68060 ERRATA

Rev 4.0 - 10/18/96

060 Production Masksets

	68060 68LC060 68EC060	68060 68LC060 68EC060	68LC060 68EC060	68060
Errata #	**1F43G**	**1G65V**	**2G59Y**	**0E41J**
I11	X	X		
I12a	X	X		
I12b	X	X		
I13	X	X		
I14	X	X		
I15	X	X		
I16	X	X		
F5	X	X		
F6	X			
E2	X	X		
H4	X	X		
H5	X	X		
H6	X	X		
J2	X	X		
J3	X	X		
J4	X	X		
P4	X	X		
P9	X	X		
P12	X	X		
P13	X	X		
G3	X	X		
G4	N/A	N/A	X	N/A

X	=	Errata exists in this maskset
N/A	=	Feature being errata'ed is not available on this maskset

Notes:

LC060 and EC060s are produced off the 1F43G and 1G65V masksets until the 2G59Y maskset is available.

Integer Instruction Processing

I11 - A Move to USP instruction (opcodes of $4E60-$4E67), which moves the contents of an address register to the User Stack Pointer in Supervisor Mode, will be erroneously executed if it is the target of a wrong-way predicted, zero-cycle conditional branch instruction.

Workarounds include:

Do not use the MOVE An,USP instruction (opcodes = $4E60-$4E67) to load the User Stack Pointer - MOVEC Rn,USP ($4E7B x800) can be used as an alternative method to load the USP --OR--

Do not use the MOVE An,USP instruction (opcodes = $4E60-$4E67) as the target instruction (first instruction at the destination) of a conditional branch.

I12a - If a floating point pre-instruction exception on a fbcc instruction occurs on the same cycle that the 68060 asserts the transfer start signal (TS) for a bus write cycle during an instruction side tablewalk (update of the U bits in page descriptors of table descriptors), the referenced descriptor in memory will be corrupted.

Workarounds include:

Guarantee that no fbcc floating point pre-instruction exceptions occur --OR--

Do not use instruction side virtual addressing --OR--

Build all instruction side page tables with the used (U bit) set in the page and table descriptors.

I12b - If an operand pipeline detected fault occurs on the same cycle that the 68060 asserts the (TS) of a locked bus cycle for an instruction side tablewalk for any mode other than full-speed bus with 0 or 1 wait states, the processor will change its outputs mid-bus cycle (after the assertion of TS and before the assertion of TA or TEA) and drive undefined information onto the address, attribute, data, and control buses. This can result in the bus hanging, etc.

Operand pipeline detected exceptions are:

- Integer divide-by-zero
- Check
- TRAPcc, TRAPV
- Format Error
- Floating point post-instruction exceptions (which can be generated on fmove operations register-to-memory or on fbcc instructions)

Workarounds include:

Operate the 68060 in full-speed bus mode with 0 or 1 wait state --OR--

Do not use instruction side virtual addressing --OR--

Build all instruction side page tables with the used (U bit) set in the page and table descriptors --AND EITHER--

Do not use the bus control register (BUSCR) to force locked bus cycles --OR--

Guarantee that no instruction side tablewalks can occur while the BUSCR locked bus bit is set.

I13 - If a TRAPcc instruction is immediately followed by an instruction that does an operand read and the operand read is either cache inhibited or gets a cache miss and these two instructions are dispatched as a Superscalar pair and the trap condition is true, then the bus read cycle for the instruction following the TRAPcc will occur, i.e., this bus read will not be canceled by the fault caused by the trap instruction as it should. Additionally, if the bus is locked for this bus read cycle due to the lock bit in the BUSCR register being asserted, the processor will change its outputs mid-bus cycle and drive undefined information onto the address, attribute, control buses; this can result in the bus hanging.

Workarounds include:

If the extra bus read operations are not a problem, do not use TRAPcc instructions while using the BUSCR register to assert bus lock -- OR --

Do not use TRAPcc instructions followed by operand read instructions. (NOTE: the tpf = trap on condition false instruction may be used, since the trap condition is never true).

I14 - Execution of the following 4 instruction sequence can result in processor usage of stale data:

```
op.l    <ea>,Dx         % instruction modifying data register Dx
mov.l   Dx,<mem.a>      % move of Dx to memory address "a"
mov.l   <mem.a>,Ry      % move of the contents of memory address "a" to general register Ry
op.l    <ea>,Ry         % instruction that has Ry as the destination register
```

If this four instruction sequence is dispatched to the operand pipelines as two consecutive superscalar pairs, the copy of Ry prior to the load from memory will be erroneously used by the "op.l ,Ry" instruction. This sequence exercises three 68060 pipeline, result forwarding performance optimizations simultaneously; this specific combination encounters the failure. Note that all four instructions must be ".l" for the failure to be present.

Workaround:

Set Processor Control Register bit 5 = 1. For xF43G and xG65V masksets, PCR(5) = 1 disables one of the specific 68060 result forwarding features, store/load bypass. The store/load bypass feature, which eliminates a pipeline stall in scenarios of successive instructions storing and loading back data from the same address, should have very limited performance impact on most programs.

I15 - Execution of the following 3 instruction sequence in a cache inhibited mode can result in the 68060 ignoring externally sourced read data and using internal data instead:

```
mov.l    <ea>,<mem.b>   % move of data to memory address "b"
{either no intervening or one intervening instruction that can execute in the secondary OEP}
mov.l    <mem.b>,Rz     % move of the contents of memory address "b" to general register Rz
op.l     Rz             % instruction that uses Rz
```

If this three instruction sequence is dispatched to the operand pipelines in two consecutive machine cycles and the cache mode is cache inhibited precise or imprecise, the data used for Rz by the "op.l Rz" instruction will be an internal copy of the data stored to memory location "b", not the data read from external memory by the "mov.l <mem.b>,Rz" instruction. For example, if a program attempts to test an external memory by writing data to a location and then immediately reading the data back from that location with this instruction sequence, the result will be that the external memory is not tested. Note that all three instructions must be ".l" for the failure to be present.

Workaround:

Set Processor Control Register bit 5 = 1. For xF43G and xG65V masksets, PCR(5) = 1 disables one of the specific 68060 result forwarding features, store/load bypass. The store/load bypass feature, which eliminates a pipeline stall in scenarios of successive instructions storing and loading back data from the same address, should have very limited performance impact on most programs --OR--

Do not use cache inhibited precise or imprecise modes.

I16 - If an instruction uses a memory indirect addressing mode to specify its source and/or destination operand, and it meets ALL of the following conditions then the next sequential instruction may fail. Instructions which do not utilize any of the memory indirect addressing modes are NOT affected by this errata.

The instruction uses full-format effective addressing with memory indirection:
- Memory Indirect Postindexed Mode
- Memory Indirect Preindexed Mode
- Program Counter Memory Indirect Postindexed Mode
- Program Counter Memory Indirect Preindexed Mode --AND--

The addressing mode uses a one word base displacement --AND--

The addressing mode uses a two word outer displacement --AND--

The instruction alignment is as follows:

```
 31           16  15             0
| "instr word Z" | FF              |
| base.w         | outer.l (31:16) |
| outer.l (15:0) |                 |
```

where:
- "instr word Z" = 16-bits of the M68000 instruction immediately preceding the full-format extension word (for standard opcodes this is the 16-bit opword, for others it can be an extension word, immediate data, etc.).
- FF = full-format word
- base.w = one word base displacement
- outer.l (31:16) = upper 16 bits of two word outer displacement
- outer.l (15:0) = lower 16 bits of two word outer displacement --AND--

The bit pattern in outer.l (31:16) decodes as an opcode with multiple extensions.

The failure mode under this scenario is unpredictable - possible failures include processor lockup, unexpected exceptions and data corruption.

Workarounds include:

Avoid using memory indirect addressing modes. Note: newer compilers such as Greenhills, PCC, GNU, or Diab, do not emit memory indirect addressing modes.

--OR--

If memory indirect addressing modes must be used, either:

- Make all base displacements long in size (avoid word base displacements)

  -- OR --
- Make all outer displacements word in size (avoid 32-bit outer displacements).

  -- OR --
- Follow each instance of an instruction with memory indirect addressing with a "TPF" instruction.

Floating Point Instruction Processing

F5 - An FMOVE.P instruction, register-to-memory, using a dynamic k-factor can erroneously set the accrued overflow and inexact FPSR bits.

The execution of an FMOVE to packed decimal destination format with dynamic k-factor causes an unimplemented data type exception, invoking the 060SP emulation software package to perform the operation. If the source operand would overflow in double precision mode, the OVFL, AOVFL, and AINEX bits in the FPSR are erroneously set by the hardware (the 060SP cannot restore the accrued history bits, since they are "sticky" and could have been set by operations prior to the FMOVE.P).

F6 - Execution of the following five instruction sequence on the 68060 MAY result in the processor loading erroneous data into a floating point (FP) data register and/or result in an unexpected floating point exception:

```
        F<op.A>                         % where F<op.A> is a FP opcode with FPreg destination and with
execution time
                                %   greater  than or equal to 3 cycles
        MOVL    {mem}->Ax       % move long instruction loading Ax from memory
        MOVL    {mem}->Ay       % move long instruction loading Ay from memory
        F<op.B> (Ax)                    % any FP opcode using Ax to reference memory
        <op.C>  (Ay)                    % any opcode using Ay to reference memory
```

where:

F<op.A> = any floating point instruction with FPreg destination which takes 3 or more cycles to execute AFTER any memory operand is transferred to the floating point unit. These instructions are:

FADD, FSADD, FDADD, FSUB, FSSUB, FDSUB, FMUL, FSMUL, FDMUL, FSGLMUL, FDIV, FSDIV, FDDIV, FSGLDIV, FSQRT, FSSQRT, FDSQRT, FINT, FINTRZ, instructions with a memory or FPreg source operand of any data type, or

FABS, FCMP, FMOVE (into FPreg), FNEG, or FTST instructions if the source operand is an integer - requiring conversion to FP

MOVL = for this errata case, this second instruction must load the 32-bit address register used in the F<op.B> instruction

MOVL = for this errata case, this third instruction must load the 32-bit address register used in the <op.C> instruction

F<op.B> = any FP instruction in the 68060 instruction set that can access (read or write) an operand from memory using any form of address register indirect addressing mode

<op.C> = any instruction in the 68060 instruction set (integer or floating point) that can access (read or write) an operand from memory using any form of address register indirect addressing mode

This errata is due to an error in the 68060 operand execution pipeline control logic for "result forwarding". This logic implements a performance optimization that "forwards" the data from memory on an address register (long word) load to a subsequent instruction without incurring a "stall" (delay) to write the data from memory into the address register and read the address register back for the subsequent instruction.

In the five instruction sequence listed above, the value of the memory operand being loaded into address register (Ay) MAY be incorrectly forwarded as the base address for the memory access made by F<op.B>, (the correct value to be forwarded for F<op.B> is the value of the memory operand being loaded into address register (Ax)). When this error in register forwarding occurs, an incorrect memory address will be used to access the memory location specified in the F<op.B> instruction, and erroneous data will be stored into F<op.B>'s destination FPreg or memory location (data corruption). An unexpected FP exception may also occur. Rare cases can generate an unexpected FP exception even though correct data is stored into F<op.B>'s destination FPreg or memory location.

This errata will fail as described for cases where the initial F<op.A> instruction takes 4 or more cycles to execute (DIV, SQRT, or any of the listed instructions with an integer source operand). However, an additional condition must be met for this errata to fail where the initial F<op.A> takes exactly 3 cycles to complete execution - F<op.A> and MOVL must be dispatched internally as a "superscalar pair" by the 68060's internal, superscalar instruction dispatch hardware. A 3-cycle F<op.A> instruction which specifies a memory source operand cannot by dispatched for superscalar execution with the subsequent MOVL instruction (which also specifies a memory source operand).

This failure is present only in 68060 devices with on-chip floating point unit; 68EC060's and 68LC060's are NOT effected. Floating point instruction processing, as documented in the five instruction sequence above, is required for the failure to occur.

Workaround:

A work-around for all cases of this errata is to insert a NOP (or other 68060 pipeline synchronization instruction) after the F<op.A> instruction in the five instruction sequence described above. This will always ensure that the F<op.A> finishes execution before the execution of the subsequent MOVL instruction begins, and the error in the "result forwarding" logic will never be encountered.

Disabling superscalar dispatch in the 68060 Processor Configuration Register (setting PCR(0)=0) will insure that no 3-cycle F<op.A> instruction can incur this errata. It significantly reduces the number of errata occurrence scenarios, but does not eliminate the F<op.A> =greater than or equal to 4 cycles (DIV, SQRT, or any of the listed instructions with an integer source operand) scenarios.

Exception Processing

E2 - An address error exception caused by an exception vector table entry containing an odd byte address for any type of exception, except access error and address error, will experience erroneous generation of the vector table address for the address error. This erroneous address will result in transfer to an unpredictable address (PC) for the first instruction of the address error exception handler, and subsequently result in an unexpected exception, data corruption, the generation of a double-bus error, or other failure mode. Note, an address error on the PC fetch for an access fault or address error exception results in a double-bus error.

Workaround:

Ensure that all exception vectors in the exception vector table are aligned to an even byte address.

Hardware

H4 - In systems with at least one alternate master which asserts only BB and not TS during bus tenure the 68060 may incorrectly take over bus mastership. This occurs if BG is asserted to both the alternate master and 060 where the alternate master had, at some time in the past, used the bus, relinquished the bus and started to use the bus again with no intervening bus cycles that assert TS. In this case, the 060 may run bus cycles if an internal request is pending without waiting for the alternate master to finish using the bus and negate BB.

The following timing diagram will illustrate this case.

```
            |    |    |    |    |    |    |    |
_____      ______________________________________
AM_TS       |    |    |    |    |    |    |    |
_____      ____________    |    |    |    |    |
AM_BG       |    |    |\_________________________
_____       |    | _________    |    |    |    |
AM_BB      _______/   |    |\____________________
________   ______________________    |    |    |
68060_BG    |    |    |    |    |\_______________
________   ________________________________    |
68060_BR    |    |    |    |    |    |    |\____/
________   ________________________________    |
68060_TS    |    |    |    |    |    |    |\____/
```

Workarounds include:

Have the alternate master assert TS when it starts bus cycles

--OR--

Assure that the bus grants to an alternate master and 68060 are mutually exclusive

--OR--

Do not assert BG to the 68060 until the alternate master is done using the bus.

H5 - RSTI must be asserted for a minimum of 11 BCLK periods (instead of spec'ed value of 10 clocks) to properly reset the 68060.

H6 - In a multiple master system with the debug mode turned on (EDEBUG set in PCR register), where the 68060 implicitly owns the bus, the 68060 may erroneously drive the data bus for one clock after the clock in which bus grant is negated. (Although this can happen during the first clock of tenure by the alternate master, it is unlikely an alternate master will be transferring data during this clock.).

Workarounds include:

Do not set the EDEBUG bit in PCR register to enable debug mode --OR--

Do not allow the 68060 to implicitly own the bus while debug mode is enabled --OR--

Do not simultaneously negate bus grant to the 68060 (while the 68060 implicitly owns the bus) and assert bus grant to the alternate master, but insert an extra clock in which bus grant is negated to both the 68060 and alternate master --OR--

Guarantee the alternate master cannot drive the data bus during the first two clocks of its bus tenure.

JTAG

J2 - The HIGHZ instruction does not set chip outputs to the high impedance state on xF43G and xG65Vs. The debug pipe control mode HIZ instruction ($0D) also does not isolate chip inputs and tristate chip outputs.

J3 - When the TAP state machine is operated through the rising-edge of TCK leaving the CaptureDR state, all boundary scan registers will conduct a sample or capture operation. Since all output registers are of type BC_2, the sample for the output registers should capture the known data held in the Update register. The operation of the EXTEST instruction does not meet this specified 1149.1 functionality.

For the xF43G and xG65V mask sets, the boundary scan elements which are associated with the output signals of the Data Bus (D[31:0]) will incorrectly sample undefined data instead of the known Update register data. Any sampled information that is shifted out and associated with these cells should be ignored (don't cared). These cells are listed in Chapter 9 of the User's Manual in table 9.3 as boundary scan bit locations: 142, 144, 147, 149, 151, 153, 156, 158, 160, 162, 165, 167, 169, 171, 174, 176, 178, 180, 183, 185, 187, 189, 192, 194, 196, 198, 201, 203, 205, 207, 210, and 212.

Note: This is only a problem during the EXTEST instruction and does not affect operation of the SAMPLE instruction.

J4 - Specified 1149.1 functionality is not met for any instruction that operates the boundary scan register (EXTEST, SAMPLE/PRELOAD, LPSAMPLE) while the 68060 is in the functional LPSTOP state and the internal processor clock is stopped. {Note: LPSTOP state is indicated with a value of $16 on the processor status signals (PST[4:0]).}

When the TAP state machine is operated through the rising-edge of TCK leaving the CaptureDR state, the boundary scan register associated with the CLK input pin should conduct a sample or capture operation of the logic value applied at the signal pin. For xF43G and xG65V mask sets, the boundary scan element associated with the CLK input pin will sample a fixed value of logic 0 instead of the clock pin value if the clock is stopped and will sample the clock pin value if the clock is not stopped. Any sampled information that is associated with this cell when shifted out should be ignored (don't cared). This cell is listed in Chapter 9 of the User's Manual in table 9.3 as boundary scan bit location: 73.

Parametric

P4 - Address input hold time for spec 45a is out of spec by 0.5ns. Therefore, addresses must be held for at least 2.5ns.

P9 - The input hold time specification (spec 52) of 2ns for RSTI is marginal. RSTI should be held valid for at least 5ns after the rising edge of BCLK.

P12 - The time from BCLK to output at high impedance for D[31:0], CIOUT, LOCK, LOCKE, R/W, SIZ[1:0], TS, TLN[1:0], TM[2:0], TT[1:0], UPA[1:0], and BS[0:3] (Specs 21 and 38) are out of spec. For 50MHz parts the BCLK to output at high impedance spec should be lengthened from 12ns to 18ns.

P13 - The time from CLK to IPLx invalid (interrupt input hold time), spec 42d, is marginal. IPLx inputs should be held valid for at least 3ns after the rising edge of CLK.

General Information

G3 - LPSTOP operation is sampled but not 100% production tested.

G4 - The 68060 surface mount package has been changed from a 208 pin CQFP to a 240 pin CQFP. Pinouts for the new package are available on the World Wide Web at http://www.mot.com/hpesd. Only the 68LC060 and 68EC060 products will be offered in this 240 pin CQFP; the full 68060 with FPU will only be offered in the PGA package.

Spec Changes (These anomalies will NOT be changed in future mask sets)

S1a - If two or more virtual addresses are mapped to the same physical address where at least one of the mappings is copyback and at least one of the mappings is writethrough, and if references using different mappings can occur within a sequence of 8 consecutive instructions such that the first instruction performs a write in copyback mode and a subsequent instruction performs a read in writethrough mode, data corruption may occur.

S1b - If two or more virtual addresses are mapped to the same physical address where at least one of the mappings is copyback and at least one of the mappings is writethrough, and if a pair of instructions are dispatched to the 68060 pipeline as a superscalar pair such that the first instruction loads data from memory to a general register Ry and the second instruction of the pair uses Ry (invoking a result forwarding optimization), and if the virtual address of the load operation is mapped as writethrough to a line (physical address) resident in the data cache with its dirty bit set (previously referenced through a VA with copyback mapping), the execution of the second instruction uses erroneous data for Ry, resulting in data corruption.

Workaround:

Do not simultaneously map a given physical location as both copyback and writethrough

--AND--

If switching the mapping of a physical location from copyback to writethrough, guarantee that the DCache dirty bit for the location is reset, by either executing a CPUSH or CINV instruction to that address.

S2 - If a non-cacheable reference hits in the instruction cache (that is, an instruction cache reference with a cache-mode-mismatch), then the matching instruction cache line, which should be invalidated, may not be invalidated, and another instruction cache line may be erroneously invalidated. Note: unless software is depending on the cache-mode-mismatch condition described above to automatically invalidate an instruction cache entry prior to modifying code space, this condition will not result in failure.

Workarounds:

Never map a given instruction physical location as both cacheable (either copyback or writethrough) and non-cacheable (either non-cacheable precise or non-cacheable imprecise)

-- OR --

Never have an instruction cacheable mapping and non-cacheable mapping active at the same time -- AND --
When switching from using the cacheable mapping to the non-cacheable mapping or from using the non-cacheable mapping to the cacheable mapping, all physical locations in these mappings that may be in the instruction cache must be invalidated with either the CINV or CPUSH instruction.

S3- The nominal resistance and temperature coefficient of the internal die resistor connected across the thermal sensing pins (THERM1, THERM0) as listed in the User's Manual (rev. 1) on page 2-16 is incorrect. The correct value for the temperature coefficient is approximately 2.8 ohms/ degree C. The correct value for the nominal resistance is approximately 780 ohms at 25 degrees C.

S4 - The CLKEN input setup time specification (spec 55) listed in the data book is incorrect. The CLKEN input setup time should be specified as 8ns at 50MHz and 7ns at 66MHz.

S5 - Output hold times listed in the data book are incorrect. The minimum output hold time specification (specs 12, 19, 58, 60, and 62) is 2.5ns, not 3ns.

S6 - Certain input setup times at 66MHz listed in the data book are incorrect. The input setup times for data (spec 15), IPL (spec 41d), address (spec 44a), and RSTI (spec 51) are 2ns, not 1ns.

S7- If a CPUSHP or CINVP instruction that targets the instruction cache uses an effective address that does NOT have address bits [10:4] = 0000000, then lines from the addressed page in set 0x0 of the instruction cache will NOT be cleared. This means for 4K pages, lines 0x0 and 0x800 of the addressed page will not be cleared if in the instruction cache; and for 8K pages, lines 0x0, 0x800, 0x1000, and 0x1800 of the addressed page will not be cleared if in the instruction cache.

If the CPUSHP or CINVP instruction that targets the instruction cache uses an address that DOES have address bits [10:4] = 0000000, then the instruction functions correctly (all instruction cache lines from the addressed page will be cleared).

Workaround: Force effective address bits [10:4] = 00000000 for any CPUSHP or CINVP instruction that targets the instruction cache.

S8 - If SNOOP is asserted during a reset exception, between the time RSTI negates and the bus request for the fetch of the initial stack pointer at address 0x0 occurs, the address generation for the fetch of the initial program counter may be corrupted (resulting in a fetch from address 0x0 instead of address 0x4). The failure manifestation under this scenario is unpredictable (possible failures include unexpected exceptions or data corruption).

Workaround:
Do not assert SNOOP to the 68060 during reset exception processing.

S9 - If a bus error (TEA) occurs on a cache inhibited instruction prefetch, and this prefetch is discarded due to a previous change-of-flow instruction, a bus error exception may be erroneously taken when it should have been discarded.

Background: Bus errors for prefetchs of instructions beyond a change-of-flow instruction are specified to be ignored if the change-of-flow is taken. This may be an issue for some systems if the change-of-flow is the last instruction on the last page of virtual memory.

The scenario/instruction sequence required to create this erroneous reporting of a bus error exception on instruction fetch has a very small probability of occurrence (the 060 instruction buffer FIFO queue must be full of instructions, the change-of-flow instruction must be an entry in the buffer, followed by an instruction located at a cache inhibited address that encounters a TEA, followed by a specific combination of operand pipeline bus activity and entries being present in the branch cache).

Workarounds include:

Do not generate TEAs on cache inhibited instruction fetch accesses --OR--

Have access error exception handlers recognize that an unexplained instruction fetch bus error could have occurred due to a TEA on a cache inhibited address (that should have been ignored due to a change-of-flow) very near in location to the PC value reported in the exception stack frame.

S10 - If a bus error (TEA) occurs on a bus write operation while the 68060 is emptying the store buffer or push buffer, during a specific "one-cycle window" of pipeline instruction processing, the access error exception for the TEA will be delayed and not taken until the next 060 exception occurs.

The scenario/instruction sequence required to create this delayed TEA exception has a very small probability of occurrence (a store buffer TEA or push buffer TEA indication presented to the 060 pipeline on exactly the same cycle as a pipeline cancel takes place due to "wrong-way branch prediction"). If this happens, a pending exception internal control flip-flop is not set and no immediate exception is initiated. The occurrence of the store buffer or push buffer TEA is captured in an 060 internal "fault-type register" such that the resulting access error will be recognized at the time of the next 060 exception. System designs that have implemented address/data capture logic external to the 68060 to recover from store buffer TEA exceptions (see 68060 User's Manual, section 8.4.6) are vulnerable to this errata.

Workarounds include:

Do not generate TEAs on 060 bus writes --OR--

If operating in write-through cache mode and TEAs on store buffer bus writes are necessary, disable the store buffer --OR--

If operating in copyback cache mode and TEAs on bus writes are necessary, do not generate TEAs on push buffer/line writes.

S11- The input setup times for interrupt (spec 41d) and reset (spec 51) listed in the data book are incorrect when, and only when, leaving the LPSTOP state. The IPL inputs and the RSTI input must be valid at least 3ns before CLK, at both 50MHz and 66MHz, for the exit from LPSTOP scenario. For normal, non-LPSTOP conditions, the input setup times for interrupt (spec 41d) and for reset (spec 51) are 2ns.

Should the interrupt input setup time or the reset input setup time specifications listed above be violated during the LPSTOP state, the processor may initiate LPSTOP "exit execution" (asserting IPEND for interrupt, begin exception processing for reset) and resume bus cycle operations one clock later than during normal, non-LPSTOP, interrupt or reset operations.

S12- Clock-to-output valid times listed in the data book for data (D[31:0]) pins at 66MHz are incorrect. The "BCLK to Data Out Valid" time (spec 18) is 11.5ns, not 10.4ns, at 66MHz.

S13- The BCLK to Data-In High Impedance specification (spec 17), read followed by write, listed in the data book for data (D[31:0]) pins is more restrictive on a board design (7ns at 50MHz and 66MHz) than the 68060 requires. The new spec 17 values are 11ns at 50MHz and 9ns at 66MHz.

S14 - The power dissipation value listed in the data book (30mW) when the processor is in LPSTOP mode with the CLK stopped low is incorrect. Power dissipation may be as high as 175mW at 3.465V. The only specification change is power dissipation during LPSTOP mode when the clock is stopped. The part is in spec during normal mode or LPSTOP mode when the clock is running.

.

060 Errata Change History:

• Rev 1.0:	Initial xF43G Errata. 8/18/94
• Rev 1.1:	Updated I1, I5, P1. 8/22/94
• Rev 1.2:	Updated P2 fix date. Revised numbers to be consistent with xD11W errata. 8/31/94
• Rev 1.3:	Added I11, I12 and P5. 9/23/94
• Rev 1.4:	Added I13. 9/27/94
• Rev 1.5:	Revised I6, I12, P1 and P5. Added I14, I15, E2, H6, J3, J4, P6-P12, G3, G4 and G5. 12/19/94
• Rev 1.6:	Revised I8 wording and moved it to S1. Added I16 and S2. Moved P6, P7, P8 and P10 to S3-S6 respectively. 4/5/95
• Rev 2.x:	Version number not used here to avoid confusion with old 68060 xD11W errata rev 2.x.
• Rev 3.0:	Eliminated errata for older masksets which did not go into production. Added I17, I18 and E3. 11/3/95
• Rev 3.1:	Updated table on page 1--errata E3 does not exist on xF10H masksets. 11/16/95
• Rev 4.0:	Moved I17, I18, and E3 to S7, S8, and S9 respectively. Revised P5 and P11 and moved to S11 and S14 respectively. Added F6, P13, S10, S12, and S13. Updated 060 Production Masksets table to include only productized masksets. 10/18/96