AN1426
APPLICATION NOTE

Design Guide
PSDsoft Express and PSD4235

CONTENTS

Flash PSD4235 devices are members of a family of Flash-memory based peripherals for use with embedded microcontrollers (MCUs). These programmable system devices (PSDs) consist of memory, logic, and I/O. When coupled with a low-cost 16-bit MCU/MPU, the PSD forms a complete embedded Flash memory system that is 100% In-System Programmable (ISP) and In-Application Programmable (IAP). There are many features in the PSD silicon and in the PSDsoft development software that make using the PSD easy, regardless of how much embedded design experience you have.

This document offers two designs using an ST PSD4235G2 and a Infineon C167CR MCU. Note that a variety of 16-bit MCU/MPUs can be used in place of the Infineon part. Although the specifics of this document are based on the C167CR, this document can be used as a guide for other MCU/MPU applications. The first design is a simple system to get up and running quickly for basic applications, or to check out your prototype C167CR hardware. The second design illustrates the use of enhanced features of PSD In-System Programming (ISP) as applied to the C167CR. You can start with the first design and migrate to the second as your functional requirements grow.

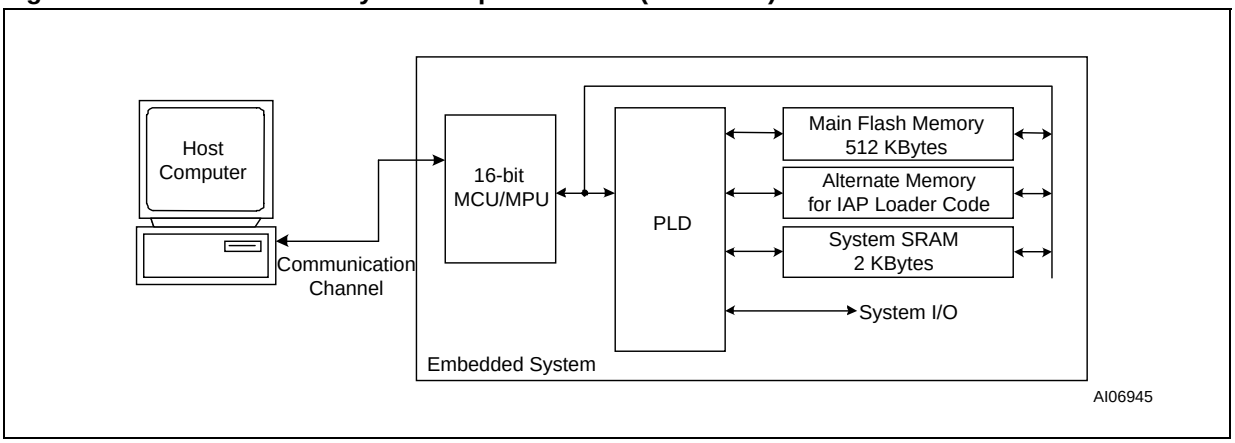
In-System Programming and In-Application re-Programming. Our industry uses the term In-System Programming (ISP) in a general sense. ISP is applicable to programmable logic, as well as programmable Non-Volatile Memory (NVM). However, an additional term will be used in this document: In-Application Programming (IAP). There are subtle yet significant differences between ISP and IAP when microcontrollers are involved. ISP of memory means that the MCU is off-line and not involved while memory is being programmed. For IAP, the MCU 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. PSD4235 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.

The IAP Problem. Typically, a host computer downloads firmware into an embedded Flash memory system through a communication channel that is controlled by the MCU. This channel is usually a UART, but any communication channel that the

C167CR supports will do. The C167CR must execute the code that controls the IAP process from an independent memory array that is not being erased or programmed. Otherwise, boot code and Flash memory programming algorithms (IAP loader code) will be unavailable to the C167CR. It is absolutely necessary to use an alternate memory array (an independent memory that is not being programmed) to store the IAP loader code.

A system designer must choose the type of alternate memory to store IAP loader code (ROM, SRAM, FLASH, or EEPROM); each type has advantages and disadvantages. This alternate memory may reside external to the MCU or on-chip. A top-level view of an embedded IAP Flash memory system with external memory is shown in Figure 1.

Figure 1. Embedded Flash System Capable of IAP (5 devices)

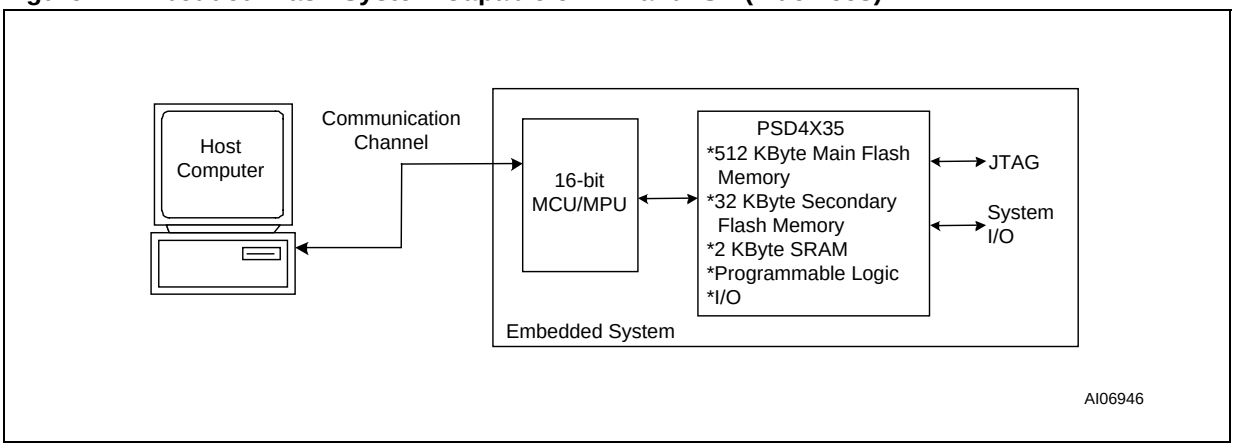


A Common Solution. Without a PSD device, implementing IAP with the C167CR and most other 16-bit MCUs can be difficult and time consuming. For IAP, some C167CR designers will use the fixed boot-loader feature of the C167CR UART to download executable code into SRAM. Then C167CR execution jumps to the SRAM to execute the remainder of the download process for programming the main Flash memory. This can be a cumbersome and error prone exercise using re-locatable code in volatile memory, which is difficult to debug, vulnerable to power outages, and not supported by all emulators. Additionally, this method restricts the designer to using a UART to implement IAP.

A Better, Integrated Solution. Previously, IAP required MCU participation to exercise a communication channel to implement a download to the main Flash memory. However, the PSD4235 offers an alternative to IAP. This method—ISP—uses a built-in IEEE-1149.1 JTAG interface, which requires no MCU participation. This means that a completely blank PSD can be soldered into place, and the entire chip can be programmed in-system using ST's FlashLINK™ JTAG cable ($59 US) and PSDsoft Express™ development software, available for free at *www.st.com/psm*.

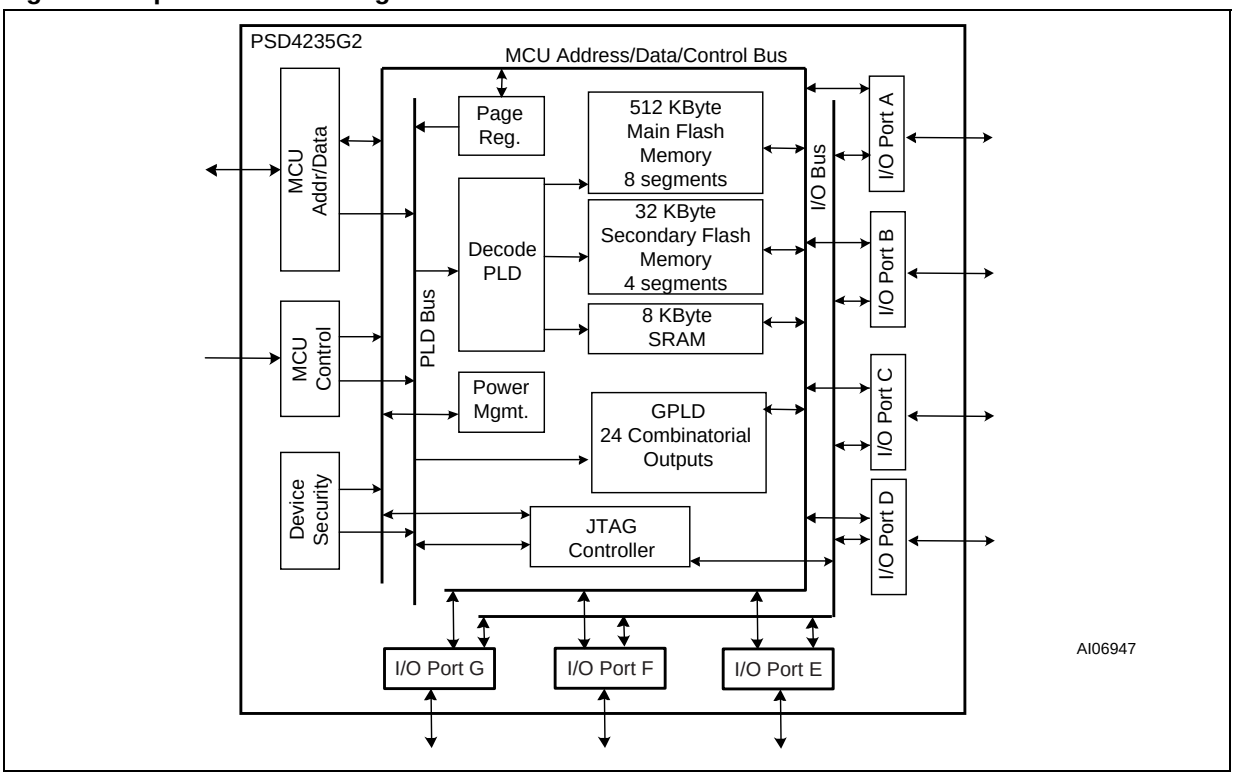
Figure 2 (next page) shows a two-chip solution using a Flash PSD. This system has ample main Flash memory, a secondary Flash memory, and SRAM. All three of these memories can operate independently and concurrently, meaning the MCU can operate from one memory while erasing/writing the other. The system has programmable logic, expanded I/O, and design security. Since the PSD4000 family is 100% ISP, a blank PSD4235 can be connected to a ROM-less MCU/MPU and initially programmed through the JTAG port. Therefore, no IAP firmware needs to be written up front. Just plug in the FlashLINK™ cable and begin programming memory, logic, and configuration. This powerful new feature of the PSD4235 allows immediate development of application code in your lab, smart manufacturing techniques, and easy field updates.

Figure 2. Embedded Flash System Capable of IAP and ISP (2 devices)



Let's take a quick look inside the Flash PSD4235, as shown in Figure 3. You can see the three independent memory arrays, which are selected on a segment basis when the proper MCU address is decoded in the Decode PLD. The page register participates in memory decoding, which greatly simplifies paging. The MCU address, data, and control signals are routed throughout the chip and can be used within the general-purpose PLD. The GPLD has 24 combinatorial logic outputs for external device chip-selects or general logic. There are 52 I/O pins that can be individually configured for many different functions. A power management scheme can selectively shut down parts of the chip and tailor special power saving mechanisms on-the-fly. The security feature can block access to all areas of the chip from a device programmer/reader. Finally, the self-contained JTAG-ISP controller allows programming of all areas of the chip.

Figure 3. Top Level Block Diagram of PSD4235

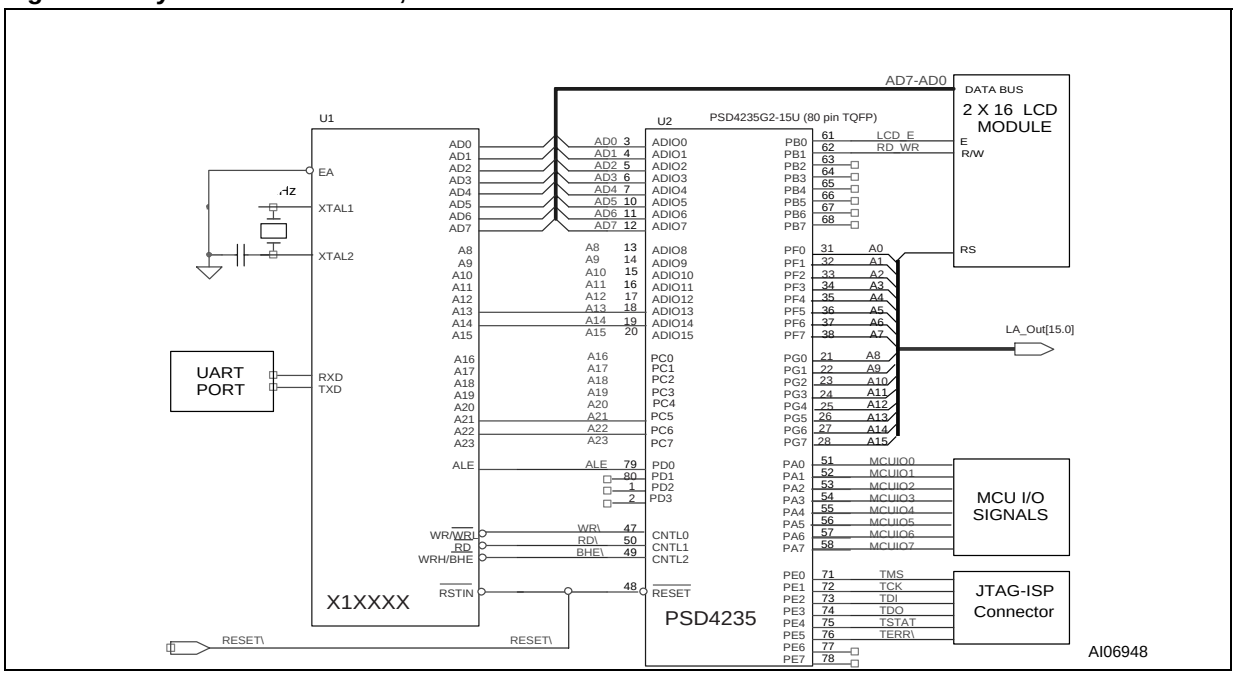


PHYSICAL CONNECTION

Connect your C167CR to the PSD4235 as shown in Figure 4. An 80-pin package is shown in the example. The same connections can be used for all of the members of the PSD4000 family. The JTAG programming channel, LCD module, latched address output, and MCU I/O signals are all optional.

This example design is similar to ST's DK4000-C167 Development kit, available for purchase ($149 US) on the web: *www.st.com/psm*. There are 11 unused PSD I/O pins in this example. Unused pins should be pulled to V_{CC} with a 100K resistor or tied to GND. See *Application Note 1153* for more information on the JTAG port.

Figure 4. Physical Connections, C167CR and PSD4235



FIRST DESIGN EXAMPLE—ISP CAPABLE SYSTEM

The first design example is only capable of ISP and not IAP. It outlines the steps required to get a Flash memory C167CR system up and running quickly. Basically, the PSD's secondary Flash memory will be programmed with the JTAG-ISP channel with code that will execute basic system tests and display some messages on the LCD. The second design example takes advantage of concurrent memory operation and IAP, using the main Flash memory in addition to the secondary Flash memory. You should become familiar with this first design before using the second.

A PSD4235G2 was used for this example. However, other members of the *Easy*FLASH™ family may be used instead, with minor changes to the sample design. See the PSD4000 series data sheets for a comparison of family members.

For this simple design, we used a PSD4235G2 with the following memories:

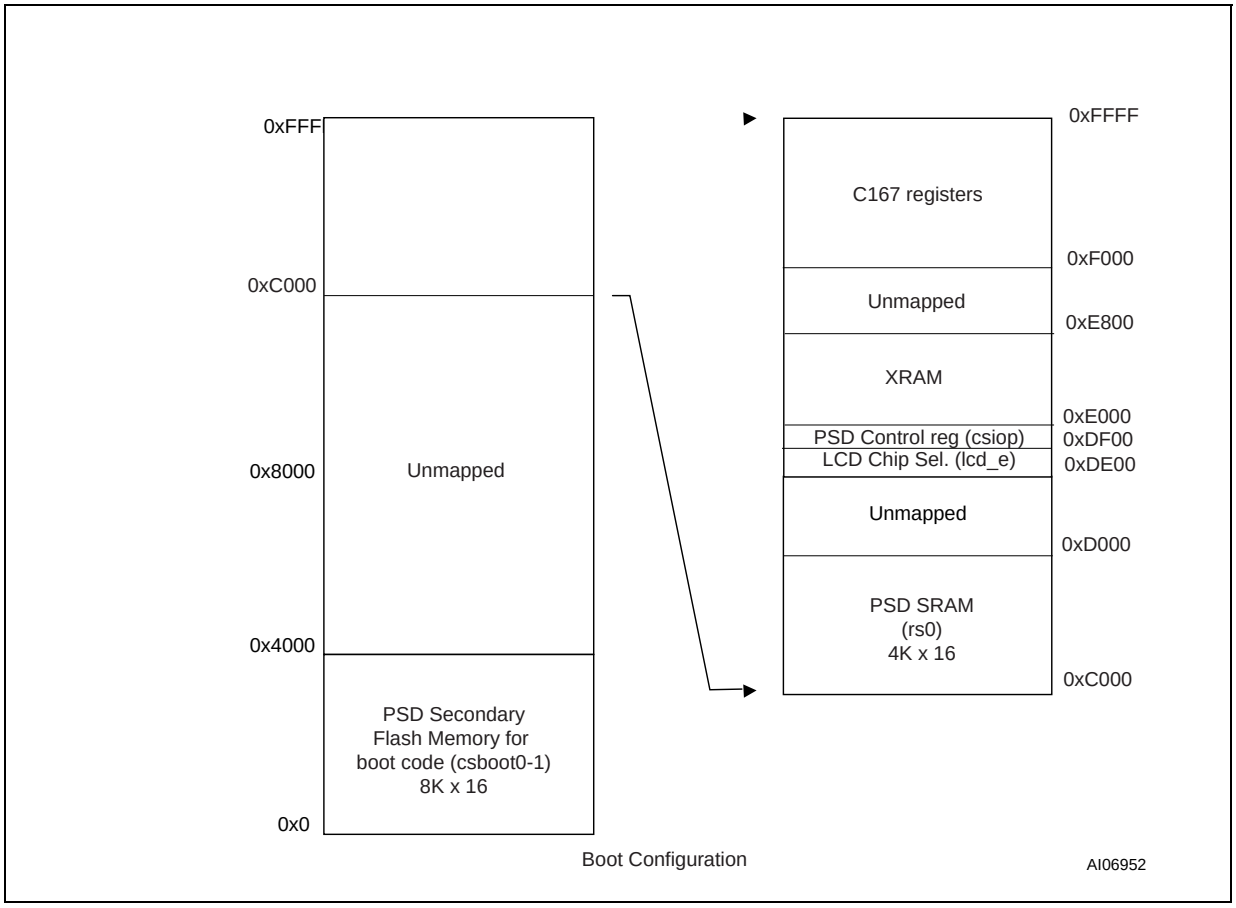
- 512 KBytes main Flash memory, broken into eight 64 KByte segments denoted $fs_{i\ (i = 0\text{-}7)}$
- 32 KBytes secondary Flash memory, broken into four 8 KByte segments denoted $csboot_{j\ (j = 0\text{-}3)}$.
- 8 KByte SRAM (rs0)
- 256-Byte PSD4235 configuration register (csiop).

Note: the PSD memory segments are defined using PSDsoft Express™.

We'll use the PSD's secondary Flash memory to hold the boot code, C167 interrupt vectors, and common firmware functions. For this example, we'll execute from the PSD's secondary Flash memory only and not use the PSD's main Flash memory.

Let's examine the sample memory map in Figure 5, below.

Figure 5. Memory Map: Simple C167CR/PSD4235 Design



Note: 1. Only C167 page 0 is used for this demonstration (no segmentation, 64k max).
2. Syscon and buscom0 used for memory access except for csiop and ice_e, which use buscon1.

Note the following about the sample memory map shown in Figure 5:

- This simple example only requires 16 address bits
- Only half (16 KBytes) of the secondary Flash memory is used
- The middle 32 KBytes of the memory map is unused
- The upper 16 KBytes is allocated for the PSD SRAM (rs0), the PSD control register (csiop), the LCD module, and the C167CR registers and RAM.

The boot memory holds the following information:

- C167CR reset vector and initialization routines
- C167CR interrupt vectors and service routines
- I/O management.

Since Figure 5 is a sample memory map, you may wish to change it. To do so, simply edit the chip select equations for the desired segments using the Design Assistant.

PSDsoft Express Design Entry

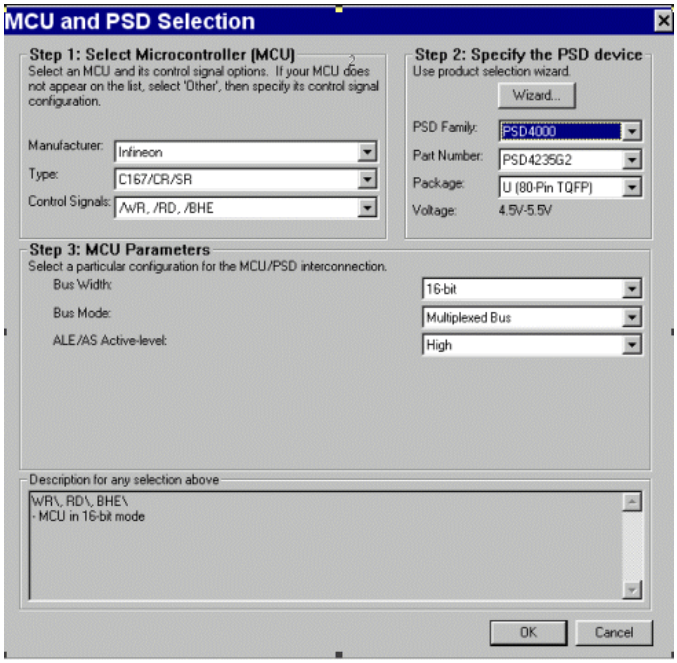
Highlights of design entry will be given here. This section is meant to show you just the essentials to get you going. Here are the steps:

Invoke PSDsoft Express and Open a New Project.

- Start PSDsoft Express.
- Create a new project.
- Select your project folder and name the project (in this example, name the project “EasyC167” in the folder PSDexpress\my_project).
- Select an MCU. In this example, we’re using an Infineon C167CR.
- Select /WR, /RD, /BHE for the control signals.
- Select the PSD4000 series for the PSD Family.
- Select a PSD4235G2 and use the 80-pin TQFP package (U package).
- Based on the above selections, the Bus Width will be set to 16-bits automatically.
- Select the Bus Mode to be multiplexed and the ALE/AS level will automatically be set to high.

Now you have your project established based on a PSD4235G2 and a C167CR. However, there are many other MCU/MPUs you could have chosen in place of the C167CR and still have use of this document. The main reason for selecting the C167CR is that it is used in our DK4000 development kit.

MCU and PSD Selection. This is what the screen should look like after you’ve made the selections:



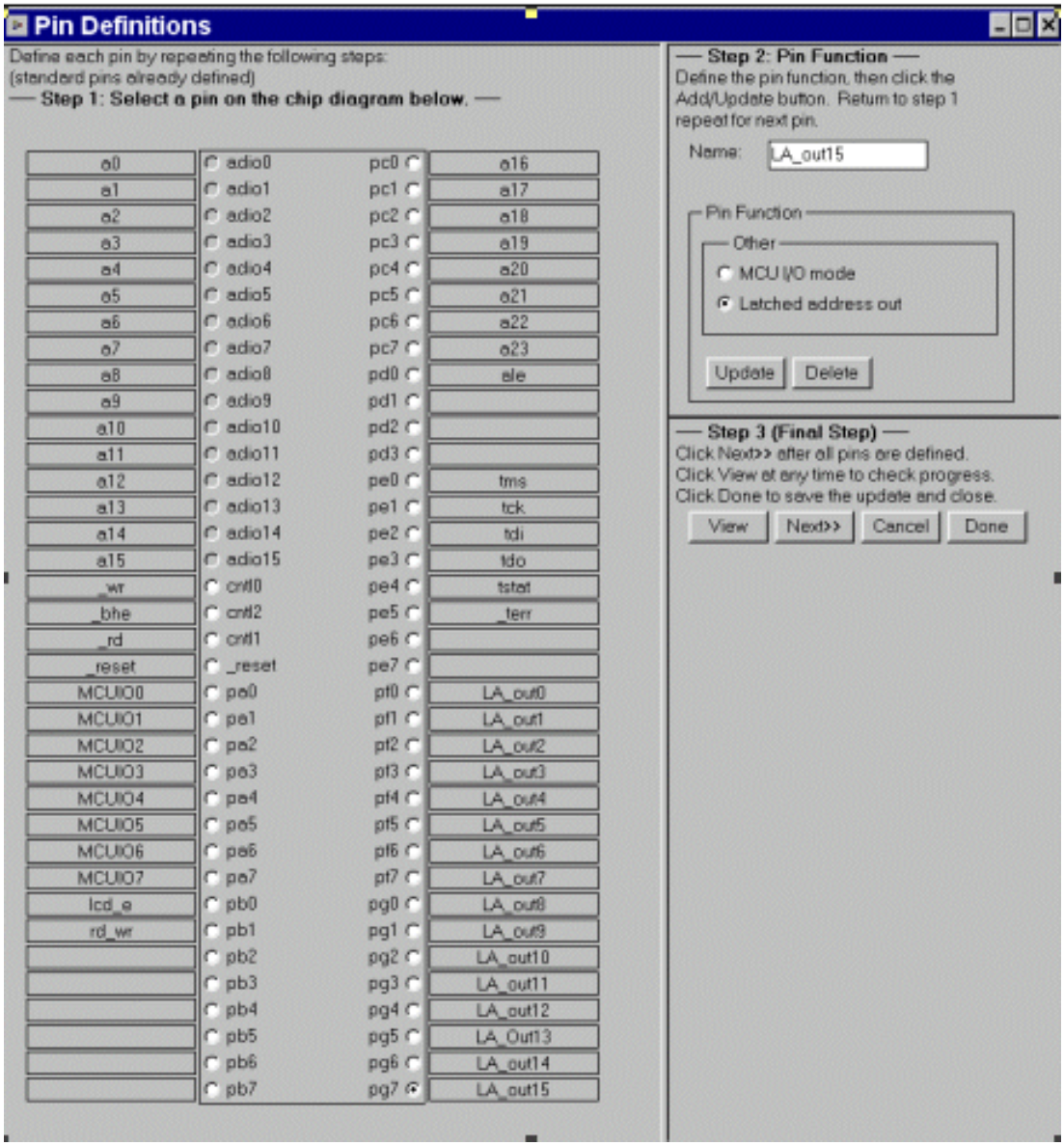
Click **OK**. Now you will be asked if you want to use the Design Assistant or a pre-defined template. Choose Design Assistant. This exercise in the Design Assistant will help you become familiar with the design flow. In the future, you may choose to use a template, which will make many of the choices for you, based on your selection of MCU and PSD.

Pin Definitions. You are immediately taken to the “Pin Definitions” screen, which allows you to define each PSD pin function on a point and click basis. Notice that some of the PSD pins that connect to the C167CR are already defined for you because their function is set. You need only define the remaining pins. We want to define the remaining pins based on the functional requirements presented in the sche-

matic. Define the pins as follows:

- Set all the pins of Port A to MCU I/O mode and label them “MCUIO0” to “MCUIO7.”
- Define pb0 and pb1 as external active-high chip selects. Name pb0 “lcd_e” and pb1 “rd_wr.”
- Set all the pins on Port C to Address Input and give them the label “a16” to “a23.”
- The Port E pins pe0 to pe5 should be assigned to dedicated JTAG signals.
- Define the Port F and Port G pins to be Latched Address Out and give them the labels “LA_out0” to “LA_out15”, where LA_out0 is assigned to pf0 and LA_out15 to pg7.

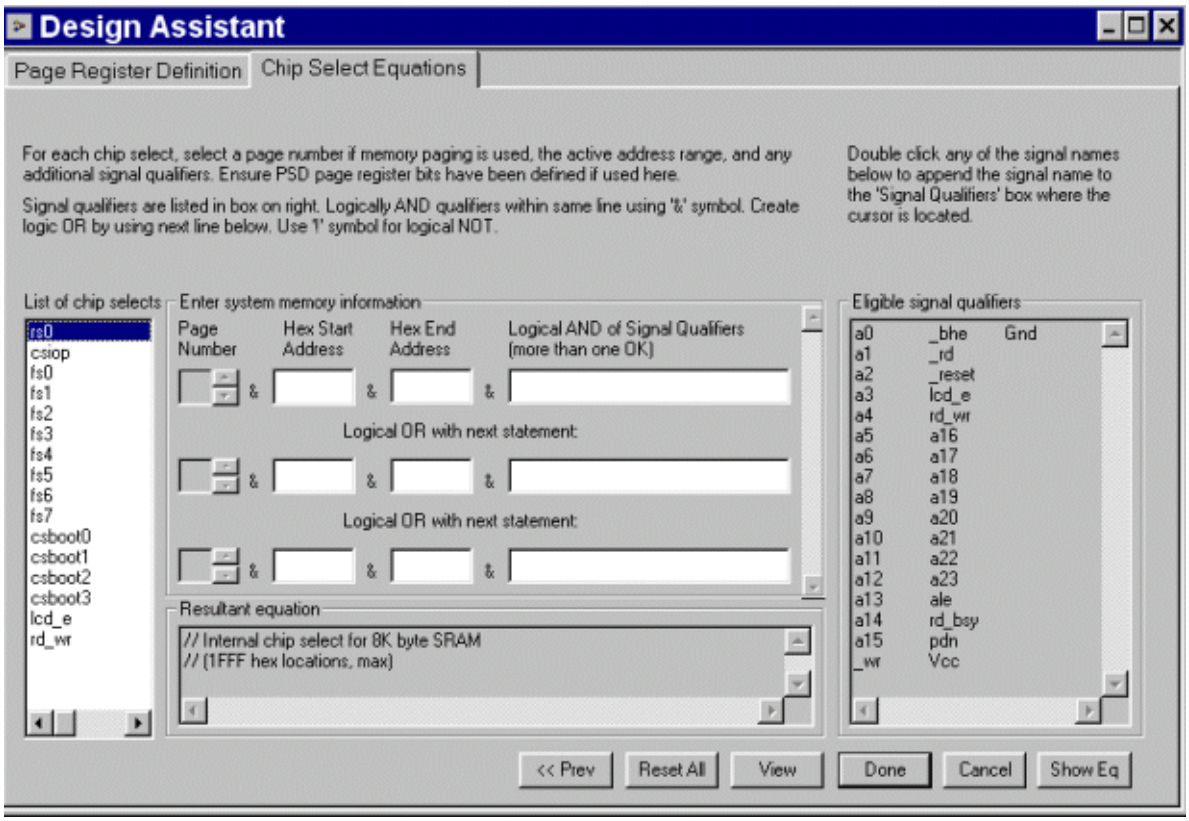
Your Pin Definition screen should now look like the screen capture below:



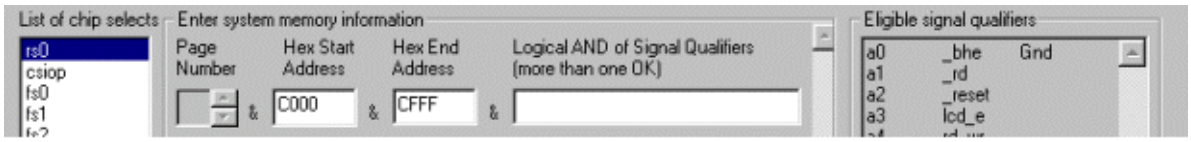
You can view a summary of your pin definitions by clicking the **View** button. When you are satisfied that you have defined all the pins correctly, click the **Next>>** button to be taken to the “Page Register Definition” screen. Since we are not using paging in this example, you can immediately click on the “Chip Select Equations” tab. It is on this screen that you define your memory map for internal and external chip-select

equations.

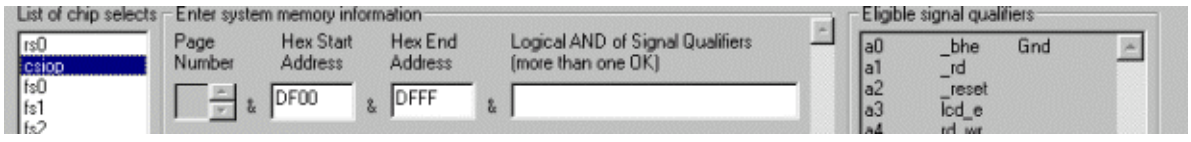
Chip Select Equations. Your screen should be similar to the capture below:



Let's start by defining the chip-select for the internal SRAM (rs0). Looking at the memory map of Figure 5, we see that 4 Kwords (8 KBytes) of address space needs to be allocated to the PSD's internal SRAM. So, we enter the Hex Start Address of C000 and the Hex End Address of CFFF. Here is a snapshot of what your screen should look like after you have entered the equations:

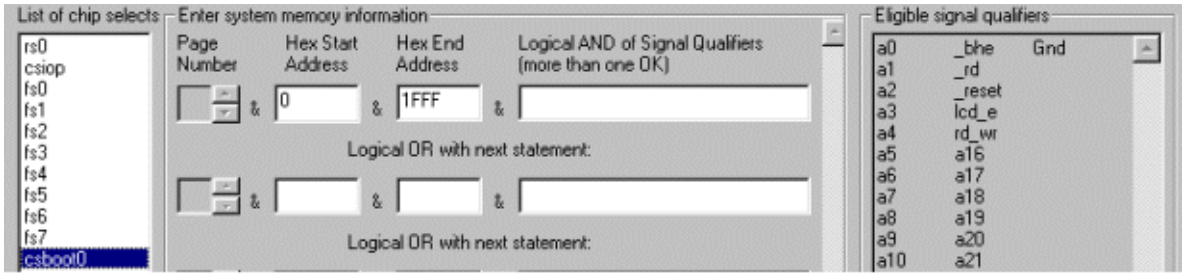


Now, click on csiop and enter the equations according to Figure 5. Note: the csiop is X 8 access.



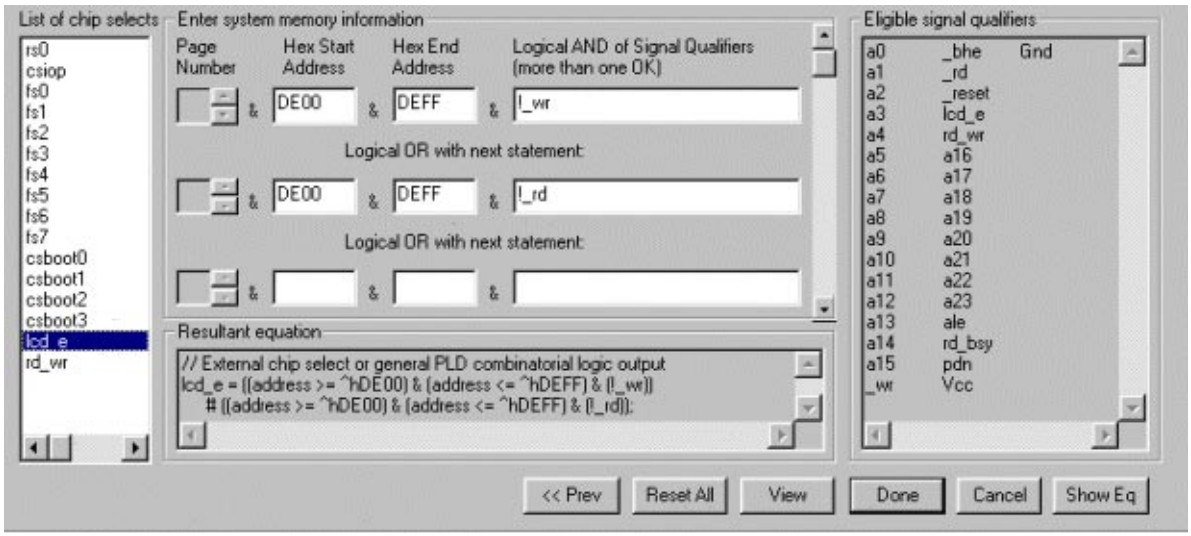
The boot code is stored in the secondary Flash memory in the lower 8 Kwords of address space from 0h

to 3FFFh in csboot0 and csboot1. The equation for csboot0 should be entered as follows:



Enter the information for csboot1 in a similar fashion where csboot1 will be valid from 2000h to 3FFFh.

Enter the external chip-select equation for the LCD module (X 8 access):



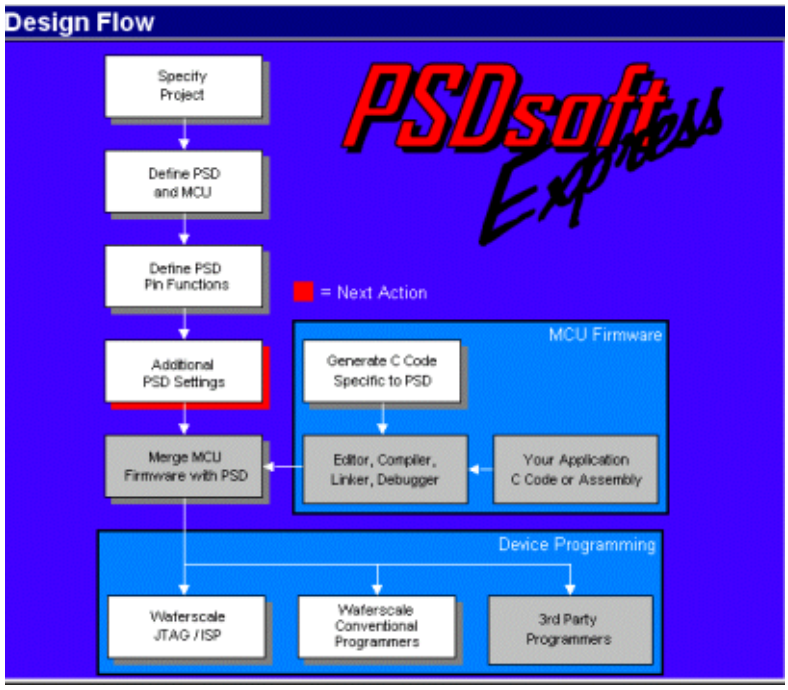
Note that since the LCD is external, we must now include the _wr and _rd control signals in the chip select equation, where this was taken care of automatically for internal chip-selects. Since the _wr and _rd signals are active-low, the ! symbol is required. Also notice how, after you have finished typing in the equation for the chip select, the end result appears at the bottom of the screen if you select that signal again.

Lastly, click on the rd_wr signal and type “Gnd” in the last column to keep the signal low at all times. If the desire was to keep the signal high at all times, you would have typed V_{CC} instead.

You can click the **View** button at any time to see a Design Assistant summary. Once you are satisfied with the results, click the **Done** button. Clicking **Done** starts a preliminary resource check of the information you have entered to ensure that there are no overlapping memory segments, among other tests. Any errors encountered will be indicated.

Design Flow. Once you have clicked on **Done**, you are taken to the “Design Flow” window. Use this window as your main navigational tool for PSDsoft Express™. Clicking on individual boxes within the flow diagram will invoke a process. A box shadowed in red identifies the next process that needs to be completed. The first three steps have been completed to this point. If you invoke a process that invalidates other processes downstream, the gray boxes indicate which processes must be invoked again and the red shadow indicates which process to invoke first.

The design flow should be in the following state:



Additional PSD Settings. Click on the “Additional PSD Settings” box. This is where you may choose to set the security bit to prevent a device programmer from examining or copying the contents of the PSD. You can also click through the other sheets on this screen to set the JTAG USERCODE value and set sector protection on PSD Non-Volatile memory segments as desired.

PSD-Specific C Code Generation. You can take advantage of the provided low-level C code drivers for accessing memory elements within the PSD by clicking on the “Generate C Code Specific to PSD” box in the design flow window. ANSI C code functions and headers are generated for you to paste into your C compiler environment. Simply tailor the code to meet your system needs and compile. C code generation can be performed anytime after a project is opened.

To generate ANSI C functions and headers, simply specify the folder(s) in which you want the header files and the C source file to be written, and name the C source file. Select the categories of functions that you would like to include, then click **Generate**. Three files will be written to your specified folder(s):

- <your_specified_name>.c ANSI-C source for all of the selected functions
- psd4235g2.h—ANSI-C header file to define particular PSD registers
- map4235g2.h—ANSI-C header file to define locations of system memory
- elements (Main/Secondary Flash 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 psd4235g2.h contains a #define statement for each individual C function within the <your_specified_name>.c file. Edit psd4235g2.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.

There are also coded examples available. Click on the “Coded Examples” tab at the top of the C Code

Generation screen. This sheet contains several examples that you may use as a basis for building your own C code application. These are complete projects (main, functions, and headers) targeted toward various MCUs. You may copy these files to some folder to browse them for ideas, or cut and paste sections from the examples into your own MCU cross-compiler environment.

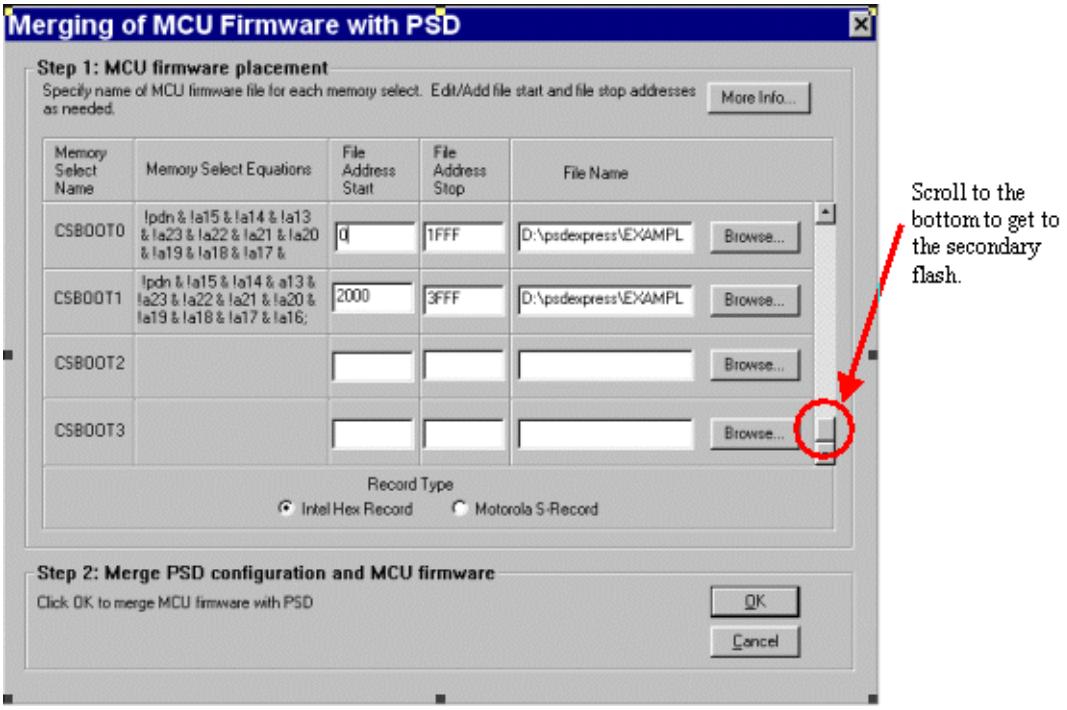
Merge MCU Firmware with PSD. Now that all PSD pins and internal configuration settings have been defined, PSDsoft Express™ will create a single object file (.obj) that is a composite of your firmware and the PSD configuration. FlashLINK™, PSDpro, and third party programmers can use this object file to program a PSD device. PSDsoft Express will create a file called “EasyC167.obj” for this design example.

During this merging process, PSDsoft Express will input firmware files from your compiler/linker in S-record or Intel HEX format. It will map the content of these files into the physical memory segments of the PSD according to the choices you made in the “Chip Select Equations” screen. This mapping process translates the absolute system addresses inside firmware files into physical internal PSD addresses that are used by a programmer to program the PSD. This address translation process is transparent. All you need to do is type (or browse) the file names that were generated from your linker into the appropriate boxes and PSDsoft Express does the rest. You can specify a single file name for more than one PSD chip-select, or a different file name for each PSD chip-select. It depends on how your linker has created your firmware file(s). For each PSD chip-select in which you have specified a firmware file name, PSDsoft Express will extract firmware from that file only between the specified start and stop addresses, and ignore firmware outside of the start and stop addresses.

Click on “Merge MCU Firmware” in the main flow diagram. First you will notice that PSDsoft Express™ will “Fit” your PSD configuration to the silicon architecture of the PSD. After the fitting process is complete, the “Merging of MCU Firmware with PSD” screen appears.

In the left column are individual PSD memory segment chip-selects (FS0, FS1, and so on). The next column shows the logic equations for selection of each internal PSD memory segment. These equations reflect the choices that you made while defining PSD internal chip-select equations in an earlier step. In the middle of the screen are hexadecimal start and stop addresses that PSDsoft Express has filled in for you based on your chip-select equations. On the right are fields to enter (browse) the MCU firmware files.

Select “Intel Hex Record” for “Record Type” as shown. Scroll all the way down to the bottom to get to the secondary Flash memory. Now, click on the Browse… button for csboot0 and csboot1 and select the firmware file, PSDexpress\examples\Tim.h86. Once you have filled in the file names, your screen should look like the one below:



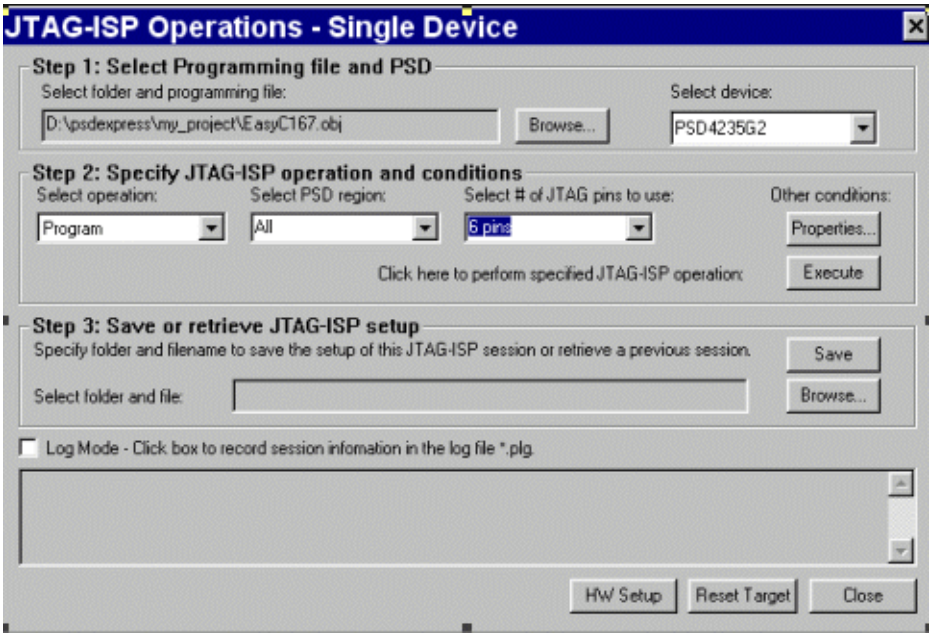
This specification places firmware in PSD secondary Flash memory segments csboot0 and csboot1. PSDsoft Express will extract any firmware that lies inside the file Tim.h86 between MCU addresses 0000 and 3FFF and place it in PSD memory segment csboot0-1. Click **OK** to generate the composite object file, EasyC167.obj.

Note: the file Tim.h86 will run on the DK4000-C167 development board from ST, and display some messages on the LCD screen to indicate a successful ISP session. For your own prototype project, create a simple firmware file that configures your system hardware and performs rudimentary tasks to check out your new hardware. After your new hardware is proven, you can add more code for advanced tasks, including IAP of the PSD Flash memories.

Programming the PSD. The EasyC167.obj file can be programmed into the PSD by one of three ways:

- The ST FlashLINK™ JTAG cable, which connects to the PC parallel port.
- The ST PSDpro device programmer, which also uses the PC parallel port.
- Third-party programmers, from Stag, Needhams, and others. See our web site at *www.st.com/psm* for compatible third-party programmers.

Programming with FlashLINK™. Connect the FlashLINK™ JTAG-ISP cable to your PC parallel port. Click the “JTAG-ISP” box in the design flow window. You will be asked how many devices are in your JTAG chain. For this example, select “Only One.” You would only select “More than One” if you had more than one ISP device in your JTAG chain. After you make your selection and clicked **OK**, you should see the following screen:



This window enables you to perform JTAG-ISP operations and also offers a loop back test for your FlashLINK™ cable. If this is your first use, test your FlashLINK™ cable and PC parallel port by clicking the **HW Setup** button, then click **LoopTest** button and follow the directions.

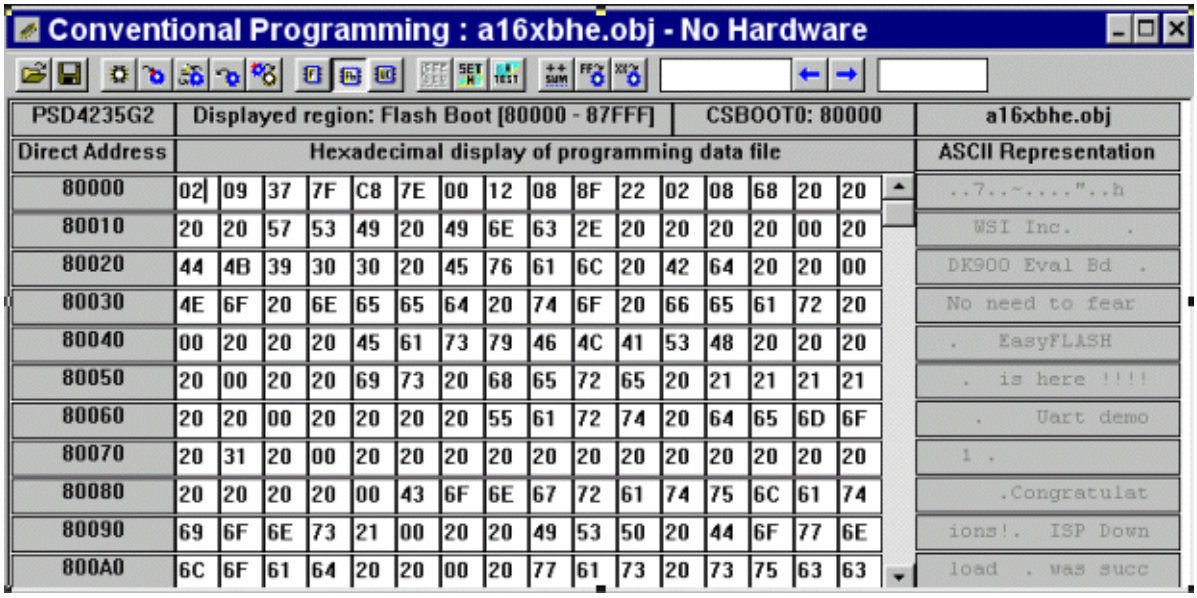
Now let's define our JTAG-ISP environment. PSDsoft Express should have filled in the folder and filename of the object file to program, the PSD device, and the JTAG-ISP operation, as shown in the screen above. For this design example, we have chosen to use all six JTAG-ISP pins (instead of four). Be sure to indicate "6 pins" as shown above to achieve minimum JTAG-ISP programming times. (Refer to *Application Note 1153* for details on six pins versus four.)

To begin programming, connect the JTAG cable to the target system, power-up the target system, and click **Execute** on the JTAG screen. The Log window at the bottom of the JTAG screen shows the progress.

There are optional choices available when the **Properties…** button is clicked. One choice includes setting the state of all non-JTAG PSD I/O pins during JTAG-ISP operations (make them inputs or outputs). The default state of all non-JTAG PSD I/O pins is "input", which is fine for this design example. The other choice allows you to specify a USERCODE value to compare before any JTAG-ISP operation starts. This is typically used in a manufacturing environment. (See the on-screen description for details.)

After JTAG-ISP operations have completed, you can save the JTAG setup for this programming session to a file for later use. To do so, click on the **Save** button in "Step 3". To restore the setup of a previous session, click the **Browse…** button in "Step 3".

Programming with PSDpro. Connect the PSDpro device programmer to your PC parallel port per the installation instructions. Click on the "Conventional Programmers" box in the design flow window. You will see this:



If this is the first use of the PSDpro, you'll need to designate the PSDpro as the device connected to your parallel port. To do this, click the **SET H** icon button at the top of the "Conventional Programming" screen and choose the PSDpro. Then click on the **H TEST** icon to perform a test of the PSDpro and the PC parallel port. After testing, place a PSD4235G2 into the socket of the PSDpro and click on the **Program** icon. (The EasyC167.obj file is automatically loaded when this process is invoked.) The messaging of PSDsoft will inform you when programming is complete.

Note: this window is also helpful even if you do not have a PSDpro device programmer. Use this window to see where the "Merge MCU Firmware" utility has placed C167CR firmware within physical memory of the PSD. For this design example, click on the secondary PSD Flash memory icon "Fb" in the tool bar to see the C167CR vector at absolute MCU addresses 0001h and 0002h, which translates to direct physical PSD addresses 80001h and 80002h, respectively. To see how all of your C167CR absolute addresses translated into direct physical PSD memory addresses, view the report that PSDsoft generates under "Reports" from the main toolbar, then select "Address Translation Report." Within the report, the Start and Stop addresses are the absolute MCU system addresses that you have specified. The addresses shown in square brackets are the direct physical addresses used by a device programmer to access the memory elements of the PSD in a linear fashion (a special device programming mode that the MCU cannot access).

ENHANCED DESIGN EXAMPLE

This section should be available in June, 2000. It will detail how to use the PSD memories concurrently to perform IAP. If you have any questions on this in the mean time, contact an applications engineer at apphelp@wsiusa.com.

CONCLUSION

These examples are just two of an endless number of ways to configure the *Easy*FLASH™ PSD for your system. Concurrent memories with a built-in programmable decoder at the segment level offer excellent flexibility. The ability to expand your system does not require any physical connection changes, as everything is configured internal to the PSD. And finally, the JTAG channel can be used for ISP anytime, and anywhere, with no participation from the MCU. All of these features are crosschecked under the PSDsoft Express™ development environment to minimize your effort to design a Flash-memory based system capable of ISP and IAP.

REFERENCES

3. PSD4000 Family Data Sheet

4. *Application Note 1153— JTAG Information—PSD8XXF* for detailed use of the JTAG channel

5. *DK4000 User Manual*—For information on the C167CR/PSD4235G2 development kit.

Table 1. Document Revision History

Date	Rev.	Description of Revision
Apr-2000	1.0	Document written (AN069) in the WSI format
03-Jan-2002	1.1	Front page, and back two pages, in ST format, added to the PDF file References to Waferscale, WSI and PSDsoft 2000 updated to ST, ST and PSDsoft Express
25-Jul-2002	1.2	Document converted to ST format.

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.