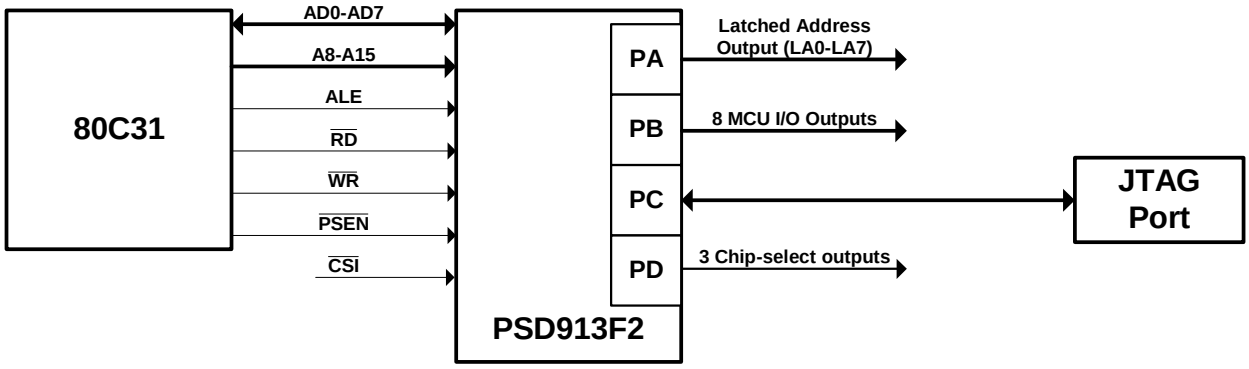
3) For the PSD3XX using PSDsoft, the EPROM segments (es0..es7) were declared and defined in the ABEL design file. With PSDsoft Express, the memory select signals are defined using the Design Assistant. For the PSD9XX, the Flash select signals are denoted fs0 to fs7. See the screen captures in Section 6.1.2.

4) If you wish, you may reduce the resolution of the CSIOP space to 256 bytes (from 2 Kbytes). Again, see the example in the previous section.

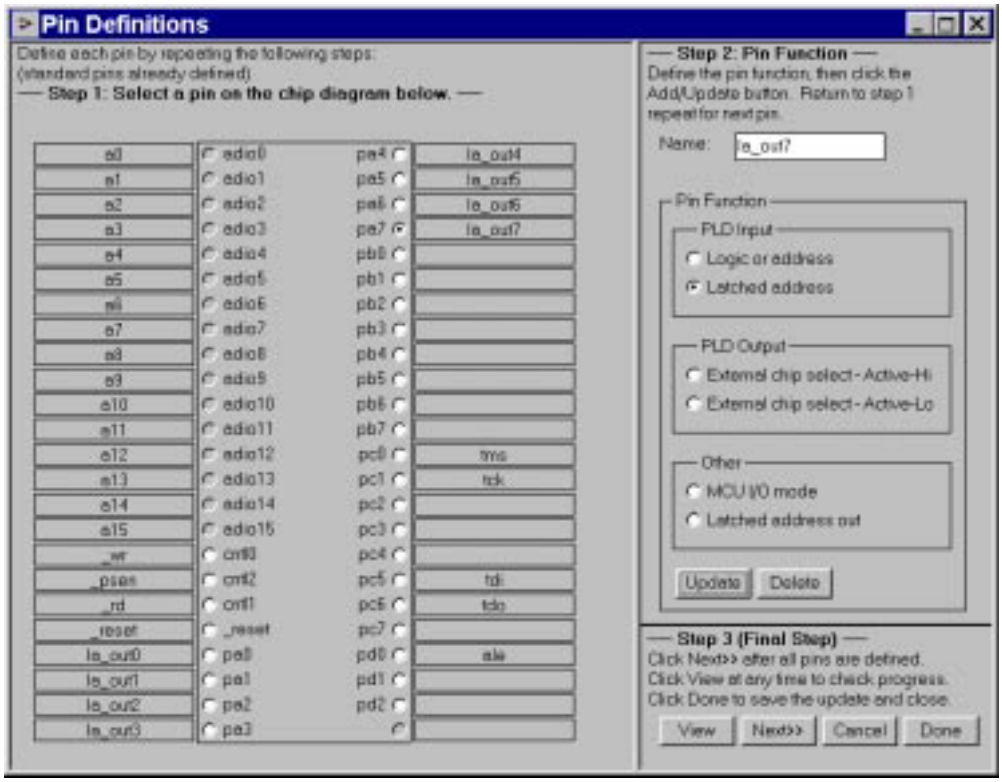
The next section builds on this example by successively adding functions to the PSD913F2 and explains the procedures.

8 Adding Functionality to the PSD913F2

This section builds on the last section and covers what must be done in PSDsoft Express to handle additional functionality for the PSD913F2. With this in mind, let's see what can be done with Port C first by adding the JTAG interface. For information regarding the JTAG interface and how to use it, see *Application Note 54—JTAG Information*.



To ensure that Port C is setup for JTAG, let's take another look at the "Pin Definitions" screen:

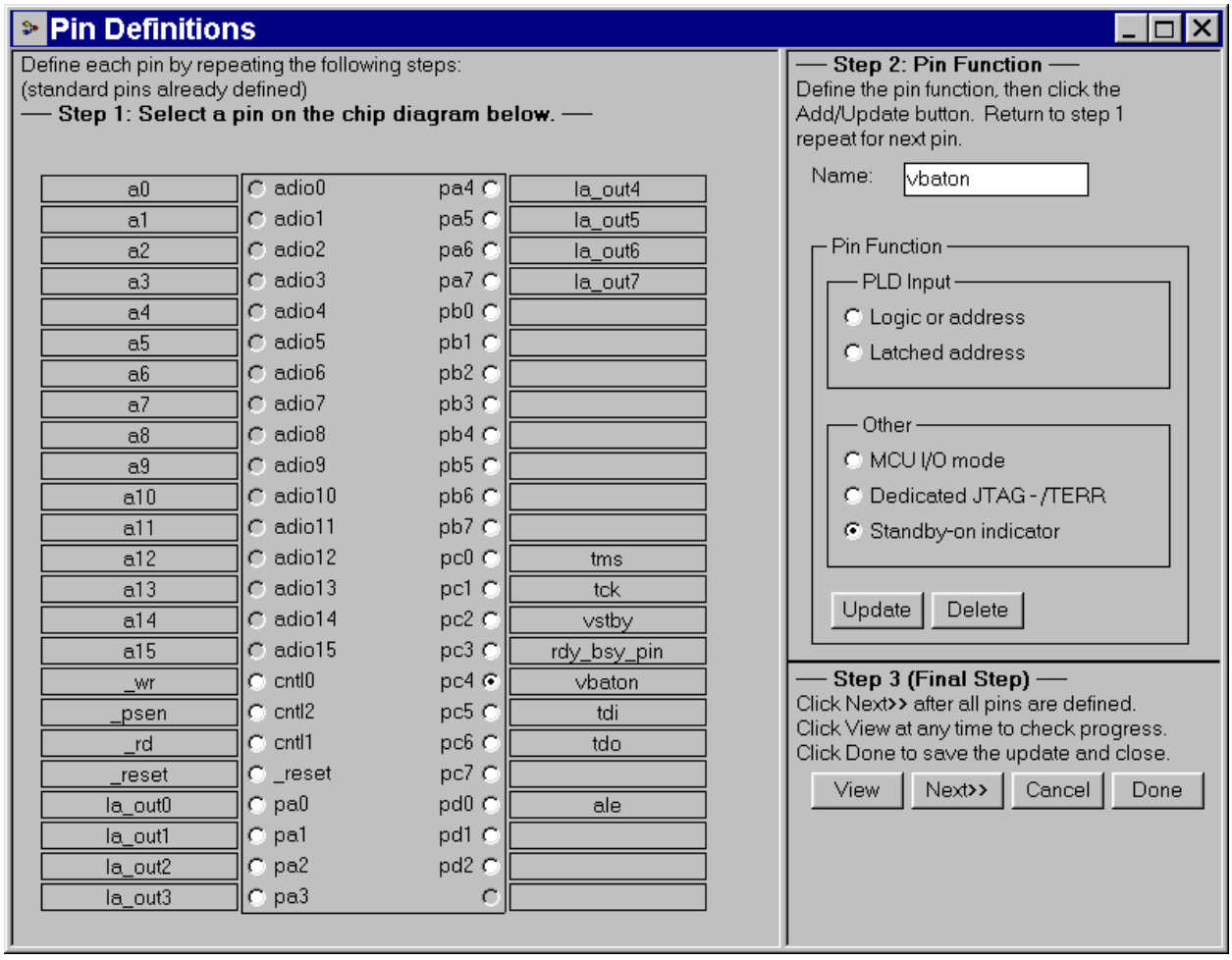


Notice how the port pins pc0, pc1, pc5, and pc6 are already setup for the main four JTAG signals: tms, tck, tdi, and tdo. Note: pins pc3 and pc4 can be used for optional (and non-standard) JTAG signals to help speed up programming.

Now, let's say that we wanted to add the following additional functions on Port C:

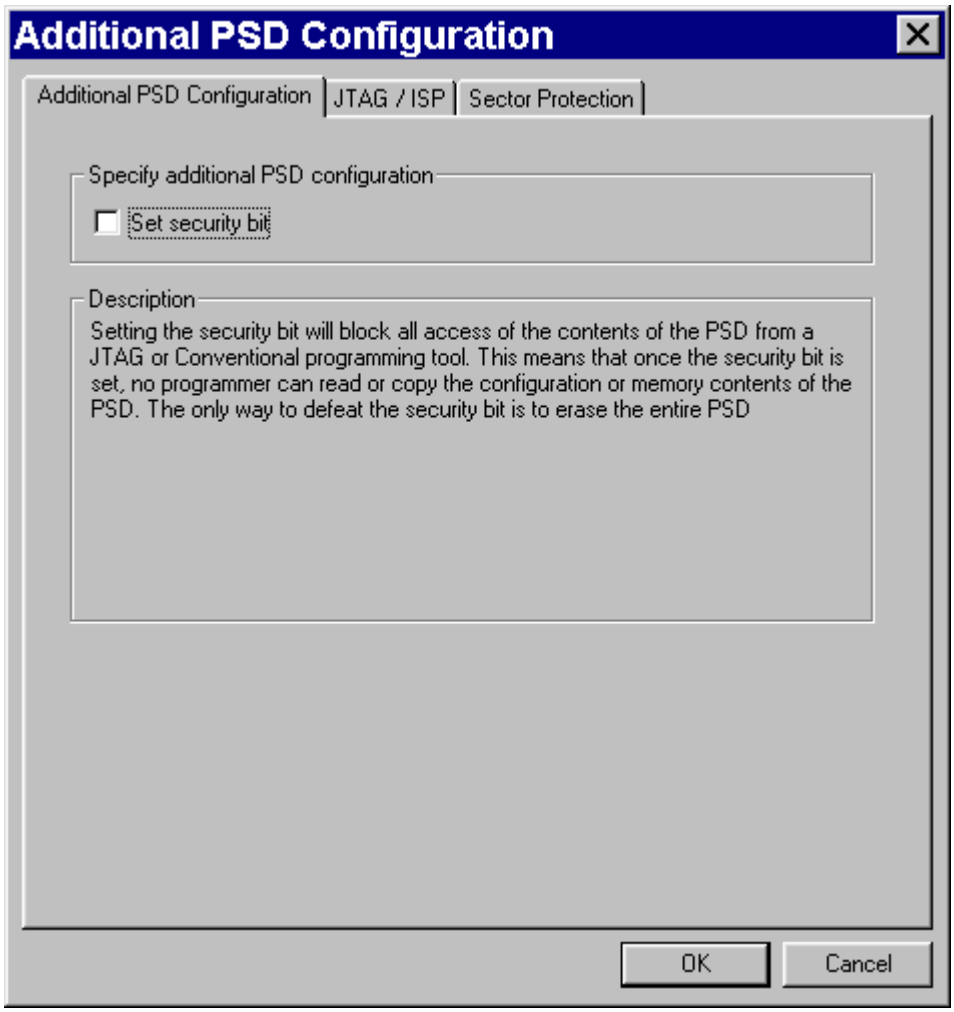
- Connect a battery-backup for the SRAM
- Have an output that indicated when the battery was being used
- Output a programming status signal (Ready/Busy—see Appendix A).

These functions are also set in the "Pin Definitions" screen:



8.1 Additional PSD Configuration

If you had a PSD3XX part, this screen would only have the “Additional PSD Configuration” tab and there would be an option to set the CMISER bit, which is no longer needed in the PSD9XX family because the devices automatically incorporate the equivalent power saving functionality. Thus, by clicking on **Additional PSD Settings** in the PSDsoft Express design flow, you are presented with this screen:



The “JTAG/ISP” tab is used to specify the JTAG User Code. See the description box for more details.

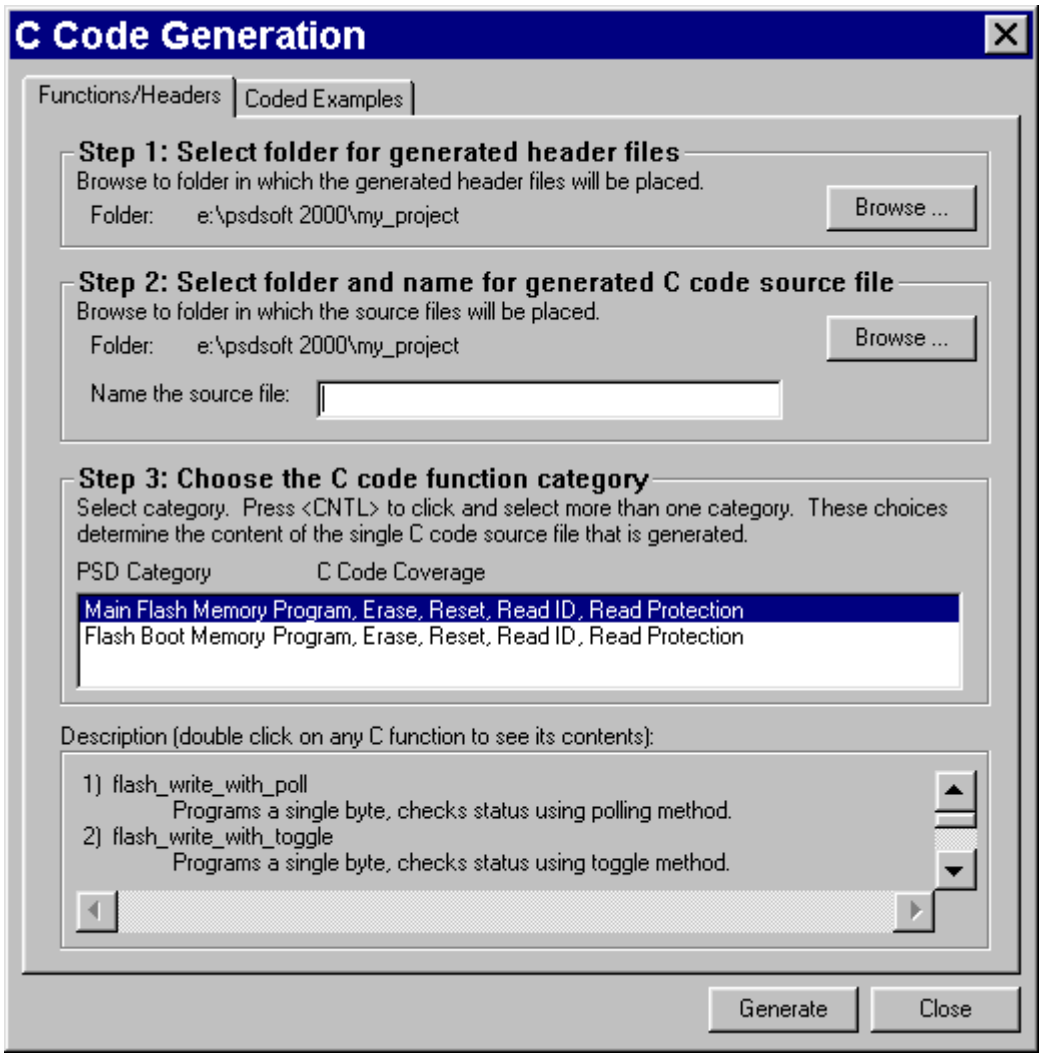
Lastly, the “Sector Protection” tab allows you to set the sector protection bits for individual memory sectors within the PSD9XX. Sector protection write protects the individual sectors and is not applicable to the PSD3XX family.

9 Generating C Code

PSDsoft Express offers C code generation for all flash-based parts, including the PSD9XX. This C code generated conforms to the ANSI-C standard and can be combined with the rest of your code to ease the transition of reading and writing the Flash within the PSD, which requires special "unlock" codes for such purposes. See the Appendix for more details on Flash memory.

The "Coded Examples" brings up a screen that allows you to select from different complete design examples that were written for our development boards.

See the *PSDsoft Express User Manual* for more information on the C code generation.



10 Conclusion—Where to go from Here

If you are interested in upgrading to the FLASH PSD9XX family, it is highly recommended that you visit our website and check out the available resources. The home page has several links that have descriptions of the Flash-based PSD products.

Other resources available on the website include:

- The PSD9XX family data sheets
- *Application Note 54—JTAG Information*
- *Application Note 62—In System Programmability—How Much Money Can it Save?*
- *Application Note 67—PSD913F2/80C32 Design Guide*
- *Application Note 68—PSD913F2/68HC11 Design Guide*
- *PSDsoft Express User Manual*

You will also want to download PSDsoft Express and check it out if you have not already done so. It is available for free on our website.

A Flash Memory Explained

This section explains how Flash memory works from a functional standpoint. More in-depth explanations can be found elsewhere, and are beyond the scope of this document. We start with a comparison of how Flash and EPROM are programmed and erased. Then, the specifics of how our PSD9XX Flash is read and written are provided. Paragraphs highlighted in gray contain information specific to the PSD9XX and may not be true for all Flash-based products.

A.1 Flash versus EPROM—Programming and Erasing

Flash is programmed and erased using embedded algorithms. With embedded algorithms, a series of commands are written to the Flash. These commands unlock the Flash so it can accept data, erase sectors, or perform other programming tasks.

EPROMs cannot be programmed while in-system, so they have no internal state machines to control erase and program algorithms. Instead, they must be programmed and verified (at higher voltage levels than Vcc) using an external programmer.

Erasing an EPROM requires the EPROM's cells be exposed to ultraviolet light for an extended period of time.

The PSD9XX devices can be programmed and erased while in-system (see section 4), and do **not** require a special programmer, an ultraviolet light source, or a voltage higher than Vcc.

A.2 Flash PSD9XX Operations

Flash Read

The read interface of Flash is identical to EPROM. A valid address selects the Flash, and the read strobe tells it to output data at the given address. Valid data would then be presented by the Flash on the data bus after an access time. Valid data is usually the contents of the given address, but it can also be status information about a program or erase in-progress.

Flash Identifier

The Flash identifier provides specific information about the type of Flash on the chip. Reading the Flash Identifier requires a specific embedded algorithm—three specific writes, followed by a read. For more information on the Flash Identifier, refer to section 9.1 of the *PSD9XX Family Data Sheet*.

Flash Sector Protection

The Flash sector can be individually protected, which disables programming and erasing of the sector being protected.

With the PSD9XX family, the sector protection is set using the PSDsoft Configuration utility. The sector protection status can be read using a series of three specific writes, followed by a read.

Flash Program and Erase

Programming and erasing the Flash memory requires special instructions. The special instructions are required in order to protect the contents of the Flash from inadvertent writes and erasure. Flash sectors must be erased before they can be programmed.

The PSD9XX Flash memory is programmed byte-by-byte. The programming instruction is a sequence of three specific write operations to predefined address locations (which "unlock" the Flash), followed by writing the address and data byte to be programmed. During programming, the memory status may be checked by reading status bits (data polling).

There are two ways to erase the PSD9XX: all at once (Bulk Erase) or by sector (Sector Erase). The Bulk Erase instruction requires six write operations, followed by a read operation of the status register bits. The Sector Erase instruction requires six write operations. After the operations, additional addresses can be written to erase other Flash sectors in parallel. During either erase operation, the memory status may be checked by reading the status bits.

Data Polling

Data polling provides a method of determining when a program or erase operation is in progress (or has finished). Data polling is provided because of the large variances in the time it takes to program or erase. Basically, polling the status of a data bit (D6 or D7) determines weather an embedded programming algorithm is still in progress or has completed.

The Ready/Busy Pin

The Ready/Busy pin provides a low overhead method of determining if an embedded programming algorithm is in progress or has completed. The pin is low (busy) during a program or erase operation, and high (ready) otherwise.

Refer to the *PSD9XX Family Data Sheet* for detailed instructions on how the PSD9XX Flash memory operates, including:

- Flash sector selects
- The Ready/Busy pin
- Flash operations
- Flash embedded programming algorithms
- Data Polling.

Table 1. Document Revision History

Date	Rev.	Description of Revision
Aug-2000	3.0	Document written (AN059) in the WSI format
03-Jan-2002	3.1	Front page, and back two pages, in ST format, added to the PDF file References to Waferscale, WSI, EasyFLASH and PSDsoft 2000 updated to ST, ST, Flash+PSD and PSDsoft Express

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

www.st.com/psm

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.

littlediode
COMPONENTS FOR ALL

Littlediode supplies new, hard to find or obsolete electronic components and semiconductors all over the world.

With over two million different components listed you are sure to find the part you need.

Feel free to visit us today at our online store:

LittleDiode.com

Looking forward to providing you with the best possible service.