Table 6-2. Exceptions and Conditions—Overview

Exception Type	Vector Offset (hex)	Causing Conditions
System reset	00100	The causes of system reset exceptions are implementation-dependent. If the conditions that cause the exception also cause the processor state to be corrupted such that the contents of SRR0 and SRR1 are no longer valid or such that other processor resources are so corrupted that the processor cannot reliably resume execution, the copy of the RI bit copied from the MSR to SRR1 is cleared.
Machine check	00200	The causes for machine check exceptions are implementation-dependent, but typically these causes are related to conditions such as bus parity errors or attempting to access an invalid physical address. Typically, these exceptions are triggered by an input signal to the processor. Note that not all processors provide the same level of error checking. The machine check exception is disabled when MSR[ME] = 0. If a machine check exception condition exists and the ME bit is cleared, the processor goes into the checkstop state. If the conditions that cause the exception also cause the processor state to be corrupted such that the contents of SRR0 and SRR1 are no longer valid or such that other processor resources are so corrupted that the processor cannot reliably resume execution, the copy of the RI bit written from the MSR to SRR1 is cleared. (Note that physical address is referred to as real address in the architecture specification.)
DSI	00300	A DSI exception occurs when a data memory access cannot be performed for any of the reasons described in Section 6.4.3, “DSI Exception (0x00300).” Such accesses can be generated by load/store instructions, certain memory control instructions, and certain cache control instructions.
ISI	00400	An ISI exception occurs when an instruction fetch cannot be performed for a variety of reasons described in Section 6.4.4, “ISI Exception (0x00400).”
External interrupt	00500	An external interrupt is generated only when an external interrupt is pending (typically signalled by a signal defined by the implementation) and the interrupt is enabled (MSR[EE] = 1).
Alignment	00600	An alignment exception may occur when the processor cannot perform a memory access for reasons described in Section 6.4.6, “Alignment Exception (0x00600).” Note that an implementation is allowed to perform the operation correctly and not cause an alignment exception.

Table 6-2. Exceptions and Conditions—Overview (Continued)

Exception Type	Vector Offset (hex)	Causing Conditions
Program	00700	A program exception is caused by one of the following exception conditions, which correspond to bit settings in SRR1 and arise during execution of an instruction: • Floating-point enabled exception—A floating-point enabled exception condition is generated when MSR[FE0–FE1] ≠ 00 and FPSCR[FEX] is set. The settings of FE0 and FE1 are described in Table 6-3. FPSCR[FEX] is set by the execution of a floating-point instruction that causes an enabled exception or by the execution of a Move to FPSCR instruction that sets both an exception condition bit and its corresponding enable bit in the FPSCR. These exceptions are described in Section 3.3.6, “Floating-Point Program Exceptions.” • Illegal instruction—An illegal instruction program exception is generated when execution of an instruction is attempted with an illegal opcode or illegal combination of opcode and extended opcode fields or when execution of an optional instruction not provided in the specific implementation is attempted (these do not include those optional instructions that are treated as no-ops). The PowerPC instruction set is described in Chapter 4, “Addressing Modes and Instruction Set Summary.” See Section 6.4.7, “Program Exception (0x00700),” for a complete list of causes for an illegal instruction program exception. • Privileged instruction—A privileged instruction type program exception is generated when the execution of a privileged instruction is attempted and the MSR user privilege bit, MSR[PR], is set. This exception is also generated for **mtspr** or **mfspr** with an invalid SPR field if spr[0] = 1 and MSR[PR] = 1. • Trap—A trap type program exception is generated when any of the conditions specified in a trap instruction is met. For more information, refer to Section 6.4.7, “Program Exception (0x00700).”
Floating-point unavailable	00800	A floating-point unavailable exception is caused by an attempt to execute a floating-point instruction (including floating-point load, store, and move instructions) when the floating-point available bit is cleared, MSR[FP] = 0.
Decrementer	00900	The decrementer interrupt exception is taken if the exception is enabled (MSR[EE] = 1), and it is pending. The exception is created when the most-significant bit of the decrementer changes from 0 to 1. If it is not enabled, the exception remains pending until it is taken.
Reserved	00A00	This is reserved for implementation-specific exceptions. For example, the 601 uses this vector offset for direct-store exceptions.
Reserved	00B00	—
System call	00C00	A system call exception occurs when a System Call (**sc**) instruction is executed.
Trace	00D00	Implementation of the trace exception is optional. If implemented, it occurs if either the MSR[SE] = 1 and almost any instruction successfully completed or MSR[BE] = 1 and a branch instruction is completed. See Section 6.4.11, “Trace Exception (0x00D00),” for more information.
Floating-point assist	00E00	Implementation of the floating-point assist exception is optional. This exception can be used to provide software assistance for infrequent and complex floating-point operations such as denormalization.
Reserved	00E10–00FFF	—
Reserved	01000–02FFF	This is reserved for implementation-specific purposes. May be used for implementation-specific exception vectors or other uses.

Table 6-5 shows the bit definitions for the MSR.

Table 6-5. MSR Bit Settings

Bit(s)	Name	Description
0–12	—	Reserved
13	POW	Power management enable 0 Power management disabled (normal operation mode). 1 Power management enabled (reduced power mode). **Note**: Power management functions are implementation-dependent. If the function is not implemented, this bit is treated as reserved.
14	—	Reserved
15	ILE	Exception little-endian mode. When an exception occurs, this bit is copied into MSR[LE] to select the endian mode for the context established by the exception.
16	EE	External interrupt enable 0 While the bit is cleared the processor delays recognition of external interrupts and decrementer exception conditions. 1 The processor is enabled to take an external interrupt or the decrementer exception.
17	PR	Privilege level 0 The processor can execute both user- and supervisor-level instructions. 1 The processor can only execute user-level instructions.
18	FP	Floating-point available 0 The processor prevents dispatch of floating-point instructions, including floating-point loads, stores, and moves. 1 The processor can execute floating-point instructions.
19	ME	Machine check enable 0 Machine check exceptions are disabled. 1 Machine check exceptions are enabled.
20	FE0	Floating-point exception mode 0 (see Table 2-10).
21	SE	Single-step trace enable (Optional) 0 The processor executes instructions normally. 1 The processor generates a single-step trace exception upon the successful execution of the next instruction. Note: If the function is not implemented, this bit is treated as reserved.
22	BE	Branch trace enable (Optional) 0 The processor executes branch instructions normally. 1 The processor generates a branch trace exception after completing the execution of a branch instruction, regardless of whether or not the branch was taken. Note: If the function is not implemented, this bit is treated as reserved.
23	FE1	Floating-point exception mode 1 (See Table 2-10).
24	—	Reserved
25	IP	Exception prefix. The setting of this bit specifies whether an exception vector offset is prepended with Fs or 0s. In the following description, *nnnnn* is the offset of the exception vector. See Table 6-2. 0 Exceptions are vectored to the physical address 0x000*n_nnnn* . 1 Exceptions are vectored to the physical address 0xFFF*n_nnnn*. In most systems, IP is set to 1 during system initialization, and then cleared to 0 when initialization is complete.

Table 6-5. MSR Bit Settings (Continued)

Bit(s)	Name	Description
26	IR	Instruction address translation 0 Instruction address translation is disabled. 1 Instruction address translation is enabled. For more information see Chapter 7, “Memory Management.”
27	DR	Data address translation 0 Data address translation is disabled. 1 Data address translation is enabled. For more information see Chapter 7, “Memory Management.”
28–29	—	Reserved
30	RI	Recoverable exception (for system reset and machine check exceptions). 0 Exception is not recoverable. 1 Exception is recoverable. For more information see Section 6.4.1, “System Reset Exception (0x00100),”and Section 6.4.2, “Machine Check Exception (0x00200).”
31	LE	Little-endian mode enable 0 The processor runs in big-endian mode. 1 The processor runs in little-endian mode.

Those MSR bits that are written to SRR1 are written when the first instruction of the exception handler is encountered. The data address register (DAR) is used by several exceptions (for example, DSI and alignment exceptions) to identify the address of a memory element.

6.2.1 Enabling and Disabling Exceptions

When a condition exists that may cause an exception to be generated, it must be determined whether the exception is enabled for that condition as follows:

- IEEE floating-point enabled exceptions (a type of program exception) are ignored when both MSR[FE0] and MSR[FE1] are cleared. If either of these bits is set, all IEEE enabled floating-point exceptions are taken and cause a program exception.
- Asynchronous, maskable exceptions (that is, the external and decrementer interrupts) are enabled by setting the MSR[EE] bit. When MSR[EE] = 0, recognition of these exception conditions is delayed. MSR[EE] is cleared automatically when an exception is taken to delay recognition of conditions causing those exceptions.
- A machine check exception can only occur if the machine check enable bit, MSR[ME], is set. If MSR[ME] is cleared, the processor goes directly into checkstop state when a machine check exception condition occurs.

6.2.2 Steps for Exception Processing

After it is determined that the exception can be taken (by confirming that any instruction-caused exceptions occurring earlier in the instruction stream have been handled, and by confirming that the exception is enabled for the exception condition), the processor does the following:

1. The machine status save/restore register 0 (SRR0) is loaded with an instruction address that depends on the type of exception. See the individual exception description for details about how this register is used for specific exceptions.
2. SRR1 bits 1–4 and 10–15 are loaded with information specific to the exception type.
3. MSR bits 16–23, 25–27, and 30–31 are loaded with a copy of the corresponding bits of the MSR. Note that depending on the implementation, additional bits from the MSR may be saved in SRR1.
4. The MSR is set as described in Table 6-7. The new values take effect beginning with the fetching of the first instruction of the exception-handler routine located at the exception vector address.

   Note that MSR[IR] and MSR[DR] are cleared for all exception types; therefore, address translation is disabled for both instruction fetches and data accesses beginning with the first instruction of the exception-handler routine.

   Also, note that the MSR[ILE] bit setting at the time of the exception is copied to MSR[LE] when the exception is taken (as shown in Table 6-7).
5. Instruction fetch and execution resumes, using the new MSR value, at a location specific to the exception type. The location is determined by adding the exception's vector offset (see Table 6-2) to the base address determined by MSR[IP]. If IP is cleared, exceptions are vectored to the physical address 0x000*n_nnnn*. If IP is set, exceptions are vectored to the physical address 0xFFF*n_nnnn*. For a machine check exception that occurs when MSR[ME] = 0 (machine check exceptions are disabled), the checkstop state is entered (the machine stops executing instructions). See Section 6.4.2, “Machine Check Exception (0x00200).”

In some implementations, any instruction fetch with MSR[IR] = 1 and any load or store with MSR[DR] = 1 may cause SRR0 and SRR1 to be modified.

6.4.5 External Interrupt (0x00500)

An external interrupt exception is signaled to the processor by the assertion of the external interrupt signal. The exception may be delayed by other higher priority exceptions or if the MSR[EE] bit is zero when the exception is detected. Note that the occurrance of this exception does not cancel the external request.

The register settings for the external interrupt exception are shown in Table 6-11.

Table 6-11. External Interrupt—Register Settings

Register	Setting Description
SRR0	Set to the effective address of the instruction that the processor would have attempted to execute next if no interrupt conditions were present.
SRR1	1–4 Cleared 10–15 Cleared 16–23 Loaded with equivalent bits from the MSR 25–27 Loaded with equivalent bits from the MSR 30–31 Loaded with equivalent bits from the MSR Note that depending on the implementation, additional bits in the MSR may be copied to SRR1.
MSR	POW 0 FP 0 BE 0 DR 0 ILE — ME — FE1 0 RI 0 EE 0 FE0 0 IP — LE Set to value of ILE PR 0 SE 0 IR 0

When an external interrupt exception is taken, instruction execution resumes at offset 0x00500 from the physical base address determined by MSR[IP].

6.4.6 Alignment Exception (0x00600)

This section describes conditions that can cause alignment exceptions in the processor. Similar to DSI exceptions, alignment exceptions use the SRR0 and SRR1 to save the machine state and the DSISR to determine the source of the exception. An alignment exception occurs when no higher priority exception exists and the implementation cannot perform a memory access for one of the following reasons:

- The operand of a floating-point load or store instruction is not word-aligned.
- The operand of **lmw**, **stmw**, **lwarx**, **stwcx.**, **eciwx**, or **ecowx** is not aligned.
- The instruction is **lmw**, **stmw**, **lswi**, **lswx**, **stswi**, or **stswx** and the processor is in little-endian mode.
- The operand of an elementary or string load or store crosses a protection boundary.
- The operand of **lmw** or **stmw** crosses a segment or BAT boundary.