AN1422
APPLICATION NOTE

Designing with Flash+PSD Memory

CONTENTS

(Introduction on next page)

Programmable Peripheral
Application Note 048
Designing with Flash Memory

Introduction

Flash Memory has gained a wide popularity as a choice of use in embedded system solutions. Many designers, who are using Microcontrollers, are using Flash Memory for program store and sometimes also for data store. The ability to reprogram Flash Memory in the system has allowed designers to change the program code in production or out in the field without a large expense of doing so. What many designers are not aware of is the complexity of designing with Flash Memory for In-System Programming (ISP). One important point to remember when reprogramming Flash Memory: the programming algorithm cannot be executed out of the same Flash Memory device that is being programmed. It must be executed out of a separate memory device such as an EPROM or SRAM device. At the time of this publication, there are no cost effective Flash Memory devices that can perform this dual function. This Application Note will address the issues of designing Flash Memory into a system and show an easy solution using ST's PSD device as a Flash Support Chip (FSP). The next few sections discuss the design issues of using Flash Memory. A summary of these issues are shown in Table 1.

Reason for Using Flash Memory

The two main reasons why designers use Flash Memory is to have the ability to cost effectively change the program code in the field, and to load the latest version of program code into the system just prior to shipping the product. Other designers may also use segmented Flash Memory to store variables or calibration data. Flash Memory is also used in manufacturing for running test programs. This can help decrease the cost of manufacturing a product.

Methods of Programming

The designer must determine the method of programming a Flash Memory the first time and reprogramming the Flash Memory in-system after the first time. The technique of performing these two functions may be very different.

The Flash Memory can be initially programmed by an EPROM programmer, or programmed in the system using a Boot EPROM or ROM, or programmed in the system using a Microcontroller with a JTAG Port. If the system boots up out of the boot sector of the Flash Memory, this boot sector must be initially programmed either by an external EPROM programmer prior to the Flash Memory being assembled on the board or by an MCU with a JTAG Port. Since most low cost Microcontrollers do not have a JTAG Port, there is an added cost of programming Flash Memory on an EPROM Programmer prior to being assembled on the board.

When reprogramming the Flash Memory in-circuit, a serial port or JTAG port must be used. If a serial port is used, the data is downloaded to SRAM prior to being programmed into the Flash Memory. Next, the Microcontroller must execute the programming algorithms to reprogram the Flash Memory with the data that was downloaded from the serial port. If the system does not have a separate boot EPROM or ROM to execute the programming algorithms, the programming algorithms must be copied from the boot sector of the Flash Memory into the SRAM. The programming algorithms must then be executed from the SRAM while programming the Flash Memory. A common mistake designers make is thinking that they can execute the programming algorithms out of the same Flash Memory they are trying to program.

Methods of Programming (cont.)

If a JTAG port is available on the MCU, this port can be used to reprogram the Flash Memory in the same manner as programming the Flash Memory the first time. The boot code can be downloaded through the JTAG port into local SRAM inside the Microcontroller while the Microcontroller is being held in reset. The boundary scan registers on the Microcontroller can also be used to program the Flash Memory in-circuit. Since most low cost Microcontrollers do not have a JTAG Port, further discussions in this application note will assume that a JTAG Port is not available.

Manufacturing Stages of Programming

Flash Memory can be programmed in various stages of the manufacturing process. If no boot EPROM or ROM is used in the system, the Flash Memory must be programmed on an EPROM Programmer prior to being assembled onto the board. As a minimum, the boot sector must be programmed at this time. Manufacturing personnel must be careful not to assemble the wrong programmed part on the board. This mistake may be costly as it will be detected in Functional Test where test time and troubleshooting is expensive. The program code may also change prior to shipping the product. If this occurs, programming at this stage will be an unnecessary cost added to manufacturing. During in-circuit test, test code may be loaded into Flash Memory. This may simplify the testability of the system. Functional Test is the most expensive stage for detecting a bad or incorrectly programmed Flash Memory. From a manufacturing point of view, the least expensive Flash solution is having the ability to assemble a blank Flash Memory device on the board and program it during the in-circuit test phase. This requires a separate boot EPROM or ROM.

System Programming Issues

There are various system programming issues that must be considered when designing Flash Memory systems. The most important issue was stated previously: the system cannot execute the programming algorithms from the same Flash Memory being programmed. Some designers will store the programming algorithms in the boot sector of the Flash Memory. When programming the Flash Memory, the system must copy the algorithms from Flash to SRAM. The system will next execute the algorithms from SRAM to program the Flash Memory.

Taking a closer look at what is involved in doing this, I will use an 80C31 for an example. The 80C31 has a separate Program Space and Data Space. Initially the Flash Memory is in the Program Space and the SRAM is in the Data Space. To copy the algorithms from Flash Memory to SRAM, the Flash Memory and SRAM must be in the Data Space. The code executing in the Flash Memory to copy the algorithms must be in the Program Space. Once the algorithms are in the SRAM, the SRAM must be moved to the Program Space while the Flash Memory remains in the Data Space. To work around this problem, both the Flash Memory and SRAM must be in both the Program and Data Space at the same time. This can be a problem if the sum of memory sizes of the Flash Memory, SRAM, and other I/O functions exceeds the 64 Kbytes of address space. The solution to this problem requires a lot of PLD logic for address decoding in both the Operational and Programming Modes along with a page register to extend the addressing capabilities of the 80C31 in the Programming Mode.

There are other concerns the designer must consider. What if power fails while the Microcontroller is executing the programming algorithm from SRAM? The programming state machine built into Flash Memories must reset itself and allow access to the boot sector when powering the system back up. The system then must have the ability to reprogram the corrupted Flash Memory. The boot block in the system must be write protected. If this boot block is corrupted for any reason, the system will be inoperative. The best protection for the boot block and programming algorithms is to store them in EPROM or ROM technology. If there is a way to reprogram the boot block, there is a way to corrupt it. Some sector protect techniques cannot guarantee that the sector will not be corrupted when the Microcontroller malfunctions and executes the wrong code (maybe unlocking and corrupting a sector).

System Programming Issues (cont.)

Table 1. Programming Flash Memory

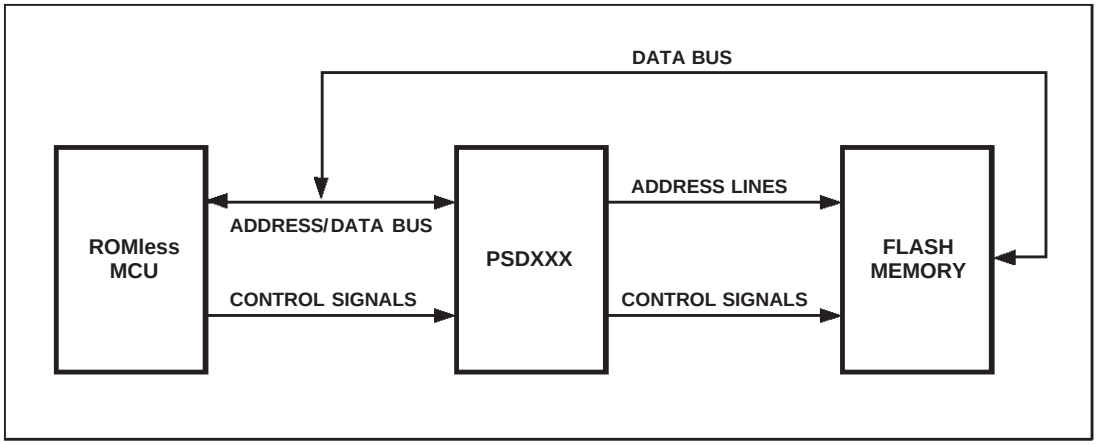
Using SRAM for Executing Programming Algorithms	Using the PSDXXX for Executing Programming Algorithms and Integrating Other Support Logic
Need to program Flash Memory the first time on a PROM Programmer.	Program one PSD device vs. programming Flash Memory and PLDs on PROM Programmer. Blank Flash Memory can be assembled on board.
Requires MCU to down load the programming algorithms from Flash Memory to SRAM and execute from SRAM to program the Flash Memory. The MCU must execute code from a memory device while down loading the programming algorithms to SRAM	Execute programming algorithm from PSD's internal EPROM. There is no requirement to down load the programming algorithm to SRAM
Must re-map the SRAM from Data Space to Code Space to execute the Programming Algorithms. May require additional PLD Logic (when using an 80C31).	No need to reconfigure SRAM.
Need PLD Logic for changing the memory map to access Flash Memory in both Program and Data Space (when using an 80C31).	Required PLD Logic is built into the PSD device.
Need PLD Logic for Address Decoding.	Built-in PLD Logic for Address Decoding.
Need PLD Logic for Paging (if exceeding addressing capability of MCU).	Built-in Page Register.
Need Address Latch (for multiplexed MCUs).	Integrates Address Latch.
Boot block in Flash Memory is at risk of being corrupted.	EPROM based boot block cannot be corrupted.
Longer development time. More complex design for defining Address Map and PLD PLD Logic functions to handle both Flash Memory and SRAM in different modes of Operation.	Partitioned design (MCU+PSD+Flash), easy to develop and debug. One PSD device integrates all logic functions and Boot EPROM. Simple Abel design entry for defining memory map and PLD logic. The PSD device is reprogrammable.
Larger size PCB required.	Integrated solution requiring smaller PCB and less traces. ZPSD device is a lower overall power solution with higher reliability.

The PSD Flash Solution

As shown in Figure 1, the basic blocks of a simple partitioned design consists of a ROMless Microcontroller, a PSD device, and a Flash Memory. The Microcontroller interfaces directly to the PSD device. The Flash Memory interfaces to the PSD device through the PSD's I/O ports. The system boots up out of the PSD's EPROM. This boot code should be able to bring the system up to a mode of operation that includes the ability to program the Flash Memory. This includes operation of a serial port to download data into local SRAM. The boot code will also interrogate the Flash Memory to determine which Flash Memory vendor is being used. Since each Flash Memory vendor uses a different programming algorithm, all algorithms will be stored in the PSD's EPROM. This will allow the designer to use the least expensive pin compatible Flash Memory.

The PSD's EPROM will store the boot code along with the Flash Memory programming algorithms for all types of Flash Memories being used in production. The PSD's General PLD (GPLD) will be used for address decoding to generate the Flash Memory Chip Selects, to generate upper Flash Memory address lines needed for paging, and serve as additional PLD logic required in the system. The PSD's I/O Ports will be used to latch the address lines when a Microcontroller is used with a multiplexed address/data bus. These I/O Ports will also be used for other port expansion required in the system. The built-in Page Register (not available on the PSD3X1 so use the I/O ports instead) will be used to extend the addressing capabilities of the Microcontroller. When using an 80C31, one bit of the Page Register will be used to change the address map from an Operational Mode to a Programming Mode.

Figure 1. Flash Support Chip Solution



80C31 Example

Examples of implementing a Flash Memory with an 80C31 will be shown with both a PSD311 and a PSD411A1. In these examples, an 80C31 is being used with 128 Kbytes of external Flash Memory and 32 Kbytes of SRAM. The memory map during the normal Operational Mode is show in Figure 2. In the Program Space, the PSD's EPROM containing the boot code and programming algorithms along with a portion of Flash Memory are common to all pages of memory. The rest of the Flash Memory is spread over three pages (or banks) of memory. The Data Space is the same for all pages of memory. When programming the Flash Memory, a bit is set in the Page Register to switch the memory map from an Operational Mode to a Programming Mode as shown in Figure 3. The PSD's EPROM that is executing the programming algorithms is still located into the Program Space while the Flash Memory has moved into the Data Space. The 32 Kbytes of SRAM is disabled since the Flash Memory is mapped over it. The external chip selects and the PSD's SRAM and I/O Ports are unaffected and can be accessed on any memory page. While executing from the PSD's EPROM, the system downloads the data from an external Serial Port to the PSD's SRAM. The data is then programmed into the Flash Memory. Note that the chip select to the Flash Memory should cover the programming command addresses (for example, the AMD29010 programming command addresses are 2AAAH and 5555H with A15 = Don't Care). See the Flash Memory vendor's data book for more details.

80C31 Example (cont.)

Figure 2. Memory Map in Operational Mode with 1 Mbit Flash Memory

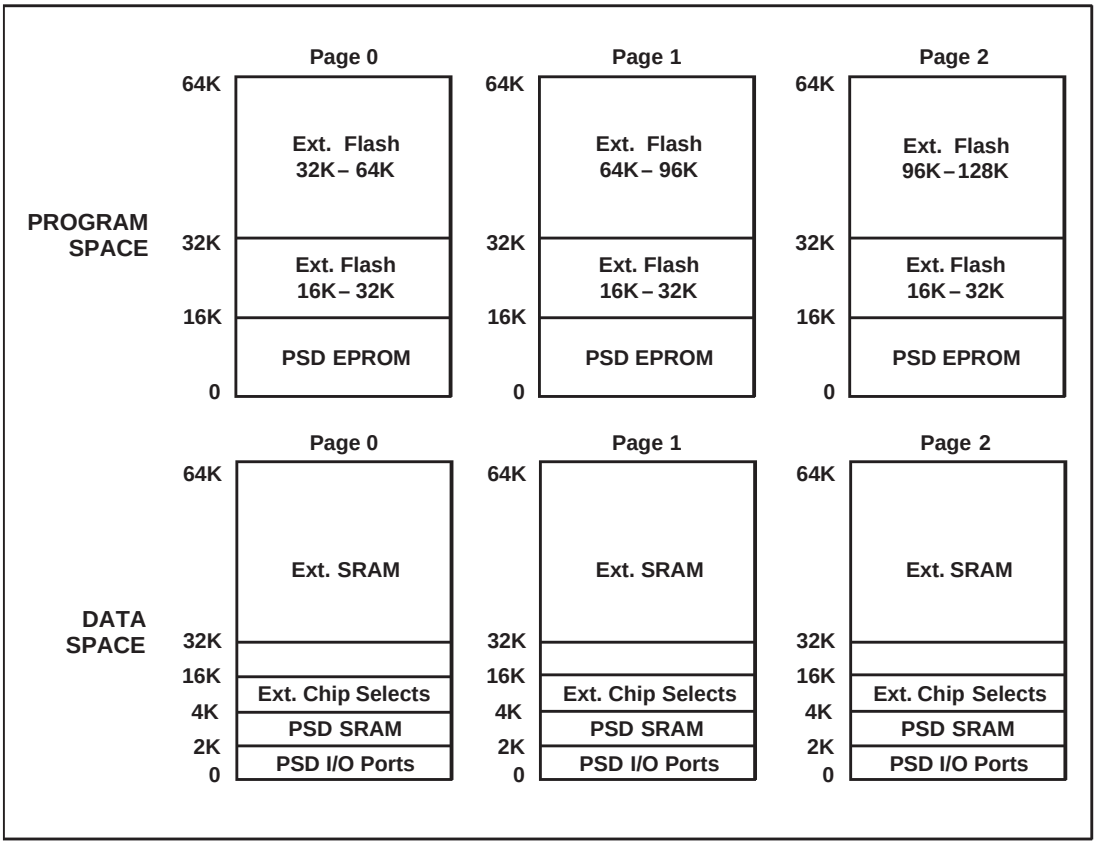
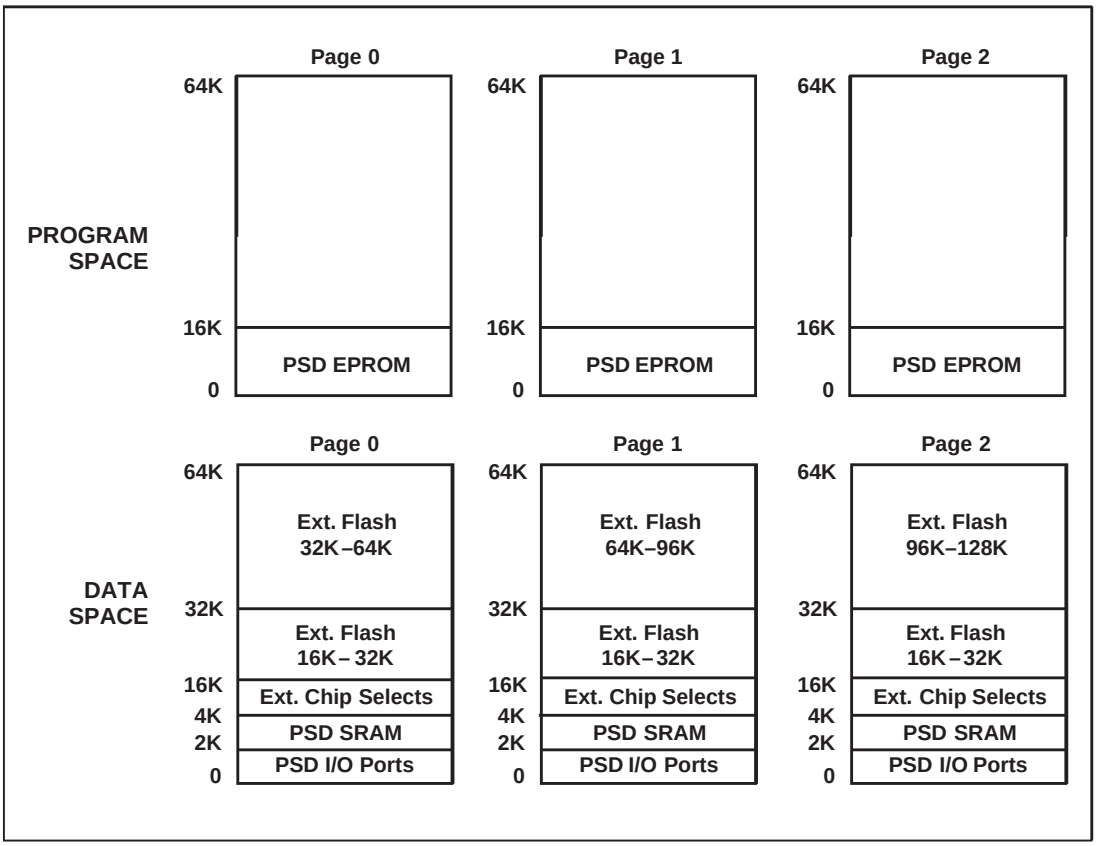


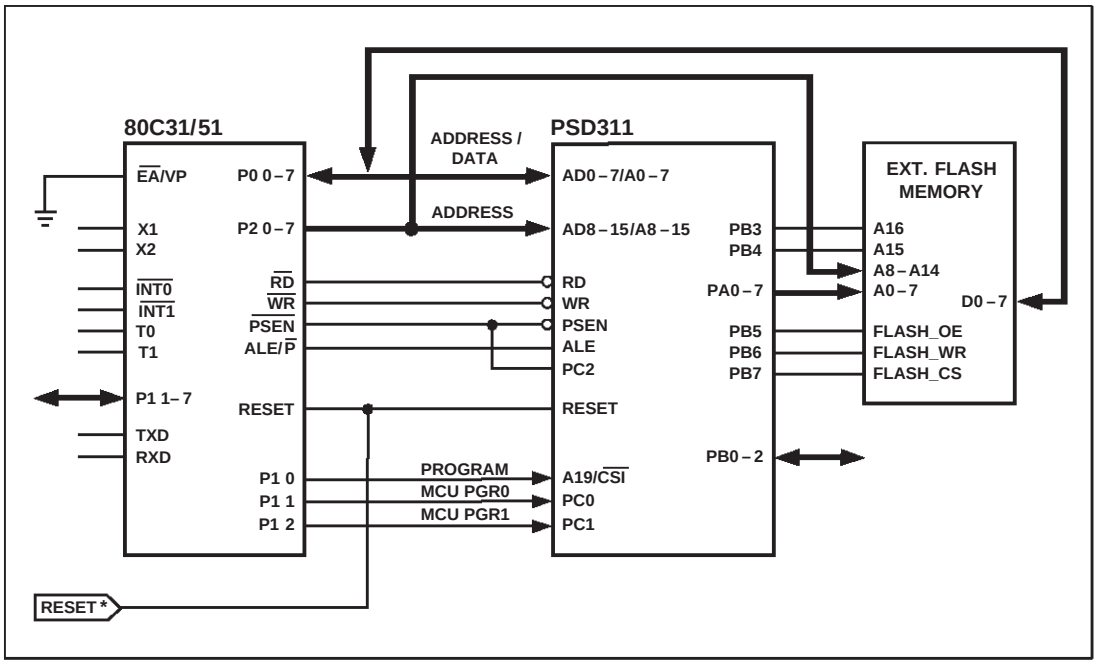
Figure 3. Memory Map in Programming Mode with 1 Mbit Flash Memory



80C31 Example (cont.)

The detailed solution using a PSD311 is shown in Figure 4. The required logic equations are shown in Table 2. When the 80C31 writes a logical "1" to the Program Bit (Port 1, bit 0 on the 80C31), the memory map is switched from the Operational Mode to the Programming Mode as show in Figures 2 and 3. The latched address (A0–7) appears on Port A. Port B provides the control signals along with the upper address lines for the Flash Memory. Controlling the upper two address lines of the Flash Memory allows the PSD to map each 16 Kbytes of Flash Memory anywhere in the 64 Kbyte address space and on any page of the memory map. Since the PSD311 does not have a built-in page register (the PSD312 device has a 4-bit page register), three I/O port pins on the 80C31 are used as a 3-bit page register. These signals are routed into the PLD inside the PSD311. Another way to implement the page register in the PSD311 is to use PB0–2 as output pins instead of using the I/O pins on the 80C31. These pins are routed back to the PSD311 through Port C and A19 in the same manner as shown in Figure 4. Note that the PSEN signal is connected to PC2. This will route the PSEN signal into PAD B (or GPLD) for external chip selects. The PSDabel file along with the Fitter Report are shown in Appendix A.

Figure 4. PSD311 with Extemal Flash Memory



*RESET Signal. The external reset must have a fast rise time from a logic gate, not a slow rise time generated from an RC circuit.

Table 2. Logic Equations for the PSD311

DPLD Equations

```
rs0   = (address > = ^h0800) & (address < = ^h0FFF);
es0   = (address > = ^h0000) & (address < = ^h0FFF);
es1   = (address > = ^h1000) & (address < = ^h1FFF);
es2   = (address > = ^h2000) & (address < = ^h2FFF);
es3   = (address > = ^h3000) & (address < = ^h3FFF);
csiop = (address > = ^h0000) & (address < = ^h07FF);
```

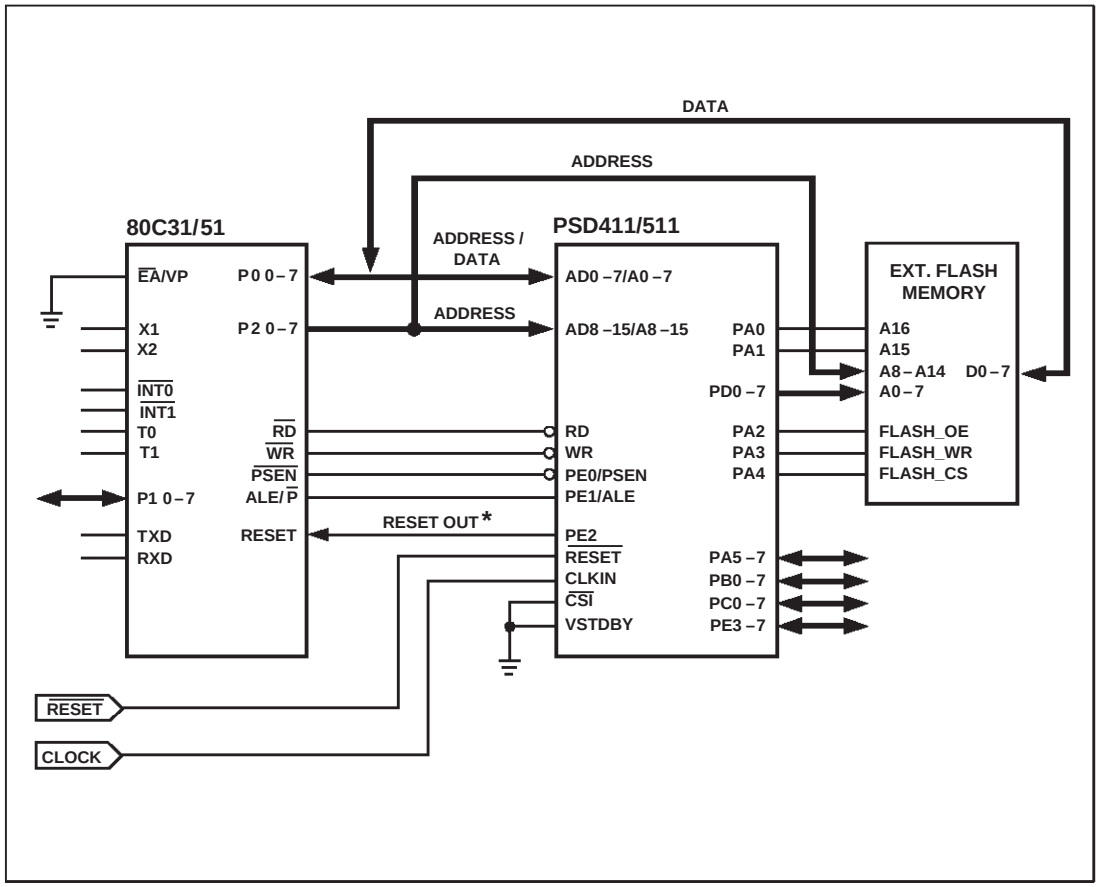
GPLD Equations

```
Flash_A16  = a15 & ((page = = 1) # (page = = 2));
Flash_A15  = a15 & ((page = = 0) # (page = = 2));
Flash_OE   = !((!program & !psen) # (program & !rd));
Flash_WR   = !(program & !wr);
Flash_CS   = !((address > = ^h4000) & (address < = ^hFFFF));
```

80C31 Example (cont.)

The detailed solution using a PSD411A1 is shown in Figure 5. The PSD411A2 or PSD511B1 can also be used in this example. The required logic equations are shown in Table 3. When the Microcontroller writes a logical "1" to the Page Register Bit 3 at address location ^h0018 (CSIOP + ^h18), the memory map is switched from the Operational Mode to the Programming Mode as shown in Figures 2 and 3. This Page Register is reset to zero when a Reset occurs on the PSD. The latched address (A0–7) appears on Port D. This must be initialized by writing a ^hFF to the Directional Register on Port D (CSIOP + ^h16). Port A provides the control signals along with the upper address lines for the Flash Memory. The PSDabel file along with the Fitter Report are shown in Appendix B. Note that CSIOP = ^h00 in the above two examples (see the definition of CSIOP in the PSDabel file shown in corresponding appendixes).

Figure 5. PSD411/511 with External Flash Memory



*RESET OUT Signal is optional. An external reset can be used to make Port Pin PE2 available for other use.

80C31 Example (cont.)

Table 3. Logic Equations for the PSD411/511

DPLD Equations

```
rs0    =  (address > = ^h0800) & (address < = ^h0FFF);
es0    =  (address > = ^h0000) & (address < = ^h1FFF);
es1    =  (address > = ^h2000) & (address < = ^h3FFF);
csiop  =  (address > = ^h0000) & (address < = ^h00FF);
```

GPLD Equations

```
Flash_A16   =  a15 & ((page = = 1) # (page = = 2));
Flash_A15   =  a15 & ((page = = 0) # (page = = 2));
Flash_OE    =  !((!program & !psen) # (program & !rd));
Flash_WR    =  !(program & !wr);
Flash_CS    =  !((address > = ^h4000) & (address < = ^hFFFF));
```

Conclusion

By using the PSD device as a Flash Memory support chip, the design is simplified into three chips. This solution does not require the programming algorithms to be moved into SRAM and executed from there. By having a separate Boot EPROM, the design is easier to complete and debug. The boot code and programming algorithms are in a secure EPROM technology thus preventing any possibility of the boot sector being corrupted. Any pin compatible Flash Memory can be used without any design change or impact to production. This solution allows a blank Flash Memory to be assembled on the board. By integrating the boot EPROM, PLD logic, and Address Latch into one device, the overall system cost including the impacts on production can be dramatically reduced.

Appendix A

```
************************************************************************************
                    W S I - PSDsoft Version 2.12
                    PSDabel Design Entry File
************************************************************************************

module flash311
title ST PSD311 design for External Flash Memory';


"PIN DECLARATIONS

"Declare all the pins that are used in the PSD3XX except Port A.
"Port A function (latched address out, I/O etc) are defined in the PSD Configuration.
"Pin numbers in this design are for the 44 pin PLCC package.

"************************** Bus Interface signals declaration **************************

"The following are multiplexed latched address inputs that are available
"as inputs to the PAD.

a15,a14,a13,a12,a11        pin 39,38,37,36,35;

"The following are bus control input signals from the 80C31
"In PSD Configuration, you need to select: 8-bit data, mux bus, and
"rd, wr, psen signal with active high ale signal

rd        pin 22;     "read line
wr        pin 2;      "write line
psen      pin 1;      "program space enable
ale       pin 13;     "address latch enable


"********************************** Port B pin declaration **********************************

"Port B pins (PB0–PB7) can be configured as:
"     1. Input or Output pin. No need to enter equations here for I/O pins. Declare the pin
"        such that the Fitter will not assign it for PAD output.
"     2. PAD output (chip select output). You need to write a logic equation for each of the
"        PAD output in the .abl file.

"Replace the reserved port pin name with your own signal name

pb0           pin 11 ;  "Port  B  pin  pb0
pb1           pin 10 ;  "Port  B  pin  pb1
pb2           pin 9  ;  "Port  B  pin  pb2
Flash_A16     pin 8  ;  "Port  B  pin  pb3
Flash_A15     pin 7  ;  "Port  B  pin  pb4
Flash_OE      pin 6  ;  "Port  B  pin  pb5
Flash_WR      pin 5  ;  "Port  B  pin  pb6
Flash_CS      pin 4  ;  "Port  B  pin  pb7
```

Appendix A
(cont.)

```
"************************************ Port C pin declaration ************************************

"Port C pins (PC0-2) pin can be configured as:
"     1. Address input (a0-a9), which can be latched by ALE. Input is considered as
"        address input if it is included in the es0–7 equations.
"     2. Logic input to PAD. A logic input participates in PAD output equations only.
"     3. PAD output (chip select output). You need to write a logic equation for each of the
"        PAD outputs in the .abl file.

"Replace the reserved port pin name with your own signal name

"The following two signals are from the 80C31 I/O ports.  These signals indicate what
" page of memory is being accessed.  This is needed to address the full range
"of the 1Mbit Flash Memory.
MCU_pgr0    pin 40;  "Port C pin pc0
MCU_pgr1    pin 41;  "Port C pin pc1

"Connect the external psen signal to PC2. This will route the psen into PAD B
"for external chip selects.
ext_psen      pin 42;  "Port C pin pc2


"********************************** A19/CSI pin declaration **********************************

"The A19/CSI pin can be configured as A19 (address or logic) input or as CSI
"(power down). Select the CSI function in the PSD Configuration. Declare A19 here if it is
"configured as an address or logic input to the PAD.

"Replace the reserved port pin name with your own signal name.
"Delete the next line if CSI function is selected.

"This is a signal from the 80C31 I/O Port indicating, when high, that the
"Flash Memory is in the program mode.
program       pin 43;


"******************** PAD A Outputs and other internal node declaration ********************


"The following are chip selects for EPROM, SRAM and I/O Port

rs0,csiop,es0,es1,es2,es3,es4,es5,es6,es7 node;

"The following are Page Register outputs (for PSD3X2, PSD3X3 only).
"pgr3, pgr2, pgr1, pgr0 node;

"*****************************************************************************************************
```

Appendix A (cont.)

```
"DEFINITIONS

X = .x.;        "Don't care
page = [MCU_pgr1,MCU_pgr0];

"address a10-a0 are not available as inputs to the PAD unless it has been assigned
"to Port C pins
address = [a15,a14,a13,a12, a11,X,X,X, X,X,X,X, X,X,X,X];


EQUATIONS


"For addresses that are assigned to Port C, they need to be latched by ALE
" in the logic equations that follow:

"a10.le = ale;        a9.le = ale;



"************************************ PAD A equations ************************************

"The PAD A acts as the address decoder and generates chip select signals
"that select the internal PSD resources. The address is latched
"internally by the ALE.  The PSD drives the 8031 data bus only if one of
"the select lines and rd or psen are active.

"The following equations are examples only. Change the address space to fit
"the selected PSD device.

"rs0 is the internal PSD SRAM chip select that is active high.
"The read and write lines are connected to the SRAM in the PSD so only the address
"is necessary in the equation. The SRAM occupies 2KB of memory space and is defined
"in 2KB boundaries.


rs0  =  ((address >= ^h0800)  &  (address <= ^h0FFF));

"es0-es7 are internal PSD  EPROM block chip selects that are active high.
"es0-es7 are defined in 4KB, 8KB or 16KB boundaries, depending on the PSD device.
"The psen signal is used internally in the PSD to qualify the chip selects
"(read can also be used to access the EPROM, see 'PSD Configuration'.


es0  =  ((address  >=  ^h0000)  &  (address  <=  ^h0FFF)) ;
es1  =  ((address  >=  ^h1000)  &  (address  <=  ^h1FFF)) ;
es2  =  ((address  >=  ^h2000)  &  (address  <=  ^h2FFF)) ;
es3  =  ((address  >=  ^h3000)  &  (address  <=  ^h3FFF)) ;


"csiop is the internal PSD chip select for the PSD I/O Port registers and is active high.
"csiop takes up 2KB bytes of memory space. Refer to the PSD Data Book
"for the address offsets of the registers.

csiop  =  ((address >=  ^h0000)  &  (address  <=  ^h07FF));
```

Appendix A
(cont.)

```
"************************************** PAD B equations ************************************

"The PAD B can implement up to 9 combinatorial logic outputs.

"The number of product terms for the PAD outputs are:
"Port B pins pb0–pb3  --  4
"Port B pins pb4–pb7  --  2
"Port C pins pc0–pc2  --  1

"Please VIEW the optimized equation file to find out the number of product terms
"used for each output signal definition.
"PAD outputs are always enabled by default.

Flash_A16  =  a15  &  ((page == 1)  #  (page == 2));
Flash_A15  =  a15  &  ((page == 0)  #  (page == 2));

Flash_OE   =  !((!program  &  !ext_psen)  #  (program & !rd));
Flash_WR   =  !(program  &  !wr);
Flash_CS   =  !((address >=  ^h4000)  &  (address <=  ^hFFFF));

end
```

Appendix A
(cont.)

****

W S I - PSDsoft Version 2.12
Output of PSD Fitter

****

TITLE : ST PSD3XX design for External Flash Memory
PROJECT : flash3xx DATE: 10/18/1996
DEVICE : PSD311 TIME : 15:06:57
FIT OPTION : Keep Current
DESCRIPTION : Interfacing Ext Flash Memory to the PSD311

****

==== 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
Flash_CS	4]	pb7	adio3	[26	Address/Data Bus ADIO_3
Flash_WR	5]	pb6	adio4	[27	Address/Data Bus ADIO_4
Flash_OE	6]	pb5	adio5	[28	Address/Data Bus ADIO_5
Flash_A15	7]	pb4	adio6	[29	Address/Data Bus ADIO_6
Flash_A16	8]	pb3	adio7	[30	Address/Data Bus ADIO_7
pb2	9]	pb2	adio8	[31	Address Bus ADIO_8
pb1	10]	pb1	adio9	[32	Address Bus ADIO_9
pb0	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	MCU_pgr2
Address Line A2	19]	pa2	pc1	[41	MCU_pgr1
Address Line A1	20]	pa1	pc2	[42	ext_psen
Address Line A0	21]	pa0	a19/csi	[43	program
rd	22]	rd	VCC	[44	

Appendix A
(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	IO	CMOS
PB1	IO	CMOS
PB2	IO	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	Input	Logic Input
PC1	Input	Logic Input
PC2	Input	Logic Input

Appendix A
(cont.)

==== 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

===== 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	3 / 8	37 %
PLD Output	5 / 8	62 %
Data Port (16 bit Non-Mux Bus)	0 / 8	0 %
Port C: (pin 40 - pin 42)		
I/O Pins	3 / 3	100 %
PLD Input	3 / 3	100 %
PLD Output	0 / 3	0 %

Appendix A (cont.)

```
========= Equations =========

DPLD  EQUATIONS :
============================
 es0   = !a15 & !a14 & !a13 & !a12;
 es1   = !a15 & !a14 & !a13 & a12;
 es2   = !a15 & !a14 & a13 & !a12;
 es3   = !a15 & !a14 & a13 & a12;
 es4   = 0;
 es5   = 0;
 es6   = 0;
 es7   = 0;
 rs0   = !a15 & !a14 & !a13 & !a12 & a11;
 csiop = !a15 & !a14 & !a13 & !a12 & !a11;

PORT B  EQUATIONS :
===================
!Flash_A16 = !a15
   # MCU_pgr0 & MCU_pgr1
   # !MCU_pgr0 & !MCU_pgr1;

!Flash_A15 = !a15
   # MCU_pgr0;

!Flash_OE = !rd & program
   # !ext_psen & !program;

!Flash_WR = !wr & program;

!Flash_CS = a15
   # a14;

PORT C    EQUATIONS :
====================
```

Appendix B

```
*************************************************************************************************
                                    W S I - PSDsoft Version 2.12
                                    PSDabel Design Entry File
*************************************************************************************************

module flash411A1
title ST PSD4XX/5XX design for External Flash';

"PIN DECLARATIONS

"Declare all the pins that are used in the PSD4XX/5XX.

"********************* Bus Interface signals declaration *********************

"The following are multiplexed latched address inputs that are available as inputs
"to the ZPLD.

a9,a8,a1,a0                 pin 67,68,8,9;
a15,a14,a13,a12,a11,a10     pin 61,62,63,64,65,66;


"The following are bus control input signals from the 80C31
"In PSDConfiguration, you need to select:8-bit data, mux bus, and
"rd, wr, psen signal with active high ale.

rd      pin 41;     "read line
wr      pin 29;     "write line
psen    pin 38;     "program space enable
ale     pin 37;     "address latch enable
reset   pin 40;     "reset input, active low
csi     pin 39;     "PSD chip select, grounded if pin is not used
clkin   pin 42;     "clock input, optional

"***************************** Port A, B, C, D, E pin declaration *****************************

"The following are I/O Port pin assignments. Port functions that need
"to be declared here are:
"       1. latched address output a0-a7 (Port C or D)
"       2. other address inputs:  a2-a7 (Port A)
"       3. MCU inputs/outputs
"       4. GPLD inputs/outputs
"Your software needs to set up the appropriate registers to enable Port functions such as
"the latched address and MCU I/O (see I/O section in the PSD Data Book for instructions).
"No equations are required if the pins are declared for non-GPLD function.

"If GPLD output is an internal feedback, you can assign it as a 'node', where the node
"number is the same as the pin number.
"For PLD pins: you may want the PSD Fitter to assign the pins for you for complex logic
"functions. In that case you need only to declare the PLD signal as 'pin' and omit
"the pin number.
```

Appendix B
(cont.)

```
"psd4xxa1:  Port A and B pins can be defined as PLD input or output
"psd4xxa2/psd5xxb1:  Port A, B and E pins can be defined as PLD input or output
"                    Port C and D pins are PLD input only

"Replace the reserved port pin name with your own signal name

"Port A pin assignments

Flash_A16    pin 27;    "Port A pin pa0
Flash_A15    pin 26;    "Port A pin pa1
Flash_OE     pin 25;    "Port A pin pa2
Flash_WR     pin 24;    "Port A pin pa3
Flash_CS     pin 23;    "Port A pin pa4
pa5          pin 22;    "Port A pin pa5
pa6          pin 21;    "Port A pin pa6
pa7          pin 20;    "Port A pin pa7

"Port B pin assignments

pb0          pin 50;    "Port B pin pb0
pb1          pin 49;    "Port B pin pb1
pb2          pin 48;    "Port B pin pb2
pb3          pin 47;    "Port B pin pb3
pb4          pin 46;    "Port B pin pb4
pb5          pin 45;    "Port B pin pb5
pb6          pin 44;    "Port B pin pb6
pb7          pin 43;    "Port B pin pb7

"Port C pin assignments

"If Port C is assigned for latched address output, use signal names
"such as addr0-addr7 (a0, a1 etc. are reserved names). You need
"to set up the Control and Direction  registers during run time
"to enable this function.

pc0          pin 17;    "Port C pin pc0
pc1          pin 16;    "Port C pin pc1
pc2          pin 15;    "Port C pin pc2
pc3          pin 14;    "Port C pin pc3
pc4          pin 13;    "Port C pin pc4
pc5          pin 12;    "Port C pin pc5
pc6          pin 11;    "Port C pin pc6
pc7          pin 10;    "Port C pin pc7

"Port D pin assignments

pd0          pin 60;    "Port D pin pd0
pd1          pin 59;    "Port D pin pd1
pd2          pin 58;    "Port D pin pd2
pd3          pin 57;    "Port D pin pd3
pd4          pin 56;    "Port D pin pd4
pd5          pin 55;    "Port D pin pd5
pd6          pin 54;    "Port D pin pd6
pd7          pin 53;    "Port D pin pd7
```

Appendix B (cont.)

```
"Port E pin assignments; pe0-1 are used as psen and ale inputs, respectively

pe2     pin 36;     "Port E pin pe2
pe3     pin 34;     "Port E pin pe3
pe4     pin 33;     "Port E pin pe4
pe5     pin 32;     "Port E pin pe5
pe6     pin 31;     "Port E pin pe6
pe7     pin 30;     "Port E pin pe7


"********** DPLD Outputs and other internal node declaration **********

"The following are DPLD outputs (chip selects) for EPROM, SRAM & I/O Port

rs0,es0,es1,es2,es3,csiop node;

"The following are Page Register outputs; they can be used as a data
"latch for GPLD input if paging is not needed

pgr3, pgr2, pgr1, pgr0 node;

"Use the pgr3 bit to determine if the system is in the flash program mode or the
"operational mode.  When this bit is a one, the system is in the flash program mode.


"DEFINITIONS

X  =  .x.;      "Don't care
clk = .c.;      "Clock pulse
page  =  [pgr1,pgr0];
program  =  pgr3;

"address a7-a2 are not available as inputs to the GPLD unless
"it has been assigned to Port A pins
address  =  [a15,a14,a13,a12,a11,a10,a9,a8,X,X,X,X,X,X,a1,a0];


EQUATIONS


"If a7-a2 are assigned to Port A, they need to be latched by ALE in the
"logic equations that follow:

"a7.le = ale;      a6.le = ale;
```

Appendix B
(cont.)

```
"****************************************** DPLD equations ******************************************

"The DPLD acts as the address decoder and generates chip select signals
"that select the  internal PSD resources. The address is latched internally by the ALE.
"The PSD drives the data bus only if one of the select lines and rd or psen are active.

"The following DPLD equations are examples only. Change the address space
"to fit the selected PSD device.

"rs0 is the internal PSD SRAM chip select that is active high.
"The read and write lines are connected to the SRAM in the PSD
"so only the address is necessary in the equation. The SRAM
"occupies 2KB of memory space and is defined in 2KB boundaries.

rs0  =  ((address >=  ^h0800)  &  (address <=  ^h0FFF));

"es0-es3 are internal PSD  EPROM block chip selects that are active high.
"The psen signal is used internally in the PSD to qualify these chip selects.

es0  =  ((address >= ^h0000) & (address <= ^h1FFF)) ;
es1  =  ((address >= ^h2000) & (address <= ^h3FFF)) ;


"csiop is the internal PSD chip select for all of the PSD configuration
"and I/O registers. csiop takes up 256 bytes of memory space and is active
"high. See 'System Configuration Section' in the PSD Data Book for the
"address offsets of the registers.

csiop  = ((address >= ^h0000) & (address <= ^h00FF));


"************************************** GPLD equations ***************************************

Flash_A16  =  a15  &  ((page == 1)  #  (page == 2));
Flash_A15  =  a15  &  ((page == 0)  #  (page == 2));

Flash_OE   =  !((!program  &  !psen)  #  (program  &  !rd));
Flash_WR   =  !(program  &  !wr);
Flash_CS   =  !((address >=  ^h4000)  &  (address <=  ^hFFFF));


end
```

Appendix B
(cont.)

```
*******************************************************************************************
                          W S I - PSDsoft Version 2.12
                          Output of PSD Fitter
*******************************************************************************************

TITLE        : ST PSD4XX/5XX design for External Flash
PROJECT      : Flash4xx                          DATE : 10/18/1996
DEVICE       : PSD411A1                          TIME : 15:08:44
FIT OPTION   : Keep Current
DESCRIPTION: Interfacing Flash Memory to the PSD411A1
*****************************************************************************

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

                                  ------------------
                                 |
                        GND | 1 ]  GND          GND  [35 |
      Address/Data Bus ADIO_7 | 2 ]  adio7        pe2  [36 | pe2
      Address/Data Bus ADIO_6 | 3 ]  adio6        pe1  [37 | ale
      Address/Data Bus ADIO_5 | 4 ]  adio5        pe0  [38 | psen
      Address/Data Bus ADIO_4 | 5 ]  adio4        csi  [39 | csi
      Address/Data Bus ADIO_3 | 6 ]  adio3      reset  [40 | reset
      Address/Data Bus ADIO_2 | 7 ]  adio2         rd  [41 | rd
Address/Data Bus ADIO_1 (a1)  | 8 ]  adio1      clkin  [42 | clkin
Address/Data Bus ADIO_0 (a0)  | 9 ]  adio0        pb7  [43 | pb7
                          pC7 | 10]  pC7          pb6  [44 | pb6
                          pC6 | 11]  pC6          pb5  [45 | pb5
                          pC5 | 12]  pC5          pb4  [46 | pb4
                          pC4 | 13]  pC4          pb3  [47 | pb3
                          pC3 | 14]  pC3          pb2  [48 | pb2
                          pC2 | 15]  pC2          pb1  [49 | pb1
                          pC1 | 16]  pC1          pb0  [50 | pb0
                          pC0 | 17]  pC0       adio15  [51 |
                              | 18]  VCC          GND  [52 |
                              | 19]  GND          pd7  [53 | pd7
                          pa7 | 20]  pa7          pd6  [54 | pd6
                          pa6 | 21]  pa6          pd5  [55 | pd5
                          pa6 | 22]  pa5          pd4  [56 | pd4
                     Flash_CS | 23]  pa4          pd3  [57 | pd3
                     Flash_WR | 24]  pa3          pd2  [58 | pd2
                     Flash_OE | 25]  pa2          pd1  [59 | pd1
                    Flash_A15 | 26]  pa1          pd0  [60 | pd0
                    Flash_A16 | 27]  pa0       adio15  [61 | Address Bus ADIO_15 (a15)
                              | 28]  VSTBY     adio14  [62 | Address Bus ADIO_14 (a14)
                           wr | 29]  wr        adio13  [63 | Address Bus ADIO_13 (a13)
                          pe7 | 30]  pe7       adio12  [64 | Address Bus ADIO_12 (a12)
                          pe6 | 31]  pe6       adio11  [65 | Address Bus ADIO_11 (a11)
                          pe5 | 32]  pe5       adio10  [66 | Address Bus ADIO_10 (a10)
                          pe4 | 33]  pe4        adio9  [67 | Address Bus ADIO_9 (a9)
                          pe3 | 34]  pe3        adio8  [68 | Address Bus ADIO_8 (a8)
                              |                          |
                                  ------------------
```

Appendix B
(cont.)

==== Global Configuration ====

Data Bus : 8-bit Multiplexed
ALE/AS Signal : ACTIVE HIGH
Security Protection : OFF

==== 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

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

Device Resources	used / total	Percentage
Port A: (pin 20 - pin 27)		
I/O Pins	8 / 8	100 %
MCU I/O or Address Out	3 / 8	37 %
Peripheral I/O	0 / 8	0 %
ZPLD Inputs	0 / 8	0 %
ZPLD Combinatorial Outputs	5 / 8	62 %
Other Information		
Buried Macrocells	0 / 3	0 %
Product Terms	7 / 25	28 %
Port B: (pin 43 - pin 50)		
I/O Pins	8 / 8	100 %
MCU I/O or Address Out	8 / 8	100 %
ZPLD Inputs	0 / 8	0 %
ZPLD Combinatorial Outputs	0 / 8	0 %
ZPLD Registered Outputs	0 / 8	0 %
Other Information		
Buried Macrocells	0 / 8	0 %
Product Terms	0 / 80	0 %
Port C: (pin 10 - pin 17)		
I/O Pins	8 / 8	100 %
MCU I/O or Address Out	8 / 8	100 %
Data Port (Non-Mux Bus)	0 / 8	0 %
Port D: (pin 53 - pin 60)		
I/O Pins	8 / 8	100 %
MCU I/O or Address Out	8 / 8	100 %
Data Port (16-bit Non-Mux Bus)	0 / 8	0 %
Port E: (pin 30 - pin 34, pin 36 - pin 38)		
I/O Pins	8 / 8	100 %
MCU I/O or Address Out	6 / 8	75 %
ZPLD Inputs	0 / 2	0 %
Control Signal Inputs	2 / 2	100 %
APD Clock Input	0 / 1	0 %

Appendix B
(cont.)

```
==== OMC Resource Assignment ====

Resources Used      User Name
-------------------------------------------------------------

Port A :
 macro cell 0       Flash_A16 (mc_pa0) => Combinatorial
 macro cell 1       Flash_A15 (mc_pa1) => Combinatorial
 macro cell 2       Flash_OE (mc_pa2)  => Combinatorial
 macro cell 3       Flash_WR (mc_pa3)  => Combinatorial
 macro cell 4       Flash_CS (mc_pa4)  => Combinatorial

Port B :

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

DPLD     EQUATIONS :
===================

 es0    =  !a15 & !a14 & !a13;
 es1    =  !a15 & !a14 & a13;
 es2    =  0;
 es3    =  0;
 rs0    =  !a15 & !a14 & !a13 & !a12 & a11;
 csiop  =  !a9 & !a8 & !a15 & !a14 & !a13 & !a12 & !a11 & !a10;


PORT A  EQUATIONS :
===================
 Flash_A16  =  a15 & !pgr1 & pgr0
    #  a15 & pgr1 & !pgr0;

 Flash_A15  =  a15 & !pgr0;

 Flash_OE  =  rd & pgr3
    #  psen & !pgr3;

!Flash_WR  =  !wr & pgr3;

Flash_CS  =  !a15 & !a14;

 [Flash_A16, Flash_A15, Flash_OE, Flash_WR, Flash_CS].OE = 1;

PORT B  EQUATIONS :
===================
```

Table 1. Document Revision History

Date	Rev.	Description of Revision
	1.0	AN048: Document written in the WSI format
30-Jan-2002	1.1	AN1422: Designing with Flash+PSD Memory Front page, and back two pages, in ST format, added to the PDF file Any 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.