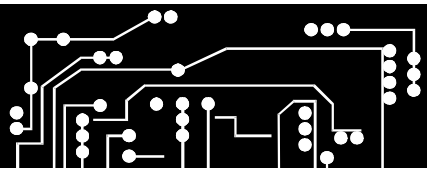


IBM PowerPC® 6xx/7xx Processor Frequently Asked Questions (FAQ)

PowerPC Applications
IBM Microelectronics
Research Triangle Park, NC
ppcsupp@us.ibm.com
http://www.chips.ibm.com

February 18, 2002 Version: 1.3

Abstract *- This document addresses topics and questions frequently asked about the PowerPC 603, 603e, 603ev, 740 and 750 processors. This document supplements the information found in the PPC603e and PPC750 User's Manuals and the associated hardware specifications.*

1. General Questions

This section discusses questions applying to more than one processor family.

1.1 What is the difference between HRESET# and SRESET#?

1.11 HRESET#

HRESET#, the hard reset signal, is considered to be a nonrecoverable, nonmaskable exception. HRESET# must be used at power-on in conjunction with the TRST# signal to properly reset the processor. HRESET# can also occur at any time and may be asserted asynchronously to the processor's input (SYSCLK) clock. It must be held asserted for a minimum of 255 clock cycles after the PLL lock time (100 us) has been met.

A hard reset will initialize latches, zero out some registers, and leave others in an indeterminate state. HID0, HID1, SRR0 and SRR1 registers will be all 0s, and the MSR will be set to 0x00000040 (IP set so that hard reset exceptions will always vector to address 0xfff00100). The BATs, segment registers, and TLBs will all be left in indeterminate states. The processor stays in this state while HRESET# is asserted. When HRESET# goes inactive, the processor will begin operations by issuing an instruction fetch from address 0xfff00100. Because caching is inhibited at startup, this will be a single beat read operation.

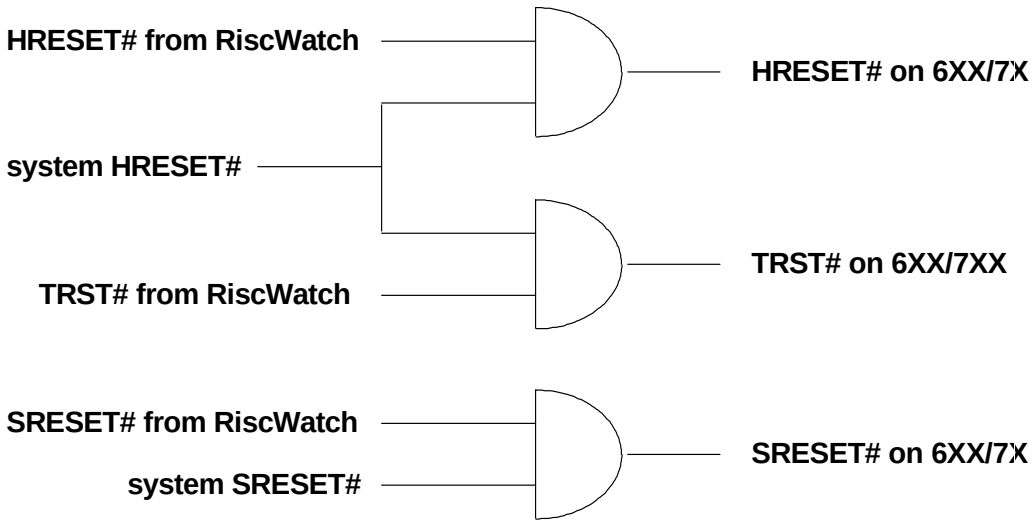
1.12 SRESET#

SRESET#, the soft reset signal, is considered to be a nonrecoverable, nonmaskable exception. SRESET# can occur at any time and may be asserted asynchronously to the processor's input (SYSCLK) clock. The SRESET# input is negative edge sensitive and may be negated two cycles after assertion.

The assertion of SRESET# will generate a system reset exception and the processor will attempt to reach a recoverable state before taking the exception. The exception taking process will modify the MSR, SRR0, and SRR1 as described in the Programming Environments Manual and Chapter 4 of the User's Manuals. SRESET# does not directly affect the states of off chip (output) signals. The MSR[IP] bit is not cleared by a SRESET# exception, so the exception handling for soft resets will vector to either high (0xfff00100) or low (0x00000100) memory, depending on system s/w design.

1.2 How should HRESET# and TRST# be wired to allow RISCWatch to be used?

TRST# is the reset input to the processor's JTAG interface. The JTAG logic must be reset during power on (hard reset) to ensure that the JTAG logic does not interfere with the operation of the processor. Basically the system must assert TRST# whenever it asserts HRESET#. When using the RISCWatch debugger, there are TRST# and HRESET# outputs from the RISCWatch JTAG probe that need to be taken into account. These signals are typically handled as follows:



This shows how to wire the HRESET#, TRST#, and SRESET# signals from the RISCWatch JTAG connector with the HRESET# and SRESET# signals generated by system logic. This allows the CPU to be reset by either the hardware (normal operation) or the RISCWatch (for debug). The addition of weak (10K ohm) pull-ups on the lines from the RISCWatch JTAG connector will prevent spurious resets when the RISCWatch is not attached.

1.3 What registers should be initialized before use?

Tables 2-19 in the 740/750 User's Manual and 4-9 in the 603e User's Manual list the register values caused by a HRESET# or a power-on condition. **All GPRs, FPRs, SRs, TLBs, and BATs come up in undefined states and should be initialized prior to use.**

Unused BATs should always be initialized to 0x00000000.

FPR initialization prior to use is strongly recommended. FPRs must be initialized prior to execution of stfd instructions to prevent potential processor hangs due to the error conditions caused by execution with undefined operands.

1.4 How is page mode translation disabled?

There is no trivial way to disable page-based virtual memory since:

- Segment registers do not have a valid bit
- There is no single bit anywhere (like in HID0) where it can be disabled.

Block (BAT) and Page translation occur simultaneously, and a successful Block translation always takes precedence. The minimum page table size is 64K bytes (PowerPC Programming Environments Manual, Chapter 7), so you always should reserve 64K (minimum) for the page table. To setup a minimal size page table with a minimal amount of effort, this block must be 64K byte aligned and filled with 8K invalid Page Table Entries (PTEs). Zeroed memory suits this purpose, since the Valid bit in the lower-addressed word will always be zero.

Once this 64K area is cleared, move the real (non-translated) address of this area into SDR1. You can do this real address copy ONLY if your block is aligned on a 64K byte boundary. This works because the HTABMASK portion of SDR1 is supposed to be all zeroes for a minimally sized page table, and since the page table is 64K byte aligned, the low 16 bits will already be zero.

There is no requirement to clear the Segment registers. Because of the mechanism by which a Page Table Entry group is selected (see section 7.6.1.4 in the Programming Environments Manual for details), it might minimize the L1 D-cache footprint to use the same value for each segment register. In any case, having them all the same will not adversely impact the cache footprint.

To recap, while translation is OFF, do the following three steps:

1. Zero the page table (64K bytes aligned on a 64K boundary)
2. Set SDR1 to point to the page table
3. Invalidate the TLBs

Optionally, clear the segment registers.

It does not matter where the page table setup occurs relative to the BAT setup, as long as translation is turned off during both setup operations.

1.5 Where can I find models for these processors?

Both IBIS and BSDL can be found on the web at:

http://www-3.ibm.com/chips/techlib/techlib.nsf/productfamilies/PowerPC

Select the specific processor and within the link and the BSDL and IBIS models will be available along with all technical documentation that supports that processor.

7xx Questions

This section presents questions specific to the 7xx processor family, including the 740, 750, and 750CX processors.

1.6 What is the difference between the 740 and 750 processors?
The 740 processor is just a 255 pin version of the 360 pin 750. The 750 has built-in support (cache tags) for a direct attached L2 cache; the pin difference between the two parts is to accommodate the L2 interface. The 750 and 740 are available in two different process technologies, each with a different speed range and core voltage. The following table outlines the different 740/750 processors:

	PID 8t	**PID 8p**
Processor Speed	200/233/266 MHz	300/333/366/400 MHz
Technology	0.25 um/0.18 Leff CMOS	0.22 um/0.12 Leff CMOS
Die Size	67 sq. mm	40 sq. mm
Power Supply (voltages)	2.5 V to 2.7 V core 3.3 V +/- 5% I/O	2.0 V to 2.1 V core (see note) 3.3 V +/- 5% I/O
Power Dissipation (typical)	5.7W @ 266 MHz (740) 5.9W @ 266 MHz (750)	3.7W @ 400 MHz (740) 4.1 W @ 400 MHz (750)
Temperature Range	0 to 105 C	0 to 105 C (see note)
Packaging	255-pin CBGA (740) 360-pin CBGA (750)	255-pin CBGA (740) 360-pin CBGA (750)

NOTE: APPLICATION CONDITIONS EXIST FOR HIGHER SPEED PARTS; REFER TO THE PRODUCT DATASHEET FOR SPECIFIC APPLICATION CONDITIONS. DATASHEETS ARE AVAILABLE AT:

http://www-3.ibm.com/chips/techlib/techlib.nsf/productfamilies/PowerPC

1.7 Bus Arbitration Questions

This section raises some 750/750CX bus arbitration issues to be considered when designing the system bus arbiter.

1.71 Will the 750CX continue to assert BR as long as it has internal bus accesses pending and BG is negated to the 750CX?

Yes, the 750/750CX will assert BR as long as it needs the bus and doesn't have a grant. Once claiming the bus, the 750/750CX will negate its BR for at least one bus clock cycle even if there are pending internal accesses.

1.72 In general, what timing should the arbiter observe when handing the bus from the 750CX to another device?

Once the arbiter has asserted BG to the 750/750CX, the arbiter must maintain BG asserted until the 750 either negates BR (i.e., decides it no longer needs the bus) or asserts TS (runs a cycle). The arbiter may negate BG to the CPU any time after either of these events, and the arbiter may assert BG to another device during the same clock cycle. Only one bus grant should be active on any rising edge of the bus clock. Note that the 750/750CX may initiate a bus cycle on the same clock as the negating BG, so the arbiter must recognize that the bus has become busy AND that the owner of the current bus cycle is the CPU.

1.73 Section 7.2.1.2 of the 740/750 User's Manual states: "The 750 may still assume bus mastership on the bus clock cycle of the negation of BG because during the previous cycle BG indicated to the 750 that it could take mastership". If this condition occurs while the bus is parked on the 750CX, so that the 750CX runs a cycle without asserting BR, how do the other devices and the arbiter recognize the bus has become busy? Should the arbiter wait for 1 more cycle after BG removal to avoid contention?

Address bus tenures are marked by TS and AACK. Fundamentally, the arbiter and any bus residents need to track the bus states to determine when the bus is idle. There is no need to wait an additional cycle to

avoid contention as long as the alternate master is capable of recognizing a busy bus from TS and AACK. The arbiter can synthesize ABB with a state machine and provide ABB to bus residents that require it.

1.74 When the arbiter is parked on the CPU, what timing should the arbiter observe when handing the bus to another device?

The arbiter should observe the same timing as in the general case.

1.75 If the arbiter negates the BG to the 750/750CX and asserts BG to a different master during the time between TS asserting and AACK asserting, will this guarantee another bus master will do the next transfer?

Yes. However, it really doesn't matter; at most the CPU will run one additional cycle, in the case where the CPU BG negates while the CPU begins a bus cycle with TS. Other masters would in general follow the same timing, so the number of bus tenures will average out with fair arbitration and similar bus demands.

1.8 Little-Endian Mode Operation

1.81 When the 750 is operating in little-endian mode, does the MMU expect the page tables to be in little-endian format as well?

The answer is no. The little-endian logic is in the load/store unit; the page table walker logic bypasses the load/store unit and talks directly to the cache & bus interface unit. So, the page tables need to be built in big-endian format.

2. 6xx Questions

This section discusses questions specific to the 6xx processor family.

2.1 What is an Em603e?

The IBM Em603e is actually a family of processors; Em603-100, Em603e-166, and Em603e-200. The higher speed processors are implemented in a faster technology

Each Em603e processor is identical to the 603e processor of the same speed, with the exception that the floating point unit (FPU) and floating point register files are not supported. The following table outlines the different Em603e family members:

GENERAL	**603e 100MHz**	**603ev 166MHz**	**603e2 200MHz**
Technology	0.5um CMOS PID6	0.35 um CMOS PID7v	0.35 um CMOS PID7t
Die Size	98 sq mm	79 sq mm	79 sq mm
# of transistors	2.6 million	2.6 million	2.6 million
Tj (junction temp)	0 - 105 deg C	0 - 105 deg C	0 - 105 deg C
Power Dissipation	3.2/4.0 W @100MHz	3.0/4.0 W @ 166MHz	4.0/5.0 W @ 200MHz
Signals	165	165	165
Package	255 CBGA 240 PFP w/MHS	255 CBGA	255 CBGA
Voltages (+/- 5%)	3.3 V core 3.3 V I/O	2.5 V core 3.3 V I/O	2.5 V core 3.3 V I/O
PVR number	0x0006	0x0007	0x0007
Part Number	IBM25EMPPC 603eBG100F, IBM25EMPPC 603ePG100F	IBM25EMPPC 603eBC166F	IBM25EMPPC 603e2BB200F

2.2 What are the limitations of the Em603e?

Proper operation of the floating point arithmetic unit is not guaranteed for the Em603e parts. Software should either be modified so that all floating point instructions are removed, or floating point library software should be used to emulate the floating point instructions. The default state of MSR[FP] = 0 should not be changed; this default setting will cause a floating point exception to be generated when any floating point load, store, or arithmetic instruction is executed. The exception handler can then call the appropriate library routine to emulate the fp instruction.

Printed in the United States of America February 2002