Programmable Peripheral Application Note 023

PSD3XX Family Programmable Microcontroller Peripheral Design Tutorial

By Mark Elliott

Introduction

The PSD3XX family devices contain several commonly used microcontroller peripheral functions combined into one package. These include EPROM, SRAM, Chip Selects, and logic functions. Some of the advantages of the PSD3XX family are: board space is reduced, power consumption is reduced, cost is competitive – usually less, and board complexity is reduced. Design risk is also reduced because fewer traces are required on the PWB and the PSD3XX devices are more flexible than a discrete component design. Mistakes or design changes pose less of a potential problem. Additionally, the PSD3XX device includes a security option which, when implemented, protects the internal configuration data from duplication.

For a particular application, the designer should learn the PSD3XX family architecture, understand the configuration software, and understand the programming process. While none of these tasks are complicated, full knowledge of them is not required to understand the PSD3XX family. This application note introduces the PSD3XX family design process by example. While going through the examples, you will learn about the entire design process including hardware, software, and programming. Those who do not have a strong background in the microcontroller field may also find themselves able to use the PSD3XX. This application note uses an 80C31 system as a model. Even if you intend to use a different microcontroller, you will find it useful to read on.

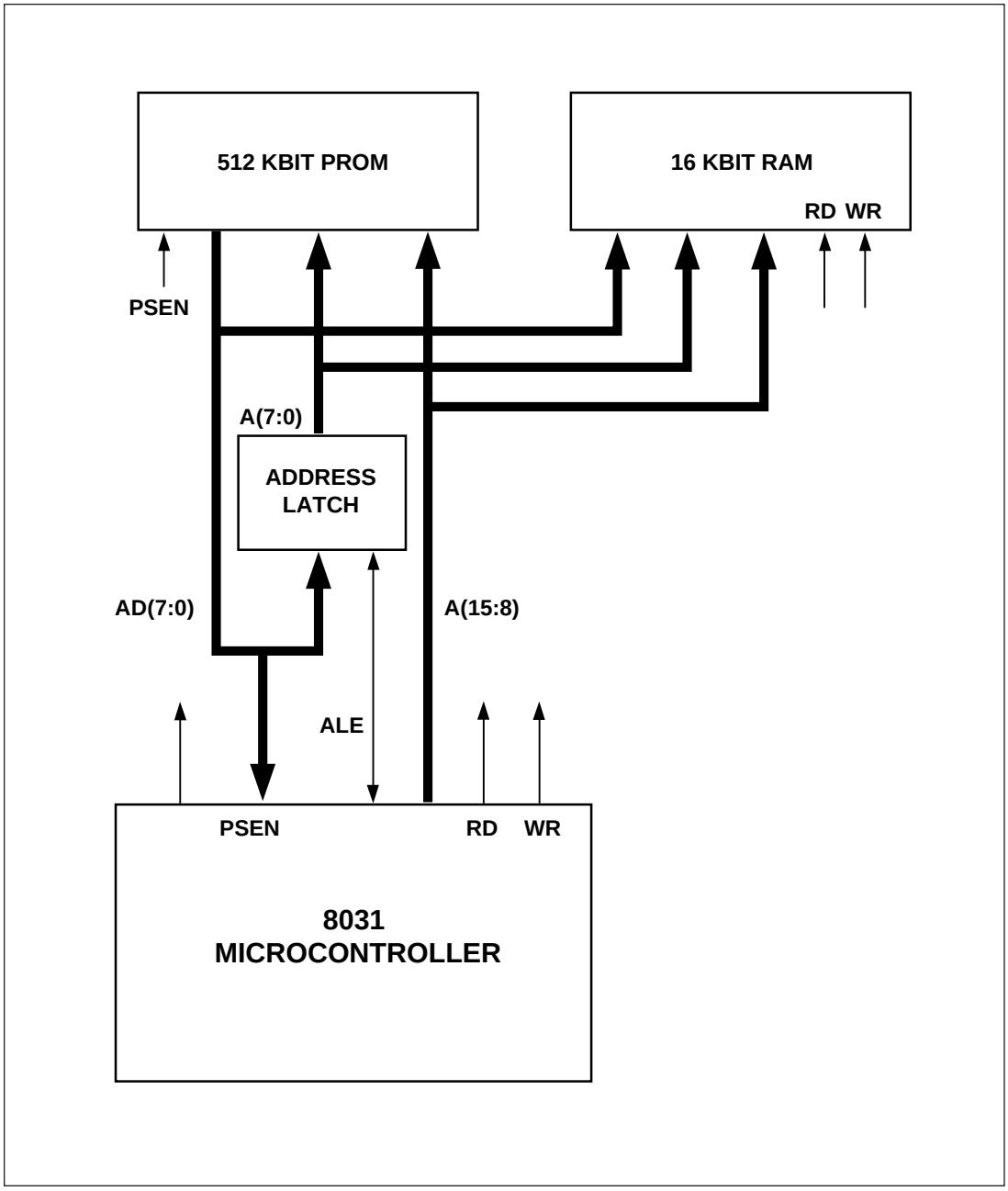
This application note demonstrates two designs using multiple packages which will be replaced by a design using the PSD312. The first example is a standard 80C31 board, one that does not make use of all the PSD312's potential. This will form a basis for understanding the product. In a second example, new functions are added to the standard design. By the end of the application note, you should have enough basic knowledge to understand the PSD312 device's use in your design.

PART I – Using the PSD312 with a Standard 8031 System.

Figure 1.1 illustrates a standard 80C31 microcontroller board design. The board contains a microcontroller, a 512 Kbit EPROM for program storage, a 16 Kbit static SRAM, and an address latch. In this example, all of the circuits on this board, excluding the microcontroller, will be replaced. Keep in mind that the PSD312 is not being used to its full advantage here. In the second example, Part 2 of this application note, you will see that the PSD312 is able to provide additional functions, replacing additional discrete packages.

The PSD312 was chosen for this task because it has the same SRAM and EPROM space as that used on the original design. A similar device, the PSD311, would be more suitable for replacing a smaller EPROM space of 256K bits or less. If a larger EPROM space is required, a PSD313 with its internal 1M bit UV EPROM could be used.

Figure 1.1: 8031 Microcontroller Standard System Block Diagram

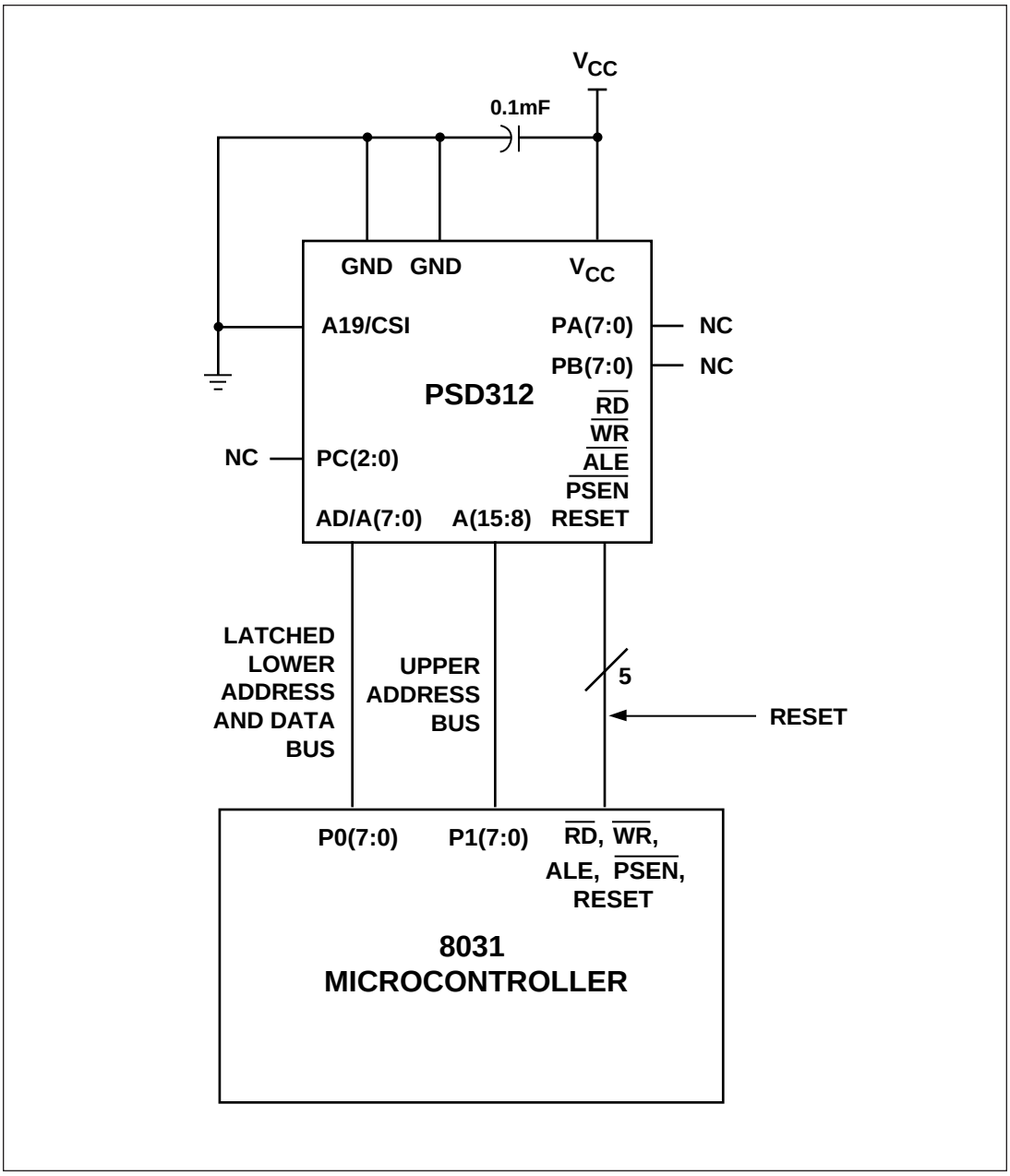


NOTE: Each Block represents one IC package.

Physical Connections

The physical connections for the new board design using the PSD312 are illustrated in Figure 1.2. The multiplexed address/data pins, 0 through 7, from the 80C31 port 0 connect to the PSD312 pins AD/A(7:0). The upper address bits, 8 through 15 from port 1 connect to the pins A(15:8) on the PSD312. The 80C31 write line is connected to "WR.Vpp or R/W", the read line to RD/E, ALE to "ALE or AS". The connections for PSEN and RESET are straightforward. The A19/CSI pin is an unused input in this application so it is tied low. Last of all, the power, grounds and the decoupling capacitor are connected. All other PSD312 pins will remain unconnected. These pins become useful for a more functional design such as in the second example.

Figure 1.2: Standard PSD312 Physical Connections



NOTE: Additional Microcontroller connections are not shown.

PSDsoft Overview

Configuration of the PSD3XX functional blocks is supported by PSDsoft, an integrated system development software tool from WSI that runs on a PC in the Microsoft Windows environment. PSDsoft consists of the following major modules:

PSDabel – The design entry tool used to define the PADs and some I/O constructs.

PSD Configuration – Used to specify the Bus Interface type and I/O port assignments.

PSD Compiler – Combines outputs from PSDabel (logic description), PSD Configuration (bus and port configuration), and MCU Program Code into one .OBJ file to program the PSD part.

PSD Programmer – Provides the programming interface to the WSI MagicPro programmer.

A typical PSD3XX development sequence is listed below:

1. Create or open a project after entering into PSDsoft.
2. Enter the PSDabel Design Entry menu, and describe your circuit using PSDabel Hardware Description Language (HDL).
3. Compile the ABEL source file.
4. Enter the PSD Configuration menu, and configure the Bus Interface and I/O Ports.
5. Enter the PSD Compile menu, and “FIT” the design. The function of the Fitter is to fit the logic functions into the PAD.
6. Select the Address Translate function, also found in the Compile menu. Address Translate combines your MCU program code files (.HEX) and the output of the Fitter into one .OBJ file.
7. Enter the PSDsoft Programmer menu, and Program the PSD3XX by downloading the .OBJ file.

Configuration Data Entry

Following the design sequence described in the previous section, we will implement our 80C31 design example. Figure 1.3 shows some of the PSDsoft windows that would be encountered when starting a new design. Selecting “New Project” opens a dialog box where project management information such as Project Name, Directory Location, Project Description, Device Family, and Part Name are entered.

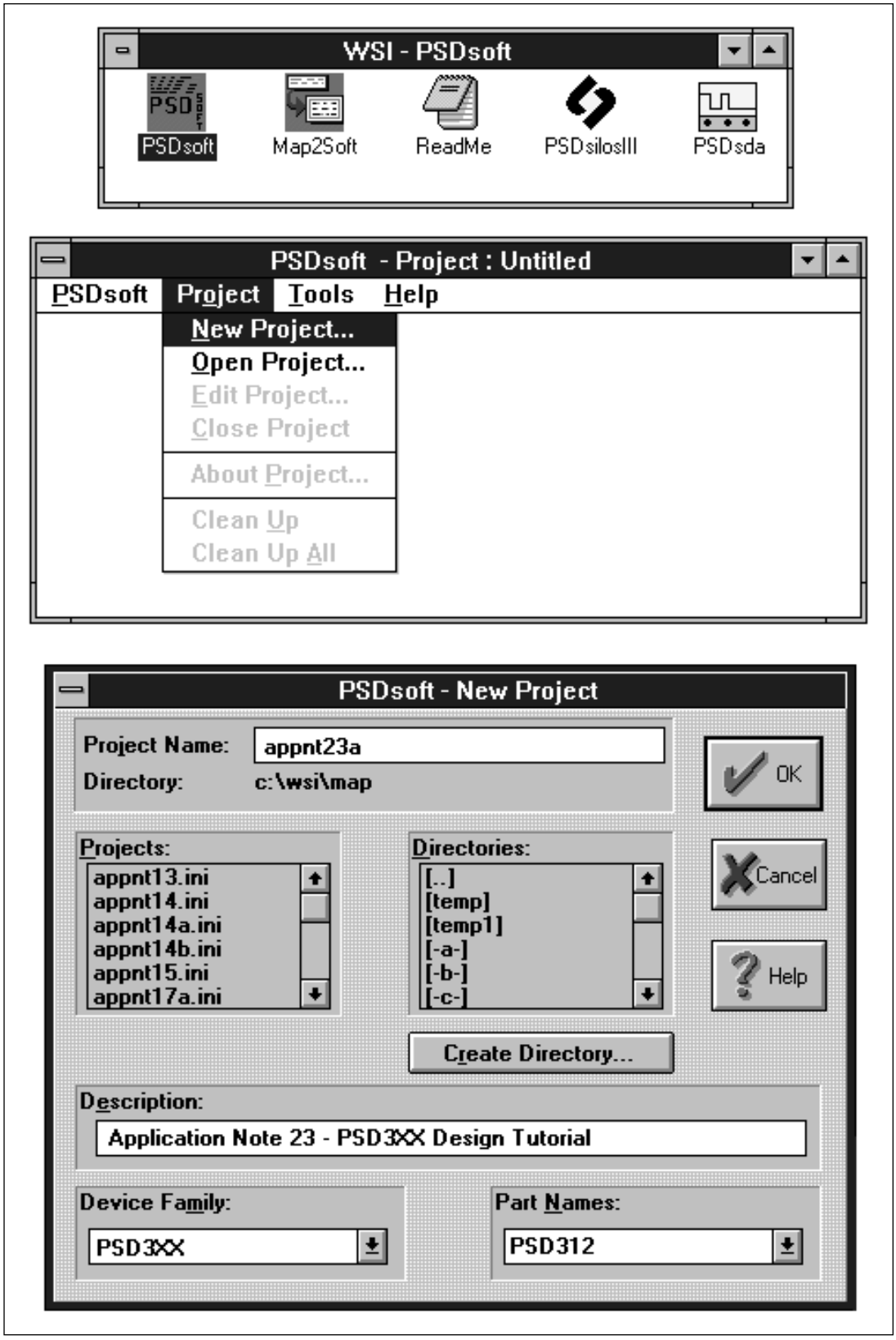
After the project has been defined and the specific PSD part selected, the user would proceed to start the actual logic design by entering the PSDabel portion of PSDsoft. Figure 1.4 shows the ABEL source file created for this example. A typical ABEL file usually consists of five sections:

1. **Header** – A Header consists of a module name and/or title.
2. **Declarations** – Declarations define signals, constants, and macros.
3. **Logic Description** – The logic description defines the PLD functions in terms of equations, truth tables, and state diagrams.
4. **Test Vectors** – The test vectors are used in logic simulation.
5. **End** – An ABEL file must end with an “END” statement.

These sections, minus the test vectors, are easily recognizable in our example source file. Note that Port B and Port C are declared as chip select outputs (CS0 – CS10) even though they are not being used (i.e. chip select logic equations are not specified). This was done to avoid having the terminate these pins if they were configured as inputs, thereby saving board routing space.

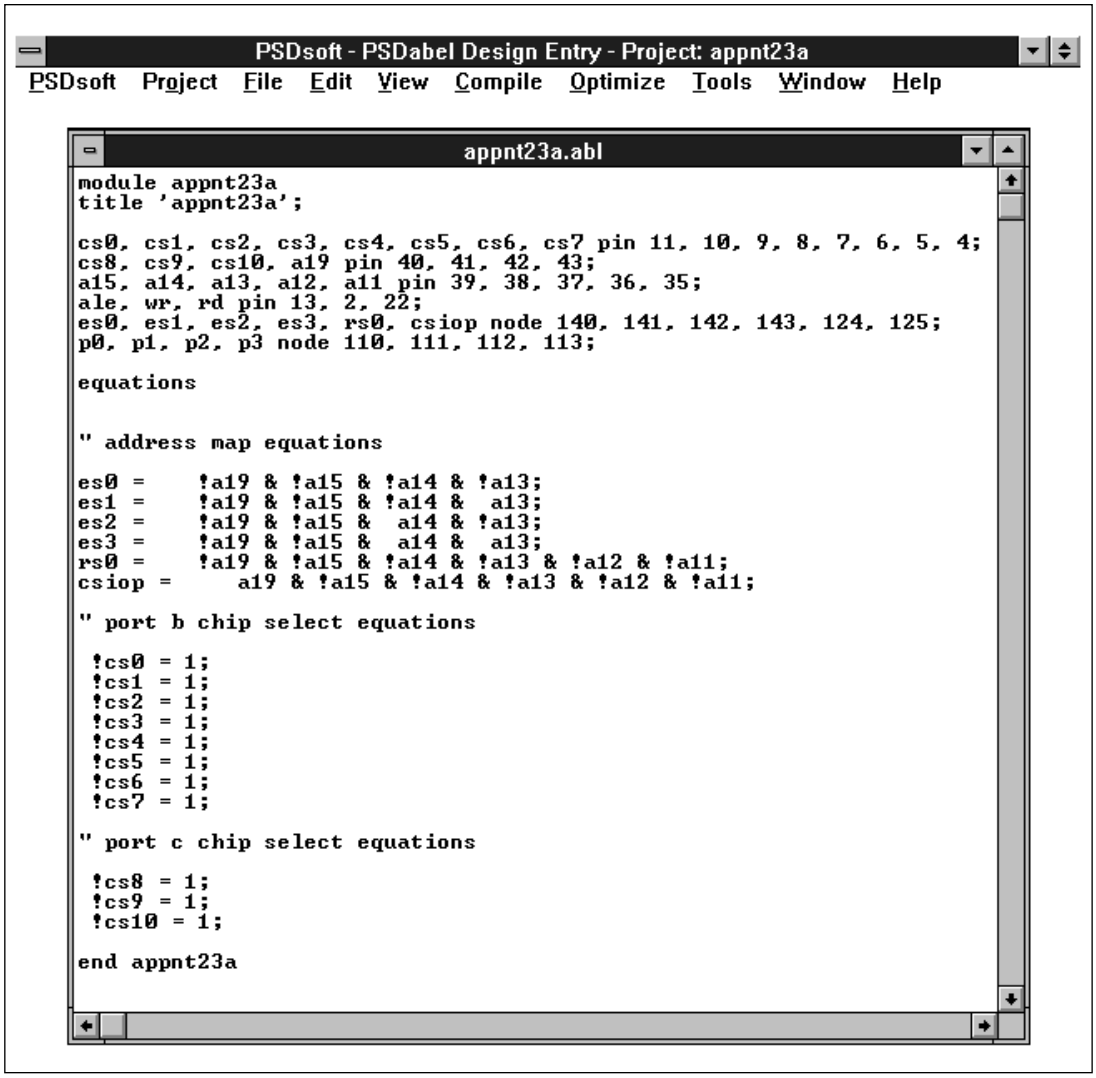
Configuration Data Entry (cont.)

Figure 1.3 PSDsoft Start-up Windows



Configuration Data Entry (cont.)

Figure 1.4 PSDsoft ABEL File

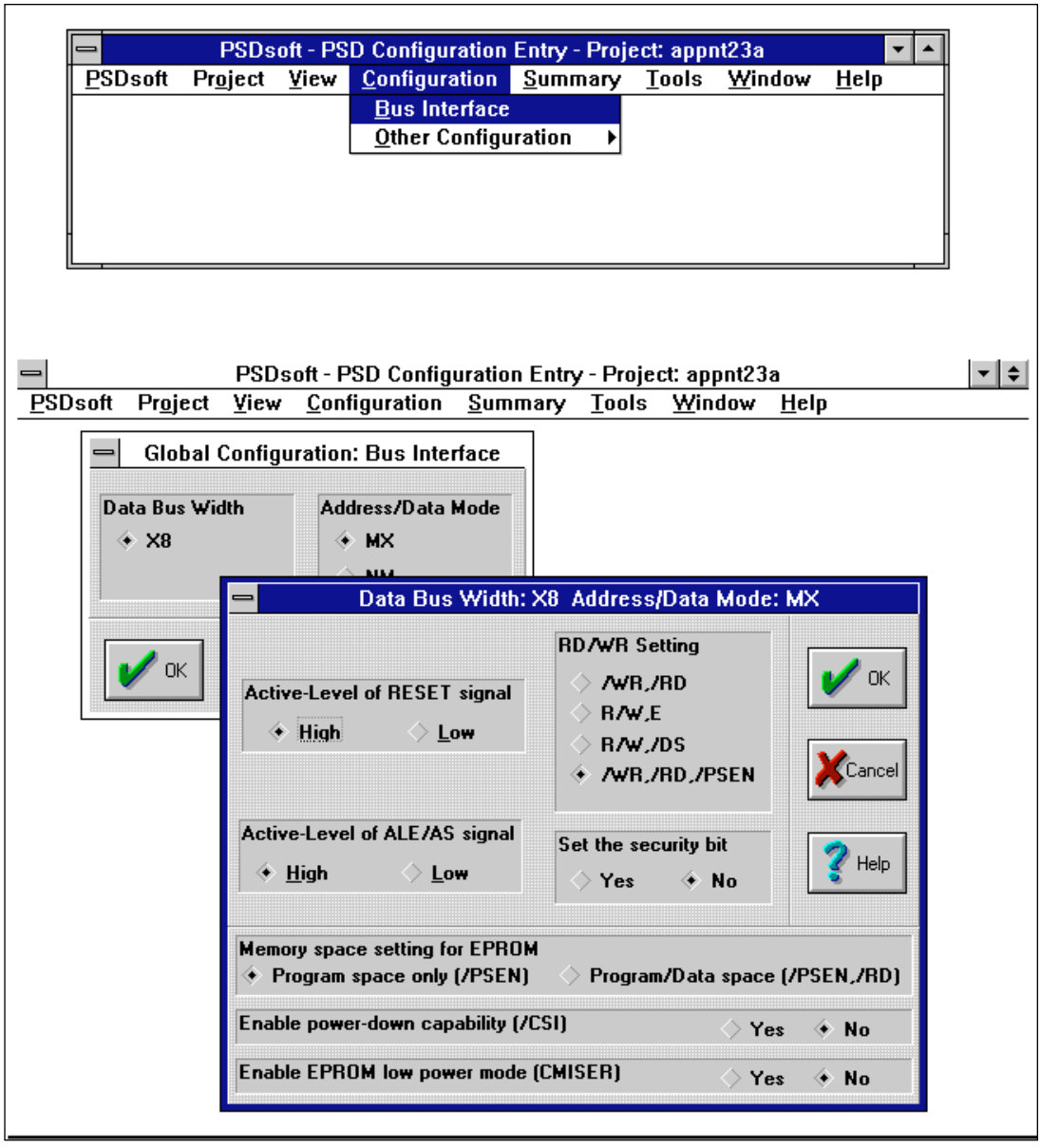


Configuration Data Entry (cont.)

Once the ABEL source file is complete, it should be checked for errors and then compiled using the compile option under the PSDsoft - PSDabel Design Entry window. For additional details on compiling, optimizing, or simulating an ABEL source file, please refer to the PSDsoft - PSDcontrol or PSDabel User Manuals.

The next step in our development sequence is the Bus and I/O Port configuration. Figure 1.5 shows the PSDsoft Configuration Entry windows which are accessed by selecting PSD Configuration from the main window. In our example, the PSD312 bus interface is configured for an 80C31 microcontroller – 8-bit multiplexed address/data bus, reset level is active HIGH, ALE/AS level is active HIGH, $\overline{WR}$, $\overline{RD}$, $\overline{PSEN}$ control mode selected, security bit disabled, EPROM controlled by $\overline{PSEN}$ only, power down feature ($\overline{CSI}$) not used, and low power CMiser feature disabled.

Figure 1.5 PSDsoft Bus Configuration

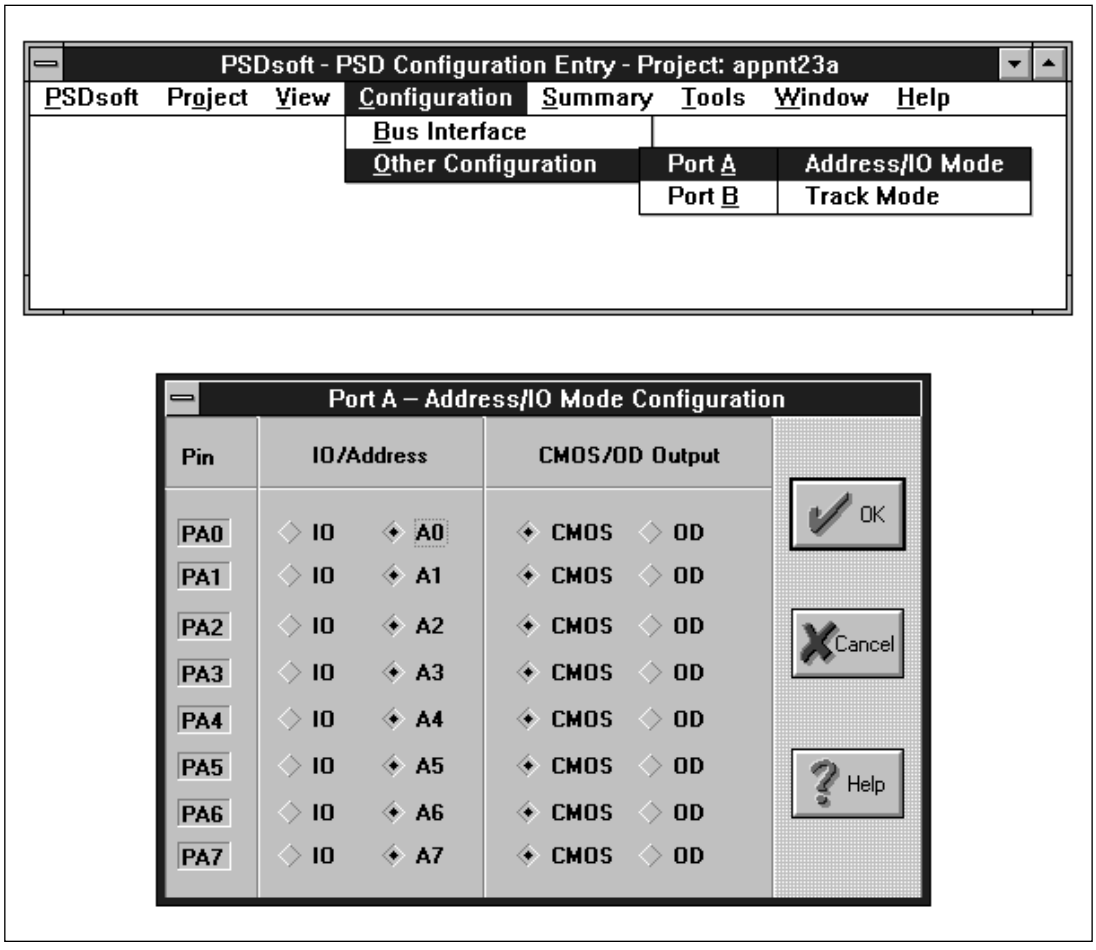


Configuration Data Entry (cont.)

Figure 1.6 shows the options available for configuring Port A. This port will not be used in this design, however the selections are shown to help the user understand Port A functions. One configuration option is Address/I/O Mode. In Address mode, the lower address bits A0 – A7 are brought out on Port A via an internal output latch. In I/O mode mode, Port A functions like a general purpose bi-directional buffer/output latch. In either mode, each pin is individually configurable and are controlled on the fly by special control registers internal to the PSD312 (refer to the PSD Data Book for details on these control registers and other internal configuration bits for the PSD3XX). In our application, Port A is configured in the Address mode with CMOS outputs.

Another mode for Port A is the Track Mode. In this mode, the entire port will track the inputs AD0/A0 – AD7/A7 depending on the specific address ranges defined by the PAD's CSADIN, CSADOUT1, and CSADOUT2 signals. This feature lets the user interface the microcontroller to shared external resources without requiring external buffers and decoders.

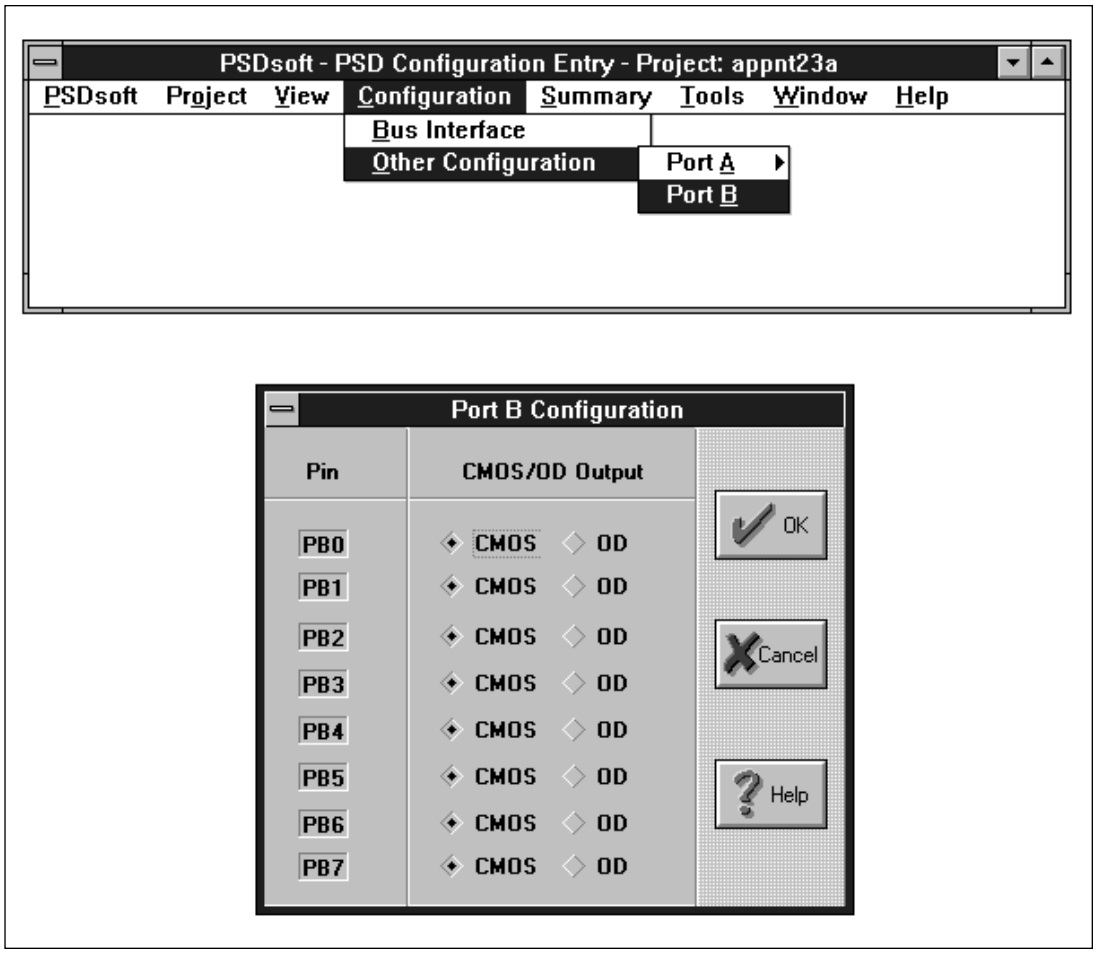
Figure 1.6 PSDsoft Port A Configuration



Configuration Data Entry (cont.)

Figure 1.7 shows the configuration options for Port B. Port B is configured similarly to Port A with the exception of the choice between Address/I/O or Track Mode. This port is a general purpose I/O port that can be used for transferring signals, either chip select outputs from PAD B, or data from the internal data bus. It is also used as the high order data byte in 16-bit, non-multiplexed applications. Since I/O is the default setting for the PSD312, the only parameter that needs to be set is CMOS or OD outputs. If all eight I/Os are not needed, the alternative configuration for the PSD312 Port B pins are chip selects. The CS pins and their associative logic equations would be specified in the ABEL source file as previously discussed. Since Port B will not be used in this example, all of the pins should be configured as outputs for the same reason given for the Port C pins.

Figure 1.7 PSDsoft Port B Configuration

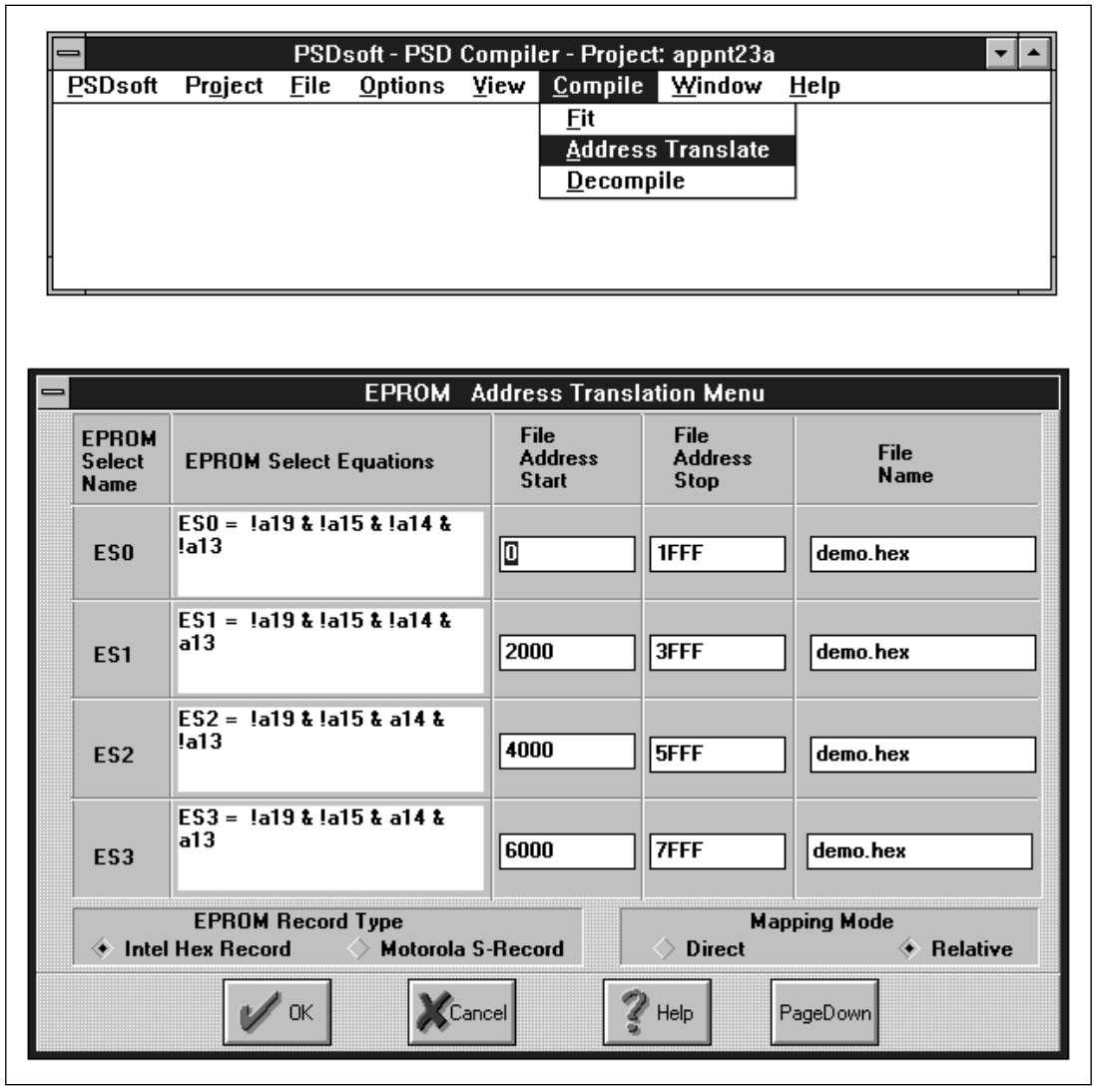


Configuration Data Entry (cont.)

The next step in our development sequence is to compile the design using the “FIT” function, found under the PSDsoft Compile menu. The function of the fitter is to fit the logic functions into the PAD. It merges the outputs from the compiled ABEL file and the configuration file.

Also under the PSDsoft Compile menu is the Address Translator. The Address Translator combines the PAD fusemap file, PSD Configuration, and the EPROM codes file into the .OBJ file which can be downloaded to an EPROM programmer for PSD312 programming. Figure 1.8 shows the windows associated with these functions. The EPROM select equations shown, are automatically entered into this table, taken directly from the ABEL source file. Instead of having one contiguous EPROM space, the PSD312 EPROM is broken up into 8 Kbyte blocks. In this application, the selects for each block are ES0 through ES3. The other selects, ES4 through ES8, will not be used in this example. A space for the individual hexadecimal files to be programmed into the PSD312 EPROM is reserved under the File Address Start, File Address Stop, and File Name sections. These parameters are entered manually by the designer.

Figure 1.8 PSDsoft “FIT” and Address Translate Functions



This completes the design entry portion of our design. Before actually programming the PSD part, the user may want to review and verify the design. A convenient way of doing this is to review some of the reports generated by the compilers. One such report is the Fitter Output Report, shown in Appendix A. This report contains a summary of all the design parameters chosen during the design.

Programming The PSD3XX

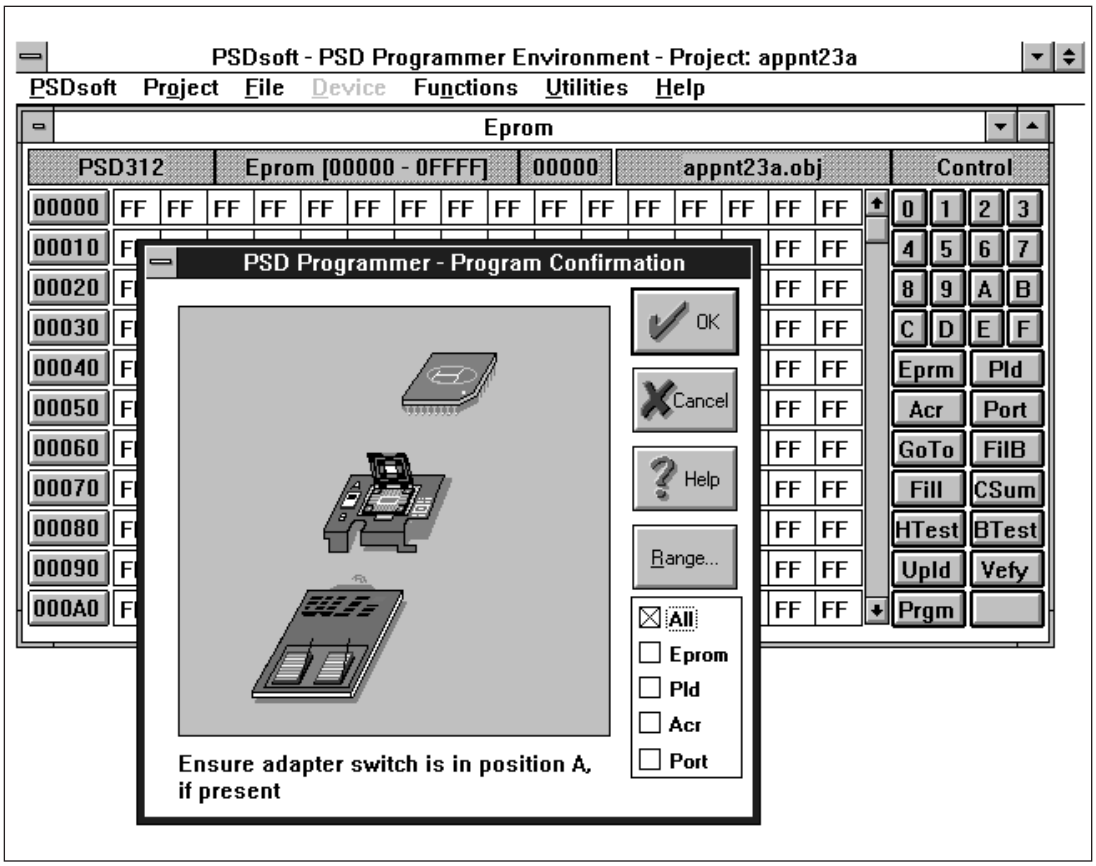
Finally, the last item in our development sequence is to actually program the PSD312 part. The PSD Programmer software, which is contained within PSDsoft, allows you to use either WSI's MagicPro PC compatible programmer or one of many industry standard equivalents to transfer the contents of a .OBJ file into a PSD device. The combined software and hardware allow you not only to program devices but also to perform the following useful functions:

- ❏ **Blank Test**
  Test a device to see if it is blank.
- ❏ **Upload**
  Transfer the contents of a device from the programmer to memory.
- ❏ **Edit**
  Edit the memory locations of the EPROM section of a .OBJ file.
- ❏ **Verify**
  Verify the contents of a device after programming.
- ❏ **Checksums**
  Compute the checksums on the file contents.

Figure 1.9 shows some of the programmer options available under the PSDsoft Programmer Environment.

WSI's PSD devices can be made secure by setting the security bit during the Bus Configuration portion of the design. Make certain that this option is not selected until after the device configuration is programmed. If the security bit is programmed before the configuration, the PSD device will fail the blank test and will require UV erasing.

Figure 1.9 PSDsoft Programmer Window



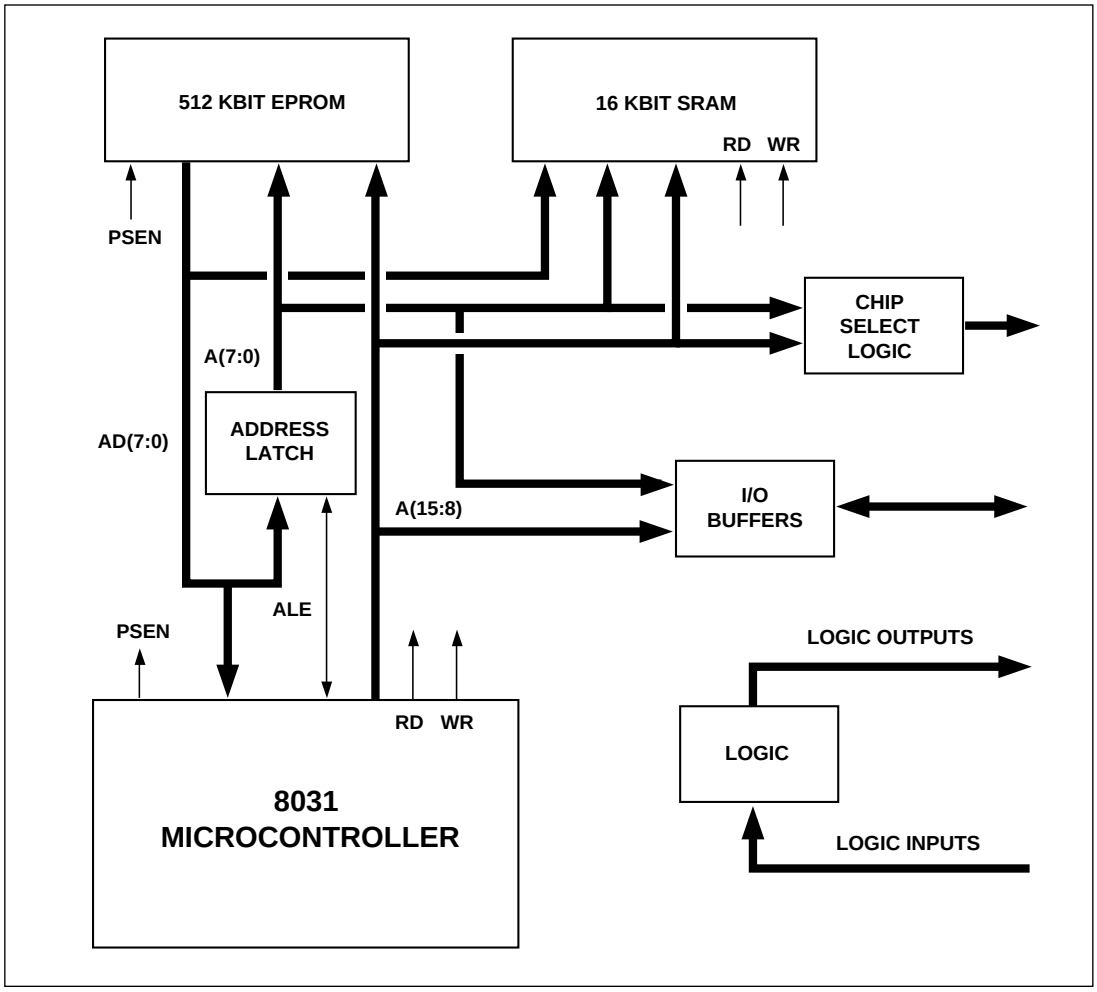
PART II. Advanced PSD3XX Family Design

By now you should have a basic understanding of the PSD3XX device so it is time to introduce some additional features. This example will solve a slightly more complicated design problem. By going through this example you should understand how to use the PSD3XX device to realize functions for your own unique designs.

Assuming the design in Part 1 has been created and saved under a project name (i.e. appnt23a), you can re-open that project to begin this new design. This will keep you from having to enter redundant information. For example, the Bus Configuration options will not require any changes in this design. In this second example, restating the method of navigating through the menus will be avoided for purposes of brevity.

The diagram in Figure 2.1 illustrates the new microcontroller board design to be replaced by a design using the PSD312. In addition to functions previously replaced in the standard board design, this board includes chip select logic, I/O buffers, and a logic chip. The logic chip does not perform microcontroller functions but is included to show the flexibility of the PSD312. An additional change is there is now an SRAM chip select which is an input to the board. Its logic select function is defined elsewhere so it does not need to be recreated in the PSD312's chip select logic. This scenario would occur when some other device has a separate SRAM of its own. This other device will decide whether the microcontroller writes to the PSD312 SRAM or its own SRAM. The configuration of this design is mostly the same as in Part 1. We are now using PORT A as an I/O buffer, all PORT C pins are now used as logic inputs, and we must specify the chip selects in PORT B to conform to our logic and chip selects.

Figure 2.1. Advanced 8031 Microcontroller System Block Diagram

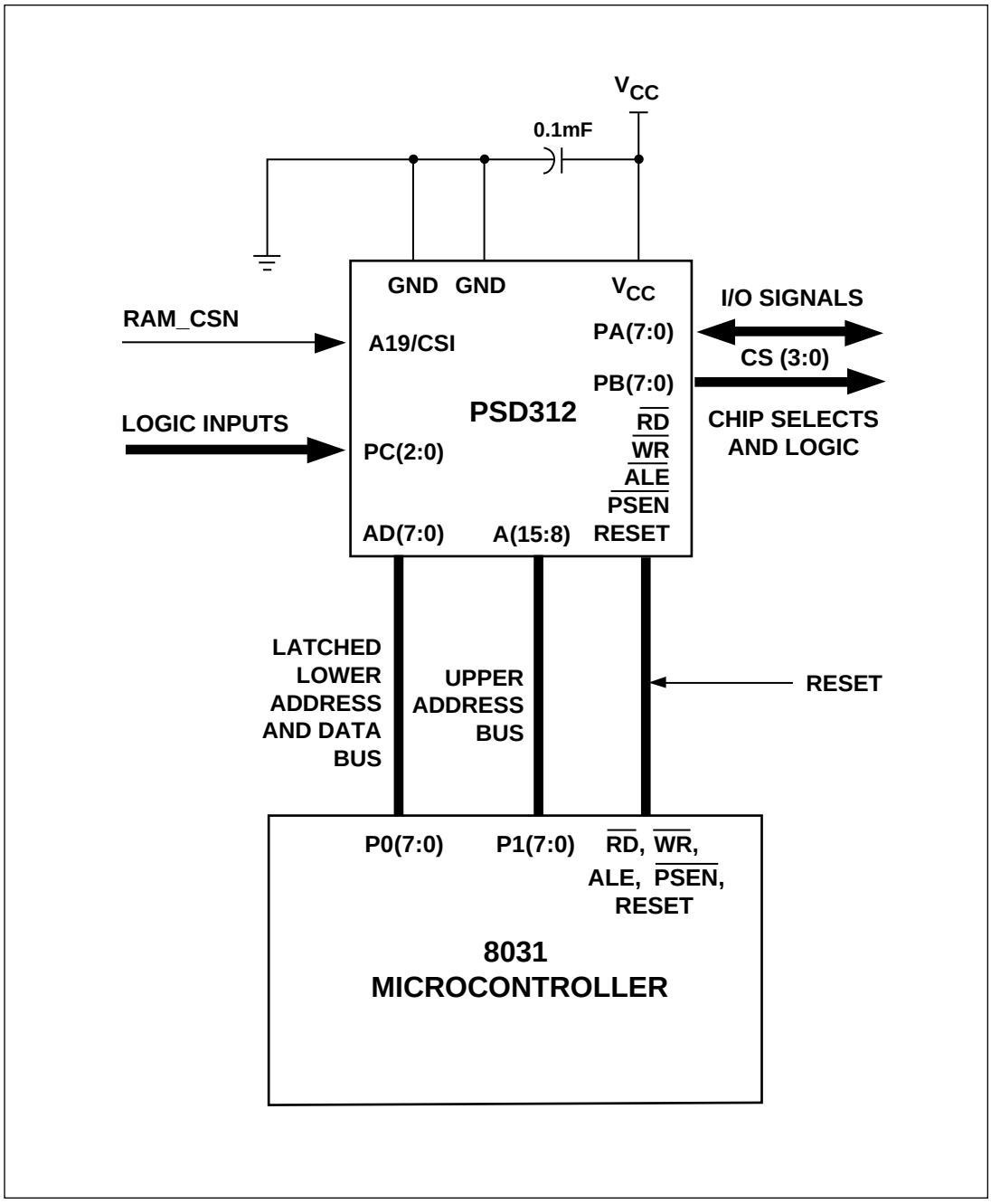


NOTE: Each Block represents one IC package.

Advanced PSD3XX Family Design (Cont.)

Figure 2.2 illustrates the physical connections to the PSD312. The SRAM chip select is input to A19 (A19/CSI), and the logic inputs are input through A(18:16). These are arbitrary assignments among the address inputs. PORT C could not be used for chip select outputs because we needed to make room for the inputs.

Figure 2.2. Advanced Design PSD312/8031 Physical Connections



NOTE: Additional Microcontroller connections are not shown.

Advanced PSD3XX Family Design (Cont.)

Figure 2.3 shows the new ABEL file for this example. Notice that Port C pins 40, 41, and 42 have been declared as inputs A16, A17, and A18 instead of chip select outputs CS8, CS9, and CS10 as in example one. Although the names of the inputs are addresses, they can be used as either logic or address inputs.

Figure 2.3. PSDsoft ABEL File – Example II

PSDsoft - PSDabel Design Entry - Project: appnt23b - [appnt23b.abl]

PSDsoft Project File Edit View Compile Optimize Tools Window Help

```
module appnt23b
title 'appnt23b';

cs0, cs1, cs2, cs3, cs4, cs5, cs6, cs7 pin 11, 10, 9, 8, 7, 6, 5, 4;
a16, a17, a18, a19 pin 40, 41, 42, 43;
a15, a14, a13, a12, a11 pin 39, 38, 37, 36, 35;
ale, wr, rd pin 13, 2, 22;
es0, es1, es2, es3, rs0, csiop node 140, 141, 142, 143, 124, 125;
p0, p1, p2, p3 node 110, 111, 112, 113;

equations

" address map equations

es0 =    !a15 & !a14 & !a13;
es1 =    !a15 & !a14 &  a13;
es2 =    !a15 &  a14 & !a13;
es3 =    !a15 &  a14 &  a13;
rs0 =    !a19 & !a15 & !a14 & !a13 & !a12 & !a11;
csiop =     a19 & !a18 & !a17 & !a16 & !a15 & !a14 & !a13 & !a12 & !a11;

" port b chip select equations

 !cs0 =    !a18 & !a17 & !a16 #
          !a18 & a17 & a16 #
          a18 & a17 & !a16;
 !cs1 =    a15 & a14 & !a13 & a12;
 !cs2 =    a15 & a14 & !a13 & !a12 & !a11;
 !cs3 =    a15 & a14 & !a13 & !a12 & a11;
 !cs4 = 1;
 !cs5 = 1;
 !cs6 = 1;
 !cs7 = 1;
end appnt23b
```

Advanced PSD3XX Family Design (Cont.)

The I/O signals connect to the PSD312 through Port A. The only configuration change from example one therefore would be to select I/O instead of A0 – A7 in the Port A configuration menu (refer back to Figure 1.6 of example one).

Port B is used for the chip select and logic outputs. The configuration remains the same as in example one (refer to Figure 1.7), however in this example the chip select equations for CS0, CS1, CS2, and CS3 are defined in the ABEL file as shown in Figure 2.3. Table 2.1 describes the function of the logic chip that will be implemented by the PSD's internal PAD (inputs A, B, C = A18, A17, A16; output = CS0). Chip selects CS1, CS2, and CS3 are used to enable components on other boards when their address ranges, as shown in Table 2.2, are encountered.

Table 2.1: Logic Truth Table

INPUTS			OUTPUT CONDITION
A	B	C	(CS0)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Table 2.2: Chip Select Address Map

CHIP SELECT	ADDRESS RANGE	ADDRESS BITS				
		15	14	13	12	11
CS1	D000 – DFFF	1	1	0	1	X
CS2	C000 – C7FF	1	1	0	0	0
CS3	C800 – CFFF	1	1	0	0	1

Referring to the address map equations shown in the ABEL source file for this example (Figure 2.3), notice that the SRAM chip select (RS0) is enabled (active low) by the external signal A19. This was done simply to allow the PSD's SRAM to be disabled so that another SRAM, external to the PSD, could be enabled and addressed by the microcontroller without conflicting with the PSD's internal SRAM. Also notice that A19 is not necessary in the logic equations that select the EPROM (ES0 – ES3) in this design example. This is because the EPROM and SRAM are controlled by the separate read strobes, PSEN and RD, and therefore can share the same address space.

The CSIOP signal is the base address for the I/O functions. Since it is also controlled by the the RD signal, it must have a different address range than the SRAM. In this example, its address range is from 80000 to 807FF (the actual microcontroller address is 0000 to 07FF with the upper address bit, A19, being provided externally).

Conclusion

A PSD3XX family device can implement many common microcontroller functions and it is flexible enough to be used on designs requiring special functions. Its use will reduce the component count, layout complexity, size, component cost, PCB cost, and power consumption of a design. Reliability is increased due to the reduced chip count. The risk of board redesign is minimal given the ease of design and the PSD3XX device's flexibility. The user friendly software makes it easy to use in any design.

Appendix A. PSDsoft Fitter Output Report

```
************************************************************
                W S I - PSDsoft Version 2.12
                    Output of PSD Fitter
************************************************************
  TITLE      : appnt23a
  PROJECT    : appnt23a                        DATE :
  DEVICE     : PSD312                          TIME :
  FIT OPTION : Keep Current
  DESCRIPTION:
************************************************************
```

==== Pin Layout for PLDCC/CLDCC Package Type ====

		----------			
psen	1]	psen	adio0 [23	Address/Data Bus ADIO_0
wr	2]	wr	adio1 [24	Address/Data Bus ADIO_1
reset	3]	reset	adio2 [25	Address/Data Bus ADIO_2
CS7	4]	pb7	adio3 [26	Address/Data Bus ADIO_3
CS6	5]	pb6	adio4 [27	Address/Data Bus ADIO_4
CS5	6]	pb5	adio5 [28	Address/Data Bus ADIO_5
CS4	7]	pb4	adio6 [29	Address/Data Bus ADIO_6
CS3	8]	pb3	adio7 [30	Address/Data Bus ADIO_7
CS2	9]	pb2	adio8 [31	Address Bus ADIO_8
CS1	10]	pb1	adio9 [32	Address Bus ADIO_9
cs0	11]	pb0	adio10 [33	Address Bus ADIO_10
	12]	GND	GND [34	
ale	13]	ale	adio11 [35	Address Bus ADIO_11 (a11)
Address Line A7	14]	pa7	adio12 [36	Address Bus ADIO_12 (a12)
Address Line A6	15]	pa6	adio13 [37	Address Bus ADIO_13 (a13)
Address Line A5	16]	pa5	adio14 [38	Address Bus ADIO_14 (a14)
Address Line A4	17]	pa4	adio15 [39	Address Bus ADIO_15 (a15)
Address Line A3	18]	pa3	pc0 [40	cs8
Address Line A2	19]	pa2	pc1 [41	cs9
Address Line A1	20]	pa1	pc2 [42	cs10
Address Line A0	21]	pa0	a19/csi [43	a19
rd	22]	rd	VCC [44	

Appendix A. PSDsoft Fitter Output Report (cont.)

==== Global Configuration ====

Data Bus :	8-bit Multiplexed
Reset Polarity :	HIGH
ALE/AS Signal :	ACTIVE HIGH
Security Protection :	OFF
Power-down capability (/CSI) :	Not Used
EPROM low power mode (CMISER) :	DISABLE
Track Mode :	OFF

==== Other Configuration ===

Port A :

Pin	IO/Address	CMOS/OD Output
PA0	Address	CMOS
PA1	Address	CMOS
PA2	Address	CMOS
PA3	Address	CMOS
PA4	Address	CMOS
PA5	Address	CMOS
PA6	Address	CMOS
PA7	Address	CMOS

Port B :

Pin	IO/Chip Select Output	CMOS/OD Output
PB0	Chip Select Output	CMOS
PB1	Chip Select Output	CMOS
PB2	Chip Select Output	CMOS
PB3	Chip Select Output	CMOS
PB4	Chip Select Output	CMOS
PB5	Chip Select Output	CMOS
PB6	Chip Select Output	CMOS
PB7	Chip Select Output	CMOS

Port C :

Pin	Input/Output	Address/Logic
PC0	Output	
PC1	Output	
PC2	Output	

==== Address & Data Bus Assignment ====

Stimulus Bus Name	Signal Description
`adiol = adio[7:0]	= Address/Data Bus ADIO_7 - ADIO_0
`adioh = adio[15:8]	= Address Bus ADIO_15 - ADIO_8
adio = adio[15:0]	= Address/Data Bus ADIO_15 - ADIO_0

Appendix A. PSDsoft Fitter Output Report (cont.)

```
                    ===== Resource Usage Summary =====

Device Resources                          used / total  Percentage
-------------------------------------------------------------------

Port A: (pin 14 - pin 21)
   I/O Pins                               8 /  8        100 %
     MCU I/O                              0 /  8          0 %
     Address Out                          8 /  8        100 %
     Data Port (Non-Mux Bus)              0 /  8          0 %
     Track Mode                           0 /  8          0 %

Port B: (pin 4 - pin 11)
   I/O Pins                               8 /  8        100 %
     MCU I/O                              0 /  8          0 %
     PLD Output                           8 /  8        100 %
     Data Port (16 bit Non-Mux Bus)       0 /  8          0 %

Port C: (pin 40 - pin 42)
   I/O Pins                               3 /  3        100 %
     PLD Input                            0 /  3          0 %
     PLD Output                           3 /  3        100 %


                    ========= Equations =========

DPLD EQUATIONS :
================================================
 es0   = !a19 & !a15 & !a14 & !a13;
 es1   = !a19 & !a15 & !a14 & a13;
 es2   = !a19 & !a15 & a14 & !a13;
 es3   = !a19 & !a15 & a14 & a13;
 rs0   = !a19 & !a15 & !a14 & !a13 & !a12 & !a11;
 csiop = a19 & !a15 & !a14 & !a13 & !a12 & !a11;

PORT B EQUATIONS :
==================
!cs0  = 1;
!cs1  = 1;
!cs2  = 1;
!cs3  = 1;
!cs4  = 1;
!cs5  = 1;
!cs6  = 1;
!cs7  = 1;

PORT C EQUATIONS :
==================
!cs8  = 1;
!cs9  = 1;
!cs10 = 1;
```