Porting software from an MC68040 to an MC68060

by Jeff Gokingco, Cliff Parrott, Robert Podnar

1.0 Credits

This article was submitted to Electronic Design, and was published in August 1994. The article you are seeing here is a draft copy only, and that the final form of the article is the propery of Electronic Design.

2.0 Introduction

The MC68060 delivers up to 3 times the performance relative to its predecessor, the MC68040. Furthermore, the MC68060 bus is very similar in comparison to the MC68040 bus. A socket adapter from Interconnect Systems is available that converts the MC68060 bus to operate in some existing MC68040 systems. For these reasons, the MC68060 is a viable upgrade processor to an existing MC68040-based system.However, to achieve the aggressive performance target, the MC68060 leaves some instructions unimplemented to streamline internal operations. This article discusses in detail the differences between the MC68060 and the MC68040 from a software perspective and discusses the software challenges that may arise when porting software to the new MC68060.

The superscalar MC68060 represents the latest generation of Motorola microprocessor products. The MC68060 provides superscalar integer performance of over 100 MIPS at 66 MHz. The MC68060 comes fully equipped with both a floating-point unit (FPU) and a memory management unit (MMU) for high-performance embedded control and desktop applications. Leveraging many of the same performance enhancements used by RISC designs as well as providing innovative architectural techniques, the MC68060 harnesses new levels of performance for the M68000 family. Incorporating 2.5 million transistors on a single piece of silicon, the MC68060 employs a deep pipeline, dual issue superscalar execution, a branch cache, a high-performance floating-point unit, eight Kbytes each of on-chip instruction and data caches, and dual on-chip demand paging MMUs. The MC68060 allows simultaneous execution of up to of two integer instructions, a floating-point instruction and one branch instruction during each clock.

In addition to substantial cost and performance benefits, the MC68060 also offers advantages in power consumption and power management. The MC68060 automatically minimizes power dissipation by using a fully-static design, dynamic power management, and low-voltage operation. It automatically powers-down internal functional blocks that are not needed on a clock-by-clock basis. Explicitly, the MC68060 power consumption can be controlled from the operating system. Although the MC68060 operates at a lower operating voltage, it directly interfaces to both 3-V and 5-V peripherals and logic.

In addition to the MC68060, two low-cost derivative products are offered. The MC68LC060 and MC68EC060 both provide the full integer performance as the MC68060. However, these products are targeted for applications that cannot justify the cost of a floating-point unit. The MC68EC060 also leaves the paged memory management unit unimplemented for applications that do not require the services of both the floating-point unit and paged memory management unit.

3.0 The M68000 Exception Model

The approach used in achieving user object code compatibility is through the use of M68000 exception processing architecture. An understanding of the M68000 Architecture is needed to fully appreciate how code compatibility can be achieved through software.

An exception is defined to be the result of a special internal or external condition that preempts normal processing. There are three external conditions that can cause an exception: Reset, Interrupts and Bus Errors (reported as an Access Error). All other exceptions are caused by internal conditions. A partial list of internal conditions that cause exceptions is as follows: Access Error (MMU-related), Illegal Instruction, A-line Instruction, F-line Instruction, Divide-by-Zero, Trap, Privilege Violation, Trace, Format Error, Floating-point Arithmetic.

When an exception occurs, an “exception Stack Frame” is created on the system stack, the “Vector Table” is used to determined where the “Exception Handler” begins.

The exception Stack Frame is placed on top of the system stack to provide information to the exception handler and to provide a way to return to normal processing. The Vector Table is a dispatch table that consists primarily of address pointers to the start of each Exception Handler. Each exception is associated with a unique entry in the Vector Table. Each entry in the Vector Table is four bytes in size. Note that it is possible for multiple exception types to share a single entry in the Vector Table. For these shared exception types, it is the responsibility of the Exception Handler to determine the actual cause of the exception. An example of this case is the F-line entry of the Vector Table. The F-line entry in the Vector Table is used when either the Floating-point Unimplemented exception, the Floating-point Disabled exception, and the F-line unimplemented exception results in an exception. Note that the Vector Table is relocatable. The VBR (Vector Base Register) points to the top of the Vector Table and can be modified through the MOVEC instruction.

The size and contents of an exception Stack Frame is a function of the exception type and the processor type.The minimum Stack Frame has a size of 8 bytes and contains the stacked SR, the stacked PC (program counter), the frame format and the vector number. The stacked SR contains the contents of the status register at the time of exception. The stacked PC may either point to the instruction that caused the exception, or to a the instruction being executed at the time the exception is reported, depending on the exception type. The frame format is used to distinguish among different stack frame types. The vector number is associated with the n-th entry of the Vector Table.

At the entry point of an Exception Handler, the system stack points to the top of the exception Stack Frame. From there, the Exception Handler determines the cause of the exception, handles the exception, and then executes the Return from Exception (RTE) instruction to return to the original program flow. When the RTE is executed, the Stack Frame format is queried to determine proper handling. If the frame format is invalid for that processor, a Format Error exception is taken. In most stack frames, the RTE instruction places the stacked SR into the status register, and instruction execution will continue at the location pointed by the stacked PC.

Let’s walk through an example of how the exception model is used to extend the instruction set of the M68000 Architecture.In this example, we will define an instruction with an instruction opcode of $ABCD. This instruction will place the value of 0 into the data register D0. the A-line exception handler starts at location $10000.

When this instruction is encountered, the processor (whether it be the MC68040 or MC68060) takes an A-line exception. The A-line exception is associated with the Vector Table entry 10. This entry translates to the location

VBR+$28 of the Vector Table. That location in memory contains the value $10000, which is the start of the exception handler.The exception handler can then execute the following code:

```
        cmpi.w     #$ABCD,[(2,A7)]
        bne        _done
        move       #0,d0
done:
        addi.l     #2,2(A7)
        rte
```

At the beginning of the exception handler, the system stack pointer points to the top of the exception stack frame. The first instruction determines whether or not the instruction opcode is correct. The second instruction performs the desired function of placing a zero into the data register D0. The third instruction increments the stacked PC by 2. When the RTE instruction is finally executed, the modified stacked PC will point to the instruction immediately after the $ABCD instruction.This way, when the RTE is executed, the A-line instruction exception is not repeated.

4.0 User Code Differences

The M68000 Architecture supports two privilege levels: User and Supervisor. These privilege levels are implemented to provide system protection.When executing in supervisor mode, all instructions are available to the processor. Otherwise, when executing in user mode, a few “privileged” instructions generate a Privilege Violation exception.Since certain rarely used instructions and addressing modes are not implemented in hardware, complete emulation of user mode instructions is the target of the MC68060 Software Package.The MC68060 Software Package is available free of charge from Motorola.

The MC68060 Software Package (MC68060SP) contains the exception handlers needed for emulation of integer and floating-point instructions to achieve user mode object code compatibility with the MC68040. In addition, the MC68060SP provides full IEEE floating-point support through another set of exception handlers. This functionality is provided by two distinct package modules: the MC68060 Integer Software Package (MC68060ISP) and the MC68060 Floating-Point Software Package (MC68060FPSP).

The MC68060ISP contains the emulation software needed for the unimplemented integer instructions listed in Table 1.

When any of these instructions is encountered, the MC68060 takes an “Unimplemented Integer Instruction” exception. By examining the exception information on the stack, the MC68060ISP determines the instruction that caused the exception and emulates it in software. The package then returns from the exception to the previously executing program. The numerical and condition code results produced by the MC68060ISP are equivalent to those produced by the MC68040 hardware.

For example, the integer divide instruction, which divides a sixty-four bit dividend by a thirty-two bit divisor, will take the Unimplemented Integer exception. If properly installed, the MC68060ISP will generate the correct result and return from the exception. Alternatively, if the divide instruction being emulated results in a divide-by-zero exception, the MC68060ISP alters the exception stack frame and vectors to that exception handler. To the user, it appears as if the “Unimplemented Integer Instruction” exception never occurred.

Of special note are the CAS and CAS2 (compare and swap with operand) instructions, which, on the MC68040, provided a means of synchronizing several processors by using read-modify-write transfer sequences. To assist the MC68060ISP in emulating these instructions, The MC68060 processor has added the ability to control the external bus lock signal from software. This functionality has been added to the MOVEC (move control register) privileged instruction. The MC68060ISP asserts the bus lock signal before initiating the first emulation read access and de-asserts it after the final write access to ensure atomic operation.

Like the MC68040FPSP for the MC68040, the MC68060FPSP makes the MC68060 floating-point arithmetic unit 100% compliant with the ANSI IEEE Standard for Binary Floating-Point Arithmetic 754-1985. In addition, the MC68060FPSP makes the MC68060 processor 100% instruction-compatible with the M68881 and M68882 floating-point co-processors. The MC68060FPSP produces this compatibility through a set of entry points that correspond to the floating-point exception vectors on the MC68060. The target operating system should fill the vector table entries for these exceptions to point to the MC68060FPSP entry points.

For example, the IEEE FP standard defines that operations resulting in an underflow condition either return a default result or take a trap so that the user can generate an alternate answer. The MC68060, like the MC68040, takes an exception for both cases. The MC68060FPSP checks the Floating-Point Control Register (FPCR) to see which option the user desires. If the default mode is selected, the package returns the default underflow result and user program execution resumes. It will appear to the user as if the exception never occurred. If the default mode is not selected, the MC68060FPSP jumps to the system-supplied code to generate the alternate result.

To maintain MC68881/2 instruction compatibility, the MC68060 and MC68040 use the “Unimplemented FP Instruction” exception and the corresponding FPSP to emulate the floating-point instructions listed in Table 2. The MC68060 processor takes this exception when one of these instructions is encountered. The MC68060FPSP emulates the instruction in software and returns to normal program execution.

The MC68060FPSP also emulates the instructions FSCC, FTRAPCC, and FDBCC, which perform functions based on the state of the floating-point condition codes. These low-use instructions were hardware-implemented on the MC68040.

Like the MC68040, any instruction whose floating-point operand is denormalized, unnormalized, or in packed decimal real format generates an “Unsupported Data Type” exception on the MC68060. The MC68060FPSP emulates the floating-point instruction and resumes normal program execution.

Finally, the MC68040 provided hardware support for immediate data extended precision operands. The MC68060 does not in order to optimize internal routing. The same is true for the immediate data move multiple control registers instruction. These instructions take the newly defined “Unimplemented Effective Address” exception. The MC68060FPSP contains an entry point for the emulation code for these instructions.

Although the MC68040FPSP and MC68060FPSP provide similar functionality to a user program executing floating-point code, the packages are significantly different. These differences are due to the simplified floating-point exception model on the MC68060 processor. Therefore, the MC68040FPSP will not function in an MC68060 system.

Both the MC68060ISP and MC68060FPSP are designed to be position-independent and re-entrant. Portability is further promoted by using “call-outs” for operating system-dependent code (such as reading/writing user space). To perform “call-outs”, the MC68060 makes subroutine calls to functions provided by the target operating system. The external routines required are very similar to those used by the MC68040FPSP except for the new functions needed by the MC68060ISP.

Although the main goal of the MC68060 Software Package is to provide instruction emulation for all user mode instructions, two user mode instruction emulation cases require special attention.

The first situation involves the “TRAPF #immediate” instruction. This instruction may occasionally cause a branch prediction error, this error is subsequently reported as an Access Error exception. The branch prediction error occurs when a taken branch instruction is executed creating a branch cache entry and then this same code is re-executed with the former branch instruction now appearing as an extension word for another opcode. In this type of sequence where the interpretation of the code stream is dynamically changed, a branch prediction error may occur. This error condition is easily identified and handled in the access error exception handler.

The second condition involves the misaligned CAS and CAS2 emulation. The MC68060 ISP requires that interrupts be masked when executing inside certain code segments. However, since the level 7 interrupt is non-maskable, the CAS misaligned and CAS2 emulation may result in a non-recoverable system error if interrupted.A possible solution involves having all level 7 interrupt handlers query the BUSCR to check whether or not a locked sequence is violated. The interrupt handler may then restore the LOCK and return to emulation software immediately. However, in typical systems, the non-maskable interrupt is used only during severe system-level error conditions, and changing the level 7 interrupt handlers may be unnecessary.

With the exception of the two specific conditions as previously mentioned, the MC68060 Software Package provides complete emulation of all user mode instructions.

5.0 Supervisor Mode Differences

Privileged instructions that are unimplemented or different are not handled by the MC68060 Software Package. Hence, supervisor mode differences merit discussion as it applies to porting system software.

There are two new registers on the MC68060, accessed via the MOVEC instruction.These registers are the Processor Control Register (PCR) and the Bus Control Register (BUSCR). As these two registers are unimplemented by the MC68040, existing MC68040 software will not affect these registers.

The PCR allows supervisor control of superscalar dispatch, floating-point disabling and the hardware debug mode. All operations resulting from accesses to these registers are fully synchronized, and may be done at any time. After a processor reset, the secondary operand pipe is disabled. To enable it, which is recommended for normal operation, the Enable Superscalar Dispatch (ESS) bit of the PCR must be set. Superscalar dispatch may be disabled to help emulation, or in certain cases for hardware debug. The on-board floating point unit of the MC68060 is enabled after reset, and may be disabled by setting the Disable Floating Point (DFP) bit in the PCR. When disabled, floating point instructions will trap as unimplemented floating-point instructions, thus emulating an MC68EC060 or MC68LC060 (and similarly, the MC68EC040 and MC68LC040), which do not have on-board floating point units. For hardware and software debug using a logic analyzer, the MC68060 can make certain internal state information available on the bus pins when the bus is idle.The EDEBUG bit of the PCR is cleared on reset. For lowest power operation, debug should not be enabled.

The BUSCR controls the external LOCK and LOCKE pins and is used primarily for CAS and CAS2 emulation.Unlike the MC68040, the MC68060 supports the CAS instruction with misaligned operands, and all CAS2 variants, only via software emulation. These instructions implicitly control the LOCK and LOCKE (LOCK End) bus pins, which are used in some designs as part of the bus arbitration logic. For emulation, direct control of these pins is given to the programmer via a new register, the BUS Control Register (BUSCR). If the Lock (L) bit of the BUSCR is set, then the LOCK bus pin is asserted on the next bus access; if the bit is cleared (=0), then LOCK is negated on the next bus access. Similarly, the LOCKE bus pin is controlled by the LockE (LE) bit. To model the MC68040, the LOCK and LOCKE pins are also negated by the processor on the next bus access which occurs when both the L and LE bits are set. The emulation code must then clear the L and LE bits, or else LOCK and LOCKE will reassert on the next bus access. All operand references while in locked mode will bypass the data cache. If an entry matches the access, it will be invalidated if clean, and pushed and invalidated if dirty. This is the normal operation for locked instructions. It is a good idea to limit exceptions that may occur during the execution of emulation code, by preprocessing memory-management faults and by raising the interrupt mask. However, if an exception occurs and the LOCK bit is set, the MC68060 will copy the L bit to the Shadow Lock (SL) bit, and then clear the L bit, thus negating the external LOCK bus pin. The LOCK pin will be negated by the processor before any exception-processing bus accesses are made. In the exception handler, the programmer must copy the state of the SL bit back to the L bit, just before returning from the exception. Therefore, the LOCK bus pin will be asserted for bus accesses related to unstacking the exception. The SL bit is never cleared by the processor; thus, nested exceptions (exceptions that occur during the processing of other exceptions) will not lose the LOCK state, but LOCK may be enabled for bus accesses other than for the emulated instruction if there are nested exceptions. The operation of LE and Shadow LockE (SLE) are similar to L and SL. BUSCR is accessed via the MOVEC instruction, and is also cleared by reset. Since BUSCR universally controls the LOCK and LOCKE pins, the user may implement any self-defined LOCK protocol, in addition to CAS and CAS2. Misaligned CAS and CAS2 emulation code is available as part of the MC68060 software package.

There are two MOVEC control registers that are implemented on the MC68040, but have added functionality by the MC68060. These registers are the Translation Control Register (TCR) and the Cache Control Register (CACR). Since these registers are defined for the MC68040, it is very likely that existing code will modify these registers as though they were the MC68040 control registers counterparts. Unfortunately, the most common error result would be that the newly-defined bits will be cleared by existing code. Care must be taken as this may be a significant porting issue.

The TCR is used to enable the Paged Memory Management Unit (MMU), and to select page size, exactly as the MC68040 All bits in the TCR are cleared by external reset. The MC68060 adds the following features:

- Freeze ATC entries: NAD (No Allocate Data ATC), and NAI (No Allocate, Instruction ATC) - when either of these bits is set, the corresponding ATC will not allocate space for any new entries, but will continue to use the entries that are resident. Accesses which hit will source the translation; accesses which miss will cause a table search to get the translation. This feature should be used cautiously, since accesses which miss will cause a table search, there may be considerable performance penalty due to those misses. Write access which find a corresponding valid read entry will update the internal and page table M-bits (Modified) and the entry will remain valid.
- Half-cache mode: FOTC (Data ATC), FITC(Instruction ATC) If either of these bits is set, the corresponding ATC will only use 32 entries instead of 64.
- Default TTR: DCO (Default Cache Mode), DUO (Default User Page Attributes, Data), DWO (Default Write Protection), DCI (Default Cache Mode, Instruction), DUI (Default UPA, Instruction) On the MC68040, if the Paged Memory Management Unit is disabled, an access that misses the TTR registers will be mapped transparently with a cache mode of cacheable write-through, and with the UPA = 0. The MC68060 adds to the

functionality provided by the MC68040 by implementing these Default TTR register bits to provide programmable default cache modes, default UPA bits definition, and default write protection.Immediately after reset, the MC68060 TTR defaults are identical to those of the MC68040.

The CACR cache enable bits which were defined by the MC68040 are unchanged for the MC68060. This register is accessed via the MOVEC instruction, and all bits in the CACR are cleared by a reset. There are a number of new features which are controlled through the CACR:

- Cache Freeze: NAD (No Allocate Data Cache), NAI (No-allocate Instruction Cache) - If the either of these bits is set, and the corresponding cache remains enabled, the cache will continue to source and accept data for accesses which hit, but will not allocate space for any new entries. Accesses which miss will be fetched from the bus. The CINV and CPUSH instructions operate normally, regardless of the state of this bit.
- ESB - Enable Store Buffer - The `060 implements a four-deep first-in-first-out (FIFO) store buffer to enhance performance. When the ESB bit is set and there is room in the buffer, then write accesses using the writethrough or cache-inhibited imprecise cache modes are queued in the buffer until the bus is free, allowing the operand pipes to continue processing. Locked accesses (e.g. TAS instruction) and accesses which use the cache-inhibited precise mode will always bypass the buffer and will wait for the buffer to empty before accessing the bus. Writes which use the store buffer are not deferred past operand reads of later instructions. Data is stored in the buffer exactly as it will be written to the bus (misaligned accesses will take two locations in the buffer). If there is a bus error on a write from the store buffer, any other pending write (from other entries in the store buffer or the pipe) will be completed before the access error fault is taken.
- Disable CPUSH Invalidation: DPI - This bit controls the invalidation mode of the CPUSH instruction. If DPI is cleared, each line pushed to the bus by the CPUSH instruction is invalidated in the data cache. This is the normal mode, and the way the MC68040 operates. If the bit is set, then CPUSH will not invalidate the line as it is pushed. Using DPI mode allows the data cache to be made coherent with external memory without invalidating the cache entries. The state of this bit is ignored by the instruction cache.
- Half cache modes: FDC (data cache), FIC (instruction cache) - If set, the corresponding cache will operate as a 4k sized, two-way set associative cache. If cleared, then they operate as an 8k sized four-way set associative cache.
- Enable Branch Cache: EBC - Setting this bit enables the branch cache and associated branch prediction strategy in the instruction pipeline. Since on average, one in four instructions is a branch, using the branch cache causes a significant performance increase. In many cases, branches may be folded into other instructions so that the branches are executed in zero clocks. Enabling the branch cache is highly recommended for most applications. The branch cache must be cleared prior to use.
- Clear entries in Branch Cache: CABC(all), CUBC(user mode only)- Writing a one to either of these bits invalidates all entries (CABC) or user mode entries (CUBC) in the Branch Cache. These bits always read as zero. It is necessary to clear the branch cache after power on, and for most context switches. Both the CABC and EBC bits (clear and enable) may be written at the same time. Since the branch cache is tightly coupled to the instruction cache, a CINV or CPUSH instruction which specifies the instruction cache will also implicitly clear the branch cache.

Some registers previously accessible through the MOVEC instruction are now unimplemented on the MC68060. These registers are the Master Stack Pointer (MSP), Interrupt Stack Pointer (ISP) and the MMUSR (MMU Status Register). Hence any MOVEC instruction access to any of these registers results in an illegal instruction exception.The MC68060 implements a single supervisor stack.   To aid systems which require a separate MSP and ISP, which were available on past MC68000 family processors, the M bit in the system Status Register is retained. An interrupt, which would normally be stacked on the ISP, will clear the M-bit; other supervisor exceptions will not. System software must set and track this bit if emulating two supervisor stacks.

Due to changes in the exception model and internal structure of the MC68060 MMU, the PTEST instruction and its associated MMUSR (MMU Status Register) are not implemented on the '060. When the MC68060 encounters a PTEST instruction, an F-line Instruction Exception is taken. On the MC68040 and previous processors, PTEST was used for two basic functions, 1) return information about an MMU translation, primarily used for fault recovery, and 2) return the physical address corresponding to a virtual one. On the '060, information about an MMU fault is available directly from the exception stack frame. Each type of fault is specifically identified in the Fault Status Long Word (FSLW). The new instruction, PLPA (Paged memory management Load Physical Address), ensures that an ATC is loaded with a valid translation, and returns the related physical address. The search conforms to all normal MMU search rules for the '040 and '060 (i.e., TTR hits have priority, MDIS operates like any other MMU access). If the applicable TTR misses, then a hit in the ATC will return that data, otherwise a table search will be initiated. A fault condition, such as invalid table descriptor, write protection violation from either a TTR or translation table, supervisor violation or bus error, will take the access error exception just like a normal access. The Program Counter on the exception stack frame points to the PLPA instruction. If system software allows faults to occur in supervisor space, then completion of PLPA means that all faults are resolved.

There are two variants, PLPAR and PLPAW, which check privilege and set the table and ATC history bits for a read and write access, respectively. The ATC is specified by the data in the DFC (Destination Function Code) register; an instruction function code in the DFC will cause PLPA to search the instruction ATC; similarly for the data ATC. It is recommended that PTEST and MOVEC references to the MMUSR, which are normally found in access error handler routines, be changed to the new PLPA functionality. However, it is possible to emulate PTEST, and suggested assembly code routines for this are available from the Motorola Bulletin Board.

For the most part, the stack frames used by the MC68060 are the same ones defined for the MC68040. The exception to this rule is that the MC68060 no longer supports the four word throwaway stack frame since this stack frame is primarily used in a dual system stack processor.The Access Error stack frame and floating-point state frames are entirely different between that used by the MC68040 and that used by the MC68060.Therefore, there may be software porting problems when using Access Error Handlers, Floating-point related exception handlers that were written with the MC68040 in mind. These handlers may need to be rewritten for use with the MC68060.

6.0 System Software Port Checklist

As mentioned previously, the MC68060 implements only a single supervisor stack.System software that requires a dual system stack architecture may require significant modifications.The system software port guidelines assume that a dual system stack architecture is not necessary.

The first exception handler that the processor executes upon boot-up is the Reset exception handler. Typically, the Reset exception handler performs initialization routines, flushing and enabling the caches, resource checking routines. The boot-up code can be used to enable certain new features that the MC68060 provides.

The PCR register can be updated in the boot-up code to enable superscalar dispatch, as this is a new MC68060 register, existing code is unlikely to inadvertently change the PCR setting. The BUSCR register needs no initialization.

To properly emulate the MC68EC040 or MC68LC040 using an MC68060, the floating-point unit must be disabled. This is done through the DFP bit in the PCR. When the DFP bit is set, any floating-point instruction encountered by the MC68060 generates a Floating-point Disabled exception using the same stack frame format written for the MC68LC040 or MC68EC040. However, the stacked effective address field of the stack frame format 4 is different when dealing with extended precision operands using the post-increment or pre-decrement addressing modes. Hence, floating-point emulation software that were written for the MC68EC040 and MC68LC040 may require some modification for full compatibility.

The TCR and the CACR registers are most likely to be encountered in boot-up code as well. To make the MC68060 TCR emulate MC68040 TCR functionality, the newly-defined TCR bits need to be cleared. Fortunately, most accesses to the MC68040 TCR has these bits already cleared, and therefore, no work is needed for this register to make it MC68040-compatible. However, for the CACR, existing code that expects the MC68040 writes zeroes to the newly-implemented CACR bits, if this is done, the MC68060 disables the branch cache and store buffer. Although existing code will execute properly under these conditions, this presents a serious performance penalty. Also note that the CACR also controls the invalidation of the branch cache. This invalidation needs to be performed at boot-up along with the invalidation of the instruction and data caches.

Software that performs resource checking commonly is done at boot-up. A common procedure in resource checking is a read of the resource address after setting up the Access Error handler.If the resource is not present, a bus error time-out circuitry causes an Access Error exception. However, since the MC68060 is a restart machine, any operand read access that results in a bus error will be restarted upon exit of the Access Error handler. A common way to resolve this situation is to use an instruction of known size. When the Access Error handler is entered, then the stacked program counter can be incremented by the size of the instruction that causes the bus error.

Installation of the MC68060 Software Package can be encountered during boot-up. This installation must completely replace the MC68040 Floating-point Software package installation.The MC68040 Floating-point Software Package is not designed to work with the MC68060. The main problem is that the MC68040 Floating-point Software Package assumes certain Floating-point state frames that are now very different compared to the MC68060 floating-point frames.

Some boot roms contain simple debuggers. These low level debuggers may reference unimplemented MOVEC registers such as the MMUSR, MSP and ISP. When the MC68060 encounters the illegal MOVEC instruction, an illegal instruction exception is taken. One possible solution to this is to provide a simple Illegal Instruction exception handler that simply skips the MOVEC instruction if these registers are encountered. Another source of problem that is related to simple debuggers may be that the debugger is expecting 2 trace bits in the SR instead of 1.

Software related to Virtual Memory can potentially cause porting problems. One of the main problems in porting Virtual Memory software is that the MC68060 implements tablewalking such that the data caches are not queried in the process. Given that an operating system may write to the memory locations that contain the table and page descriptors, and if those memory locations are designated as cacheable copy back, it is possible that the only valid copy of a descriptor to reside only in the data cache. If this situation arises, then the MC68060 tablewalker will use stale descriptors.This is unlike the MC68040, in which the tablewalk hardware checks and uses the descriptor information that reside inside the data caches.So, if the operating system touches the descriptors only once at boot up and never touches the descriptors again, system software may simply push the dirty contents of the cache out to memory via the CPUSH instruction. Another alternative is to designate the pages that contain the descriptors to be cacheable, but write-through, instead of copyback.

When a page fault results in an exception, it enters the Access Error handler. The Access Error frame is entirely different between the MC68040 and MC68060. One main difference is that the MC68040 access error frame contains write-back slots that expect the handler to write out to memory, if desired. The MC68040 increments the program counter when a write has been posted. This means that on an MC68040, the stacked program counter always contain the instruction address of an instruction that follows the instruction that caused the write page fault. On the other hand, the MC68060 Access Error stack frame does not have any write-back slots. A write page fault does not cause the program counter to increment, hence, the program counter points to the instruction that caused the write page fault. From the MC68060 perspective, once the write page fault has been taken care of, the entire instruction can be restarted.

On an unrelated matter, when using the MMU on a supervisor or write-protect application, a write or supervisor protection fault does not increment the program counter either. Hence, where it was possible for the MC68040 to direct the software to discard the write cycle, the MC68060 cannot do so, since the program counter points to the instruction that caused the exception. Executing an RTE simply restarts the instruction and may result in an infinite loop.A possible software workaround is to examine the instruction, and based on the instruction type and effective address, determine the number of bytes of that instruction. The stacked PC can then be incremented by that instruction length amount to skip that instruction entirely when the RTE is executed.

The PTEST instruction is no longer implemented on the MC68060. The PLPA instruction provides an alternative to the PTEST instruction. The Virtual Memory access error handler needs to be rewritten to replace all PTEST instructions to the PLPA instruction. References to the MMUSR via the MOVEC instruction must also be avoided to prevent the illegal instruction exception. If the MC68060 encounters the PTEST instruction, an F-line exception is taken. For applications that require knowledge of the cache mode, the PTEST instruction may be emulated by writing software that performs the tablewalk function.

Context Switch Interrupt handlers may contain porting issues. Context Switch handlers must realize that the branch cache is virtual. Hence, if the same virtual address is mapped onto multiple physical address locations, the Context Switch handler must flush the branch cache.For systems that translate transparently, flushing the branch cache is unnecessary. Context Switch interrupt handlers that are implemented using the non-maskable level seven interrupt may result in data corruption problems when using the MC68060 Software Package CAS2 emulation. It is highly suggested that either hardware support be implemented such that no interrupts are allowed to e reported during times in which the LOCK pin is asserted, or that the Context Switch be placed in a lower level interrupt that is maskable.

Input/Output Device Drivers is another class of applications that merit discussion. In general device drivers that work for the MC68040 would work for the MC68060.The common method by which Input/Output devices are mapped is by the Cache mode = 11. Although this cache mode means “cache-inhibited serialized” when using an MC68040, this cache mode means “cache-inhibited precise” when using the MC68060.This difference should not cause any problems since the MC68060 always treats external non-cacheable accesses as serialized already.

Some guidelines can be used to avoid boundary cases that may cause double reads or double writes to I/O devices. It is advisable to use instructions that can only incur a single page fault per instruction. For instance, a "MOVE <ea>,D0", that use aligned data, can incur a page fault on only the source read. Using a "MOVE <ea>,<ea>" instruction on the other hand can incur up to four page faults when dealing with misaligned operands on both the source and destination.Only aligned accesses should be used in accessing I/O devices. Abiding by these guidelines help ensure that an I/O device is not unnecessarily written/read multiple times by the MC68060.

When the MC68060 designates a page as "imprecise" it can simply be assumed that for that "imprecise" page, a write bus error results in lost data, unless there is an external hardware latch that picks up the write data. This should not be a severe limitation, since the memory areas that are "imprecise" are usually memory devices. A write bus error is considered by the MC68060 as a severe system error since the bus error would have to be reported after the MMU has designated that area as a valid page. Another argument for designating write bus errors as a system error is that a write bus cycle that has been posted to the system usually has no cause for a bus error since there is no such thing as a parity error when the processor performs write cycles. Also, some other MC68060 write bus errors are already designated as unrecoverable (bus error on a push buffer write or a store buffer write).

Given the guidelines outlined here, a system software port from an existing MC68040 to an MC68060 may require some effort, but when the system port is done, applications software that use only user-mode instructions requires no recompilation.For some systems, the hardware challenge may be trivial by using the Interconnect Systems adapter. Hence, your existing MC68040 system may be days away from an overall system performance boost via the MC68060.

7.0 Tables and Illustrations

DIVU.L	<ea>,Dr:Dq	64/32 ⇒ 32r,32q
DIVS.L	<ea>,Dr:Dq	64/32 ⇒ 32r,32q
MULU.L	<ea>,Dr:Dq	32*32 ⇒ 64
MULS.L	<ea>,Dr:Dq	32*32 ⇒ 64
MOVEP	Dx,(d16,Ay)	size = W or L
MOVEP	(d16,Ay),Dx	size = W or L
CHK2	<ea>,Rn	size = B, W, or L
CMP2	<ea>,Rn	size = B, W, or L
CAS2	Dc1:Dc2,Du1:Du2,(Rn1):(Rn2)	size = W or L
CAS	Dc,Du,<ea>	size = W or L, misaligned <ea>

Table 1: Unimplemented Integer Instructions

General Monadic Operations	
FACOS	FLOGN
FASIN	FLOGNP1
FATAN	FMOVECR
FATANH	FSIN
FCOS	FSINCOS
FCOSH	FSINH
FETOX	FTAN
FETOXM1	FTANH
FGETEXP	FTENTOX
FGETMAN	FTWOTOX
FLOG10	FLOG2
General Dyadic Operations	
FMOD	FREM
FSCALE	—
Conditionals	
FTRAPcc	FDBcc
FScc	–
Unimplemented Effective Address	
FMOVEM.X (dynamic register list)	FMOVEM.L #immediate of 2 or 3 control regs
F<op>.X #immediate,FPn	F<op>.P #immediate,FPn

Data Formats/Data Types	SGL	DBL	EXT	DEC	Byte	Word	Long
Normalized	S	S	S	U	S	S	S
Zero	S	S	S	U	S	S	S
Infinity	S	S	S	U	—	—	—
NAN	S	S	S	U	—	—	—
Denormalized	U	U	U	U	—	—	—
Unnormalized	—	—	U	U	—	—	—

Where:
S = Implemented Data Format, handled by the MC68060
U = Unimplemented Data Format, handled by the M68060SP

Table 2. Unimplemented Floating-point Instructions and Data Types

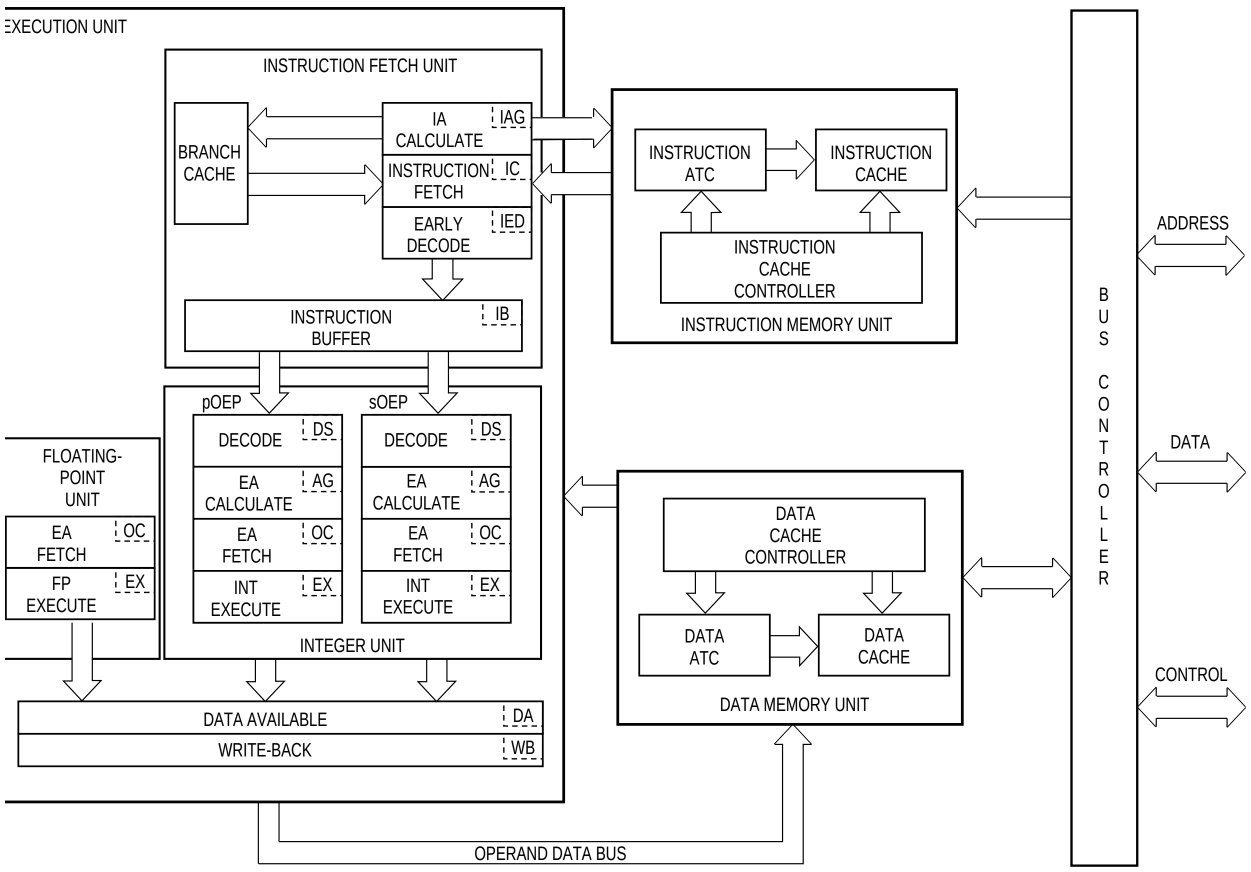

Figure 1. This represents the block diagram of the MC68060 processor. The superscalar dispatch of the MC68060 is evident in the dual pipeline shown in the Integer Unit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15 8	7	6 2	1	0
0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	Revision Number	EDEBUG	Reserved	DFP	ESS

Processor Configuration Register (PCR)

31	30	29	28	27 0
L	SL	LE	SLE	Reserved for Future Use

Bus Control Register (BUSCR)

Figure 2: The PCR and BUSCR registers are newly-defined registers for the MC68060. These registers are accessible through the MOVEC instruction.

31	30	29	28	27	26		24	23	22	21	20				16	15	14	13	12												0
EDC	NAD	ESB	DPI	FOC	0	0	0	EBC	CABC	CUBC	0	0	0	0	0	EIC	NAI	FIC	0	0	0	0	0	0	0	0	0	0	0	0	0

Cache Control Register (CACR)

31															16	15	14	13	12	11	10	9 8	7 6	5	4 3	2 1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	E	P	NAD	NAI	FOTC	FITC	DCO	DUO	DWO	DCI	DUI	0

Translation Control Register (TCR)

Figure 3: The CACR and TCR were defined for previous microprocessors like the MC68040. However, some of the register bits are newly-defined exclusively for the MC68060.

Deleted MOVEC Registers	
MMUSR	associated with PTEST instruction
ISP	MC68060 supports single stack pointer - SSP
MSP	MC68060 supports single stack pointer - SSP
Deleted Supervisor Instructions	
PTEST	replaced with PLPA
Added Instructions	
LPSTOP	low power stop
CPUSH	added no invalidate mode
PLPA	translates logical address to physical address

Figure 4. Other Supervisor Model Differences