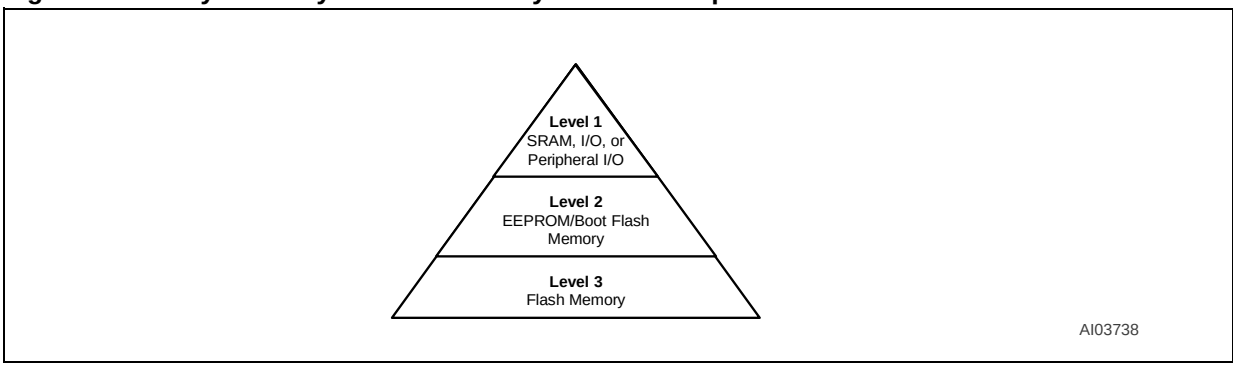
AN1237
APPLICATION NOTE

Driving an M88x3Fxx FLASH+PSD Device From an M68HC11

This document contains an example memory-map when using a FLASH+PSD device with an M68HC11. It also includes an associated minimal connection schematic and a PSDabel file. The suggested memory map is given for a M8813F1x part. Many other memory configurations are possible; however, this example provides the best utilization of the FLASH+PSD Decode PLD (DPLD). (The memory map for a M8813F2x can look just like a map for the M8813F1x with EEPROM areas replaced by a secondary Flash memory). Figure 1 shows the priority level of the PSD memory and I/O components. If a component on a lower level overlaps one on a higher level, priority will be given to the item on the lower level. For example, if the PSD's SRAM is mapped to 8000h – 87FFh, and its Flash memory segment 3 (fs3) is mapped to 8000h – BFFFh, any address in the 8000 to 87FFh range would access the SRAM, effectively reducing the size of that Flash memory segment by 2 Kbytes, from 8800h to BFFFh.

Figure 1. Priority Level Pyramid of Memory and I/O Components



Most of the Motorola 68HC11 family of microcontrollers have only 64 KBytes of address space. Because of this, a simple paging scheme is required to utilize all of the memory in the FLASH+PSD and to allow In-System-Programming (ISP). The PSDabel template file provided uses 4 of the 8 PSD page register bits in the following way:

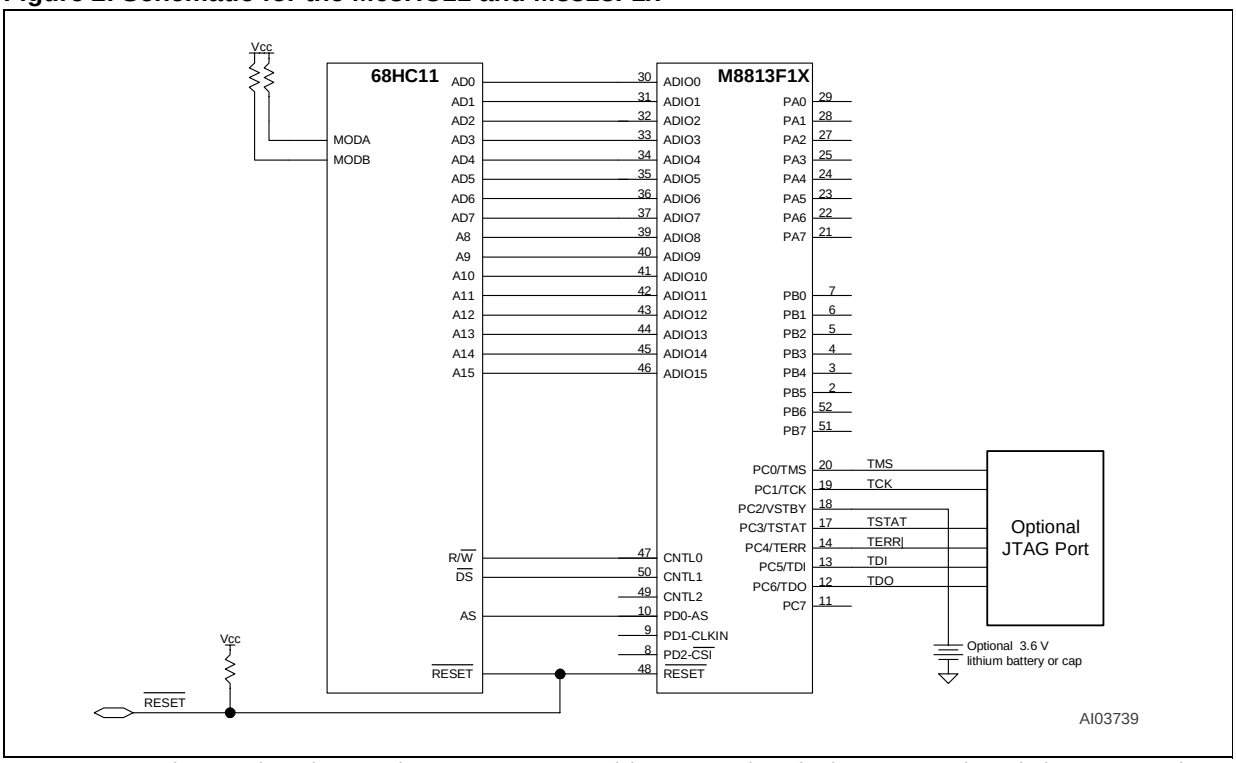
- 2 of the page register bits are decoded to allow up to four memory pages
- 1 page bit is used for swapping EEPROM segments with Flash memory segments ("swap" bit)
- 1 page bit is used for EEPROM security, limiting access to the boot code in the EEPROM ("unlock" bit)

For this example, at power-up, the MCU fetches a reset vector from high memory (FFFE – FFFF) which is stored in PSD EEPROM. This reset vector forces a jump to location 0xC000 (also in PSD EEPROM), which contains code for initialization (boot), UART download protocol, and access to Flash memory. The very first instruction that the HC11 executes in the initialization routine is one which relocates the internal HC11 SRAM and registers to a new location, 0x8000, from the default location, 0x0000. The HC11 allows this, which makes our mapping scheme easier to work with because we can create a contiguous "common" memory area from 0x8000 to 0xFFFF. This common area includes the reset vectors in high memory (0xFFFE), and the HC11 registers and memory starting at 0x8000. This leaves the address range 0x0000 to 0x7FFF open for paged memory, as shown in Figure 3.

After boot and initialization, and while the HC11 is executing out of EEPROM, the Flash memory can be checked for valid contents (using the checksum) or it can be programmed with UART data from a host computer if needed. After the Flash memory contents are validated, HC11 execution will jump to Flash memory (fs0), then set the swap bit inside the PSD page register to swap a portion of Flash memory (fs7) with a portion of EEPROM (ees0 and ees1). After the swap, the HC11 is executing completely out of Flash memory. If desired, the original boot code in EEPROM segments ees0 and ees1 can be modified with new code received over the UART. For this to happen, the HC11 must first set the “unlock” bit in the PSD page register before modifying the EEPROM boot code. The remainder of EEPROM (ees2 and ees3) can be treated as general data and is independent of the unlock bit.

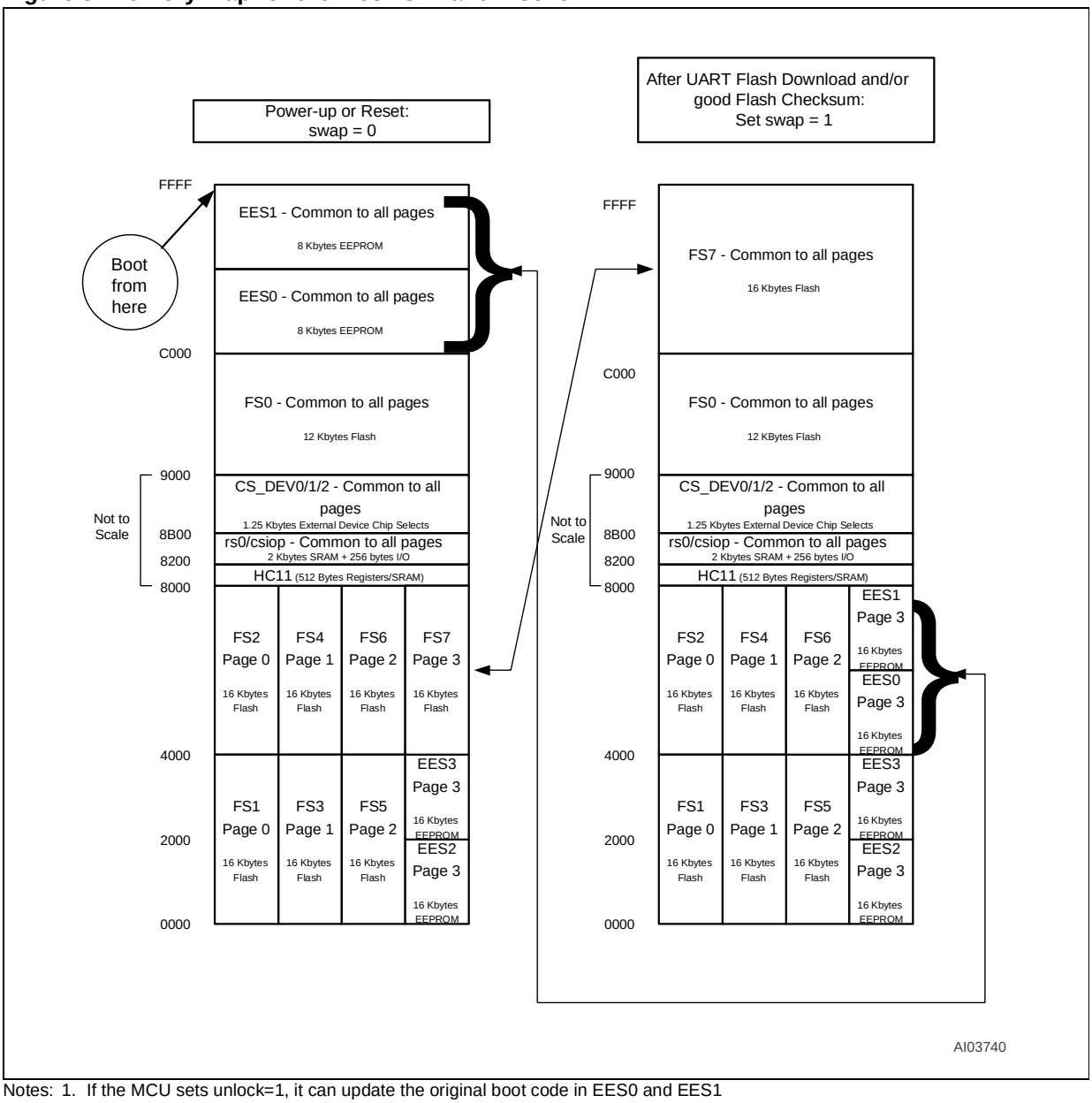
At the next reset or power cycle, this sequence will be repeated (HC11 boots from EEPROM).

Figure 2. Schematic for the M68HC11 and M8813F1x



Notes: 1. Connections are shown between the MC68HC11 MCU and the PLCC package for the M8813F1x, but only the necessary pins on the MCU are shown.

Figure 3. Memory Map for the M68HC11 and M8813F1x



Notes: 1. If the MCU sets unlock=1, it can update the original boot code in EES0 and EES1

TEMPLATE FOR THE M68HC11 AND M8813F1X

```
title ‘ST M88 FLASH+PSD design template for the 68HC11 family
of microcontrollers (multiplexed bus versions)’;

“***** Important: 52 pin PQFP package users... regard all pin
“***** numbers in this template as 52 pin PLCC package pin
“***** assignments.

“***** Template rev date 9/17/99

“***** This PSDabel file is used to:
“*****
“***** 1. Define the chip select equations for internal PSD components.
“*****    These equations are implemented on the DPLD to select:
“*****      * Main Flash memory
“*****      * Secondary NVM programmable memory (EEPROM for
“*****          M8813F1, Flash for M88X3F2)
“*****      * SRAM
“*****      * PSD control registers
“*****
“***** 2. Define external chip select equations.
“*****
“***** 3. Define equations for the CPLD application specific logic.
“*****
“***** Refer to Application Note AN1171 - ‘PSD CPLD Primer’ for more
“***** information and examples. Also, see Application note AN1170 for
“***** memory map information.

“***** This template makes the following assumption about the
“***** setting of the HC11:
“***** - The SRAM and registers internal to the HC11 are moved
“*****   from their default location (address 0x0000) to address
“*****   0x8000. The HC11 allows this move within the first 64
“*****   E clocks after reset is released.


“PIN DECLARATIONS

“***** Declare pin names for all I/O signals. Pin numbers are not
“***** necessary if it is desired to let the PSDsoft Fitter utility
“***** choose the best fit. If reserved signal names are used,
“***** predetermined pin numbers are automatically assigned. Some
“***** signals must remain on certain pins (microcontroller addr, data,
```

```
“**** control, etc.). See PSDsoft users manual for complete list of
“**** reserved names.
“****
“**** Delete any pin declaration that is not used in this template.
“**** Remove the comment delimiter from any statements that will
“**** be included in this compilation.
“****
“**** Important: All pin numbers generated from this template
“**** (either from reserved signal names or from direct pin
“**** assignments) are for the 52 pin PLCC package.


“*********** MCU Bus Interface signal declarations *****************


“**** The following are HC11 bus input signals to the FLASH+PSD PLDs.

“psen  pin 49;  The PSEN pin is not used in Motorola MCU
r_w      pin;“CNTL0 Input:(pin 47)- read/write indicator
e        pin;“CNTL1 Input:(pin 50)- E clock
//name   pin 49;  “CNTL2 Input:(pin 49)- not used as control for HC11,
                 “this pin can be used for a general input to PLD.
                 “Insert your signal name.
as       pin;“PD0 Input:(pin 10)- address strobe
reset    pin;   “Input:(pin 48)- system reset
a15..a0  pin;   “Input:(pins 46..39,37..30)- demuxed address

“**** In addition to making these declarations, use the PSD
“**** Configuration utility and make these selections:
“****    * 8-bit muxed data bus
“****    * R/W, E for control setting
“****    * Active high level for ALE/AS
“****    * Enable CSi if used in application


“******** Port A, B, C, D pin declaration ****************************

“**** Pin or node names must be declared for all signals. Replace
“**** reserved pin signal names in this template with your own names
“**** if desired. Remember, the PSDsoft Fitter utility will assign
“**** predetermined pin numbers to signals with reserved pin signal
“**** names. See PSDsoft user’s manual for list of reserved names.
“****
“**** If a CPLD output is to be used as internal feedback, declare
```

```
“**** the signal as a node instead of a pin, using your own signal
“**** name. You can specify a certain node number to force the node
“**** to a particular Output MicroCell, or you can let the Fitter
“**** utility choose a node. See PSDsoft users manual for node
“**** numbers.
“****
“**** If any Output MicroCell is used as a registered output, or any
“**** Input MicroCell is used as a registered input, the associated
“**** pin or node must be declared as a register (istype ‘reg’).
“****
“**** For MCU I/O mode and Address Out mode, no equations are
“**** necessary. Just declare the pin names so the Fitter utility
“**** will not use them. These modes must be configured at run-time
“**** by the microcontroller writing to PSD control registers, see
“**** data sheets for reg definition.
“****
“**** See the M8813F1 tutorial (app note AN1154) and the FLASH+PSD CPLD
“**** Primer (App note AN1171) for details and examples related to
“**** these issues.

“**** Port A pin assignments

“**** Use these reserved names or edit with your own names.

//pa0   pin; “I/O (pin 29)- Port A pin pa0
//pa1   pin; “I/O (pin 28)- Port A pin pa1
//pa2   pin; “I/O (pin 27)- Port A pin pa2
//pa3   pin; “I/O (pin 25)- Port A pin pa3
//pa4   pin; “I/O (pin 24)- Port A pin pa4
//pa5   pin; “I/O (pin 23)- Port A pin pa5
//pa6   pin; “I/O (pin 22)- Port A pin pa6
//pa7   pin; “I/O (pin 21)- Port A pin pa7

“**** Port B pin assignments

//pb0   pin; “I/O (pin 7)- Port B pin pb0
//pb1   pin; “I/O (pin 6)- Port B pin pb1
//pb2   pin; “I/O (pin 5)- Port B pin pb2
//pb3   pin; “I/O (pin 4)- Port B pin pb3
//pb4   pin; “I/O (pin 3)- Port B pin pb4
//pb5   pin; “I/O (pin 2)- Port B pin pb5
//pb6   pin; “I/O (pin 52)- Port B pin pb6
//pb7   pin; “I/O (pin 51)- Port B pin pb7
```

```
“**** Port C pin assignments

“**** Port C pins can be used for I/O just like Port A or
“**** Port B. However, some of the pins on Port C can be used for
“**** special functions such as the IEEE 1149.1 JTAG interface.
“**** Declare these pins for desired special functions with your
“**** own signal names, then use the PSD Configuration utility
“**** to enable the functions. See M88 tutorial.

//pc0   pin; “I/O (pin 20)- Port C pin pc0, or JTAG TMS
//pc1   pin; “I/O (pin 19)- Port C pin pc1, or JTAG TCK
//pc2   pin; “I/O (pin 18)- Port C pin pc2, or VSTBY
//pc3   pin; “I/O (pin 17)- Port C pin pc3, or JTAG TSTAT, or Stdby On
//pc4   pin; “I/O (pin 14)- Port C pin pc4, or JTAG TERR\, or Rdy/Busy
//pc5   pin; “I/O (pin 13)- Port C pin pc5, or JTAG TDI
//pc6   pin; “I/O (pin 12)- Port C pin pc6, or JTAG TDO
//pc7   pin; “I/O (pin 11)- Port C pin pc7


“**** Port D pin assignments

“pd0 (pin 10) is assigned above as the ALE signal from the MCU
“and is not available for use as general I/O.

clkin  pin;    “Port D pin pd1 (pin 9) can be used as a common
               “clock (clkin) to the PLDs and the power down
               “circuitry. If not used as a common clock, this
               “pin may be used as general I/O.

//pd2    pin;  “Port D pin pd2 (pin 8) can be used as general I/O
               “or the global PSD chip select (CSi). If CSi is
               “desired, do not declare the pin, go to the PSD
               “Configuration utility and enable CSi.

“**** Following are examples of pin and node assignments that are
“**** not tied to any particular PSD pin or node. The Fitter utility
“**** will choose. See the Fitter report (*.frp) for actual pin and
“**** node assignments.

device_cs                      pin;     “Combinatorial output pin.
term_cnt                       node;    “Combinatorial output node.
Latched_Input7..Latched_Input0 pin istype ‘reg’;    “Registered pins.
half_clkin                     node istype ‘reg,buffer’; “Reg’rd nodes
```

```
Down_Count3..Down_Count0       node istype ‘reg,buffer’;
Init_Count3..Init_Count0       node istype ‘reg,buffer’;


“**** Example of WSIPSD PROPERTY statement for Output Microcells.
“**** A four bit self-reloading down-counter (Down_Count[3:0])
“**** is used internally as buried nodes. The initial count
“**** (Init_Count[3:0]) is loaded by the microcontroller . This
“**** property statement aligns the individual bits of Init_Count
“**** with the desired microcontroller data bus bits.

WSIPSD PROPERTY ‘DataBus_OMC D[3:0]:Init_Count[3:0]’;




“**** Example of WSIPSD PROPERTY statement for Input Microcells.
“**** Eight signals on Port B will be latched by a common clock
“**** as they enter the PSD. Input Microcells will be used to
“**** implement this. This property statement will assign pins on
“**** Port B to be aligned with the desired microcontroller data
“**** bus bits.

WSIPSD PROPERTY ‘DataBus_IMC D[7:0]:Latched_Input[7:0] PortB’;


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

“**** The following are DPLD outputs (chip selects) for the
“**** main FLASH, alternate programmable NVM, SRAM and PSD
“**** control registers. Include or comment out the appropriate
“**** declarations depending on which M88 device you are using.

“**** Main Flash memory segments ************************
fs7..fs0 node;  “This declaration is supported by all M88 devices

“**** Alternate NVM programmable segments ***************
ees3..ees0 node;   “EEPROM: This declaration is supported by M8813F1
                   “devices only.

//csboot3..csboot0 node; “Flash: This declaration is supported by M88X3F2
                   “devices.

“**** SRAM *********************************************
```

```
rs0 node;  “This declaration is supported by all the M88 family.

“**** PSD Control Registers ****************************
csiop node;   “This declaration is needed for all M88 devices.

“**** Internal PSD Page Register bits
pgr1..pgr0 node;  “This declaration is supported by all M88
                  “devices. For this template example, only two page
                  “bits are used as address extension (four pages).
                  “Declare only the page bits needed for your
                  “application. Up to eight bits are available, (pgr7
                  “..pgr0). Always start with pgr0 and progress
                  “upward. The future mapping feature of PSDsoft
                  “relies on this binary weighting.

“**** Important. If page register bits are not used as address
“****            extension bits (such as pgr0, pgr1, etc), but are
“****            used to manipulate access of memory (i.e. swapping
“****            boot memory with main memory), then they should be
“****            declared as individual page bit node numbers to be
“****            compatible with the future mapping feature of
“****            PSDsoft. Do not use the reserved page bit names
“****            (such as pgr0 .. pgr7) so that the mapping feature
“****            will not count them as part of the address.
“****            It is recommended to use bits starting at the most
“****            significant end (node 117) and work downwards.
“****            Here are the node numbers associated with page
“****            register bits.
“****
“****              Page register bit        Internal node number
“****                   pgr7                     node 117
“****                   pgr6                     node 116
“****                   pgr5                     node 115
“****                   pgr4                     node 114
“****                   pgr3                     node 113
“****                   pgr2                     node 112
“****                   pgr1                     node 111
“****                   pgr0                     node 110

“**** The following page register bit definitions are an example of
“**** how to manipulate memory to facilitate In-System-Programming.
“**** See App note AN1237.
```

```
swap node 117;    " This page register bit, pgr7, will be used for
                  " swapping memory segments after a firmware
                  " download from the HC11 UART port has completed.
                  " swap = 0, secondary NVM occupies boot area for
                  " ISP, swap = 1, primary NVM occupies boot area.

unlock node 116;  " This page reg bit, pgr6, will allow access (read
                  " and write) to the secondary NVM (EEPROM or Boot
                  " Flash) after it has been swapped out of the boot
                  " address area (after swap = 1).
                  " This protects this secondary NVM boot area from
                  " unwanted writes while unlock = 0. The MCU must
                  " set unlock = 1 before it can update the
                  " secondary NVM boot code.


"**** JTAG port select ********************************

jtagsel node;    "Selects JTAG port active using a product term

"****************************************************************

"DEFINITIONS

DCOUNT = [Down_Count3..Down_Count0];  "buried down counter
INIT = [Init_Count3..Init_Count0];    "initial count value
LINPUTS = [Latched_Input7..Latched_Input0]; "latched input signals

X  = .x.;        "Don't care symbol

page = [pgr1,pgr0]; "You can use up to eight bits for memory paging.
                    "Here, only two bits are used to define four
                    "memory pages. The remaining six page register
                    "bits can be used for general registered logic
                    "that the microcontroller can write to. All
                    "eight page register bits are inputs to the PLDs.

address = [a15..a0];  "De-muxed microcontroller address signals

EQUATIONS


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

```
“**** The following DPLD equations are examples only. If desired,
“**** change the address ranges and page definition to suit your
“**** design. Delete any equation that is not used.

“**** Generate active high chip selects for the main Flash segments.
“**** Each segment is 16K bytes for the M88X3FX devices.
“**** All M88 devices support fs7..fs0.

fs0 = (address >= ^h9000) & (address <= ^hBFFF) & (page == X);
fs1 = (address >= ^h0000) & (address <= ^h3FFF) & (page == 0);
fs2 = (address >= ^h4000) & (address <= ^h7FFF) & (page == 0);
fs3 = (address >= ^h0000) & (address <= ^h3FFF) & (page == 1);
fs4 = (address >= ^h4000) & (address <= ^h7FFF) & (page == 1);
fs5 = (address >= ^h0000) & (address <= ^h3FFF) & (page == 2);
fs6 = (address >= ^h4000) & (address <= ^h7FFF) & (page == 2);

fs7 = ((address >= ^h4000) & (address <= ^h7FFF) & (page == 3) & !swap)
    # ((address >= ^hC000) & (address <= ^hFFFF) & (page == X) &  swap);


“**** Generate active high chip selects for the EEPROM segments.
“**** Each segment is 8K bytes for the M88 devices.
“**** Only M8813F1 devices support ees3..ees0.

ees0 = ((address >= ^hC000) & (address <= ^hDFFF) & (page == X) & !swap)
     # ((address >= ^h4000) & (address <= ^h5FFF) & (page == 3) &  swap & unlock);

ees1 = ((address >= ^hE000) & (address <= ^hFFFF) & (page == X) & !swap)
     # ((address >= ^h6000) & (address <= ^h7FFF) & (page == 3) &  swap & unlock);

ees2 = (address >= ^h0000) & (address <= ^h1FFF) & (page == 3) ;
ees3 = (address >= ^h2000) & (address <= ^h3FFF) & (page == 3) ;


“**** Generate active high chip selects for the alternate Flash memory
“**** segments. Each segment is 8K bytes for the M8813F2x
“**** devices. Only M88X3F2 devices support
“**** csboot3..csboot0.

//csboot0 = ((address >= ^hC000) & (address <= ^hDFFF) & (page == X) & !swap)
//        # ((address >= ^h4000) & (address <= ^h5FFF) & (page == 3) &  swap &
unlock);

//csboot1 = ((address >= ^hE000) & (address <= ^hFFFF) & (page == X) & !swap)
```

```
//        # ((address >= ^h6000) & (address <= ^h7FFF) & (page == 3) &  swap &
unlock);

//csboot2 = (address >= ^h0000) & (address <= ^h1FFF) & (page == 3) ;
//csboot3 = (address >= ^h2000) & (address <= ^h3FFF) & (page == 3) ;


“**** Generate active high chip select for the FLASH+PSD SRAM (2K bytes).

rs0 = (address >= ^h8200) & (address <= ^h89FF) & (page == X);


“***** Generate active high chip select for the PSD control registers.
“***** 256 contiguous bytes must be decoded for all M88 devices.

csiop= (address >= ^h8A00) & (address <= ^h8AFF) & (page == X);



“******** CPLD/ECSPLD Equations ***********************************

“**** The CPLD provides 16 Output MicroCells to implement
“**** combinatorial or sequential logic for signals, nodes, and
“**** state machines. Some examples:

“**** Active low chip select for an external I/O device:
!device_cs = (address >= ^h8C00) & (address <= ^h8FFF) & (page == X);


“**** Simple clock divider:
half_clkin := !(half_clkin.fb);  “Register input is negated reg output
half_clkin.clk = clkin;          “Assign clock input.
half_clkin.re = !reset;          “Node cleared by reset input.


“**** This is a self-reloading down counter. The terminal count can
“**** be loaded by the microcontroller at runtime. Loading is achieved
“**** by writing to the ‘INIT’ bits that were created with the Output
“**** MicroCell registers. Once ‘INIT’ is loaded, the down-counter
“**** (‘DCOUNT’) will free run and reload each time terminal count is
“**** reached.

INIT.c = 0;           “The clock for these four registers should
                      “be tied to an inactive source to prevent
```

```
                    “PSDsoft Fitter from connecting these clocks
                    “to the common clock (CLKIN). If CLKIN is
                    “the source, the data loaded by the MCU will
                    “be over-written.

DCOUNT.c = half_clkin;        “Assign clock input.
DCOUNT.re = !reset;           “Counter value is cleared at reset.
term_cnt = (DCOUNT.fb == 0);  “Terminal count occurs when DCOUNT = 0.
WHEN (term_cnt) THEN DCOUNT := INIT;  “Load/reload after terminal cnt
    ELSE DCOUNT := DCOUNT.fb - 1;     “Implements the down count
                                      “on rising edge of half_clkin.


“**** There are 24 Input MicroCells available. Here is an example of
“**** how eight are used to latch input signals by a common clock as
“**** they enter the PSD at pins on Port B. The microcontroller may
“**** read the state of these signals at any time, or the signals
“**** may be used as sampled PLD logic inputs. The WSIPSD PROPERTY
“**** statement above has declared this function. Only need to assign
“**** source of Input MicroCell clock here.

LINPUTS.ld = !clkin;  “Inputs are latched into Input MicroCells on
                      “falling edge of clkin. This clock can be any
                      “signal available to PLD. Use ‘.le’ extension
                      “(LINPUTS.le) for transparent latch function.


“******** State Machine ******************************************

“**** If you have a state machine, define the ‘state value’ and
“**** ‘state registers’ in the definition section. Refer to the
“**** PSDabel manual for additional information on implementing
“**** State Machine.

“**** Enter ‘state_diagram’ and state descriptions here:



“******** Test Vectors ********************************************

“**** Write test vectors to verify PLD logic equations or state
“**** machine functions. The test vector format is:
“**** [input signals logic level] -> [output signals’ expected value];
“**** Enter ‘X’ for the expected values if you want to observe the
```

```
“**** simulation result instead of comparing it.
“**** Here’s an example:

test_vectors([address,page,swap]
     -> [fs7,fs6,fs5,fs4,fs3,fs2,fs1,fs0,ees3,ees2,ees1,ees0,rs0,csiop])

[^hFFFE,X,0] -> [0,0,0,0,0,0,0,0,0,0,1,0,0,0];
[^hFFFE,X,1] -> [1,0,0,0,0,0,0,0,0,0,0,0,0,0];
[^h2000,2,1] -> [0,0,1,0,0,0,0,0,0,0,0,0,0,0];
[^h8300,X,X] -> [0,0,0,0,0,0,0,0,0,0,0,0,1,0];
[^h8A00,X,X] -> [X,X,X,X,X,X,X,X,X,X,X,X,X,X];


end
```

For current information on FLASH+PSD products, please consult our pages on the world wide web:
www.st.com/flashpsd

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

apps.flashpsd@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.