System Management Mode

Transmeta Corporation
3940 Freedom Circle
Santa Clara, CA 95054

Title: System Management Mode
Date: 2002/04/25
Purpose: This document describes the fundamental aspects of System Management Mode. It provides the State Save Map layout, a description of the initial processor state after SMI, details on how to deal with external interrupts during SMM, and information on resume features like auto-halt restart and I/O instruction restart. However, a full description of how to use SMM is beyond the scope of this document.

System Management Mode

Property of:
Transmeta Corporation
3940 Freedom Circle
Santa Clara, CA 95054
USA
(408) 919-3000
http://www.transmeta.com

The information contained in this document is provided solely for use in connection with Transmeta products, and Transmeta reserves all rights in and to such information and the products discussed herein. This document should not be construed as transferring or granting a license to any intellectual property rights, whether express, implied, arising through estoppel or otherwise. Except as may be agreed in writing by Transmeta, all Transmeta products are provided "as is" and without a warranty of any kind, and Transmeta hereby disclaims all warranties, express or implied, relating to Transmeta's products, including, but not limited to, the implied warranties of merchantability, fitness for a particular purpose and non-infringement of third party intellectual property. Transmeta products may contain design defects or errors which may cause the products to deviate from published specifications, and Transmeta documents may contain inaccurate information. Transmeta makes no representations or warranties with respect to the accuracy or completeness of the information contained in this document, and Transmeta reserves the right to change product descriptions and product specifications at any time, without notice.

Transmeta products have not been designed, tested, or manufactured for use in any application where failure, malfunction, or inaccuracy carries a risk of death, bodily injury, or damage to tangible property, including, but not limited to, use in factory control systems, medical devices or facilities, nuclear facilities, aircraft, watercraft or automobile navigation or communication, emergency systems, or other applications with a similar degree of potential hazard.

Transmeta reserves the right to discontinue any product or product document at any time without notice, or to change any feature or function of any Transmeta product or product document at any time without notice.

System Management Mode

Introduction

When a System Management Interrupt (SMI) is recognized, the processor saves a substantial part of its current state in the State Save Map (SSM), initializes some registers to provide the SMM execution environment, and then begins execution inside SMM.

The execution inside SMM ends when the processor encounters the next RSM instruction. At that point it restores a substantial part of its state from the SSM, and then resumes operation. Optionally, the resume process may include the use of the auto-halt restart or I/O instruction restart feature.

State Save Map Layout

The State Save Map (SSM) consists of a 512 byte region that starts at offset 7E00h and ends at offset 7FFFh, relative to the SMM entry point, which in turn is equal to SMBASE + 8000h.

Not all of the processor's state is held by the SSM; the following are some of the items that are not part of the SSM, and thus need to be saved and restored separately:

- control register CR2
- debug registers DR0, DR1, DR2, and DR3
- FPU and MMX registers
- Model-Specific Registers (MSRs)

If software uses SMM for the purpose of implementing a power management state during which power is removed from the processor, then the entire 512 byte SSM must be saved and restored, to ensure proper operation. Locations that are not listed in the following table should be treated as reserved.

offset	size	description
7EF8h	dword	SMBASE field
7EFCh	dword	SMM revision ID field
7F00h	word	I/O instruction restart field
7F02h	word	auto-halt restart field
7F88h	dword	GDTR base
7F94h	dword	IDTR base
7FA8h	dword	ES selector
7FACh	dword	CS selector
7FB0h	dword	SS selector
7FB4h	dword	DS selector
7FB8h	dword	FS selector
7FBCh	dword	GS selector
7FC0h	dword	LDTR selector
7FC4h	dword	TR selector
7FC8h	dword	DR7
7FCCh	dword	DR6
7FD0h	dword	EAX
7FD4h	dword	ECX
7FD8h	dword	EDX
7FDCh	dword	EBX
7FE0h	dword	ESP
7FE4h	dword	EBP
7FE8h	dword	ESI
7FECh	dword	EDI
7FF0h	dword	EIP
7FF4h	dword	EFLAGS
7FF8h	dword	CR3
7FFCh	dword	CR0

System Management Mode

SMM revision ID field							
31	reserved	18	17	16	15	SMM revision identifier	0

bit	description	value
17	SMBASE relocation is supported	1
16	I/O instruction restart is supported	1
15...0	SMM revision identifier	0002h

I/O instruction restart field					
31	reserved	16	15	I/O instruction restart	0

bit	description
15...0	0000h if processor should not restart I/O instruction 00FFh if processor should restart I/O instruction

auto-halt restart field			
31	reserved	1	0

bit	description
0	0 if processor was not in halt state, 1 if processor was in halt state 0 if processor should not resume to halt state, 1 if processor should resume to halt state

Initial Processor State after SMI

The following registers are set to fixed values after an SMI, to provide the initial SMM execution environment. Treat any registers that are not listed in this table as reserved.

register	value after SMI
CS selector	3000h
CS base	SMBASE
CS limit	FFFF_FFFFh
CS access rights	009Bh [#1]
SS, DS, ES, FS, GS selector	0000h
SS, DS, ES, FS, GS base	0000_0000h
SS, DS, ES, FS, GS limit	FFFF_FFFFh
SS, DS, ES, FS, GS access rights	0093h
EFLAGS	0000_0002h
EIP	0000_8000h
CR0	PE, EM, TS, and PG cleared
CR4	0000_0000h
DR7	0000_0400h
TEMP_DR6 (if supported)	0000_0000h
TSC	continues to increment
IN_REP	false
IN_SMM	true
IN_HLT	false
IN_SHUTDOWN	n/a
IN_FP_FREEZE	false
SUPPRESS_INTERRUPTS	false
BLOCK_INIT	true
BLOCK_SMI	true
BLOCK_NMI	true
LATCH_INIT	true if INIT recognized together with SMI, else false
LATCH_SMI	false
LATCH_NMI	true if NMI recognized together with SMI, else false
FERR#	unmodified
A20M#	unmasked [#2]

processor caches	coherency is ensured, no WBINVD is required

notes	description
#1	Whether or not CS is writable depends on the processor mode. In real and virtual 8086 mode, CS is always writable; but CS can never be written in protected mode.
#2	Changing the state of A20M# is not supported while executing inside SMM.

External Interrupts in SMM

The value of EFLAGS after SMI implies that INTRs are disabled. They can be enabled inside SMM by setting EFLAGS.IF=1. If SMI is recognized inside SMM, then one occurrence is latched, and serviced back-to-back after RSM has been executed. Finally, both NMI and INIT are blocked inside SMM. Again one occurrence of either interrupt can be latched, while servicing NMI and INIT can be enabled by executing an IRET or RSM instruction.

Resume Features

SMBASE Relocation

The base address of the SMRAM is contained in an internal register called the SMBASE register. The default value after a processor RESET is 0003_0000h (refer to the Initial Processor State documentation for details). The SMBASE register is part of the SSM.

The contents of the SMBASE register can be changed by writing the SMBASE field’s value in the SSM, and executing a RSM instruction. The new SMBASE value takes effect on the next assertion of SMI.

The SMBASE value must be aligned to a 32 KB boundary; trying to load a misaligned value with the RSM instruction causes a processor shutdown.

Auto-Halt Restart

The processor may have executed a HLT instruction prior to servicing an SMI; that is, the processor may have been in the halt state. If so, then this fact is recorded in the auto-halt restart field of the SSM.

The SMM handler may clear the auto-halt restart field prior to executing the RSM instruction, to prevent entering the halt state again after RSM. Entering the halt state again after RSM provokes a halt bus transaction. This results in multiple halt bus transactions for the same HLT instruction.

I/O Instruction Restart

The processor may have attempted to execute an I/O instruction just prior to servicing an SMI. For example, the I/O instruction may have tried to access a powered-down device, in response to which the system logic asserted the SMI signal to the processor, without completing the I/O instruction itself. If so, then the SMM handler may choose to re-execute the I/O instruction after RSM.

The SMM handler may set the I/O instruction restart field prior to executing the RSM instruction, in order to re-execute an I/O instruction. It is the responsiblity of the SMM handler to detect the I/O instruction and to control the I/O instruction restart field.

The processor stores additional information in the SSM, to support the I/O instruction restart feature for the INS and OUTS instructions.

The behavior of setting the I/O instruction restart field without actually resuming to an I/O instruction is unspecified.

END