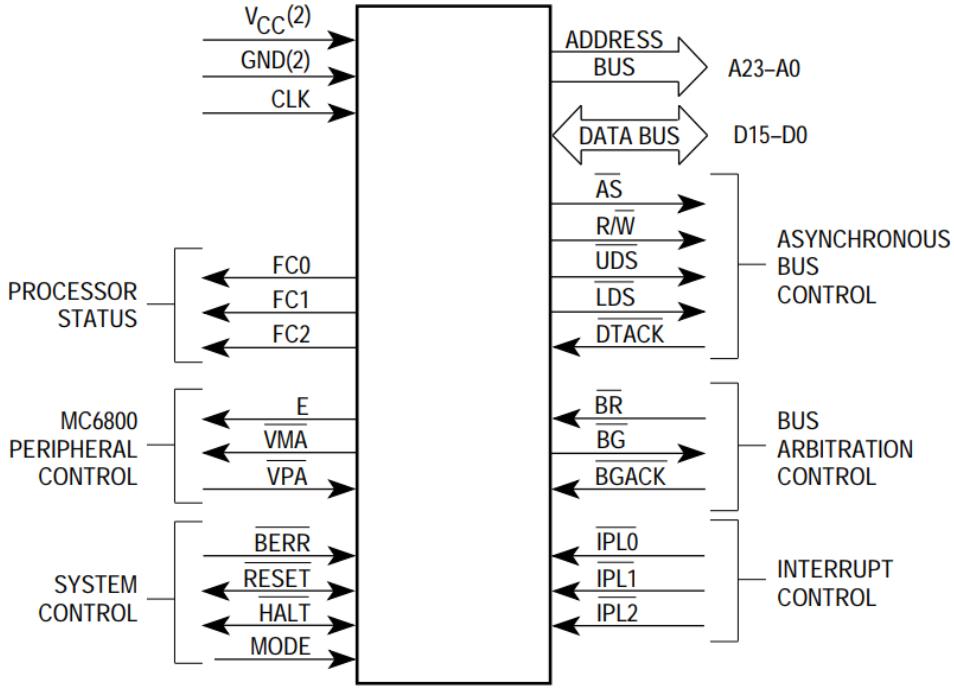
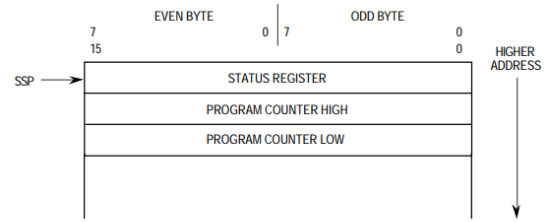


Function Code Output			
FC2	FC1	FC0	Address Space
0	0	0	(Undefined, Reserved)*
0	0	1	User Data
0	1	0	User Program
0	1	1	(Undefined, Reserved)*
1	0	0	(Undefined, Reserved)*
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	CPU Space

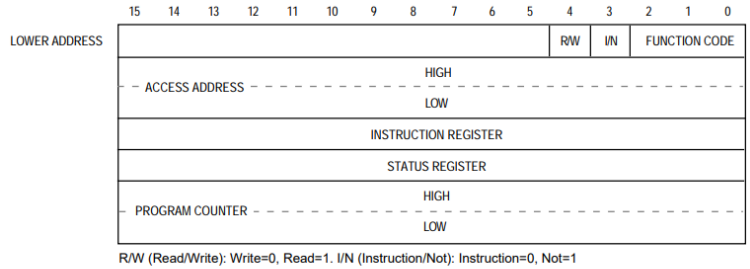
*Address space 3 is reserved for user definition, while 0 and 4 are reserved for future use by Motorola.

Vectors Numbers		Address			
Hex	Decimal	Dec	Hex	Space[6]	Assignment
0	0	0	000	SP	Reset: Initial SSP[2]
1	1	4	004	SP	Reset: Initial PC[2]
2	2	8	008	SD	Bus Error
3	3	12	00C	SD	Address Error
4	4	16	010	SD	Illegal Instruction
5	5	20	014	SD	Zero Divide
6	6	24	018	SD	CHK Instruction
7	7	28	01C	SD	TRAPV Instruction
8	8	32	020	SD	Privilege Violation
9	9	36	024	SD	Trace
A	10	40	028	SD	Line 1010 Emulator
B	11	44	02C	SD	Line 1111 Emulator
C	12[1]	48	030	SD	(Unassigned, Reserved)
D	13[1]	52	034	SD	(Unassigned, Reserved)
E	14	56	038	SD	Format Error[5]
F	15	60	03C	SD	Uninitialized Interrupt Vector
10–17	16–23[1]	64	040	SD	(Unassigned, Reserved)
		92	05C		—
18	24	96	060	SD	Spurious Interrupt[3]
19	25	100	064	SD	Level 1 Interrupt Autovector
1A	26	104	068	SD	Level 2 Interrupt Autovector
1B	27	108	06C	SD	Level 3 Interrupt Autovector
1C	28	112	070	SD	Level 4 Interrupt Autovector
1D	29	116	074	SD	Level 5 Interrupt Autovector
1E	30	120	078	SD	Level 6 Interrupt Autovector
1F	31	124	07C	SD	Level 7 Interrupt Autovector
20–2F	32–47	128	080	SD	TRAP Instruction Vectors[4]
		188	0BC		—
30–3F	48–63[1]	192	0C0	SD	(Unassigned, Reserved)
		255	0FF		—
40–FF	64–255	256	100	SD	User Interrupt Vectors
		1020	3FC		—

Group	Exception	Processing
0	Reset Address Error Bus Error	Exception Processing Begins within Two Clock Cycles
1	Trace Interrupt Illegal Privilege	Exception Processing Begins before the Next Instruction
2	TRAP, TRAPV, CHK Zero Divide	Exception Processing Is Started by Normal Instruction Execution



Group 1 and 2 Exception Stack Frame



R/W (Read/Write): Write=0, Read=1. I/N (Instruction/Not): Instruction=0, Not=1

Supervisor Stack Order for Bus or Address Error Exception

NOTES:
1. Vector numbers 12, 13, 16–23, and 48–63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.
2. Reset vector (0) requires four words, unlike the other vectors which only require two words, and is located in the supervisor program space.
3. The spurious interrupt vector is taken when there is a bus error indication during interrupt processing.
4. TRAP #n uses vector number 32+ n.
5. MC68010 only. This vector is unassigned, reserved on the MC68000 and MC68008.
6. SP denotes supervisor program space, and SD denotes supervisor data space.

Exception Processing Sequence

In the **first step** of exception processing, an internal copy is made of the status register. After the copy is made, the S bit of the status register is set, putting the processor into the supervisor mode. Also, the T bit is cleared, which allows the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated appropriately.

In the **second step**, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor bus cycle classified as an interrupt acknowledge cycle. For all other exceptions, internal logic provides the vector number. This vector number is then used to calculate the address of the exception vector.

The **third step**, except for the reset exception, is to save the current processor status. (The reset exception does not save the context and skips this step.) The current program counter value and the saved copy of the status register are stacked using the SSP. The stacked program counter value usually points to the next unexecuted instruction. However, for bus error and address error, the value stacked for the program counter is unpredictable and may be incremented from the address of the instruction that caused the error. Group 1 and 2 exceptions use a short format exception stack frame. Additional information defining the current context is stacked for the bus error and address error exceptions.

The **last step** is the same for all exceptions. The new program counter value is fetched from the exception vector. The processor then resumes instruction execution. The instruction at the address in the exception vector is fetched, and normal instruction decoding and execution is started.

Reset

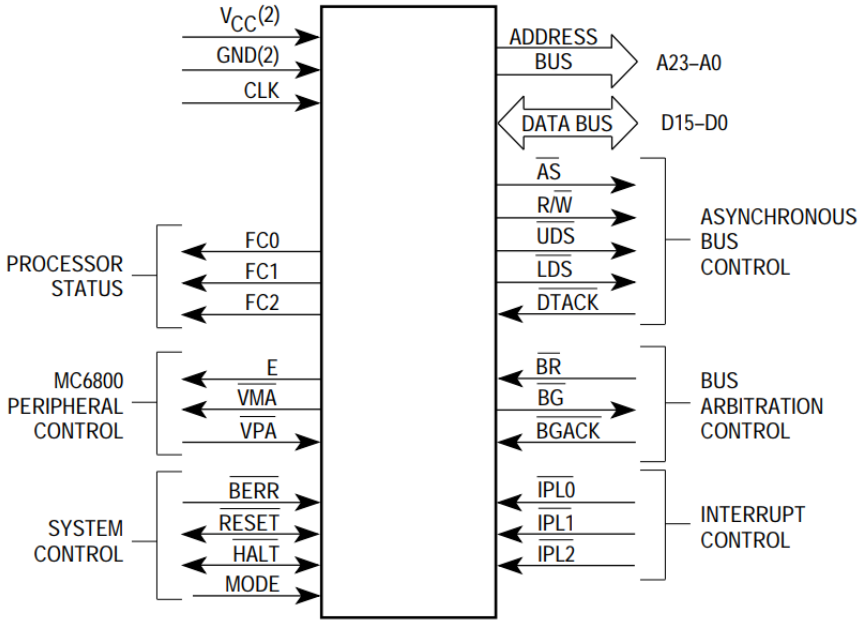
The reset exception corresponds to the highest exception level. The processing of the reset exception is performed for system initiation and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state, and the trace state is forced off. The interrupt priority mask is set at level 7. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the SSP, neither the program counter nor the status register is saved. The address in the first two words of the reset exception vector is fetched as the initial SSP, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, instruction execution is started at the address in the program counter. The initial program counter should point to the powerup/restart code. The RESET instruction does not cause a reset exception; it asserts the RESET signal to reset external devices, which allows the software to reset the system to a known state and continue processing with the next instruction.

Address Error

An address error exception occurs when the processor attempts to access a word or longword operand or an instruction at an odd address. An address error is similar to an internally generated bus error. The bus cycle is aborted, and the processor ceases current processing and begins exception processing. The exception processing sequence is the same as that for a bus error, including the information to be stacked, except that the vector number refers to the address error vector. Likewise, if an address error occurs during the exception processing for a bus error, address error, or reset, the processor is halted. The user must be certain that the proper corrections have been made to the stack image and user registers before attempting to continue the instruction. With proper software handling, the address error exception handler could emulate word or long-word accesses to odd addresses if desired.

Bus Error

Exception processing for a bus error follows the usual sequence of steps. The status register is copied, the supervisor mode is entered, and tracing is turned off. The vector number is generated to refer to the bus error vector. Since the processor is fetching the instruction or an operand when the error occurs, the context of the processor is more detailed. To save more of this context, additional information is saved on the supervisor stack. The program counter and the copy of the status register are saved. The value saved for the program counter is advanced 2–10 bytes beyond the address of the first word of the instruction that made the reference causing the bus error. If the bus error occurred during the fetch of the next instruction, the saved program counter has a value in the vicinity of the current instruction, even if the current instruction is a branch, a jump, or a return instruction. In addition to the usual information, the processor saves its internal copy of the first word of the instruction being processed and the address being accessed by the aborted bus cycle. Specific information about the access is also saved: type of access (read or write), processor activity (processing an instruction), and function code outputs when the bus error occurred. The processor is processing an instruction if it is in the normal state or processing a group 2 exception; the processor is not processing an instruction if it is processing a group 0 or a group 1 exception. If a bus error occurs during the last step of exception processing, while either reading the exception vector or fetching the instruction, the value of the program counter is the address of the exception vector. Although this information is not generally sufficient to effect full recovery from the bus error, it does allow software diagnosis. Finally, the processor commences instruction processing at the address in the vector. It is the responsibility of the error handler routine to clean up the stack and determine where to continue execution. If a bus error occurs during the exception processing for a bus error, an address error, or a reset, the processor halts and all processing ceases. This halt simplifies the detection of a catastrophic system failure, since the processor removes itself from the system to protect memory contents from erroneous accesses. Only an external reset operation can restart a halted processor.



Instruction Traps

Traps are exceptions caused by instructions; they occur when a processor recognizes an abnormal condition during instruction execution or when an instruction is executed that normally traps during execution. Exception processing for traps is straightforward. The status register is copied; the supervisor mode is entered; and tracing is turned off. The vector number is internally generated; for the TRAP instruction, part of the vector number comes from the instruction itself. The program counter, and the copy of the status register are saved on the supervisor stack. The saved value of the program counter is the address of the instruction following the instruction that generated the trap. Finally, instruction execution commences at the address in the exception vector. Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception and is useful for implementing system calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a run-time error, which may be an arithmetic overflow or a subscript out of bounds. A signed divide (DIVS) or unsigned divide (DIVU) instruction forces an exception if a division operation is attempted with a divisor of zero.