# RAD750 SUROM White Paper

**Ray Caperoon**
**October 28, 2003**
**Version 1.0**

## Overview and Background

This white paper contains a record of my investigations and discoveries about the SUROM that is resident on the BAE RAD750 flight processor.

Like other space flight processors we at NRL have experience with, the SUROM is implemented with a reprogrammable EEPROM (The flight version of the RAD750 contains Maxwell part # 28LV010RPFS-20), rather than a one time programmable PROM. This speeds up the development process and allows us to change the contents of the boot code at a later point in the spacecraft development. However, with this convenience we have to take extra care to ensure that the contents of the SUROM are secure as possible during flight. One error in SUROM can result in an unusable computer.

(Note: I may use the terms EEPROM and SUROM interchangeably in this paper, but in general when I use the term "EEPROM", I am referring to a EEPROM components; when I use the term "SUROM", I am referring to the collection of EEPROM parts that make up the Startup ROM in the RAD750. I cannot vouch for the consistent use of these terms in the quotations I have included in this paper.)

During GLAST development, I learned that the EEPROM that makes up the SUROM is write-protected only by a programmable register in the PPCI (Power PCI) bridge chip. A simple 1 or 0 in the appropriate bit in the register will enable or disable writes to the EEPROM parts. This means there is no hardware (such as a jumper, as on the Harris SSPM) to protect the EEPROM parts from entering a write enable mode. This led me to check on the details of programming and maintaining the contents of the SUROM on the BAE RAD750.

The investigation led me to identify the following areas of concern:

- Corruption of EEPROM parts during $V_{cc}$ transition,
- Unintended re-programming of SUROM by flight software or hardware and
- Intended re-programming of SUROM by software.

## RAD750 Overview

The BAE RAD750 processor had the following functional components relevant to this paper:

- Power PC Processor (PPC) - This is the main processor resident on the RAD750. Almost all the requirements allocated to the flight software would be implemented in software running on this processor. It has access to the SDRAM, SUROM and the PCI though the Power PCI Bridge.
- EMC Processor - This processor provides some low-level initialization and supervisory support to the PPC. It is responsible for initializing the memory controller, PPC and the Power PCI Bridge registers during power up. It is also responsible for implementing the watchdog-reset function on the RAD750.
- Power PCI Bridge - This functional component provides the interface between the processors PPC, EMC, memory devices (RAM and SUROM) and the PCI bus. It contains the registers to setup and access the PCI bus and configures the memory along with other support functions, such as timers, UART and discrete digital I/O. These registers are accessible by both the EMC and the PPC.
- RAM - The RAD750 typically has 128 megabytes of EDAC protected SDRAM.
- SUROM - The RAD750 has a SUROM is made up of three EEPROM parts (Maxwell 28LV010RPFS-20. These three parts provide 256 kilobytes of storage with EDAC protection, and are the focus of this white paper.

## Corruption of EEPROM parts due to $V_{cc}$ transition.

From the Maxwell data sheet for the 28LV010:

> *"When VCC is turned on or off, noise on the control pins generated by external circuits, such as CPUs, may turn the EEPROM to programming mode by mistake. To prevent this unintentional programming, the EEPROM must be kept in unprogrammable state during VCC on/off by using a CPU reset signal to RES pin."*

In the application note from Maxwell titled "Hitachi 1Mb EEPROM - Hitachi Die HN58C1001" further explanation is provided:

> *"The proper use of the RESET input must be used to ensure data integrity. During an unexpected power down sequence the RESET input must be held VSS. The RESET input must be less than or equal to 0.5V. If the RESET input is held at the proper level during power down the data will be protected even if the input signals CE and WE are not* properly *controlled."*

It is very important that the RES\ pin on the EEPROM parts be properly controlled to prevent corruption of the SUROM during power-up and power-down of the RAD750. The RAD750 board itself does not provide any circuitry to do so. The RAD750 receives a signal from the PCI backplane (POR_N) that is tied to the RES\ pin of the EEPROM parts. It would also be desirable that circuitry that controlled the POR_N signal is able to monitor the voltage and control the RES\ pins to the EEPROM parts (by way of the POR_N signal) during any under voltage of the $V_{cc}$ provided to the RAD750.

According to BAE, *"If this is a 3U card, then the POR input from the back panel comes in through the PCI connector pin, goes to the POR input on the Power PCI ASIC, the RES inputs on the 3 EEPROM's. There is no other circutry on it, it is completely controlled from the system side."* This is good; it provides the system direct control of the EEPROM RES\ pins from the backplane.

The commercial compact PCI chassis we are using at NRL do not drive the POR_N reset signal on power up or power down. This is currently putting our SUROM EEPROM at risk of corruption. I have not observed any EEPROM corruption, but I have not systematically looked for any either.

I have included a circuit that could be used to drive the POR_N signal of the RAD750. This circuit is courtesy Amy Hurley, Dennis Silver and the SECCHI program.

In summary, the RAD750 receives the POR_N signal to protect the EEPROM on the RAD750. This signal must be controlled properly by external circuitry to prevent corruption of the EEPROM on the RAD750, or the mission will be at risk.

## Unintended re-programming of SUROM by flight software or hardware.

Another possible source of corruption of the SUROM is based on the fact that the EEPROM that makes up the SUROM can be write enabled in a register resident on the Power PCI chip on the RAD750. To write data to the EEPROM parts the follow conditions must occur:

1. The SUROM is mapped for writing to the PPC or EMC. (Only the PPC provides a method to map the SUROM in a read-only mode, through use of the DBATS. EMC access would be read/write.)
2. The SUROM (bank 0) is enabled for writing in the Memory Bank Write Enable Register (at 0xBF80006C). The PPC and EMC normally have access to the register. These registers can be mapped to the PCI bus, but this would not normally be done.
3. Data is written to the address that is mapped to the SUROM.

I conducted the two experiments to simulate the effects of rogue software or hardware on the SUROM:

In the first experiment, I wrote code on the Power PC that mapped and enabled the SUROM for write-access, then wrote to SUROM locations. I did not change the processor clock speed. The contents of the targeted locations were altered, but since the processor clock was running at 33Mhz, the data written was not properly stored. This first experiment shows that the when the SUROM is in the proper mode, PPC can corrupt the contents.

In the second experiment, I tried to corrupt the SUROM contents over the PCI bus. I wanted find out if a device on the PCI bus could go crazy and both put the SUROM in a write state and modify the SUROM contents, from improperly programmed or executed DMA. I used a PCI bus analyzer to access the PowerPCI bridge chip over the PCI bus. I had to take one extra step to enable access to the PowerPCI registers from the PCI bus. Once this was done, I was able to enable the SUROM for write access, but was not able to access the SUROM locations. Access to the SUROM locations resulted in a Master Abort on the PCI bus. This experiment convinced me that the SUROM is fairly safe from re-programming from the PCI bus. You have to take too many steps to put the RAD750 in appropriate state to corrupt the SUROM.

The following mitigation steps are available to prevent the above sequence from occurring as a result of a software or firmware error:

On the PPC:

- Do not map the SUROM for any accesses not needed in the executing mode of the software.

On the EMC:

- Keep the EMC in MONITOR mode. While in MONITOR state, the EMC does not access the SUROM. This is the normal operating state.

On the PCI:

- Do not map Power PCI registers to the PCI bus. This is the normal operating state.

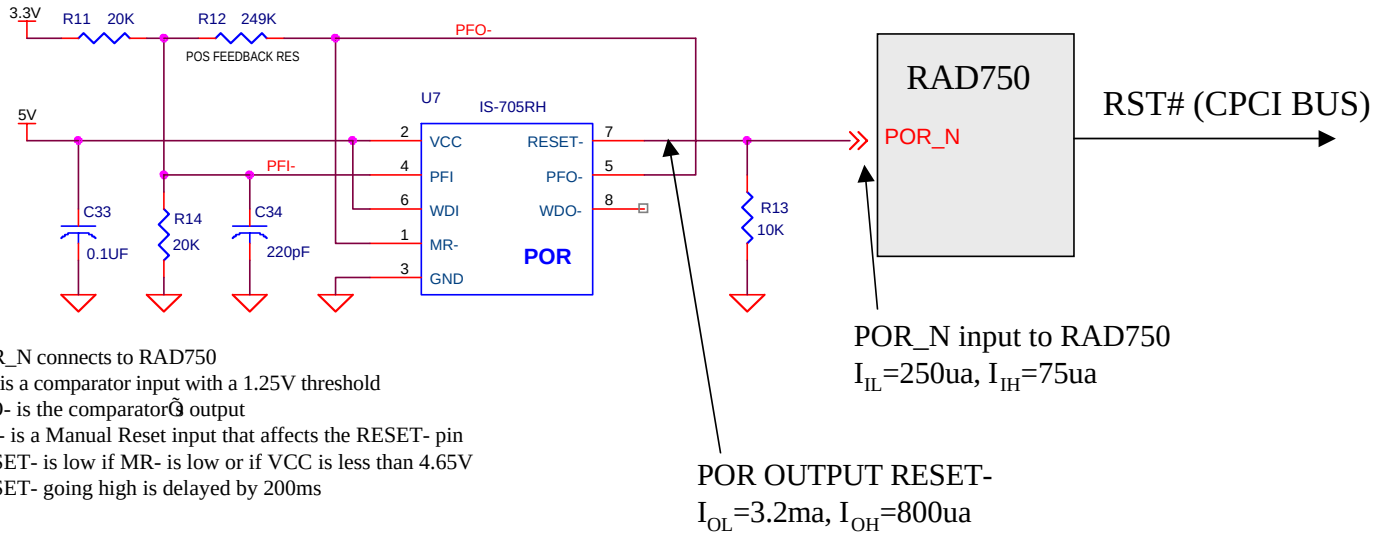
## Intended re-programming of SUROM by software.

In case anyone is uncomfortable with the protections available to keep the SUROM from being written, code can be written to re-program the contents of the SUROM on flight. This can be accomplished by the procedure described below in "The Proper Programming Procedure".

I do not recommend including this feature. The risk of corruption due to software/firmware error or improper handling of the EEPROMs during power-up or power-down should be mitigated by the methods described above.

You could develop and test this feature and have it ready to load.

# SECCHI POR Circuit




POR_N connects to RAD750
PFI is a comparator input with a 1.25V threshold
PFO- is the comparator output
MR- is a Manual Reset input that affects the RESET- pin
RESET- is low if MR- is low or if VCC is less than 4.65V
RESET- going high is delayed by 200ms

ASSUME +5V SUPPLY IS GREATER THAN 4.65V

1. If the 3.3V supply is less than 2.5V, then PFO- signal will be low. PFO- signal is feedback to MR- input. Since MR- input is low, then RESET- will be low.
2. If the 3.3V supply rises above 2.5V, then the PFO- output will go high. MR- input will then be high and 200ms later RESET- will be driven high

ASSUME +5V SUPPLY IS LESS THAN 4.65V

1. RESET- will be driven low as soon as the +5V supply is at least 1V
2. RESET- will remain low until as long as the +5V supply is less than 4.65V.
3. When the +5V supply reaches 4.65V, a 200ms timer is started. After 200ms, RESET- is driven high.

R12 provides Positive Feedback adding Hysteresis to input compare
C34 provides some basic filtering

## ICM Background

During the ICM program at NRL, we encountered a problem with the boot EEPROM in the ICM Spacecraft Controller (ISC). The processor in the ISC was the Harris Standard Spacecraft Processor Module (SSPM). The SSPM had boot EEPROM installed that contained the boot code. The EEPROMs were Austin Semiconductor AS58C1001, which contain the EEPROM die from Hitachi (HN58C1001). In the course of integration and test of the ICM, we had corrupted the boot EEPROM many times. The first few times we detected a corrupted boot EEPROM was during ISC box level testing. During investigation of the corruption, we were able to repeat the corruption many times.

The corruptions encountered during ICM were a result of the boot EEPROM parts not being properly held in reset during power-up. Simply powering on the ISC could result in corrupting the EEPROM. Circuitry was added to the ISC to solve this problem, and the EEPROM corruptions went away.

In the ISC chassis, the write protect on the boot EEPROM (wired to WE\ on the EEPROM part) could only be enabled by an external hardware jumper. To program the boot EEPROM in the ISC, we had to short the external WE\ pin to ground to enable EEPROM programming. We could do this in the spacecraft harness or in a VME chassis that provided access to that signal. Then special software was loaded into RAM to program the boot EEPROM. This process was always done at Harris for the ICM project when programming the flight units.

The EEPROM parts used on the SSPM for ICM had another data retention problem. Data written to the EEPROM parts would "fade" after a matter of days. Harris screened the EEPROM parts by holding the parts at 80° for 200 hours. During this time, data retention tests were constantly run. Parts that failed this testing were not used on the SSPMs delivered to NRL for use on ICM.

## SIB Solutions

The Spacecraft Interface Board (SIB) developed by Dennis Silver for use on GLAST and SECCHI has two banks of EEPROM installed. This EEPROM uses parts that also contain the Hitachi EEPROM die.

The SIB provides protection against corruption during $V_{cc}$ transition by using a circuit similar to the circuit shown below using the Intersil IS-705RH part. This circuit starts working between 0.7V and 1.0V and holds the EEPROM parts on the SIB in reset until 200 milliseconds after the voltage reaches 4.65V. This circuit works independent of how the POR_N on the backplane is driven.

The SIB board provides protection against accidental write enable by requiring that a specific 12-bit pattern be written to a control register before the WE\ pin on the EEPROM parts can driven to an active state. This 12-bit pattern helps reduce the occurrence of "random" events enabling the EEPROM for write.

## The Proper Programming Procedure

This is a discussion about how to program the SUROM EEPROM during flight from Craig Hatfield:

*The EEPROM parts have requirements for writing them successfully. Each part is divided into 128 byte pages. One may write in "page mode" as long as the address stays within the page bounday and the time between writes is greater than ~1us and less than 30us. When crossing a page boundary one must wait 10ms. If one waits longer than 30us, one must wait the entire 10ms. Waiting 10ms between every write is writing in "non-page mode".*

*The SUROM EEPROM on the RAD750 X2000 board is mapped across three parts. Each 32-bit write is converted into a read-modify write of 64-bits. Each of the three parts, two data, one ECC, is written four times. It does no good to attempt to write half words (16 bits) because ECC for SUROM is generated on 32-bit words and all memory accesses are 64 bits wide.*

*To answer the question, yes, I believe the X2000 board can be configured to write SUROM with a program running on the RAD750. The sequence will look something like the following:*

- *Ensure that neither the EMC, nor the RAD750 is accessing EEPROM for any other reason.*
- *Slow the system clock to add delay between the two automated writes (Adding wait cycles to ROM accesses should also add enough delay, but we haven't done it that way. We slow the clock when writing from the JTAG slave interface.) Be sure to use the EMC routines to slow the clock if the PLL is engaged. Changing the system clock speed with the RAD750 PLL multiplier enabled can cause damage.*
- *Remap bank 0 to an address other than 0xFFF00000, perhaps 0x70000000 (The PowerPC always treats the 0xFFF00000 address range as ROM, no writes allowed.)*
- *Make sure a DBAT or page entry enables access to the remapped ROM space.*
- *Enable writes to bank 0.*
- *Disable interrupts to avoid disturbances to the length of the delays.*
- *Write to blocks of 256 bytes (2 data parts * 128 bytes) and then delay for 10ms before starting the next block (If this doesn't work, try non-page mode, that is wait 10ms between every 32-bit write. Page mode should work.*

*Use caution. If you are dependent on booting the board to program EEPROM and the EEPROM becomes corrupted, you may not be able to boot the board to fix the problem.*

## Correspondence with BAE.

Most of my information has come from RAD750 documentation, Maxwell documentation, and Hitachi documentation and from e-mail correspondence with BAE. I have included the contents of three e-mails from BAE along with a few comments of my own in response to the e-mails.

**From: Bob Tiede**
**Date: Thursday, July 31, 2003**
**Subject: Response to SUROM question**

> *Anne Cady does not recommend changing the present design with a single wire, her approach to your questions would be to add PROM to eliminate the risk of EEPROM code corruption.*
>
> *A redesign which included PROM in place of EEPROM would not only require a respin of the Printed Circuit Board but also a requalification of the new design. My belief was that we wanted to stay with an off the shelf product so that little NRE was required. This would also negate the delivery dates that have been contracted.*
>
> *(This is an edited response by Anne Cady)*
>
> *I question why anyone would actually tie this off in hardware rather than relying on the double level of software protection we have. The Maxwell parts are so well known to be able to be corrupted in space (a power glitch below 2.7V could corrupt all contents regardless of how the control signals are held per application notes on Maxwell's website), that I'd always want the ability to write them in case of a corruption occurring. Without that, if a corruption occurs the mission is at risk.*
>
> *I wonder why the customer doesn't ask for the board to just be redesigned with the option of using Radhard PROM for flight cards instead. Sure would be safer.*

A few of my own editorial comments:

- There is no circuitry on the RAD750 board to control the RESET signal of the EEPROMs during power up. It is up to something on the backplane to control the POR reset signal. Dennis Silver has sent me the circuit used for the SECCHI program. This circuit monitors the 5V and 3.3V power supplies and drives the POR_N input to the RAD750. I'm not sure that the resistor values correspond to the recommended voltages for EEPROM reset. (The documented reset voltage in the circuit is 2.5V, but Anne Cady talks about 2.7V)
- "Without that, if a corruption occurs the mission is at risk." We are talking about boot code. If we have a corruption during the mission, it will be very unlikely we will have working code to reprogram the EEPROM even if we wanted to. I have already seen a corruption that made our board un-programmable by even the JTAG interface.

This next e-mail is a response to questions I sent to BAE. My questions are in plain text, answers in italics. [ceh] is Craig Hatfield from BAE.

**From: Bob Tiede**
**Date: 8/27/2003**
**Subject: RE: EEPROM questions**

1) Proper EEPROM programming technique

The following steps are taken in the emcLd.c program before writing the EEPROM.

Stop the EMC (no documentation why this is done) Put PPC in reset ("to avoid PLL problems while changing clock") Disable PLL (done when changing clock) Set Clock to lowest rate ("to allow sufficient time for EEPROM writes") Write Enable

Is it necessary to have the EMC in STOP mode while programming the EEPROM? If so, why?

I understand why it is necessary to keep the PPC in reset while changing clock speeds, but is it necessary to leave the PPC in reset while programming the EEPROM? If so, why?

*[ceh] The EMC and the PowerPC are stopped or held in reset to keep them from accessing EEPROM during the write process. They typically don't read EEPROM after the initial portion of boot, but some programmer may decide to loop in some EEPROM code. With both CPU's stopped, there is no danger from their interference.*

*Proper technique involves setting the system clock speed or the EEPROM wait cycles in the memory controller to ensure that accesses are separated by at least 400ns and no more than 30us. This allows multiple writes to each part within the "page write" timing paraemters. Since all writes are 64 bits wide and there are two 8-bit wide parts (ignoring EDAC for the moment), four write commands will be sent to each part every time we write 32-bits to SUROM. Also note that when each crossing 128 byte boundary on a part one must wait 10ms before resuming write commands. Our loader is slow enough hat all 32-bit writes are separated by at least 10ms so we don't track pages in the program.*

2) In the email from Bob Tiede, dated Thursday, July 31, 2003, Subject: Response to SUROM question, Anne Cady speaks of "double level of software protection we have". Please describe this "double level of protection".

*[Cady, Anne] The double level of protection is that if a user doesn't want the SUROM to be writable, they should have the Bank 0 control register set to disable writes to the SUROM parts. Once that is done, TWO events must happen to cause a write. First, something (SEU or other mechanism) must occur to flip the bit in the Bank 0 control register to make it writable (if its an SEU, we will get a parity error flagged on the bank control reg). Second, once the bit is flipped, then the user will also have to actually try to do a write to an address that is mapped to the SUROM range.*

*[ceh] Normally there is not a DBAT configured for ROM space. This means that the programmer would also have to configure a DBAT or a page table to access the ROM address range. This makes three actions to have software corrupt bank0 EEPROM.*

3) Is there circuitry on the RAD750 board to help protect the EEPROM from a power glitch? Is there circuitry on the RAD750 board to properly control the RESET signal of the EEPROMs during power up? Are the schematics for those circuits available?

*[Cady, Anne] The POR signal from the backpanel is tied to the RES signal on the EEPROM's. The user must keep POR active for power on and power off. The 3U board does NOT have any built in power detection circuitry to generate the RES itself.*

*Other than the capacitors on the card which help a bit in filtering out glitches, major power transients must be detected externally and POR activated to hold RES active to the parts.*

4) What are the exact part number of the EEPROM used on both the flight and commercial RAD750 units used on GLAST?

*[Cady, Anne] The 3U RAD750 cards (EM/Flight grade) use the 28LV010RPFS-20. I'm not sure what is on the commercial boards.*

*[Jason K. Jones] The commercial boards have either 28LV010RPFS-20 or HN58C1001FP-15 parts, according to the released BOM. We tend to use the HN58C1001FP-15 parts since they are cheaper plastic packaged parts. Note that these are both the same die. All Maxwell EEPROMs that we use are the same die, the screens are different for voltage, speed, and rad hardness, but it is the same Hitachi die in all of the parts.*

5) What will happen if a write to the EEPROM addresses occurs on the PCI bus? Is it possible to write to the EEPROM from a PCI device?

*[Cady, Anne] If the row is enabled for writes, then yes, a write can occur to the EEPROM. If the timings aren't set-up to support EEPROM writes (which are slow operations), then the correct data may not get written. If the row is disabled, then a memory error will be flagged and the write will not be performed.*

*[ceh] Normally bank0 (EEPROM) is configured at address 0xFFF00000. This is outside of the BAR1 address space of 0 to 0x7FFFFFFF. One could change the bank0 address registers to move EEPROM below 0x80000000, but without that change, EEPROM would not be accessible to the PCI bus.*

6) I understand the radiation test report from the RAD750 will be available on September 25, 2003. Are there any preliminary results available to us that deal with radiation effects/performance of the EEPROM? Are there radiation test reports available to us for any tests run on just the EEPROM parts used on the RAD750?

*[Cady, Anne] Maxwell (SEi) has quite a bit of data on the part and we did do some extra TID testing for JPL. What is the specific area of concern? TID or SEU? For SEU, the array is basically immune and upsets are generally in the support logic (ie - no scrubbing is needed back to the array and the normal ECC will correct the data passed on to the requestor). For TID, the read performance is per the Maxwell (SEi) spec, but write performance is much lower (as low as 15-20 Krads at the die level). If you let us know the specific concerns we can get more data to send you either from Maxwell or from the X2000 TID testing.*

A few comments:

- Anne Cady speaks of "the double level of software protection". When I asked what this double level of software protection was, the response I got from Anne Cady was that it takes two errors to corrupt the EEPROM (first to write-enable BANK0, second to write to a EEPROM address). In my mind, I can concoct a scenario in with one error can cause this to happen, so I feel this "double level of software protection" is a misleading statement.

This next e-mail is a response to a second set of questions I sent to BAE. My questions are in plain text, answers in italics. [ceh] is Craig Hatfield from BAE.

**From: Bob Tiede**
**Date: 9/8/2003**
**Subject: FW: EEPROM questions**

(Ray - 7) A follow-up on my question number one and a comment from Anne Cady:

If BAE is recommending including the ability to reprogram the boot EEPROM on flight, I need you to recommend the proper way to do it.

*[Cady, Anne] Just to clarify, we don't recommend that you ever plan to write SUROM in flight, but if you are flying EEPROM instead of PROM, then we would recommend you have the ability in case any corruption of the parts occurs. In all likelihood you won't need it, but having the ability is prudent. Craig should be able to provide you details on how it would need to be done.*

*As per Craig's info, it sounds like we cannot be executing from the EEPROM in question while programming the part, so our program will have to execute from RAM. We also need a way to enforce that the EMC is in the STOP state. Please outline an algorithm to accomplish on-flight programming of the boot EEPROM.*

*[ceh] The EEPROM parts have requirements for writing them successfully. Each part is divided into 128 byte pages. One may write in "page mode" as long as the address stays within the page bounday and the time between writes is greater than ~1us and less than 30us. When crossing a page boundary one must wait 10ms. If one waits longer than 30us, one must wait the entire 10ms. Waiting 10ms between every write is writing in "non-page mode".*

*The SUROM EEPROM on the RAD750 X2000 board is mapped across three parts. Each 32-bit write is converted into a read-modify write of 64-bits. Each of the three parts, two data, one ECC, is written four times. It does no good to attempt to write half words (16 bits) because ECC for SUROM is generated on 32-bit words and all memory accesses are 64 bits wide.*

*To answer the question, yes, I believe the X2000 board can be configured to write SUROM with a program running on the RAD750. The sequence will look something like the following:*

*- Ensure that neither the EMC, nor the RAD750 is accessing EEPROM for any other reason.*

*- Slow the system clock to add delay between the two automated writes (Adding wait cycles to ROM accesses should also add enough delay, but we haven't done it that way. We slow the clock when writing from the JTAG slave interface.) Be sure to use the EMC routines to slow the clock if the PLL is engaged. Changing the system clock speed with the RAD750 PLL multiplier enabled can cause damage.*

*- Remap bank 0 to an address other than 0xFFF00000, perhaps 0x70000000 (The PowerPC always treats the 0xFFF00000 address range as ROM, no writes allowed.)*

*- Make sure a DBAT or page entry enables access to the remapped ROM space.*

*- Enable writes to bank 0.*

*- Disable interrupts to avoid disturbances to the length of the delays.*

*- Write to blocks of 256 bytes (2 data parts * 128 bytes) and then delay for 10ms before starting the next block (If this doesn't work, try non-page mode, that is wait 10ms between every 32-bit write. Page mode should work.*

*Use caution. If you are dependent on booting the board to program EEPROM and the EEPROM becomes corrupted, you may not be able to boot the board to fix the problem.*

(Ray - 8) Please provide more details about the circuitry between POR input to the RAD750 and EEPROM RES\ pins. Is there logic driving this signal? I again ask for schematics detailing this circuitry between POR reset and EEPROM RES\, even if it simply consists of wires and capacitors.

*[Cady, Anne] If this is a 3U card, then the POR input from the backpanel comes in through the PCI connector pin, goes to the POR input on the Power PCI ASIC, the RES inputs on the 3 EEPROM's. There is no other circutry on it, it is completely controlled from the system side. We can go back and pull the actual wire lengths from the routing if you need them. [Cady, Anne] The POR signal should be held active whenever power is outside of spec (ie - during power up and down), not just to remove concerns about EEPROM contents, but to also keep the ASICs and all other devices in the system from transitioning through undefined states when only partially powered. This signal would generally either be created by the Power Supply to the box, or can be generated using a device like the Intersil IS-705RH (DSCC part number 5962-00538) radiation hardened reset generation circuit, and then sent to all the power On/off sensitive parts in the box.*

(Ray - 9) In Anne's answer to question 2, she states that "(if its an SEU, we will get a parity error flagged on the bank control reg)". Are these registers scrubbed in way to check parity? How is software notified of this parity error? Is this error indicated by the Internal Parity Error bit in the Status 2 register?

*[Cady, Anne] The static control registers are monitored continously (ie - every clock cycle) for correct parity. A parity error on an internal register is a critical error. For this example, that would activate bit 19 in the MCTL Status register in the Power PCI, which would generate the MCTL_CR_INT signal, which causes a Vector Interrupt 7 to the EMC in the chip.*

*[ceh] I am missing the context of the question, so I may be off base. Anne's latest answer addresses parity errors in the memory controller's registers. If the question is referring to the reporting of RAM parity errors by the registers, the response is not as severe. Correctable errors are reported to the CPU via maskable interrupt. Uncorrectable errors found during a CPU access are reported as an interrupt and a Machine Check. Uncorrectable errors found during a DMA access are reported to the CPU by interrupt and to the initiator as a PCI target abort. Uncorrectble errors found during EMC execution result in an EMC vector 6 which is normally programmed to reset the board.*

(Ray - 10) When the EMC processor is in the MONITOR state, is it accessing the EEPROM? If the EMC processor is in STOP state, how does the PPC take it out of STOP state?

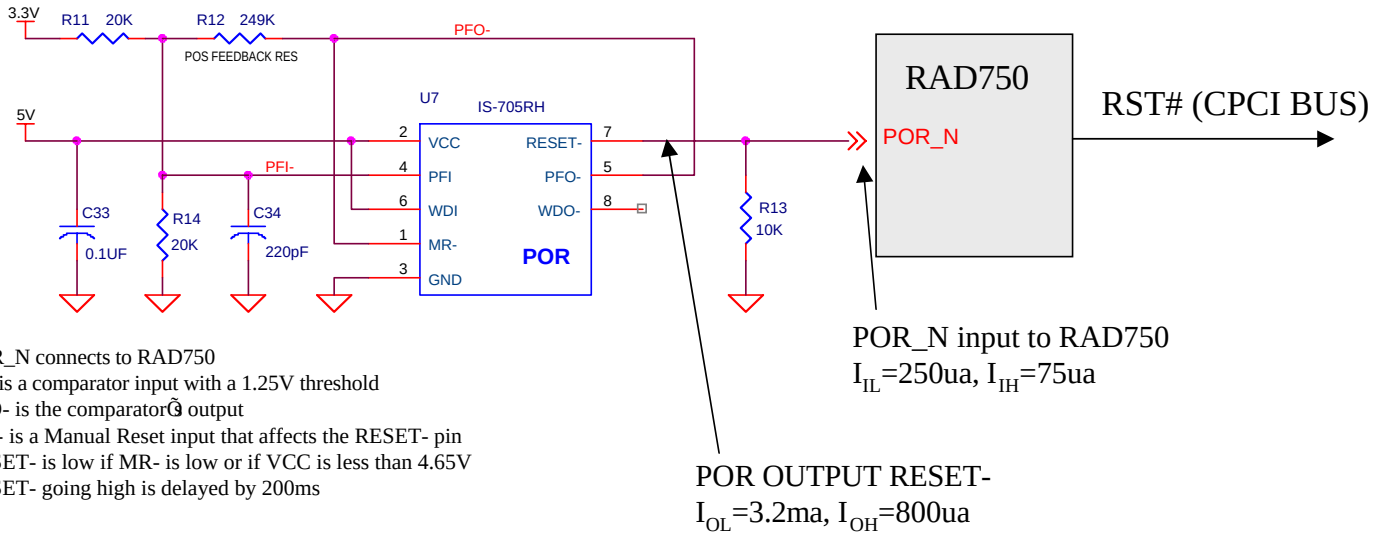
*[ceh] When the EMC is in MONITOR state, it does not access anything. It can be awoken by an unmasked or unmaskable vector interrupt. When the EMC is in a STOP state it can only be restarted by a debug command or a reset. A debug command must be enabled in the JTAG slave core. In other words, outside of the lab, the STOP state can only be entered by a STOP instruction and can only be exited by a chip reset.*

(Ray - 11) A follow-up to the answers to my question 7 about accessing EEPROM from the PCI bus: BAR1 is not setup for access to the EEPROM address range. What about BAR2? Will a write beyond the documented 1 megabyte range of BAR2 cause an error on the bus, or will data be presented to the EEPROM?

*[ceh] BAR2 only gives access to Power PCI registers, nothing more.*

My comments:

- Craig claims "The PowerPC always treats the 0xFFF00000 address range as ROM, no writes allowed." My experimentation shows that is not the case.




POR_N connects to RAD750
PFI is a comparator input with a 1.25V threshold
PFO- is the comparator output
MR- is a Manual Reset input that affects the RESET- pin
RESET- is low if MR- is low or if VCC is less than 4.65V
RESET- going high is delayed by 200ms

ASSUME +5V SUPPLY IS GREATER THAN 4.65V

1. If the 3.3V supply is less than 2.5V, then PFO- signal will be low. PFO- signal is feedback to MR- input. Since MR- input is low, then RESET- will be low.
2. If the 3.3V supply rises above 2.5V, then the PFO- output will go high. MR- input will then be high and 200ms later RESET- will be driven high

ASSUME +5V SUPPLY IS LESS THAN 4.65V

1. RESET- will be driven low as soon as the +5V supply is at least 1V
2. RESET- will remain low until as long as the +5V supply is less than 4.65V.
3. When the +5V supply reaches 4.65V, a 200ms timer is started. After 200ms, RESET- is driven high.

R12 provides Positive Feedback adding Hysteresis to input compare
C34 provides some basic filtering