SYNOVA

1333 Gateway Drive, Suite 1017, Melbourne FL 32901, (407) 726-0796
Visit us at our web site http://www.synova.com

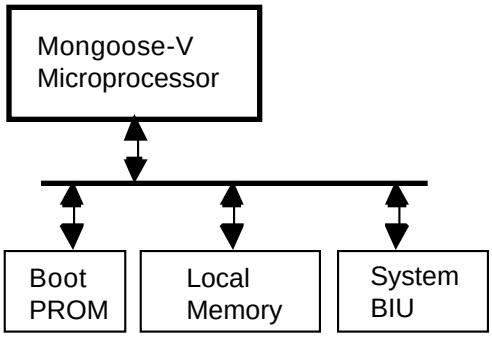
Mongoose-V 32-bit MIPS Microprocessor

Architecture Description

January 30, 1997 (rev 1.3)

1.0 PROCESSOR OVERVIEW{ TC "1.0 PROCESSOR OVERVIEW" \l 1 }

The Mongoose-V is a single chip rad-hard microprocessor with integrated cache memory and integrated peripheral functions. As shown in the diagram below, the Mongoose-V can be used to form a complete rad-hard computer by the straight-forward addition of local memory and system-specific interface logic.



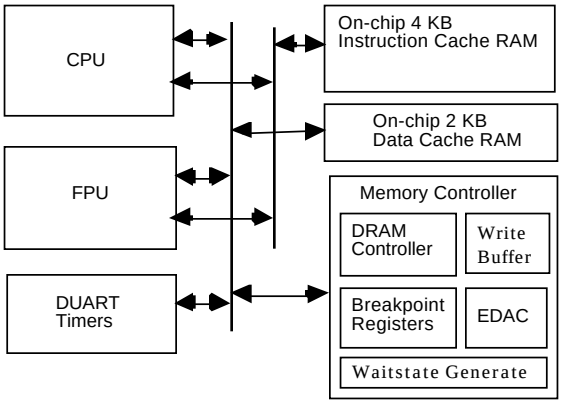
The Mongoose-V implements the MIPS R3000 instruction set including the MIPS floating point hardware. On-chip caches provide high bandwidth memory access to keep the CPU operating efficiently. The data cache size is 2 KB and the instruction cache size is 4 KB.

The on-chip Memory Controller provides the necessary interface control between the CPU core and the local bus. The Memory Controller supports both SRAM and DRAM systems and provides on-chip wait-state control. In addition, the following functions are provided on-chip.

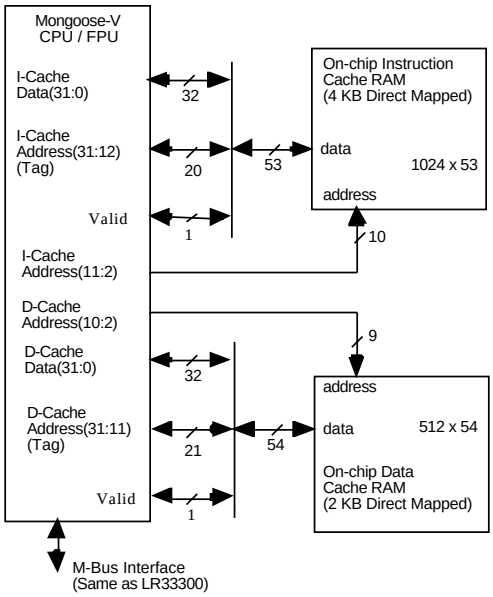
- Error Detection and Correction (EDAC)
- Expansion Interrupts
- Dual UART
- Timers
- Debug Registers
- Write Buffer

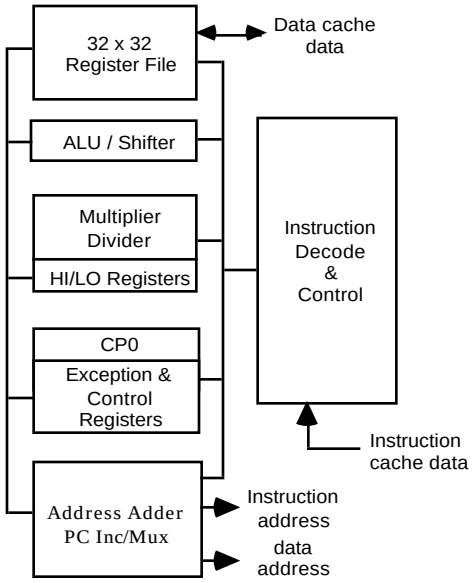
2.0 CACHE ARCHITECTURE{ TC "2.0 CACHE ARCHITECTURE" \l 1 }

The Mongoose-V provides internal instruction and data caches. The data cache size is 2 KB and the instruction cache size is 4 KB. The cache architecture for both instruction and data caches is direct mapped as shown in the following diagram.



This document contains preliminary information that has not yet been verified with production devices.





The data cache line consists of a single 32-bit data word, the valid flag and the cache tag as shown below.

Valid	Tag	Data
1	21	32

The instruction cache line consists of a single 32-bit instruction word, the valid flag and the cache tag as shown below.

Valid	Tag	Data
1	20	32

3.0 CPU ARCHITECTURE{ TC "3.0 CPU ARCHITECTURE" \l 1 }

The CPU consists of a RISC architecture with a five stage pipeline. The functional blocks of the processor are shown in the diagram below. The Register File has 32 words of 32 bits each. The ALU/shifter block performs arithmetic and logical operations while the multiplier/divider block can perform these functions in parallel with the ALU operation. Coprocessor zero (CP 0) handles exceptions and control functions.

Instruction Pipeline

The CPU instruction pipeline consists of five stages without interlocking. The five stages are described below.

IF	RD	ALU	MEM	WB
1	2	3	4	5

IF	Instruction Fetch.
RD	Read operands from Register File and Decode the instruction.
ALU	ALU operation.
MEM	Read or write the data cache and memory if needed.
WB	Write Back result from ALU instruction or the result from a Load instruction.

Delayed Instruction Slots

Instructions normally execute in one clock cycle and the result from one instruction is available to the next instruction without delay. There are two cases where this is not the case due to pipeline delays.

The result from a Load instruction is not available to the next instruction and is delayed one cycle. In addition, a Jump or Branch has a delay of one cycle before the program counter is actually changed. The result is that the instruction immediately after a Jump or Branch is always executed before the Jump or Branch occurs.

CPU Registers

The CPU registers include the 32 General Registers and other specialized groups of 32-bit registers. The HI/LO register pair is used for passing operands and results between the Multiplier/Divider and the Register File.

General Registers
r0 = hardwired zero
r1
r2
.....
r31

Multiply - Divide Registers
HI
LO

Program Counter
PC

Endianess

Both Big and Little Endian data formats are supported in the Mongoose-V. The table below shows the byte ordering for each format.

	Byte Numbering			
	31:27	23:16	15:8	7:0
Big Endian	3	2	1	0
Little Endian	0	1	2	3

Coprocessor Zero Registers

Coprocessor zero contains a set of specialized registers that are related to CPU control functions. These are summarized below:

Status Register - Contains all the major status bits for the CPU.

Cause Register - This register indicates the cause of the last exception.

Bad Address Register - Saves the memory address that caused an addressing error. Does not apply to the bus error exception.

Target Address Register- Saves return address for restarting after an exception.

Exception PC Register- Holds address where processing resumes after an exception.

Revision ID Register- Contains ID number for this processor version.

Debug/Cache Invalidate Control Register- Contains cache test and breakpoint control bits.

Breakpoint PC Register- Contains instruction breakpoint address.

Breakpoint PC Mask Register- Contains instruction breakpoint address mask.

Breakpoint Data Address Register- Data tracepoint address.

Breakpoint Data Mask Address Register- Data tracepoint address mask

Command-Status Registers

In addition to the CPU registers, there is a group of memory mapped command-status registers used to control and configure the Mongoose-V. These are listed in the section on Memory Organization.

4.0 CPU INSTRUCTION SET{ TC "4.0 CPU INSTRUCTION SET" \l 1 }

All instruction are 32 bits wide. There are three basic formats for the CPU instructions. These formats are shown below. The CPU instruction set is summarized on the following page.

I-Type (Immediate) Instruction Format			
opcode	**rs**	**rt**	**immediate**
31:26	**25:21**	**20:16**	**15:0**

J-Type (Jump) Instruction Format	
opcode	**target**
31:26	**25:0**

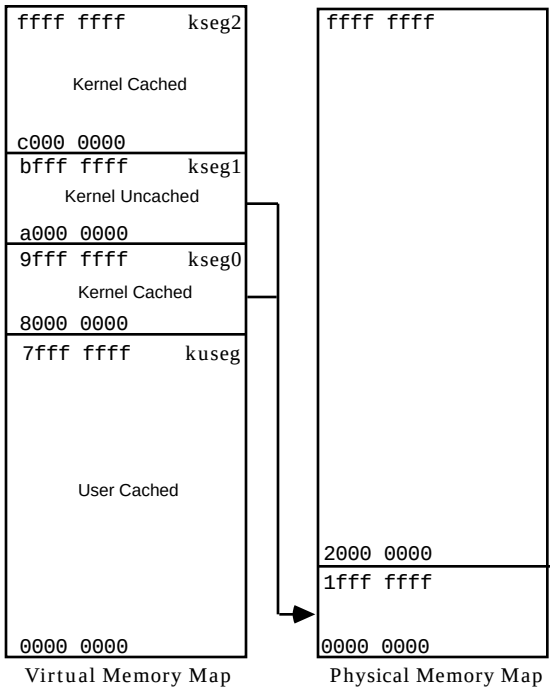
R-Type (Register) Instruction Format					
opcode	**rs**	**rt**	**rd**	**re**	**funct**
31:26	**25:21**	**20:16**	**15:11**	**10:6**	**5:0**

CPU Instruction Set Summary

Load / Store Instructions		Multiply - Divide Instructions	
LB rt,offset(base)	Load Byte	MULT rs,rt	Integer Multiply
LBU rt,offset(base)	Load Byte Unsign	MULTU rs,rt	Integer Mult Unsign
LH rt,offset(base)	Load Halfword	DIV rs,rt	Integer Divide
LHU rt,offset(base)	Load Halfword Unsign	DIVU rs,rt	Integer Divide Unsign
LW rt,offset(base)	Load Word	MFHI rd	Move From HI
LWL rt,offset(base)	Load Word Left	MFLO rd	Move From LO
LWR rt,offset(base)	Load Word Right	MTHI rd	Move To HI
SB rt,offset(base)	Store Byte	MTLO rd	Move To LO
SH rt,offset(base)	Store Halfword	Jump Instructions	
SW rt,offset(base)	Store Word		
SWL rt,offset(base)	Store Word Left	J target	Jump
SWR rt,offset(base)	Store Word Right	JAL target	Jump and Link
ALU Instructions		JR rs	Jump Register
		JALR rs,rd	Jump and Link Reg
ADDI rt,rs,imm	Add Immediate	Branch Instructions	
ADDIU rt,rs,imm	Add Immediate Unsign		
SLTI rt,rs,imm	Set on < Immed	BEQ rs,rt,offset	Branch on Equal
SLTIU rt,rs,imm	Set on < Unsigned Imm	BNE rs,rt,offset	Branch on Not Equal
ANDI rt,rs,imm	And Immediate	BGTZ rs,offset	Branch on > 0
ORI rt,rs,imm	Or Immediate	BLTZ rs,offset	Branch on < 0
XORI rt,rs,imm	Exclusive Or Immed	BGEZ rs,offset	Branch on >= 0
LUI rt,rs,imm	Load Upper Immed	BLEZ rs,offset	Branch on <= 0
ADD rd,rs,rt	Add	BLTZAL rs,offset	Branch on < 0 & Link
ADDU rd,rs,rt	Add Unsigned	BGEZAL rs,offset	Branch on >= 0 & Link
SUB rd,rs,rt	SUB	Special Instructions	
SUBU rd,rs,rt	SUB Unsigned		
SLT rd,rs,rt	Set on <	SYSCALL	System Call
SLTU rd,rs,rt	Set on < Unsign	BREAK	Breakpoint
AND rd,rs,rt	Logical And	MTC0 rt,rd	Move To CP0
OR rd,rs,rt	Logical Or	MFC0 rt,rd	Move From CP0
XOR rd,rs,rt	Logical Exclusive Or	RFE	Return from Exception
Shift Instructions			
SLL rd,rt,shamt	Shift Left Logical		
SRL rd,rt,shamt	Shift Right Logical		
SRA rd,rt,shamt	Shift Right Arithmetic		
SLLV rd,rt,rs	Shift Left Logical Var		
SRLV rd,rt,rs	Shift Right Log Var		
SRAV rd,rt,rs	Shift Right Arith Var		

5.0 MEMORY ORGANIZATION{ TC "5.0 MEMORY ORGANIZATION" \l 1 }

The processor virtual address is 32 bits wide with the low two bits reserved for byte addressing. The virtual address space is divided into four segments. These four segments are mapped into physical memory as shown in the diagram below.



Processes executing with kernel privileges are allowed access to all four segments. Processes executing in User Mode are restricted to accessing the Kuseg only. All segments are cached except for the kseg1. In addition, kseg0 and kseg1 are hardwired to be mapped to the low 0.5 GB of physical memory. kseg2 and kuseg are directly mapped to physical memory.

The physical memory arrangement of the Mongoose-V processor is divided into several segments.

SPEC0 (IOSEL) - This 16 MB section is controlled by the SPEC0 waitstate generator and is used for IO device address space.

SPEC1 (EPSEL) - This 16 MB section is controlled by the SPEC1 waitstate generator and is used for EPROM address space.

SPEC2 - This 16 MB section is controlled by the SPEC2 waitstate generator and is available for general use.

SPEC3 - This 16 MB section is controlled by the SPEC3 waitstate generator and is available for general use.

SRAM - This 192 MB section is controlled by the SRAM waitstate generator and is used for SRAM memory address space.

DRAM - This 256 MB section is controlled by the DRAM Controller and is used for Dynamic RAM memory address space.

ffff ffff fffe 0000	Command-Status Registers
0000 0000 2000 0000	
1fff ffff 1f00 0000	SPEC0 IOSEL (16 MB)
1eff ffff 1e00 0000	SPEC1 EPSEL (16 MB)
1dff ffff 1d00 0000	SPEC2 (16 MB)
1cff ffff 1c00 0000	SPEC3 (16 MB)
1bff ffff 1000 0000	SRAM (192 MB)
0fff ffff 0000 0000	DRAM (256 MB)

Command-Status Registers - The topsection of physical memory contains the memory mapped command-status registers. These registers are used to control the Dynamic Automatic Waitstate Generator (DAWG) as well as other functions in the Mongoose-V processor. The command-status registers are listed in the table below.

Name	Memory Address
Timer 0 Initial Count	0x fffe.0000
Timer 0 Control	0x fffe.0004
Timer 1 Initial Count	0x fffe.0008
Timer 1 Control	0x fffe.000c
Refresh Timer Initial Count	0x fffe.0010
SRAM Configuration	0x fffe.0100
SPEC0 Configuration	0x fffe.0104
SPEC1 Configuration	0x fffe.0108
SPEC2 Configuration	0x fffe.010c
SPEC3 Configuration	0x fffe.0110
DRAM Configuration	0x fffe.0120
BIU/Cache Configuration	0x fffe.0130
Command	0x fffe.0180
Status	0x fffe.0184
Clear Interrupt	0x fffe.0184
Interrupt Cause	0x fffe.0188
Set Interrupt	0x fffe.0188
Interrupt Mask	0x fffe.018c
EDAC Error Address	0x fffe.0190
EDAC Parity	0x fffe.0194
MAVN Test	0x fffe.01b4
MAVN Access Privilege	0x fffe.01b8
MAVN Access Violation	0x fffe.01bc
MAVN Range 0	0x fffe.01c0
MAVN Range 1	0x fffe.01c4
MAVN Range 2	0x fffe.01c8
MAVN Range 3	0x fffe.01cc
MAVN Range 4	0x fffe.01d0
MAVN Range 5	0x fffe.01d4
UART 0 Rx Buffer	0x fffe.01e8
UART 0 Tx Buffer	0x fffe.01ec
UART 0 Baud Rate	0x fffe.01f0
UART 1 Rx Buffer	0x fffe.01f4
UART 1 Tx Buffer	0x fffe.01f8
UART 1 Baud Rate	0x fffe.01fc

6.0 FLOATING-POINT UNIT{ TC "6.0 FLOATING-POINT UNIT" \l 1 }

The on-chip floating-point unit implements IEEE standard floating-point arithmetic in a five stage pipelined processor architecture. The FPU interfaces to the CPU as Coprocessor number one (CP 1). The FPU supports IEEE single (32 bit) and double precision (64 bit) operations. There are sixteen 64-bit general purpose floating-point registers.

The FPU instruction pipeline consists of sfive stages. The stages are described below.

IF	RD	ALU	MEM	WB
1	2	3	4	5

IF	Instruction Fetch.
RD	Read operands from Register File and Decode the instruction.
ALU	Floating-point ALU operation.
MEM	Perform a coprocessor load or store operation.
WB	Write Back result from ALU instruction.

The FPU executes three types of instructions. Load / Store instructions are used to move data between the CPU Register File and the data cache. ALU instructions perform floating-point operations on operands in the Register File. Branch instructions are used to test FPU conditions. The FPU instruction set is shown in the table below.

Load / Store Instructions	
LWC1 ft,offset(base)	Load Word
SWC1 ft,offset(base)	Store Word
MTC1 rt,fs	Move to FPU
MFC1 rt,fs	Move from FPU
CTC1 rt,fs	Move Control to FPU
CFC1 rt,fs	Move Control from FPU
ALU Instructions	
ADD.fmt fd,fs,ft	Add
SUB.fmt fd,fs,ft	Subtract
MUL.fmt fd,fs,ft	Multiply
DIV.fmt fd,fs,ft	Divide
ABS.fmt fd,fs	Absolute Value
MOV.fmt fd,fs	Move
NEG.fmt fd,fs	Negate
CVT.S.fmt fd,fs	Convert to Single
CVT.D.fmt fd,fs	Convert to Double
CVT.W.fmt fd,fs	Convert to Integer
C.cond.fmt fs,ft	Compare
Branch Instructions	
BC1T	Branch on FPU True
BC1F	Branch on FPU False

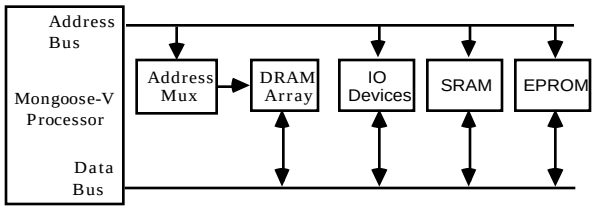
7.0 MEMORY INTERFACE FUNCTIONS{ TC "7.0 MEMORY INTERFACE FUNCTIONS" \l 1 }

The memory controller provides a high level of functional integration. The memory controller includes the following functions:

- DRAM Controller
- 4-deep Write Buffer
- Error Correction and Detection
- Byte Gather-Scatter
- Memory Access Violation

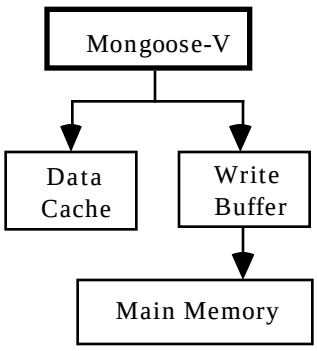
DRAM Controller

The Mongoose-V contains a DRAM controller that allows the Mongoose-V to directly control DRAM arrays with the addition of an external address multiplxer. The DRAM controller also provides a Refresh Timer. The DRAM controller is configured through the DRAM Configuration Register. The Mongoose-V supports systems that combine DRAM, SRAM, IO Devices and EPROM as shown in the diagram below:



Write Buffer

An internal four deep write buffer is provided to buffer the writes between the CPU and main memory. as shown in the diagram below.



The CPU utilizes a write-through cache policy where the word written from a Store instruction is written to the data cache and also written to the write buffer. The write buffer then outputs the word to main memory when the memory becomes available. In the event that the Write Buffer becomes full, the CPU will stall when a write is pending.

Error Detection and Correction (EDAC)

Single error correction and double error detection (SEC/DED) using a modified Hamming code are provided for the memory interface. Eight check bits are added to the 32-bit memory data bus to provide the error protection. The check bits IO signals are named PARITY(7:0){ XE "sPARITY(7:0)" }. Note that the byte parity signals that are part of the LR33300 memory bus interface are also available if byte parity is desired but are not used for the EDAC function.

Read / Write Operation

On a memory write, the check bits are generated and written to memory with the data word. On a memory read, the check bits are fetched with the 32-bit data and tested for consistency with the data. If a single bit error has occurred, it is corrected and passed to the processor and the EDAC Cerror{ XE "csEDAC Cerror" } bit is set in the Status Register{ XE "rStatus Register" }. The corrected data is not written back to memory and software scrubbing is responsible for this function. If a double bit error has occurred, the error is detected but can not be corrected. In this case, the transaction is terminated as a bus error if enabled in the Mask Register and the EDAC Uerror{ XE "csEDAC Uerror" } bit in the Status Register is set. If not enabled, the double error is ignored.

EDAC Interrupts

The EDAC interrupts can be controlled by setting the EDAC Cerror and EDAC Uerror mask bits in the Interrupt Mask Register. When the EDAC Cerror bit is set to one in the Interrupt Mask Register, a Cerror Interrupt will occur whenever the EDAC is enabled and a correctable single error is detected by the EDAC. When the EDAC Uerror bit is set to one in the Interrupt Mask Register, a Uerror Interrupt will occur whenever the EDAC is enabled and an uncorrectable error is detected by the EDAC. A Cerror Interrupt and a Uerror Interrupt is indicated in the Interrupt Cause Register and the Status Register as discussed in the Peripheral Function Interrupts section of this document. Software must clear EDAC interrupts by writing a zero to the appropriate interrupt bit in the Interrupt Cause register.

Note that if the Uerror Interrupt is enabled and an uncorrectable error is detected, the current bus transaction will be terminated as a bus error and a bus error exception will occur. If the Uerror Interrupt is not enabled and an uncorrectable error is detected, the current bus transaction is completed normally and the uncorrectable error is ignored.

Error Address Capture

When either a single or double bit error occurs, the word address of the transaction that caused the error is stored in the Error Address register. The Error Address Register is a sticky register and software must read this register before it is enabled to capture another error address. Reading this register does not clear EDAC interrupts. The Error Address Register is shown below.

Error Address	00
31:2	1:0

Enabling EDAC

The error correction function must first be enabled by setting the EDAC Enable{ XE "cEDAC Enable" } bit in the Command Register. In addition, external devices that do not support error correction and detection can disable the feature by deasserting the EDAC_EN input pin, which can be done with an external address decode function. Devices that do not support EDAC can also be located in the upper 256 MB of physical address space; addresses in the range 0x0fff.ffff to 0xf000.0000 automatically disable EDAC checking on memory reads.

EDAC is enabled whenever the EDAC Enable bit in the Command Register is set and the EDAC_EN input pin is asserted and the address is not in the upper 256 MB of physical address space. When EDAC is disabled, the Mongoose-V ignores the check bits on read.

Counting Errors

When a single bit error occurs, the EDAC_CERROR output pin is pulsed to indicate that a correctable error has occurred. When an uncorrectable error occurs, the EDAC_UERROR output pin is pulsed to indicate that a double error has occurred. These output pins can be used to strobe external counters in order to count the number of errors that have occurred. These error counter strobes are active whenever EDAC is enabled (see "Enabling EDAC" paragraph). They are unaffected by the state of the Interrupt Mask.

Non-Word Memory Writes

Non-word memory writes (byte, halfword and tribyte) that occur in the physical address space where EDAC is active require that the Mongoose-V interface perform a read-modify-write operation in order to correctly update the parity check bits. The read-modify-write transaction is implemented as a memory read cycle followed by a memory write cycle in order to maintain bus comparability with the LR33300.

Interface Functions that are Incompatible with EDAC

Some operating modes of the memory interface are not generally compatible with the EDAC function. These include byte gather/scatter and halfword gather/scatter because these modes do not naturally support EDAC. In cases where EDAC conflicts with an existing feature, the user should disable EDAC for that transaction or provide for parity generation on read.

Parity Test Mode

A test mode is provided that allows software to read/write the parity bits for test purposes. To write the parity bits, software writes to the .EDAC Parity Register shown below, and sets the EDAC Write Override in the Command Register.

NU	Parity
31:10	7:0

The EDAC unit then uses the contents of the EDAC Parity Register to drive the memory parity output signals. Software must reset the EDAC Write Override in order to return to normal parity operation.

To read the parity bits, software resets the EDAC Write Override bit in the Command Register. This disables write override and enables the parity read function. On each memory read, the parity from the read is placed in the EDAC Parity Register. Software may read the EDAC Parity Register to determine the value of the parity bits last read into the Mongoose-V device.

Memory Protection

The memory access violation notification feature provides a set of six address ranges that are monitored by the hardware. For each address range, software can specify the access privileges allowed for that range. The range is constrained such that the size of the address range size and the address range alignment must be a power of two.

A single MAVN Range Register specifies the page size and the starting address of each address range. There are six MAVN Range Registers. The size of each range is specified by n, which represents a page size of 2^n KB for $9 \leq n \leq 21$. Each range register has the following format. The "n" value is specified by the Page Size Code. A single MAVN Range Register is shown below.

Start Address	NU	PS Code
31:9	8:5	4:0

The PS Code is used to generate the Page Size Mask as shown in the following table:

PS code	Size bytes	Page Size Mask
9	512	11111111111111111111111000000000
10	1 K	11111111111111111111110000000000
11	2 K	11111111111111111111100000000000
12	4 K	11111111111111111111000000000000
13	8 K	11111111111111111110000000000000
14	16 K	11111111111111111100000000000000
15	32 K	11111111111111111000000000000000
16	64 K	11111111111111110000000000000000
17	128 K	11111111111111100000000000000000
18	256 K	11111111111111000000000000000000
19	512 K	11111111111110000000000000000000
20	1 M	11111111111100000000000000000000
21	2 M	11111111111000000000000000000000

In order to determine if a given address, A, is in a range specified by a Page Size Mask, M, and a Page Start Address, P, the following test is made:

If((A and M) equals (P and M)) then A is in the page defined by {M,P}

Access Violation Handling

The MAVN Access Privilege Register specifies the access privileges for the address ranges. There are two Access Privilege bits for each address range and two Access Privilege bits for the Global Access privilege. The access bits are Read Enable and Write Enable. On reset, all privileges in the MAVN Access Privilege Register are set to Enabled. The MAVN Access Privilege Register is shown below.

Reserved
31:20

Read Access Privilege		
Global	Reserve	Range
19:19	18:16	15:10

Write Access Privilege		
Global	Reserve	Range
9:9	8:6	5:0

The MAVN Enable bit in the Command Register must be set in order to enable memory violation detection.

When a violation occurs on a write or read access a MAVN exception is generated if MAVN exceptions are enabled in the Interrupt Mask Register. The MAVN Read Violation or MAVN Write Violation bit in the Status Register and the Cause Register is set as discussed in the Peripheral Function Interrupts section of this document. On reset, all memory checking is disabled and the Access Privilege Register is set to 0x7fff.ffff.

The violating address is captured in a sticky MAVN Access Violation Register{ XE "rMAVN Access Violation Register" } which is accessible from software. Memory access violation detection is disabled upon the first violation and the MAVN bit in the Command Register is reset. Software must set the MAVN Enable bit in order to re-enables memory violation detection. The MAVN Access Violation Register is locked with the violation address value until software re-enables the MAVN. Re-enabling the MAVN does not clear MAVN related interrupts.

The MAVN Access Violation Register has the following format:

Violation Address	00
31:2	1:0

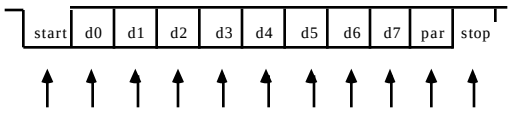
When a memory access occurs that is outside the range of all ranges, the Global Access Privilege bits are examined to determine if the access is allowed. When a memory access occurs that is inside the range of one or more Range Registers, the Access Privilege bits for the ranges involved are or'ed together to determine if the access is allowed.

8.0 COMMUNICATIONS PERIPHERALS{ TC "8.0 COMMUNICATIONS PERIPHERALS" \l 1 }

DUART Serial Ports

The Dual UART provides two full duplex communication channels for asynchronous communication with RS-232 compatible devices. Each UART is accessed through a set of software programmable registers. Each UART provides the basic RS-232 asynchronous protocol.

The data format supported by each UART is: asynchronous, 1 Start bit, 8 Data bits, 1 Parity bit, 1 Stop bit. The order of transmission of data bits is LSB first, MSB last. The Start bit is active low. The Data bits are active high. The Parity bit is even or odd depending on the Command Register programming. The Stop bit is active high.



The diagram above shows the data format and the receive sample timing. The receiver first detects a high to low transition on the data input line which starts the receive sampler timer. One half of a data period later the receiver samples the data input line again in order to confirm the start bit. If the start bit is high at the midpoint sample, the start bit is ignored and the receiver looks for a new start bit. If the start bit is low at the midpoint sample, the receiver samples each of the data bits and parity bit at the midpoint. The stop bit is then sampled at the midpoint. If the stop bit is low at the midpoint, a Framing Error is signaled.

Note that all bit field names referenced in this section are duplicated in the Status Register, once for UART 1 and a second time for 2.

Transmit Operation

The UART Tx Buffer Register{ XE "rUART Tx Buffer Register" } is loaded by software with the eight bit character to be transmitted. The transmit buffer is buffered such that software can write a new character while the old character is in the process of being transmitted. The Tx Buffer Register is shown below.

Reserved	Data
31:8	7:0

The UART serializes and transmits the character over the U_OUT{ XE "sU_OUT" } signal line. The Transmit Enable{ XE "Transmit Enable" } bit in the Command Register must be set before attempting to transmit a character. If the UART Tx Done{ XE "csUART Tx Done" } is one in the Interrupt Mask Register, the UART will generate a Tx Done Interrupt when the character has been transmitted and the UART Tx Done bit in the Status Register and Interrupt Cause Register will be set.

The transmit baud rate is set by the internal baud rate generator described below;. The transmitter generates either odd or even parity depending on the setting of the UART Parity Even{ XE "cUART Parity Even" } flag in the Command Register.

Parity enable / disable causes the following bit sequence to be sent. When parity is enabled, the following bit sequence is transmitted.

Start	Data bits	Parity	Stop
1	8	1	1

When parity is disabled, the following bit sequence is transmitted.

Start	Data bits	One	Stop
1	8	1	1

When the UART transmitter is not transmitting a character, the data output is held in the Marking state (high).

Receive Operation

The UART receives an eight bit character over the U_IN signal line and stores it in the UART Rx Buffer Register. The receive buffer is buffered such that a new character can be received while the buffer waits for software to read the old character. Software must read the old character before the new character receive has been completed in order to prevent an overrun. The Rx Buffer Register is shown below.

Reserved	Data
31:8	7:0

If the UART Rx Done{ XE "csUART Rx Done" } bit in the Interrupt Mask Register is one, the UART will generate an Rx Done Interrupt after the character has been received and the UART Rx Done bit in the Status Register and the Interrupt Cause Register will be set. The Receive Enable{ XE "Receive Enable" } bit in the Command Register must be set before the UART will receive characters.

If a new character is received before software has unloaded the previous character from the Rx Buffer Register, the UART Rx Overrun{ XE "csUART Rx Overrun" } flag is set in the Status Register and Interrupt Cause Register indicating that a character has been lost. The receiver does not check for framing errors and does not check parity.

The receive clock is provided by the internal baud rate generator. Also, the receiver serial input has cascaded flip-flops to reduce metastable behavior.

UART Interrupts

All interrupts must be enabled by software by writing to the Interrupt Mask Register. In addition, all edge sensitive interrupt status bits remain asserted until software resets the status bits by writing to the Interrupt Cause Register.

Modem Control

Each UART supports U_CTS{ XE "sU_CTS" } (Clear To Send) and U_RTS{ XE "sU_RTS" } (Request To Send) IO signals for modem control. Both signals are active low.

The RTS signal is an output that is programmed either high or low by software through the Command Register; a high value in the Command Register causes the output to be driven to the active state (active low output pin).

The transmitter waits until the CTS input signal (active low input pin) is sampled low before transmitting the next character.

Test Operation

The UART provides a loopback path for testing the transmit and receive functions. When the UART Loopback{ XE "cUART Loopback" } flag is set in the Command Register, the receive serial input is connected to the transmit serial output. In addition, the CTS input to the transmitter is connected to the CTS Test command bit to allow software to control the CTS input in loopback mode.

Baud Rate Generator

Each UART has a pair of baud rate generators that create the internal Rx and Tx clocks for the UART. The baud rate generator is programmed by writing to the Baud Rate Register as shown below. The Baud Rate Code is an unsigned integer.

reserved	Rx Baud Rate Code
31:31	30:16

reserved	Tx Baud Rate Code
15:15	14:0

The baud rate generator is driven by the processor clock. The Baud Rate Code depends on the desired baud rate (R) and on the processor operating clock frequency (F). The Baud Rate Code, C is computed by C = round(F/R - 1). For example,

F	R	C
10 MHz	9600	1041
5 MHz	9600	520

In general, the UART is intended to support baud rates such as 9600. Keep C > 100 for best receiver operation. Software response time will impose baud rate limitations, especially at higher baud rates, in order to prevent receive overruns.

UART Command Register Fields

Each uart is controlled by the following bits in the Command Register.

Bit	Description
15	UART 1 Tx Set Parity to Even
14	UART 1 Tx Enable Parity
13	UART 1 Ready to Send
12	UART 1 Tx Enable
11	UART 1 Rx Enable
10	Reserved
9	UART 0 Tx Set Parity to Even
8	UART 0 Tx Enable Parity
7	UART 0 Ready To Send
6	UART 0 Tx Enable
5	UART 0 Rx Enable
4	Reserved
3	Loopback Tx data to Rx data

2	**Connected to CTSN in loopback mode**
1	**Reset both Uarts**

Parity Even - When set, transmit parity is even, otherwise transmit parity is odd.

Parity Enable - Enable transmit parity. Parity is not checked on receive.

RTS - Drives the Request to Send output pin. When set, drives CTS pin low.

Transmit Enable - When set transmitter is enabled and will transmit a character as soon as one is written to the Tx Register. When reset, the uart will transmit all characters pending but not transmit new characters.

Receive Enable - When set, the receiver will accept incoming characters.

Loopback - Connects the transmitter data output to the transmitter data input for test purposes. Also connects the CTSN Test command bit to the CTS input pin.

CTSN Test - Used for loopback mode testing.

Reset - When high, holds both uarts in reset.

Status Register and Interrupt Cause Register

Each uart has the following set of status flags in the Status Register and the Interrupt Cause Register.

Bit	Description
23	**UART 0 Rx Ready**
22	**UART 0 Tx Ready for Next Character**
21	**UART 0 Tx Transmitter Empty**
20	**UART 0 Rx Overrun Error**
19	**UART 0 Rx Frame Error**
18	**Reserved**
17	**UART 1 Rx Ready**
16	**UART 1 Tx Ready for Next Character**
15	**UART 1 Tx Transmitter Empty**
14	**UART 1 Rx Overrun Error**
13	**UART 1 Rx Frame Error**
12	**Reserved**

Tx Ready - High when the transmitter is ready to transmit another character. This bit automatically resets when the Tx Register is written.

Tx Empty - High when the transmitter has completed transmission of all characters and has no characters pending. This bit automatically resets when the Tx Register is written.

Rx Ready - High when the receiver has a character ready in the Rx Buffer. This bit automatically resets when the Rx Register is read.

Rx Overrun Error - High when the receiver completes reception of another character before software has read the character in the Rx Register. The Rx Register is overwritten with the new character. Software must reset this bit by writing to the Clear Interrupt Register.

Rx Frame Error - High when the receiver completes reception of a character that has an invalid stop bit. Software must reset this bit by writing to the Clear Interrupt Register.

9.0 SYSTEM FUNCTIONS{ TC "9.0 SYSTEM FUNCTIONS" \l 1 }

Debug Registers

Two pairs of debug registers are provided to allow instruction and data breakpoints to be set.

General Timers

Two general 32-bit timers are provided.

10.0 INTERRUPTS{ TC "10.0 INTERRUPTS" \l 1 }

The Peripheral Function Interrupts service all the peripheral functions, including the ten external expansion interrupts and the other peripheral functions. All peripheral function interrupts are or'ed together and connected to the processor interrupt number Int05. The peripheral function interrupts can be enabled by setting the corresponding mask bit to one in the peripheral function Interrupt Mask Register{ XE "rInterrupt Mask Register" }. When a peripheral function

interrupt is signaled to the LR33300, software can examine the peripheral function Interrupt Cause Register{ XE "rInterrupt Cause Register" } to determine the source of the peripheral function interrupt.

The peripheral function Interrupt Cause Register indicates which interrupt(s) have occurred when the LR33300 interrupt indicates that a peripheral function interrupt has occurred. This register is defined below. The Cause Register is the same as the Status Register but does not show events that are masked by the current Mask Register setting.

Peripheral Function Interrupt Mask Register

Bit	Description
31	A correctable error has occurred
30	An uncorrectable error has occurred
29:24	Reserved
23	UART 0 Rx Ready
22	UART 0Tx Ready for Next Character
21	UART 0Tx Transmitter Empty
20	UART 0 Rx Overrun Error
19	UART 0 Rx Frame Error
18	Reserved
17	UART 1 Rx Ready
16	UART 1 Tx Ready for Next Character
15	UART 1 Tx Transmitter Empty
14	UART 1 Rx Overrun Error
13	UART 1 Rx Frame Error
12	Reserved
11	Memory Write Access Violation
10	Memory Read Access Violation
9:0	Expansion Interrupts

Peripheral Function Interrupt Cause Register

Bit	Description	Type
31	Correctable Error	pulse
30	Uncorrectable Error	pulse
29:24	Reserved	-
23	UART 0 Rx Ready	level
22	UART 0 Tx Ready	level
21	UART 0 Tx Transmitter Empty	level
20	UART 0 Rx Overrun Error	pulse
19	UART 0 Rx Frame Error	pulse
18	Reserved	pulse
17	UART 1 Rx Ready	level
16	UART 1 Tx Ready	level
15	UART 1 Tx Transmitter Empty	level
14	UART 1 Rx Overrun Error	pulse
13	UART 1 Rx Frame Error	pulse
12	Reserved	pulse
11	Write Access Violation	pulse
10	Read Access Violation	pulse
9:0	Expansion Interrupts	level

Expansion Interrupts

Ten external expansion interrupts are provided in addition to the normal LR33300 external interrupts. The ten expansion interrupts are level sensitive and are sampled on the rising edge of system clock. The expansion interrupts are positive polarity. The external interrupts can be masked through the Interrupt Mask Register. The expansion interrupts input pins are named X_INT(9:0){ XE "sX_INT(9:0)" }.

Servicing Peripheral Function Interrupts

All peripheral function interrupts are or'ed into CPU INT05. When this interrupt occurs, software should read the XCause Register to determine which interrupt(s) caused the event. Software then determines which interrupt(s) should be serviced next and performs the necessary service. Before returning from the interrupt service routine, software should clear the pulse type interrupt(s) which have been serviced by writing to the Clear Interrupt address. Writing a one to the particular bit location corresponding to the interrupt to be cleared causes that interrupt to be cleared without affecting other pending interrupts.

Note that the clear interrupt operation will not actually clear the interrupt if the hardware signal that caused the interrupt is still asserted. For this reason, software should cause the interrupting device to deassert the interrupt before executing a clear interrupt operation. In

the case of level type interrupts generated by internal peripheral functions, the peripheral functions are responsible for resetting the interrupt cause bit.

11.0 COMMAND/STATUS REGISTER{ TC "11.0 COMMAND/STATUS REGISTER" \l 1 }

Command Register

Bit	Description (when bit = 1)
31	Enable EDAC
30	Override EDAC Parity Output
29:16	Reserved
15	UART 1 Tx Set Parity to Even
14	UART 1 Tx Enable Parity
13	UART 1 Ready to Send
12	UART 1 Tx Enable
11	UART 1 Rx Enable
10	Reserved
9	UART 0 Tx Set Parity to Even
8	UART 0 Tx Enable Parity
7	UART 0 Ready To Send
6	UART 0 Tx Enable
5	UART 0 Rx Enable
4	Reserved
3	Loopback Tx data to Rx data
2	Drives CTSN in loopback mode
1	Reset both Uarts
0	Reserved

Status Register

Bit	Description (when bit = 1)
31	Correctable EDAC error
30	Uncorrectable EDAC error
29:24	Reserved
23	UART 0 Rx Ready
22	UART 0 Tx Ready for Next Character
21	UART 0 Tx Transmitter Empty
20	UART 0 Rx Overrun Error
19	UART 0 Rx Frame Error
18	Reserved
17	UART 1 Rx Ready
16	UART 1 Tx Ready for Next Character
15	UART 1 Tx Transmitter Empty
14	UART 1 Rx Overrun Error
13	UART 1 Rx Frame Error
12	Reserved
11	Memory Write Access Violation
10	Memory Read Access Violation
9:0	Expansion Interrupts

12.0 MONGOOSE-V VERSUS LR33300{ TC "12.0 MONGOOSE-V VERSUS LR33300" \l 1 }

The Mongoose-V employs an allocate-on-write policy with respect to byte writes to the data cache. This implies that each Store instruction writes to the data cache without first reading the previous entry in the data cache. Due to this change, the CACHD signal is not supported in Mongoose-V.

Mongoose-V generates a read-modify-write bus transaction at the memory interface for byte writes in order to allow EDAC check bits to be updated. The read-modify-write sequence is implemented as a memory read followed by a memory write on the memory bus in order to maintain bus compatibility with the LR33300.

Mongoose-V omits the load scheduling and instruction streaming features of the LR33300. Mongoose-V adds the FPU on-chip. Mongoose-V adds the peripheral function registers at addresses 0xfffe.0xxx.

Mongoose-V is packaged in a 256 pin Ceramic Quad Flat Pack.

The Highz_n signal tristates LR33300 outputs and the peripheral function outputs.

13.0 PACKAGING{ TC "13.0 PACKAGING" \l 1 }

This device is packaged in a 256 pin ceramic leaded chip carrier.

14.0 RELATED PUBLICATIONS{ TC "14.0 RELATED PUBLICATIONS" \l 1 }

(1) Kane and Heinrich. MIPS RISC Architecture, Prentice Hall, 1992.

(2) Farquhar and Bunce. The MIPS Programmer's Handbook, Morgan Kaufmann, 1994.

(3) LR33300 Self-Embedding Processor's User's Manual, LSI Logic, 1993.

16.0 IO SIGNALS{ TC "16.0 IO SIGNALS" \l 1 }

Pin	Name	Description	Pol	Type	Drive
1	VSS	Ground	-	GND	-
2	VDD	Power	-	PWR	-
3	D_31	Data Bus	P	B	6 ma
4	D_30	Data Bus	P	B	6 ma
5	D_29	Data Bus	P	B	6 ma
6	D_28	Data Bus	P	B	6 ma
7	D_27	Data Bus	P	B	6 ma
8	D_26	Data Bus	P	B	6 ma
9	D_25	Data Bus	P	B	6 ma
10	D_24	Data Bus	P	B	6 ma
11	D_23	Data Bus	P	B	6 ma
12	D_22	Data Bus	P	B	6 ma
13	D_21	Data Bus	P	B	6 ma
14	D_20	Data Bus	P	B	6 ma
15	D_19	Data Bus	P	B	6 ma
16	D_18	Data Bus	P	B	6 ma
17	D_17	Data Bus	P	B	6 ma
18	D_16	Data Bus	P	B	6 ma
19	D_15	Data Bus	P	B	6 ma
20	D_14	Data Bus	P	B	6 ma
21	D_13	Data Bus	P	B	6 ma
22	D_12	Data Bus	P	B	6 ma
23	D_11	Data Bus	P	B	6 ma
24	D_10	Data Bus	P	B	6 ma
25	D_9	Data Bus	P	B	6 ma
26	D_8	Data Bus	P	B	6 ma
27	D_7	Data Bus	P	B	6 ma
28	D_6	Data Bus	P	B	6 ma
29	D_5	Data Bus	P	B	6 ma
30	D_4	Data Bus	P	B	6 ma
31	D_3	Data Bus	P	B	6 ma
32	D_2	Data Bus	P	B	6 ma
33	D_1	Data Bus	P	B	6 ma
34	D_0	Data Bus	P	B	6 ma
35	VSS	Ground	-	GND	-
36	RESET_N	System Reset	N	I	-
37	Unused	Unused	-	-	-
38	PEN_N	Parity Enable	N	I	-
39	DP_3	Data Parity	P	B	6 ma
40	DP_2	Data Parity	P	B	6 ma
41	DP_1	Data Parity	P	B	6 ma
42	DP_0	Data Parity	P	B	6 ma
43	Unused	Unused	-	-	-
44	INT_5	Interrupt	P	I	-
45	INT_4	Interrupt	P	I	-
46	INT_2	Interrupt	P	I	-
47	INT_1	Interrupt	P	I	-
48	INT_0	Interrupt	P	I	-
49	INTMASK	Interrupt Mask	P	I	-

50	STALL_N	Stall	N	OZ	6 ma
51	Unused	Unused	-	-	-
52	FRCM_N	Force Cache Miss	N	I	-
53	FPUINT	FPU Interrupt	P	OZ	6 ma
54	Reserved	Connect to ground	P	IGND	-
55	PERR	Parity Error	P	OZ	9 ma
56	Unused	Unused	-	-	-
57	PARAM_OUT	Parametric AND Tree	P	O	6 ma
58	Reserved	Connect to ground	P	IGND	-
59	T2_TOP	Timer 2 Timeout	P	O	6 ma
60	T2ENP	Timer 2 Enable	P	I	-
61	T1ENP	Timer 1 Enable	P	I	-
62	Unused	Unused	-	-	-
63	VSS	Ground	-	GND	-
64	VDD	Power	-	PWR	-
65	VSS	Ground	-	GND	-
66	VDD	Power	-	PWR	-
67	EDACEN	EDAC Enable	P	I	-
68	EDACUE	EDAC Uncorrectable Error	P	O	9 ma
69	EDACCE	EDAC Correctable Error	P	O	9 ma
70	EP_7	EDAC Parity	P	B	6 ma
71	EP_6	EDAC Parity	P	B	6 ma
72	EP_5	EDAC Parity	P	B	6 ma
73	EP_4	EDAC Parity	P	B	6 ma
74	EP_3	EDAC Parity	P	B	6 ma
75	EP_2	EDAC Parity	P	B	6 ma
76	EP_1	EDAC Parity	P	B	6 ma
77	EP_0	EDAC Parity	P	B	6 ma
78	Reserved	Connect to ground	P	IGND	-
79	CPCONDP_3	Coprocessor Condition	P	I	-
80	CPCONDP_2	Coprocessor Condition	P	I	-
81	CPCONDP_0	Coprocessor Condition	P	I	-
82	CACHD_N	Cacheable Data	N	I	-
83	BWIDE_N	Byte-Wide Port	N	I	-
84	BRTKN	Branch Taken	P	B	6 ma
85	BREQ_N	Bus Request	N	I	-
86	Reserved	Leave unconnected	P	ONC	9 ma
87	BGNT	Bus Grant	P	OZ	9 ma
88	BFTCH_N	Block Fetch	N	I	-
89	BFREQ	Block Fetch Request	P	B	9 ma
90	BERR_N	Bus Error	N	I	-
91	BENDN	Big Endian	P	I	-
92	Reserved	Connect to ground	P	IGND	-
93	Reserved	Connect to ground	P	IGND	-
94	Reserved	Connect to ground	P	IGND	-
95	Reserved	Connect to ground	P	IGND	-
96	Reserved	Connect to ground	P	IGND	-
97	Reserved	Connect to ground	P	IGND	-
98	Reserved	Connect to ground	P	IGND	-
99	Reserved	Connect to ground	P	IGND	-
100	VSS	Ground	-	GND	-
101	Reserved	Connect to ground	P	IGND	-
102	VSS	Ground	-	GND	-

103	SYSCLK	System Clock	P	I	-
104	VSS	Ground	-	GND	-
105	Reserved	Connect to power	N	IPWR	-
106	Reserved	Connect to ground	P	IGND	-
107	HIGHZ_N	High Impedance	N	I	-
108	INIT8_N	Initial Value for 8WIDE	N	I	-
109	INIT16_N	Initial Value for 16WIDE	N	I	-
110	DMAR_N	DMA Request	N	I	-
111	Reserved	Connect to ground	P	IGND	-
112	WEPSEL_N	EPROM Select	N	O	9 ma
113	WIOSEL_N	IO Select	N	O	9 ma
114	Reserved	Connect to ground	P	IGND	-
115	DT_N	Data Transaction	N	B	9 ma
116	DRDY_N	Data Ready	N	I	-
117	Reserved	Connect to ground	P	IGND	-
118	MXS_N	Memory Transaction Start	N	OZ	9 ma
119	Reserved	Connect to ground	P	IGND	-
120	RD_N	Read Strobe	N	B	9 ma
121	RT_N	Read Transaction	N	B	9 ma
122	Unused	Unused	-	-	-
123	WR_3_N	Write Enable	N	OZ	9 ma
124	WR_2_N	Write Enable	N	OZ	9 ma
125	WR_1_N	Write Enable	N	OZ	9 ma
126	WR_0_N	Write Enable	N	OZ	9 ma
127	VSS	Ground	-	GND	-
128	VDD	Power	-	PWR	-
129	VSS	Ground	-	GND	-
130	VDD	Power	-	PWR	-
131	RTACKP	Refresh Timer Acknowledge	P	I	-
132	XRTO_N	Refresh Timeout	N	O	9 ma
133	VSS	Ground	-	GND	-
134	XDRAS_N	DRAM Row Access Strobe	N	O	9 ma
135	VSS	Ground	-	GND	-
136	XDCAS_N	DRAM Column Access Strobe	N	O	9 ma
137	VSS	Ground	-	GND	-
138	XDMXS	DRAM Mux Select	P	O	9 ma
139	VSS	Ground	-	GND	-
140	XDMAC_N	DMA Cycle	N	O	9 ma
141	XDOE_N	DRAM Output Enable	N	O	9 ma
142	U1TXRS	UART 1 Transmit Req to Send	P	O	9 ma
143	U1TXDA	UART 1 Transmit Data	P	O	9 ma
144	U1TXCS	UART 1 Transmit Clear to Send	P	I	-
145	U1RXDA	UART 1 Receive Data	P	I	-
146	U0TXRS	UART 0 Transmit Req to Send	P	O	9 ma
147	U0TXDA	UART 0 Transmit Data	P	O	9 ma
148	U0TXCS	UART 0 Transmit Clear to Send	P	I	-
149	U0RXDA	UART 0 Receive Data	P	I	-
150	Reserved	Leave unconnected	-	ONC	-
151	Reserved	Leave unconnected	-	ONC	-
152	VSS	Ground	-	GND	-
153	Reserved	Leave unconnected	-	ONC	-
154	VSS	Ground	-	GND	-
155	Reserved	Connect to ground	-	IGND	-

156	Reserved	Leave unconnected	-	ONC	-
157	Reserved	Connect to ground	-	IGND	-
158	Reserved	Connect to ground	-	IGND	-
159	Reserved	Connect to ground	-	IGND	-
160	Reserved	Leave unconnected	-	ONC	-
161	VSS	Ground	-	GND	-
162	Reserved	Connect to ground	-	IGND	-
163	VSS	Ground	-	GND	-
164	XINT_9	Expansion Interrupt	P	I	-
165	XINT_8	Expansion Interrupt	P	I	-
166	XINT_7	Expansion Interrupt	P	I	-
167	XINT_6	Expansion Interrupt	P	I	-
168	XINT_5	Expansion Interrupt	P	I	-
169	XINT_4	Expansion Interrupt	P	I	-
170	XINT_3	Expansion Interrupt	P	I	-
171	XINT_2	Expansion Interrupt	P	I	-
172	XINT_1	Expansion Interrupt	P	I	-
173	XINT_0	Expansion Interrupt	P	I	-
174	Unused	Unused	-	-	-
175	Reserved	Leave unconnected	-	ONC	-
176	Reserved	Leave unconnected	-	ONC	-
177	Reserved	Leave unconnected	-	ONC	-
178	Reserved	Leave unconnected	-	ONC	-
179	Reserved	Leave unconnected	-	ONC	-
180	Reserved	Leave unconnected	-	ONC	-
181	Reserved	Leave unconnected	-	ONC	-
182	Reserved	Leave unconnected	-	ONC	-
183	Reserved	Leave unconnected	-	ONC	-
184	Reserved	Leave unconnected	-	ONC	-
185	Reserved	Leave unconnected	-	ONC	-
186	Reserved	Leave unconnected	-	ONC	-
187	Reserved	Leave unconnected	-	ONC	-
188	Reserved	Leave unconnected	-	ONC	-
189	Reserved	Leave unconnected	-	ONC	-
190	Reserved	Leave unconnected	-	ONC	-
191	VSS	Ground	-	GND	-
192	VDD	Power	-	PWR	-
193	VSS	Ground	-	GND	-
194	VDD	Power	-	PWR	-
195	Reserved	Leave unconnected	-	ONC	-
196	Reserved	Leave unconnected	-	ONC	-
197	Reserved	Leave unconnected	-	ONC	-
198	Reserved	Leave unconnected	-	ONC	-
199	Reserved	Leave unconnected	-	ONC	-
200	Reserved	Leave unconnected	-	ONC	-
201	Reserved	Leave unconnected	-	ONC	-
202	Reserved	Leave unconnected	-	ONC	-
203	Reserved	Leave unconnected	-	ONC	-
204	Reserved	Leave unconnected	-	ONC	-
205	Reserved	Leave unconnected	-	ONC	-
206	Reserved	Leave unconnected	-	ONC	-
207	Reserved	Leave unconnected	-	ONC	-
208	Reserved	Leave unconnected	-	ONC	-

209	Reserved	Leave unconnected	-	ONC	-
210	Reserved	Leave unconnected	-	ONC	-
211	Reserved	Leave unconnected	-	ONC	-
212	Reserved	Leave unconnected	-	ONC	-
213	Reserved	Leave unconnected	-	ONC	-
214	Reserved	Leave unconnected	-	ONC	-
215	Reserved	Leave unconnected	-	ONC	-
216	Reserved	Leave unconnected	-	ONC	-
217	Unused	Unused	-	-	-
218	VSS	Ground	-	GND	-
219	Reserved	Connect to ground	-	IGND	-
220	VSS	Ground	-	GND	-
221	AS_N	Address Strobe	N	B	9 ma
222	Reserved	Connect to ground	P	IGND	-
223	A_31	Address Bus	P	B	6 ma
224	A_30	Address Bus	P	B	6 ma
225	A_29	Address Bus	P	B	6 ma
226	A_28	Address Bus	P	B	6 ma
227	A_27	Address Bus	P	B	6 ma
228	A_26	Address Bus	P	B	6 ma
229	A_25	Address Bus	P	B	6 ma
230	A_24	Address Bus	P	B	6 ma
231	A_23	Address Bus	P	B	6 ma
232	A_22	Address Bus	P	B	6 ma
233	A_21	Address Bus	P	B	6 ma
234	A_20	Address Bus	P	B	6 ma
235	A_19	Address Bus	P	B	6 ma
236	A_18	Address Bus	P	B	6 ma
237	A_17	Address Bus	P	B	6 ma
238	A_16	Address Bus	P	B	6 ma
239	A_15	Address Bus	P	B	6 ma
240	A_14	Address Bus	P	B	6 ma
241	A_13	Address Bus	P	B	6 ma
242	A_12	Address Bus	P	B	6 ma
243	A_11	Address Bus	P	B	6 ma
244	A_10	Address Bus	P	B	6 ma
245	A_9	Address Bus	P	B	6 ma
246	A_8	Address Bus	P	B	6 ma
247	A_7	Address Bus	P	B	6 ma
248	A_6	Address Bus	P	B	6 ma
249	A_5	Address Bus	P	B	6 ma
250	A_4	Address Bus	P	B	6 ma
251	A_3	Address Bus	P	B	6 ma
252	A_2	Address Bus	P	B	6 ma
253	A_1	Address Bus	P	B	6 ma
254	A_0	Address Bus	P	B	6 ma
255	VSS	Ground	-	GND	-
256	VDD	Power	-	PWR	-