IN-33

187787

33P

NASA
Reference
Publication
1319

September 1993

Mongoose ASIC Microcontroller Programming Guide

Mongoose

Brian S. Smith

(NASA-RP-1319) MONGOOSE ASIC MICROCONTROLLER PROGRAMMING GUIDE (NASA) 33 p

N94-13871

Unclas

H1/33 0187787

NASA Reference Publication 1319

September 1993

Mongoose ASIC Microcontroller Programming Guide

Brian S. Smith
Goddard Space Flight Center
Greenbelt, Maryland

National Aeronautics and Space Administration

Scientific and Technical Information Branch

PAGE ii INTENTIONALLY BLANK

Table of Contents

PRECEDING PAGE BLANK NOT FILMED

List Of Figures

Preface

Audience

This technical paper is intended for use by program managers, command and data handling system designers, and software designers as a reference to the operation of the Mongoose microcontroller.

This document contains Mongoose–specific documentation. Information generic to the MIPS R–3000, or the LSI Logic Inc. LR33000 was intentionally left absent from this paper, and must be obtained from the references listed in Appendix A.

Introduction

NASA Goddard Space Flight Center's Flight Data Systems Branch of the Electrical Engineering Division develops spacecraft computers, and is the sponsor of this work.

The Mongoose microcontroller was designed by the Branch to satisfy the need for a processor capable of operating in a space–radiation environment. A previous effort resulted in Goddard's first Command and Data Handling System aboard a spacecraft to use a 32-bit microprocessor: the Intel i386 family used in the Small Explorer Data System (SEDS). While this was a very successful program, many difficulties were encountered in designing the system to be tolerant of Single Event Upsets (SEUs) due to heavy–ion cosmic radiation. Since the i386 family of processors are not inherently rad-hard, extensive design–time was required for system–level "watchdogs," recovery methods, and defense of the methods in design reviews.

Before SEDS was first launched in July 1992, an effort was underway to develop a rad-hard microcontroller. The MIPS R–3000 architecture was chosen for several reasons, including:

—the availability of development software from several vendors;
—supporting hardware debug equipment; and
—the Reduced Instruction Set Computer (RISC) architecture.

To gain an understanding of what is meant by "RISC," it is recommended that the reader refer to Chapter 1 of G. Kane's "mips RISC ARCHITECTURE."

The RISC architecture of the R-3000 makes it attractive as a candidate for radiation hardening due to its inherent small size, measured in the number of gates necessary for implementation. Currently available "rad–hard" IC processes have a larger area per gate than do common commercial processes.

Mongoose is referred to as a microcontroller because of its architecture and intended application. Several functional elements which are normally peripheral to the processor in a spacecraft data system architecture have been "brought inside" on the same silicon as the processor core. This reduces the number of components necessary to implement a given design, while increasing the reliability of the system.

This Reference Publication is derived from the Draft Specification which was developed as a design guide to the gate–array designers who turned functional specs into physical implementation. Because of this, there are references to internal signals and functions which may be ignored by those implementing Mongoose–based software.

Implementation

Mongoose was manufactured on the LSI Logic Inc. LRH20K process, and uses approximately 50,000 gates, with a die size of .450" x .450". The chip–level design was performed by a highly qualified team of Harris GASD and LSI Logic designers.

1.0 Mongoose Chip Architecture

The Mongoose ASIC is partitioned into the major blocks shown in Figure 1. The Mongoose chip contains the CPU core, I–cache interface logic, D–cache interface logic, and Mongoose Support Functions (MSF). Software designers need to understand the MSF Architecture to implement the Mongoose–specific portions of their code.

1.1 MSF Architecture

The Mongoose support function block, or MSF, includes the D–side interface, the I–side interface, a timer/counter block, the memory interface block and the debug/control block. The D–side interface handles uncached data accesses and provides a DMA channel that operates between D–cache and memory or between memory and memory. The D-side interface also handles the interrupt expansion.

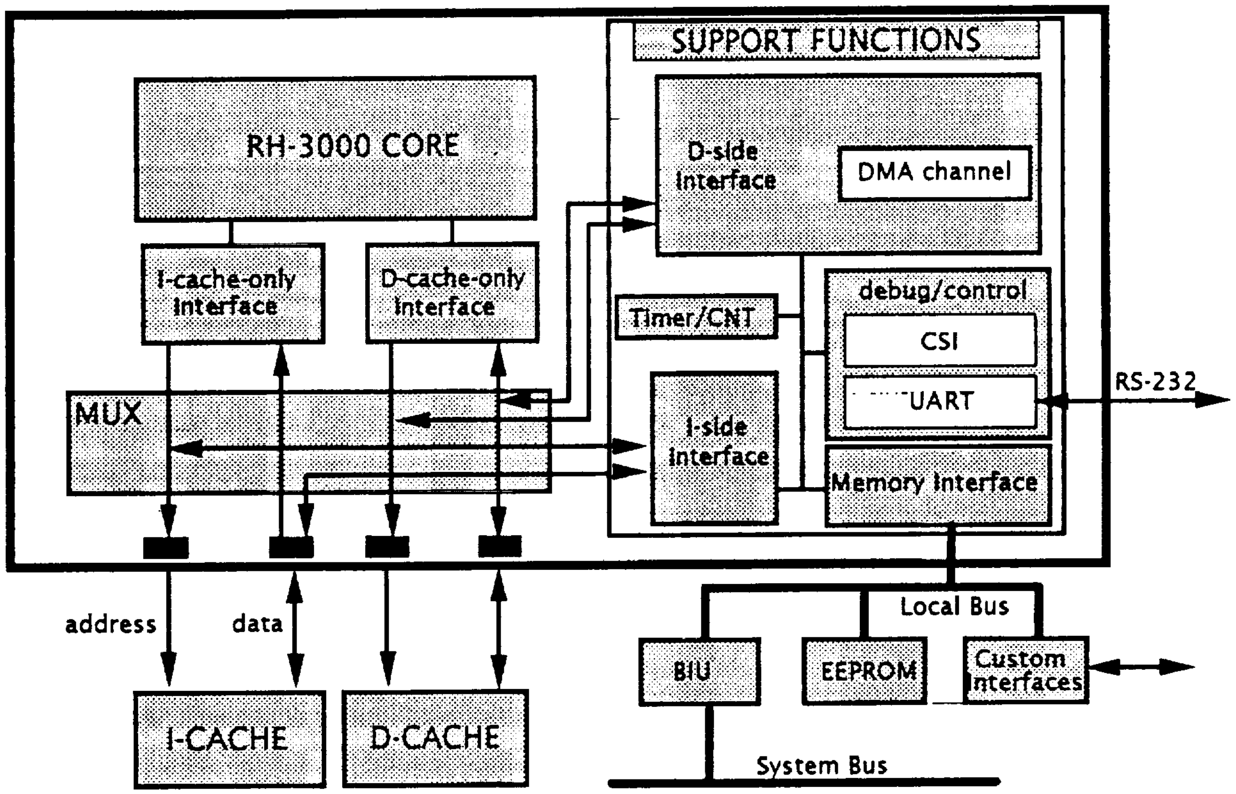

Figure 1. The Mongoose Processor Architecture.

The I–side interface handles uncached instruction fetches and allows the I–cache to be read/written under program control. The timer/counter block provides two 32–bit counter/timers. The memory interface handles all transactions to local memory, global memory and external I/O devices. The debug/control block contains the system clock generator, the UART which controls a single–channel RS–232 port and the command/status interface (CSI).

The programmer controls the MSF by reading and writing a set of registers through the Command/Status Interface (CSI). All reads/writes of the CSI occur in uncached kernel–mode address space from the D–side bus.

2.0 Memory Maps

2.1 Virtual Memory Map

The Mongoose implements the virtual memory map shown below. Note that since the Mongoose operates in cache–only mode, all accesses to cached space ignore the upper address bits; the upper address bits are used only to determine if the access is cached or uncached. Uncached addresses are limited to the portion of KSEG1 allocated for Global uncached address space; the physical address output from the Mongoose will address 508 MB (all of KSEG1 except the CSI space).

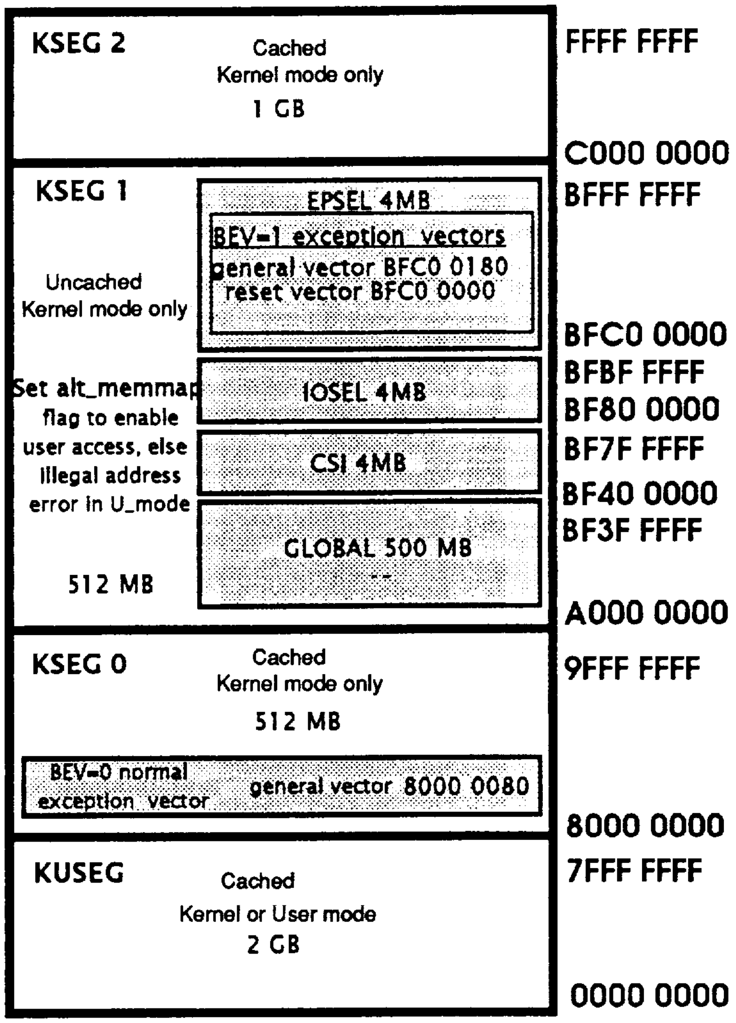

Figure 2. Virtual Memory Map.

KUSEG—This is used for normal cached code and cached data, and is accessible in both Kernel and User modes.

KSEG 0—Kernel mode cached space. This segment is special because it contains the normal exception vectors (set BEV flag to zero). This segment must be used when running exception code from cache. The general exception vector is at location 0x8000

0080. There is an unused exception vector at location 0x8000 0000, which was the UTLB exception vector in the LR33000.

KSEG 1—Kernel–mode uncached space. This is the segment in which all uncached accesses reside. Because the Mongoose operates in cache–only mode, all memory accesses outside the cache must be uncached and hence, must be in KSEG1.

The normal MIPS R–3000 and LR–33000 memory maps cause an illegal address error when a User mode process accesses this region. By setting the alt_memmap flag, the Mongoose allows user–mode processes to access this region and an address error is not generated.

KSEG1 is subdivided in a 4–MB EPROM space, a 4–MB I/O space, and a 4–MB CSI space for the Command/Status Interface of the Mongoose support functions and a 500–MB space for global uncached accesses.

Address 0xBFFF FFFC is the microboot address from which the Mongoose fetches one word on reset and configures the Mongoose cache chip selects.

EPSEL Subblock of KSEG 1—This is the 4–MBboot EPROM space; the EPSEL chip select is activated for a memory address in this region. The M_epsel_n signal is valid when M_as_n is active. This space contains the boot exception vectors (set BEV=1 to enable bootstrap exception vectors). The reset exception vector is at location 0xBFC0 0000 (Virtual); 0x1FC0 0000 (Physical). There is an unused exception vector at 0xBFC0 0100, which was previously the UTLB exception location. The general exception vector is located at 0xBFC0 0180. Normally, a boot prom will be connected to this chip select.

IOSEL Subblock of KSEG 1—This is the 4–MB I/O space; the IOSEL chip select is activated for a memory address in this region. This is useful for any I/O devices connected to the Mongoose and simply provides a pre-decoded address for external use. The M_iosel_n signal is valid when M_as_n is active.

CSI Subblock of KSEG 1—This is the 4–MB CSI space. This space is where the timer/counter control registers, the DMA control registers, the external exception cause register and all other MSF CSI registers reside. The low bits of the word address are decoded and used to address the set of CSI registers in the Mongoose support function block. All CSI registers must be accessed as 32–bit words.

MEMSEL Subblock of KSEG 1—This is the 500–MB uncached global space consisting of all uncached space which is not EPSEL, IOSEL or CSI space. The M_memsel_n signal is valid when M_as_n is active.

KSEG 2—Cached kernel mode only.

2.2 Cache–only Mode Memory Memory Map Considerations

The Mongoose operates in cache–only mode where the caches are tagless and all virtual accesses to cached memory are mapped to the physical memory in the caches. All other memory in the system must be in uncached space. Both the I and D caches have byte addresses between 0 and cachesize–1. Programs should be linked so that the Data and Instruction areas are non–overlapping in the virtual address space to satisfy the development system. With 32Kx8 SRAMs, the maximum cache size is 512KB, so the I–cache is linked to 0x80000. The hardware will ignore the upper address bits that make the two regions of memory distinct and will correctly access data from the D–cache and instructions from the I–cache.

Also note that because the upper address bits are ignored on a cached access (except the bits that determine the KSEG), all cache addresses in virtual address space map to some physical word in cache. This many–to–one mapping is determined by the physical size of each cache. In fact, each cached virtual address maps to a word in D–cache and a word in I–cache. The mapping between cached virtual space and the D–cache may differ from the map from virtual space to I–cache because the caches can be different sizes. In both cases, the addresses simply wrap around on boundaries defined by each cache size.

3.0 D–side Interface Block

The D-side interface block is the interface between the MSF and the D–cache/D-cache interface. The block is responsible for the following general functions.

- o Cache addressing, chip–select generation
- o Cache bus arbitration
- o CSI data bus interface
- o Driving CPU control port
- o Handling uncached reads and writes
- o Cache–memory, memory–cache, and memory–memory DMA

3.1 D–Cache Accesses

The MSF always operates in cache–only mode where the caches are tagless. All read accesses and all word write accesses are one cycle and a cache miss cannot occur. Partial word writes are handled by a read/modify/write operation within the D–cache interface and the MSF never needs to perform these types of operations. Only the lower address bits are used in cache–only mode and the upper virtual address is ignored. Values other than 0 in the upper unused bits are allowed and will be disregarded by the hardware. All external memory accesses outside the caches are limited to uncached type accesses. Uncached accesses are limited by the address space supported by the memory interface. The cache line in cache–only mode has the format shown below.

Uncached and Cached Access Data Path

Data
32

Internal Address Path

Index	Byte number	Access type
29	2	2

Cached Internal Address Path

Index	Byte number	Access type	Chip selects
18	2	2	1,2 or 4

The cache address path uses the byte number and access type internally—these signals are not used by the MSF for cached accesses. The uncached address path uses the byte number and access type internally and lane write strobe lines are available as pins on the Mongoose local bus.

Partial Word Accesses—The MSF always accesses the caches on a word basis and never needs to perform a partial word read or write. The MSF reads/writes full words for all DMA operations. The MSF does perform partial word operations at the Memory Interface. The MSF will present the proper lane write strobes for partial word writes and provides the access type and byte address for external decoding if necessary. For partial word reads, read data must be presented in the correct byte lanes, no lane switching is done internally, with the exception of byte–gathered reads.

Cache Read—The CPU always reads a 32–bit word in the cache in a single clock cycle.

Uncached Read—The CPU does not attempt to read the cache for an uncached read, although the uncached address will appear on the cache address bus. The CPU asserts a mem_access signal. The MSF decodes the virtual address and determines that the access is uncached. The MSF then requests a memory read operation from the Memory Interface. When completed, the MSF signals Data_ready to the CPU and drives the data bus. The CPU is the default driver of the cache bus.

One–Cycle Cache Write—During a one–cycle cache write, the CPU writes to the D-cache, and the MSF performs no function.

Uncached Write—The CPU does not write the cache for an uncached write; the CPU indicates a write_cycle to the MSF. The MSF decodes the virtual address and determines that the access is uncached. The MSF then requests a memory write operation to the Memory Interface as soon as possible. The MSF supplies the CPU the Wr_buffer_full signal to stall the CPU if necessary. A one–word buffer is used for writes, and the writes always complete before a subsequent read or write is processed.

Read/Write Invalidate—Immediately following any read/write operation, the CPU may assert the mem_request_inv signal to indicate that the read/write operation was cancelled after the instruction issued the read/write. The cancellation is caused by a late exception or interrupt. In some cases, duplicate reads/writes can occur at the data cache interface: one before the exception handler is entered, and one after the exception handler is exited. Unless an I/O device is connected to the data cache bus (such as a FIFO), this presents no problem for normal operation. Duplicate read/writes will never occur at the external memory interface.

Move–to–Coprocessor and Move–from–Coprocessor—This function is not supported, except for system control processor (CP0). An attempt to perform such an instruction will result in a coprocessor unusable exception.

Bus Idle—During any idle cycle on the cache bus, the CPU drives the cache bus and the MSF does not use the cache interface. All signals must be driven by some chip on every clock cycle.

3.2 DMA Channel

The DMA subblock controls a single DMA channel and issues the reads/writes to the D–cache and the Memory Interface Block needed to perform block movement of data in the general address space. The single DMA channel can be operated in one of three modes: cache–memory DMA, memory–cache DMA, and memory–memory DMA.

The DMA operation allows data to be transferred between the D–cache and uncached memory space or allows transfers within uncached memory space. The programmer sets up DMA origin, destination, block size and type to initiate a transfer. The DMA block size can be any size up to 16M words. The origin address must be a word address but is not required to be aligned to a page boundary. The DMA block issues the read/write commands as needed to execute the DMA transfer. When the DMA operation is complete, the DMA done bit is set and a DMA–done interrupt is issued. Byte–gathering is not supported during DMA.

Transfers are conducted on a request/grant basis with DMA given the lowest priority. The D–side interface requests access to the D–cache and waits until the D–cache interface block grants the access. The MSF D–side interface then performs a single word access to the D–cache and releases the D–cache for CPU use. The D–side interface then requests the D–cache again and waits until the CPU again grants access.

All DMA actions are non–preemptive and give the CPU and external bus requests higher priority. The end of the DMA operation is signaled by the MSF setting the DMA-done bit or the DMA–error bit in the MSF memory–cause register and by generating a DMA–done or error interrupt to the CPU, provided the DMA interrupt enable is set. Only one of the DMA–done and error bits will be set on the end of a DMA sequence. Polling the DMA–done bit is undesirable, since this can stall the DMA activity. When disabled, the DMA type is set to 00. The DMA error bit is set when a bus error or bus timeout occurs at the external memory interface during a DMA transaction. The Write bus error–interrupt bit in the MSF cause register will also be set on a write bus error to indicate in which cycle (read or write) the DMA error occurred. Bus errors cannot occur on the data cache bus.

DMA can be stopped and restarted at any time. To stop a DMA in progress, simply write DMA type = 0 in the configuration register. The DMA controller will complete the current read/write sequence and stop after the write is finished. The DMA–done bit will not be set. To restart the DMA, simply rewrite the proper DMA type in the configuration register.

In order to begin a DMA sequence the following actions should be taken:

1—Write DMA type = 0 in the configuration register.
2—Clear the DMA–done and error bits in the MSF cause register, and enable interrupts if desired.
3—Write the DMA origin address register.
4—Write the DMA destination address register.
5—Write the DMA blocksize (word count) register.
6—Write the appropriate DMA type (1, 2, 3) in the configuration register.

DMA Registers—The DMA address register format is shown below. Both registers are 30–bit counters and will both increment simultaneously on completion of the current DMA read/write sequence. The DMA executes split transactions; it will arbitrate for the origin bus, read the data word into a DMA temporary holding register, then arbitrate for the destination bus and write the word from the DMA temporary holding register to the destination bus. The DMA temporary holding register is not accessible from software.

DMA Registers

DMA Origin Register

DMA Origin	zero
31:2	1:0

DMA Destination Register

DMA Destination	zero
31:2	1:0

The DMA blocksize register format is shown below. This is simply the count of words to be transferred. A blocksize of 1 will transfer 1 word from source to destination.

DMA Blocksize Register

zero	DMA Blocksize
31:24	23:0

The DMA type field is decoded as follows.

Case	DMA Type
No DMA	00
Cache to Memory	01
Memory to Cache	10
Memory to Memory	11

4.0 I–side Interface Block

The I–side interface block is the interface between the MSF and the I–cache/I–cache interface. The block is responsible for the following general functions.

- o Cache addressing, generating chip–selects
- o Driving CPU control port (CPU microboot, I–side exceptions)
- o Controlling Microboot/Reset
- o Handling uncached reads
- o Performing read/write of cache line

4.1 I–Cache Accesses

The MSF always operates in cache–only mode where the caches are tagless. All read accesses and all word write accesses are one cycle—a cache miss cannot occur. There are no partial word writes. Only the lower address bits are used in cache–only mode and the upper virtual address is ignored. Values other than 0 in the upper unused bits are allowed and will be disregarded by the hardware. All external memory accesses outside the caches are limited to uncached type accesses. The cache line in cache–only mode has the format shown below.

Uncached and Cached Access Instruction Data Path

Data
32

Internal Address Path

Index
29

Cached Address Path

Index	Chip selects
18	1,2 or 4

Partial Word Cache Accesses

The MSF always accesses the caches on a word basis and never needs to perform a partial word read or write.

Cache Read—The CPU always reads the cache in a single clock cycle.

Uncached Read—The CPU does not attempt to read the cache for an uncached read, and although the uncached address will appear on the cache address bus, the CPU asserts cache miss. The MSF decodes the virtual address and determines that the access is uncached. The MSF then requests a memory read operation from the Memory Interface. When completed, the MSF signals Data_ready to the CPU and drives the data bus.

4.3 I–Cache Read/Write

These commands allow the instruction in the I–cache to be read and written by the CPU through the CSI. The command specifies the physical cache index to be accessed.

I-Cache Read/Write Registers

I-Cache Address Register

R	W	zero	Instruction Address	zero
31	30	29:20	19:2	1:0

I-Cache Data Register

Instruction Data
31:0

CSI Read I–cache—The following sequence is used to read the I–cache through the CSI.

1—Programmer writes the address and sets R=1, W=0 in the address register.
2—MSF hardware stalls CPU.
3—MSF hardware reads the addressed I–cache location into the data register.
4—Programmer reads data register through the CSI.

CSI Write I–cache—The following sequence is used to write the I–cache through the CSI.

1—Programmer writes the data register through the CSI.
2—Programmer write the address and sets R=0, W=1 in the address register.
3—MSF hardware stalls CPU.
4—MSF hardware writes the data register to the addressed I–cache location.

It is important to note that the I–cache Data Register is actually two 32–bit registers at the same address: one used for CSI I–cache reads, and one used for CSI I–cache writes. A value written to the I–cache data register will not return the same value as a subsequent data read from the I–cache data register. It will read a value from the previous CSI I–cache read sequence.

5.0 Interrupts and Exceptions

There are three classes of interrupts and exceptions, covered in the sections that follow. There are a total of eight interrupt pins externally available. Five of these interrupts (the exceptions are interrupts 3a,3b,3c) are sampled by the "stall" clock and sent directly to the CPU. Stall–clock sampling means that the interrupt will only be recognized when the CPU is not in an internal stall (uncached access, multiply stall, etc). These interrupts must be held active externally until the CPU (software) clears them in order for them to be properly recognized. The remaining three external interrupts (interrupts 3a,3b,3c) are sampled by the internal clock and registered, stored, and cleared internally by the MSF CSI registers.

External Interrupt Sources

External Interrupts	Mask location	Sent to CPU by
Interrupt0	CPU	core int0
Interrupt1	CPU	core int1
Interrupt2	CPU	core int2
Interrupt3a	CPU&MSF	MSF_interrupt / core int3
Interrupt3b	CPU&MSF	MSF_interrupt / core int3
Interrupt3c	CPU&MSF	MSF_interrupt / core int3
Interrupt4	CPU	core int4
Interrupt5	CPU	core int5

Timer, UART, error, and DMA events generate internal interrupts within the MSF. Internal MSF interrupts are sent through CPU interrupt number 3. The table below summarizes the internal and external interrupts.

Internal Interrupt Sources

Internal Interrupts	Mask location	Sent to CPU by
Timer1 EOC	CPU&MSF	MSF_interrupt / core int3
Timer2 EOC	CPU&MSF	MSF_interrupt / core int3
DMA Done	CPU&MSF	MSF_interrupt / core int3
UART RX Ready	CPU&MSF	MSF_interrupt / core int3
UART TX Ready	CPU&MSF	MSF_interrupt / core int3
Bus Timeout	CPU&MSF	MSF_interrupt / core int3
I-side Bus Error	CPU&MSF	MSF_interrupt / core int3
D-side Bus Error	CPU&MSF	MSF_interrupt / core int3
DMA Error	CPU&MSF	MSF_interrupt / core int3
Write Bus Error	CPU&MSF	MSF_interrupt / core int3

External interrupts 0,1,2,4, and 5 are sent directly to the core and must be held by the interrupting device. When the CPU services the interrupt, software would normally send a command to the interrupting device to clear the interrupt before reenabling interrupts to the CPU (in CP0 register). MSF interrupts can be masked by the MSF mask register (individual) or by the CPU status register (global). When a MSF interrupt is serviced, the CPU should disable the master MSF interrupt (CPU interrupt 3) as it does all external interrupts.

The third class is machine exceptions. All machine exceptions are sent directly to the CPU except a write bus error, which is signaled as an interrupt by the MSF.

Interrupt Mask Register

Bit Type	Function	Event Type	Bit
Enable	Timer 1 End of Count (EOC)	Interrupt	15
Enable	Timer 2 EOC	Interrupt	14
Enable	DMA Done	Interrupt	13
Enable	UART RX Ready	Interrupt	12
Enable	UART TX Ready	Interrupt	11
Enable	External Memory Bus Timeout	Exception	10
Enable	Instruction Read Bus Error	Exception	9
Enable	Data Read Bus Error	Exception	8
Enable	DMA Bus Error	Interrupt	7
Enable	Write Bus Error	Interrupt	6
Enable	External Interrupt 3a	Interrupt	5
Enable	External Interrupt 3b	Interrupt	4
Enable	External Interrupt 3c	Interrupt	3

The bus error bits (10,9,8) are redundant with information provided by the CPU status register and do not need to be enabled during normal operation. These error events will automatically cause a CPU exception. These bits are intended for diagnostic purposes only.

The write bus error (bit 6) is necessary and is the method to determine whether a bus error has occurred on an uncached write transaction. Since uncached writes are posted by the CPU, a bus error on a write will not automatically cause an exception. This interrupt is intended for this purpose and should always be enabled under normal circumstances.

When an interrupt or exception occurs, software can check the cause by examining the MSF extended–cause (Xcause) register accessible through the CSI. The Xcause register is defined below. All MSF interrupts listed in the table below are *or'ed* into the MSF master interrupt to the CPU, which is interrupt number 3. After the reset sequence, all maskable MSF interrupts/exceptions are cleared and the masks are set to disabled.

Interrupt Cause Register

Bit Type	Function	Event Type	Bit
Xcause	Timer 1 End of Count (EOC)	Interrupt	15
Xcause	Timer 2 EOC	Interrupt	14
Xcause	DMA Done	Interrupt	13
Xcause	UART RX Ready	Interrupt	12
Xcause	UART TX Ready	Interrupt	11
Xcause	External Memory Bus Timeout	Exception	10
Xcause	Instruction Read Bus Error	Exception	9
Xcause	Data Read Bus Error	Exception	8
Xcause	DMA Bus Error	Interrupt	7
Xcause	Write Bus Error	Interrupt	6
Xcause	External Interrupt 3a	Interrupt	5
Xcause	External Interrupt 3b	Interrupt	4
Xcause	External Interrupt 3c	Interrupt	3

After an interrupt is serviced, software must reset the interrupt bit in the interrupt register by writing a one to it through the CSI. The action of writing a one causes that bit in the interrupt register to be reset and all other bits to remain unchanged. It is important that if the interrupting source is an external interrupt or the UART, the interrupt must be cleared in the interrupting device first, followed by clearing the bit in the Xcause register.

6.0 Memory Interface Block

The Memory nterface block performs all interface functions between the local bus and the MSF. The memory interface receives read/write requests with physical addresses from other MSF blocks and controls the access to all the physical address space.

6.1 Local Bus Operation

The local bus, or M–bus, is where all devices other than cache SRAM are connected. The MSF local bus has the following format. Note that the byte number and access type are used internally for non-word accesses. Lane write strobes are available for byte lane determination on writes. Byte–lane on reads can be determined by access type and byte address.

Local Bus Pinout

Data	Physical Byte Index	Lane Write Enables	Output Enable	Address Strobe	Write	I-side Access	Chip Select
32	29	4	1	1	1	1	3

Bus Request / Grant / Busy—The MSF has absolute control over the local bus. External masters can assert M_busrequest_n to the MSF, which will assert M_busgrant when the bus is available. The external master can use the local bus until it releases the M_busrequest_n signal or until the bus timeout timer expires. The bus timer can be disabled by deasserting the M_enabletimeout_n signal.

When the MSF has granted the bus and subsequently needs the bus, the MSF asserts M_busbusy. This is an indication to external masters that the CPU is blocked due to the bus wait. Normally, external masters should release the local bus within a few clock cycles of observing the M_busbusy signal to avoid stalling the CPU. M_busbusy is asserted at any time when the CPU/DMA requires the bus or is using the bus.

It is important to note that CSI registers are treated as any other external device; i.e., they are not accessible during an external bus grant or a DMA operation.

Local Bus Priority—The DMA channel, external devices and the CPU all vie for access to the local bus. External bus requests receive the highest priority. CPU read/write requests have second priority. The internal DMA channel has third priority.

Bus Error Termination—At any time during a bus–read or bus–write transaction, an external device can assert M_buserror_n to signal an abnormal termination of a bus transaction. The MSF generates a local bus error exception with an I–side general exception if the error was caused by an Instruction fetch. The MSF generates a local bus error exception with a D–side general exception if the error was caused by a Data fetch. On a bus–write transaction, an external device can assert M_buserror_n to signal an abnormal termination. On a write–bus error, no exception is generated within the MSF, but an interrupt is generated and this condition must be signaled to software in all normal circumstances. On a bus error, the MSF terminates the memory access and clears a pending read or write. Bus–error termination can also be caused by a bus timer timeout.

6.2 Read/Write Function

Partial Word Accesses—The memory interface supports byte and half–word reads and writes. Reads occur as a normal full–word fetch that is passed unmodified to the CPU; the CPU performs the necessary alignment operations. The external device is expected to drive the full 32–bit data bus and to present the data in the proper lanes; pullups should be used if necessary.

The Mongoose asserts the appropriate lane's write strobe(s) on a partial word write. The table below identifies the bytes accessed by the possible types of partial word accesses. This table applies to non–word cache accesses as well as non–word memory accesses.

Byte Lane Definition

Access type	**Byte no.**	**Bytes accessed**							
	va1, va0	**Big Endian 31 ------- 0**				**Little Endian 31 ------- 0**			
Word -11	**00**	0	1	2	3	3	2	1	0
Tri-byte	**00**	0	1	2	-	-	2	1	0
10	**01**	-	1	2	3	3	2	1	-
Half-word	**00**	0	1	-	-	-	-	1	0
01	**10**	-	-	2	3	3	2	-	-
Byte	**00**	0	-	-	-	-	-	-	0
00	**01**	-	1	-	-	-	-	1	-
	10	-	-	2	-	-	2	-	-
	11	-	-	-	3	3	-	-	-

Wait–State Generator and Data Ready—A provision is included in the MSF to generate 0 to 15 wait states to accommodate slower devices for reading and writing in uncached space. The programmer can set the wait–state value through the MSF configuration register in the CSI. On reset, the wait count defaults to 15. The programmed wait is generated whenever the M_enablewait_n signal is asserted by an external device; this applies to EPROM space, I/O space and global memory space (M_epsel_n, M_iosel_n, M_memsel_n). Asserting M_enablewait_n enables the timeout of the counter; not the count itself.

The M_data_rdy_n external pin can also be used to control the timing of a memory or I/O transaction. When M_enablewait_n is inactive, the memory transaction ends when M_data_rdy_n is active or a local bus timeout or bus error occurs. When M_enablewait_n is active, the wait–state generator controls the end of the transaction and the M_data_rdy_n signal can be used to shorten the access. The transaction ends if M_data_rdy_n is asserted anytime before the wait–state generator reaches the end of count. The transaction also terminates if a bus error or bus timer timeout occurs.

The table below summarizes the possible cases for bus cycle termination.

Local Bus Cycle Termination Conditions

Length of Access	**Start of Access** M_data_rdy_n is	**During Access** M_enablewait_n is	**End of Access** M_data_rdy_n is
Fixed delay	Inactive	Active	Inactive
Variable delay	Inactive	Inactive	Active
Fixed delay with early termination	Inactive	Active	Active

Local Bus Timeout Error—If M_data_rdy_n is not asserted within the bus timeout period, a local bus timeout exception will occur if enabled. The MSF has a 16–bit bus timer that is set to the maximum value at the beginning of each bus access. When this timer is enabled externally by M_enabletimeout_n, the counter decrements until the access terminates or the zero value is reached. M_enabletimeout_n functions as a count enable for the bus timer.

When the timeout timer reaches zero, the counter stops and generates the internal timeout exception signal and external M_bustimeout. This signal is gated with the M_enabletimeout_n input. If M_enabletimeout_n is used to disable the timeout timer, the timer will stop counting. The exception is inhibited by the M_enabletimeout_n signal (basically a timer count enable).

External Memory Interface Chip Selects—Three decoded chip selects are available at the external memory interface. These chip selects are straight decodes of the current address and are not internally disabled when no accesses are occurring. These chip selects can be gated with M_as_n if they need to be disabled when no accesses are occurring.

EPROM Access Options (M_epsel_n)—Both word–wide and byte–wide EPROM interfaces are supported. When the M_bytewide signal is asserted, four bytes are sequentially fetched to form each 32–bit word. The M_epsel_n chip select signal is generated for EPROM address space. In byte–wide mode, the data is read from M_data (7:0) and pull–ups are expected to hold the upper data bits M_data (31:8). CPU uncached data/instruction reads support byte–gather operation (this is intended to be used for the EPROM interface but byte gathering is not inhibited for M_memsel_n or for M_iosel_n type accesses). DMA operations do not support byte gathering. Partial word reads are also supported in byte gathering. The first byte address will be the starting byte address of the partial word, and a total of four byte reads will always occur, regardless of the partial word–read data size. Write–scattering on writes to byte–wide devices does not occur.

I/O Address Space (M_iosel_n)—The M_iosel_n chip select is generated for memory mapped I/O address space.

Global Memory Address Space (M_memsel_n)—The M_memsel_n chip select is generated for global memory address space.

7.0 Timer/Counter Block

This block provides two programmable timer/counters that are accessible to the programmer through the memory–mapped CSI. The block consists of two 32–bit counters. All timer/counter registers are preloaded with a start count through the CSI. Upon end–of–count, the Done flag is set and a CPU interrupt is generated if the appropriate timer interrupt enable is set. On interrupt, software can interrogate the status register to determine which timer event(s) caused the interrupt and can read the timer value to determine any residual count. The following CSI registers control the Timer/Counter Block.

Timer Registers

Timer 1 count register	Timer 1 Count
	32

Timer 2 count register	Timer 2 Count
	32

MSF configuration register	Timer 2 enable	Timer 1 EOC	Timer 2 EOC
	1	1	1

Interrupt mask register	Timer 1 interrupt enable	Timer 2 interrupt enable
	1	1

Interrupt cause register	Timer 1 cause	Timer 2 cause
	1	1

7.1 Timer 1 Function (Watchdog Timer)

Timer 1 is a 32–bit timer designed as a watchdog timer. The timer is preset to the maximum count (0x FFFF FFFF) on reset and decrements on the system clock. Timer 1 cannot be enabled/disabled by software. When the count reaches 0x 0000 0000, the end–of–count signal is generated on the Mongoose output pin T_watch1_out_n. If the timer 1 interrupt is enabled, the end–of–count also generates a maskable interrupt to the CPU. Software can read and write the timer at any time in the count sequence. The timer does not stop counting at end of count; it rolls over to 0xFFFF FFFF. The timer is enabled by the T_enablewatch1_n pin, which is intended for test purposes.

The T_watch1_out_n output can be used to reset the processor in the event that the software fails to reset the watchdog timer before end–of–count occurs. For debugging purposes, it may be desirable to have an external jumper that disables this reset. The maskable timer 1 interrupt can be used to verify correct operation of the watchdog without resetting the board; the interrupt can be masked to disable the test function. The watchdog timer is not intended to be used with the interrupt other than for test purposes.

Timer1 Count Register

Timer 1 Count
31:0

7.2 Timer 2 Function (General–Purpose Timer/Counter)

Timer 2 is a 32–bit general–purpose timer/counter. The timer is preset to the maximum count (0x FFFF FFFF) on reset and decrements on the system clock, if enabled. The starting count is set up by writing the count value register with the initial count value. Counting begins when the count–enablebit of the configuration register is set and the T_enabletimer2_n pin is active. The counter can be read/written at any time in an enabled or disabled mode. The counter counts down until zero is signaled, which generates the end-of-count (EOC). The counter rolls over and continues decrementing until disabled. The end–of–count is signaled as a maskable interrupt as well as a pulse on the T_timer2out pin.

Timer2 Count Register

Timer 2 Count
31:0

8.0 Control and Debug Block

This block provides system control functions.

8.1 RS–232 Port

A standard full–duplex two–line RS–232 interface is provided in the MSF and is compatible with the Intel 8251 (drivers must be supplied externally to conform to the RS–232 electrical specification). No modem control is provided. An externally supplied baud clock is provided through the C_uartclockin pin. The serial data is sent out over the U_uartdata_out pin and serial data is received through the U_uartdata_in pin.

8.2 System Clock Generation

Mongoose requires only a 2x system input clock. The clock is internally divided by 2.

8.3 System Power–on Reset Generation

The Mongoose should reset immediately on power–up and should have an active clock input to ensure that tristate controls are properly reset.

8.4 MSF Configuration Register

The MSF configuration register is located in the debug/control block. This register contains configuration information for the memory block, D–side and I–side interface blocks, and the timer/counter block. The command type bits in the configuration register are read/write accessible through the CSI.

MSF Configuration Register

Function Type	Function		Bit Type	Bit
not used				31
timer	Timer 2 enable		command	29
not used				28
DMA	DMA type		command	27:26
	00	No DMA		
	01	D-Cache to Memory		
	10	Memory to D-Cache		
	11	Memory to Memory		
not used				25:21
mem	Wait state count (0 to 15 wait states)		command	20:17
not used				16:15

8.5 MSF Microboot

The MSF microboot register is located in the MSF. When Ctl_reset_n transitions from low to high, the MSF initiates the microboot sequence to initialize critical configuration parameters that are not accessible under software control. The microboot initialization can require up to 30 clock cycles. The microboot word is stored in EPROM with the format given here. The MSF microboot register has the same format as the microboot EPROM word. After a reset is initiated, the microboot address (virtual address 0xBFFF FFFC; physical address 0x1FFF FFFC) is output on the external–memory address bus with the M_epsel_n and all proper control signals. This will be a 15–cycle read access and will automatically terminate itself. Microboot cannot be terminated with either a M_datardy_n or a M_buserror_n. Normally, the microboot data will be the last word in the boot prom.

Microboot Word and Register Format

PROM Bit number	Name / Description
7:5	Cache chip select address mux decode (2:0) 000 = decode address(13:12) 001 = decode address(14:13) 010 = decode address(15:14) 011 = decode address(16:15) 100 = decode address(17:16) 101 = decode address(18:17) 110 = decode address(19:18) 111 = all chip selects not active
4:3	I-cache number of chip select decode (1:0) 00 = 1 chip select in use 01 = 2 chip selects in use 11 = 4 chip selects in use
2:1	D-cache number of chip select decode (1:0) 00 = 1 chip select in use 01 = 2 chip selects in use 11 = 4 chip selects in use
0	Not used.

The microboot data is sent to the CPU through the I–side control port during the reset sequence. There are 36 bits of microboot data sent to the CPU through the control port as nine nibbles of four bits. Only the bits listed above may be configured by the user; all others are hard–wired, as they are not used by Mongoose.

9.0 Command/Status Interface (CSI) Registers

The CSI subsection of KSEG1 contains all the control registers for the Mongoose support functions. All CSI registers are read and write accessible by word addresses (low two–address bits are 00). All CSI addresses take the form shown below. Bits marked with x are not decoded.

1011	1111	01xx	xxxx	xxxx	xxxx	xxnn	nn00

The register number given in the following table is the nnnn subfield in the CSI address.

CSI Register Summary

Register Name	Register Number	CSI Address	
Interrupt Mask	0x00	0x BF40 0000	
Extended Cause	0x01	0x BF40 0004	
Configuration	0x02	0x BF40 0008	
DMA Origin/Type	0x03	0x BF40 000C	
DMA Destination	0x04	0x BF40 0010	
DMA Blocksize	0x05	0x BF40 0014	
Timer1 Count	0x06	0x BF40 0018	
Timer2 Count	0x07	0x BF40 001C	
Not used	0x08	0x BF40 0020	
Uart Command/Status	0x09	0x BF40 0024	
Uart Data	0x0A	0x BF40 0028	
I-cache Address	0x0B	0x BF40 002C	
I-cache Data	0x0C	0x BF40 0030	
Not used	0x0D to 0x0F		

CSI Register Definition

Interrupt Mask Register

Bit Type	Function	Event Type	Bit
Enable	Timer 1 End of Count (EOC)	Interrupt	15
Enable	Timer 2 EOC	Interrupt	14
Enable	DMA Done	Interrupt	13
Enable	UART RX Ready	Interrupt	12
Enable	UART TX Ready	Interrupt	11
Enable	External Memory Bus Timeout	Exception	10
Enable	Instruction Read Bus Error	Exception	9
Enable	Data Read Bus Error	Exception	8
Enable	DMA Bus Error	Interrupt	7
Enable	Write Bus Error	Interrupt	6
Enable	External Interrupt 3a	Interrupt	5
Enable	External Interrupt 3b	Interrupt	4
Enable	External Interrupt 3c	Interrupt	3

Interrupt Xcause Register

Bit Type	Function	Event Type	Bit
Xcause	Timer 1 End of Count (EOC)	Interrupt	15
Xcause	Timer 2 EOC	Interrupt	14
Xcause	DMA Done	Interrupt	13
Xcause	UART RX Ready	Interrupt	12
Xcause	UART TX Ready	Interrupt	11
Xcause	External Memory Bus Timeout	Exception	10
Xcause	Instruction Read Bus Error	Exception	9
Xcause	Data Read Bus Error	Exception	8
Xcause	DMA Bus Error	Interrupt	7
Xcause	Write Bus Error	Interrupt	6
Xcause	External Interrupt 3a	Interrupt	5
Xcause	External Interrupt 3b	Interrupt	4
Xcause	External Interrupt 3c	Interrupt	3

MSF Configuration Register

Function Type	Function		Bit Type	Bit
not used				31
timer	Timer 2 enable		command	29
not used				28
DMA	DMA type		command	27:26
	00	No DMA		
	01	D-Cache to Memory		
	10	Memory to D-Cache		
	11	Memory to Memory		
not used				25:21
mem	Wait state count (0 to 15 wait states)		command	20:17
not used				16:15

DMA Origin Register

DMA Origin	zero
31:2	1:0

DMA Destination Register

DMA Destination	zero
31:2	1:0

DMA Blocksize Register

zero	DMA Blocksize
31:24	23:0

Timer1 Count Register

Timer 1 Count
31:0

Timer2 Count Register

Timer 2 Count
31:0

Uart Command/Status Register

zero	Command/Status
31:8	7:0

Uart Data Register

zero	Data
31:8	7:0

I-Cache Address Register

R	W	zero	Instruction Address	zero
31	30	29:20	19:2	1:0

I-Cache Data Register

Instruction Data
31:0

Appendix A: References

Kane, Gerry, *mips RISC ARCHITECTURE*, New Jersey, Prentice Hall, 1989. ISBN 0-13-584749-4

LSI Logic Corporation, *LR33000 Self–Embedding Processor User's Manual*, California, LSI Logic Corporation Literature Distribution, 1991.

Intel Corporation, *Microcommunications Handbook (i8251)*, Illinois, Intel Literature Sales, 1991.

For additional information, please contact:

Brian S. Smith
NASA Goddard Space Flight Center
Code 735.2
Greenbelt, MD 20771
(301) 286–9601 (FAX x9214)
email: brian@Mongoose.gsfc.nasa.gov or bssmith@gsfcmail.gsfc.nasa.gov

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1993	3. REPORT TYPE AND DATES COVERED Reference Publication

4. TITLE AND SUBTITLE Mongoose Application–Specific Integrated Circuit (ASIC) Microcontroller Programming Guide	5. FUNDING NUMBERS Code 735
6. AUTHOR(S) Brian S. Smith	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Goddard Space Flight Center Greenbelt, Maryland 20771	8. PERFORMING ORGANIZATION REPORT NUMBER 93E02366
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, D.C. 20546–0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA RP–1319

11. SUPPLEMENTARY NOTES

Brian S. Smith: Data Processing Devices Section, Flight Data Systems Branch, NASA–GSFC.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified–Unlimited Subject Category 33 Report available from the NASA Center for AeroSpace Information, 800 Elkridge Landing Road, Linthicum Heights, MD 21090; (301) 621–0390.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The "Mongoose" ASIC microcontroller is a radiation–hard implementation of the R3000 microprocessor. This document describes the internals of the microcontroller in a level of detail necessary for someone implementing a software design.

14. SUBJECT TERMS Flight Data Systems, Rad–hard, Microcontroller, Microprocessor	15. NUMBER OF PAGES 34
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT Unlimited

National Aeronautics and
Space Administration
Code JTT
Washington, D.C.
20546-0001

Official Business
Penalty for Private Use, $300

SPECIAL FOURTH–CLASS RATE
POSTAGE & FEES PAID
NASA
PERMIT No. G27

S3 001 RP-1319 930921 S090569 A
NASA
CENTER FOR AEROSPACE INFORMATION
ACCESSIONING
800 ELKRIDGE LANDING ROAD
LINTHICUM HEIGHTS MD 210902934

POSTMASTER: If Undeliverable (Sectio 158,
Postal Manual) Do Not eturn

GREENBELT MD OCT 1 93

U.S. OFFICIAL MAIL
U.S. POSTAGE
PENALTY FOR PRIVATE USE $300
00.98
H METER 53993