esa
estec

european space agency
european space research
and technology centre

WSD/JG/426
Issue 1
December 1997
Original: English

ERC32 VMEbus Interface (EVI32)

Functional Specification

Prepared by J. Gaisler

Spacecraft Control and Data Systems Division (WS)
Keplerlaan 1 - Noordwijk - The Netherlands
Mail address: Postbus 299 - 2200 AG Noordwijk - The Netherlands

Table of contents

1 INTRODUCTION

1.1 Scope

This document describes the ERC32 VMEbus Interface (EVI32) device. EVI32 provides a complete master and slave VME bus interface for an ERC32 based computer.

1.2 Applicable documents

AD1	*TSC691E: SPARC RT Integer Unit User's Manual*, Rev. H, Temic Semiconductors (F), 1996
AD2	*TSC692E: SPARC RT Floating Point Unit User's Manual*, Rev. H, Temic Semiconductors (F), 1996
AD3	*TSC693E: SPARC RT Memory Controller MEC Rev. A, Device Specification,* MCD/SPC/0009/SE Issue 4, Saab Ericsson Space (S), 1997
AD4	*IEEE Standard for a versatile backplane bus: VMEbus*, IEEE 1014-1987
AD5	*VMEbus Specification Manual,* Rev. C, VMEbus Manufacturers Group, 1985

Note that AD1, AD2 and AD3 can be obtained via the World Wide Web (WWW) from the ERC32 home page at **http://www.estec.esa.nl/wsmwww/erc32.** The home page contains also additional documents and information regarding ERC32.

2 GENERAL DESCRIPTION

The EVI32 device is a 32-bit ERC32 interface circuit designed to interface the ERC32 processor chip set (AD1, AD2 and AD3) to the VMEbus. The EVI32 device fully adheres to the IEEE 1014-1987 VMEbus standard (AD4), and is compatible with the commercial VMEbus specification (AD5). EVI32 can act as a system controller and provides both master and slave interfaces.

EVI32 implements the following functions:

- A32/A24/D32/D16/D8 master and slave interface;
- Interrupt handler;
- Interrupter;
- Single level arbiter (SGL);
- VME bus timer;
- Optimised D16 interface;
- Four mailboxes for multi-processor communication;
- Minimised usage of external buffers;
- On-chip error-detection.

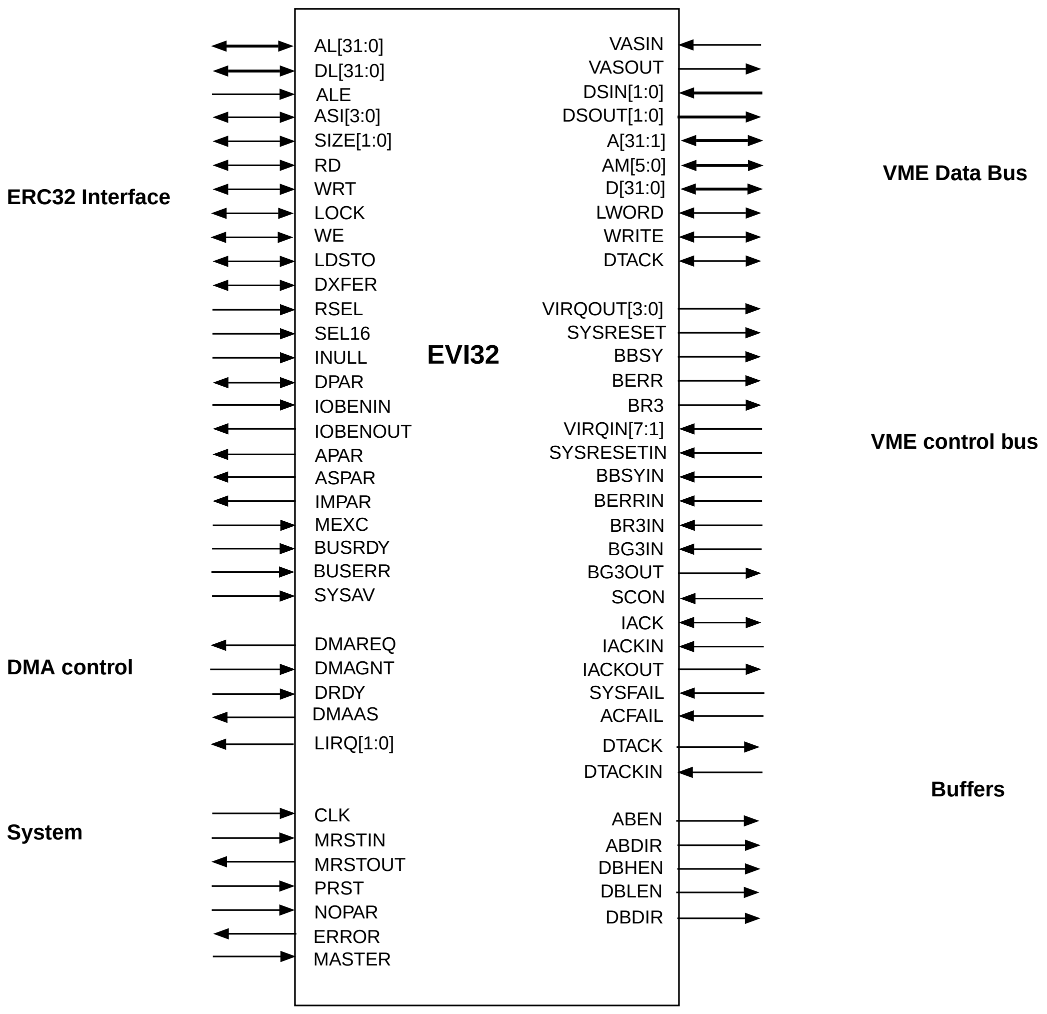


Figure 1: *EVI32 pin definitions (214 signal pins)*

2.1 EVI32/ERC32 interface

The EVI32 interfaces directly to the address, data and control bus of the ERC32, requiring no external components. The EVI32 control registers are accessed by asserting **RSEL** and are typically mapped to one of the ERC32 I/O areas. The VME bus is mapped to the ERC32 extended general area. During slave cycles, the controller preforms DMA cycles to and from the ERC32 memory. If EVI32 is connected to a 16 bit VME data bus (D16), 32-bit and 64-bit ERC32 accesses can be transformed to multiple 16-bit transfers.

2.2 VME interface

The EVI32 provides signals for the VME control bus, address bus and data bus. These signals do not have adequate driving strength to drive the VME bus directly and therefore need external buffers. Depending on the width of the address and data bus of the attached VME bus, 6 to 11 external buffers are required.

2.3 Master cycles

Master cycles are generated by read and write accesses to the extended general area. The ERC32 ASI bits are used to generate the desired VME address modifier. During 8- and 16-bit accesses, the EVI32 performs byte swapping to align ERC32 and VME data. An ERC32 double access (64-bit) will generate two 32-bit accesses.

2.4 Slave cycles

During slave accesses, the EVI32 will perform DMA cycles to the ERC32 memory. Both 8-, 16- and 32-bit accesses can be performed. Block access are allowed for all slave accesses. The internal slave select generator is used to select which VME address the controller will respond to.

2.5 Mailboxes

Four mailboxes are provided for inter-processor communication. The mailboxes consist of a 16-bit register mapped in the short VME address space. An interrupt can be generated upon reading or writing the mailbox.

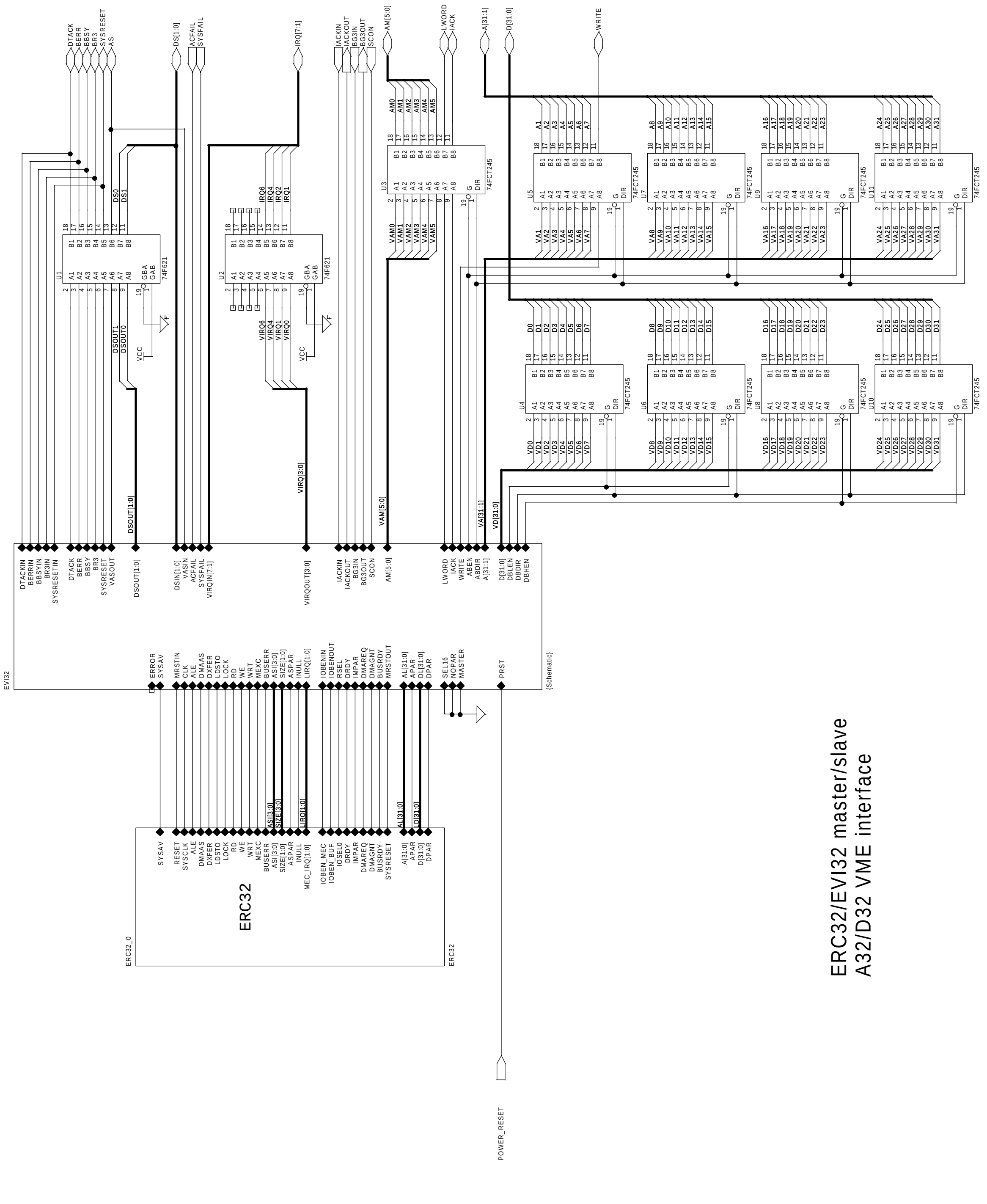
2.6 Interrupt handler and interrupter

The interrupt handler can handle any of the seven VME interrupts. A VME interrupt will generate a local ERC32 interrupt. VME Interrupt acknowledge cycles are performed by ERC32 by reading the interrupt ID of the interrupting device.

Commanded by the ERC32, the interrupter can generate four VME interrupts. The interrupt ID can be individually programmed for all four interrupts.

2.7 ERC32/EVI32 schematic

The following schematic shows how EVI32 should be connected to ERC32.



3 FUNCTIONAL DESCRIPTION

3.1 Performance and operating conditions

The EVI32 device supports as a minimum an operating frequency of 25 MHz with respect to the ERC32 microprocessor interface, targeting 33 MHz.

The EVI32 device supports an optimal data transfer rate on the VME bus.

The EVI32 device withstands as a minimum 50 kRad of total dose radiation, has a low sensitivity to single event upsets, and is immune to heavy ion and proton induced latch-up.

The device can operate under the same electrical and environmental conditions as the ERC32 chip set.

3.2 EVI32 registers

3.2.1 General

EVI32 provides 14 internal registers to control the operation. The registers are accessed by asserting **RSEL**. **RSEL** should be connected to any of the four I/O select signals on the MEC. The registers can be read with any data size but only written with store word (32 bit wide).

3.2.2 Register address map

Table 1 shows the EVI32 registers and their corresponding address.

Register	Function	Access	Address
MSTREG	Master control register	Read/Write	0x00
SLVREG	Slave control register	Read/Write	0x04
SLAREG	Slave address register	Read/Write	0x08
ICREG	Interrupt configuration register	Read/Write	0x0C
IDREG0	Interrupt identification register	Read/Write	0x10
CMDREG	Interrupt command register	Write only	0x14
ISTREG	Interrupt status register	Read/Write	0x18
MBCREG	Mailbox control register	Read/Write	0x1C
MB0REG	Mailbox 0 register	Read/Write	0x20
MB1REG	Mailbox 1 register	Read/Write	0x24
MB2REG	Mailbox 2 register	Read/Write	0x28
MB3REG	Mailbox 3 register	Read/Write	0x2C
SSTREG	System status register	Read/Write	0x30
SRREG	System reset register	Read/Write	0x34

Table 1: *EVI32 registers*

3.3 Master interface

3.3.1 Operation

A VME cycle is started when an access is performed to the decoded part of the extended general area as defined in the master configuration register. The master interface supports one-, two and four-byte transfers as defined in table 2. Un-aligned VME master cycles cannot be generated by ERC32 and are not supported. An ERC32 load (store) double cycle will generate two consecutive quad byte transfers on the VME bus. The VME bus will not be released between the cycles. If a VME cycle fails due to **BUSERR** being asserted, **MEXC** will be generated at the end of the ERC32 cycle. The error cause will be indicated in the system status register.

ERC32 cycle	VME cycle
load byte	read byte
store byte	write byte
load halfword	read double byte
store halfword	write double byte
load word	read quad byte
store word	write quad byte
load double	read quad byte (twice)
store double	write quad byte (twice)
load-store byte	read-modify-write byte
swap	read-modify-write quad word

Table 2: *ERC32 versus VME cycles*

The fields *IOS[4:0]* and *MVA[14:0]* in the master configuration register control (c.f. figure 2) how ERC32 addresses are mapped on VME addresses. The VME bus is mapped on the ERC32 extended general area starting at address 0x80000000. The *IOS* field indicates how much of the extended general area is used. The VME address is generated from the ERC32 address and the *MVA* field. The *MVA* field contains the most significant part of the VME address which is not derived from the ERC32 address. The decoded I/O area can be from 16M (IOS=000) to 2G (IOS=111), programmable in binary steps. Table 3 shows the use of the *MVA* field in relation to the *IOS* field.

IOS	Used bits in MVA	Used I/O area
000	7:0	16 M
001	7:1	32 M
010	7:2	64 M
011	7:3	128 M
100	7:4	256 M
101	7:5	512 M
110	7:6	1024 M
111	7	2048 M

Table 3: *Extended general area mapping*

The VME address modifier (*AM*) is derived from the ERC32 address space identifier (*ASI*). Table 4 shows the mapping between the ERC32 *ASI* codes and the VME address modifier during master access. Two user-defined *AM* codes can be used as defined in the master configuration register.

ASI[3:0]	AM[5:0]	VME Cycle type
0x0	0x29	short non-privileged access
0x1	0x2D	short supervisory access
0x2	0x3F	interrupt acknowledge cycle
0x3	AM0	user defined
0x4	AM1	user defined
0x8	0x3A	standard non-privileged program access
0x9	0x3E	standard supervisory program access
0xA	0x39	standard non-privileged data access
0xB	0x3D	standard supervisory data access
0xC	0x0A	extended non-privileged program access
0xD	0x0E	extended supervisory program access
0xE	0x09	extended non-privileged data access
0xF	0x0D	extended supervisory data access

Table 4: *Master ASI versus AM mapping*

3.3.2 D16 access

The EVI32 includes accelerated D16 access. This is done by converting single or double ERC32 access to multiple double byte VME accesses. This feature is enabled in two ways; if the *D16* bit is set in the master configuration register then all single and double ERC32 accesses are converted to D16 VME accesses. If *D16* is not set, then the **SEL16** input has to be asserted when accelerated D16 access is required. **SEL16** can be connected to an unused **ASI** signal or to an address signal. The accelerated D16 accesses are not possible for quad-byte read-modify-write cycles (ERC32 swap instruction) or interrupt acknowledge cycles.

3.3.3 Block transfer

VME bus block transfers will be performed if the *BT* bit is set in the master control register and an ERC32 cycle would result in more than one VME cycle. As an example, a double load ERC32 access will result in a block transfer of two quad-bytes if the *BT* bit is set, or four double-bytes if accelerated D16 access is set. Block transfer will only be performed when the ERC32 **ASI** is set to 0x8 - 0xF (standard and extended address access).

3.3.4 ERC32 bus time-out control

An ERC32 access to an I/O area will be aborted by the MEC if **BUSREADY** have not been generated within 255 clocks after the start of the access. To avoid aborting a VME transaction due to MEC time-out, the master interface includes a time-out counter.

The time-out counter will terminate the ERC32 VME access by generating **BUSERR** if:

- the VME bus have not been granted to the master interface within 128 clocks
- the VME bus have been granted but the previous slave did not released the bus within 160 clocks
- the addressed slave did not respond within 248 clocks

If the VME access is aborted due to the last case, the VME **BERR** signal will also be asserted.

3.3.5 Master control register

The master configuration register controls the master interface. The register is set to 00000000_H during reset.

26	25	24	23	22 17	16 11	10 8	7 0
RR	ME	D16	BT	AA0[5:0]	AA1[5:0]	IOS[2:0]	MVA[7:0]

MVA	: most significant part of VME address
IOS	: I/O area size
AA0, AA1	: Alternative AM codes
BT	: Block transfer enable
D16	: Accelerated D16 access enable
ME	: Master interface enable
RR	: Release bus on request

Figure 2: *Master configuration register*

3.4 Slave interface

3.4.1 Operation

The slave interface provides access to the ERC32 local memory from an external VME master. The slave address is programmed in the EVI32 slave configuration register. The following cycle types are supported:

VME cycle
address only (no action)
read single, double & quad byte
read single, double & quad byte block
write single, double & quad byte
write single, double & quad byte block
read-modify-write single, double &quad byte
read & write unaligned

Table 5: *Supported slave cycles*

On single and double byte write accesses, the slave interface will perform a read-modify-write cycle to the ERC32 since the DMA interface only allows 32-bit accesses.

Slave decoding is done using the slave address (*SA*) field in the slave address register and the slave size fields (*ESZ* & *SSZ*) in the slave configuration register. During standard address (A24) accesses, *SA[6:0]* is compared to bit [23:17] of the VME address. If equal, the slave is selected. The size field (*SSZ*) defines how many of the bits (starting from the left-most bit) shall be compared. In this way, the size of the slave is between 128k and 16M, and mappable anywhere in the VME A24 address space on an aligned block boundary. The extended area (A32) is decoded in the same way but using *SA[14:0]* and VME address bits [31:17]. The most significant part of the local address is generated in similar fashion and taken from the *LMA* field in the slave address register. The mapped VME area can thereby be mapped on any block aligned address in the full ERC32 address space.

If a master access selects its own slave area, the cycle is terminated with a bus error (**BUSERR**) and the error type is indicated in the system status register. Likewise, if a slave access is done to the part of the extended general area which is used for master VME access, the VME access will be terminated with a bus error (**BERR**). The *ST*, *EX*, *SV*, *NP*, *PE* and *DE* bits in the slave configuration register defines which address modifiers the slave will respond to. Table 6 shows how the *ASI* is generated for different address modifiers.

If the keep bus bit *(KB*) in the slave configuration register is set, then the ERC32 DMA request will be kept during a whole block transfer and during a complete read-modify-write cycle. If *KB* is not set, the local bus will be released to ERC32 between each bus access. The *KB* bit increases the transfer rate but halts the ERC32 for a longer time.

ASI[3:0]	AM[5:0]	VME slave access type	ERC32 cycle type
0x8	0x3A, 0x0A	non-privileged program access	user program access
0x9	0x3E, 0x0E	privileged program access	supervisor program access
0xA	0x39, 0x3B 0x09, 0x0B	non-privileged data access	user data access
0xB	0x3F, 0x3D 0x0F, 0x0D	privileged data access	supervisor data access

Table 6: *Slave access ASI versus AM mapping*

3.4.2 Slave configuration register

The slave configuration register defines the operation of the slave interface. The register is set to 00000000_H during reset.

15	14	13	12	11	10	9	8	7	6 4	3 0
KB	SE	PE	NP	SV	BE	EX	ST	DE	SSZ[2:0]	ESZ[3:0]

SSZ/ ESZ	A24 & A32 size	ESZ	A32 size
0000	128k	1000	32M
0001	256k	1001	64M
0010	512k	1010	128M
0011	1M	1011	256M
0100	2M	1100	512M
0101	4M	1101	1G
0110	8M	1110	2G
0111	16M	1111	4G

KB	: keep bus during block and RMW cycles
SE	: slave enable
PE	: program access enable
SV	: enable supervisor access
BE	: block transfer enable
EX	: enable extended address decoding
NP	: enable non-privileged access
ST	: enable standard address decoding
DE	: data access enable
SSZ	: decoded slave area size for standard addressing
ESZ	: decoded slave area size for extended addressing

Figure 3: *Slave configuration register*

3.4.3 Slave address register

The slave address register defines the address selection and generation of the slave interface. The register is set to 00000000_H during reset.

29 15	14 0
LMA[14:0]	SA[14:0]

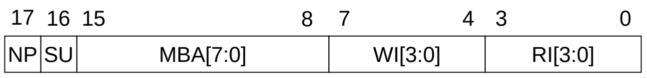
SA	: slave address
LMA	: local address (most significant part)

Figure 4: *Slave address register*

3.5 Mailboxes and system status register

3.5.1 General

The four mailboxes provide a mean for inter-processor communication over the VME bus. Each mailbox consist of a 16-bit register mapped into the short VME address space (A16). The mailbox registers can be accessed with A16/D8/D16 transfers. The mailbox configuration register controls the operation of the mailboxes. The mailbox address field (*MBA*) defines at which address the mailboxes appear in address space, aligned at 256-byte blocks. The *SU* and *NP* bits enable supervisor (AM = 0x2D) and non-privileged (AM=0x29) access. Interrupts can optionally be generated to the local processor when the individual mailboxes are read or written. The *RI* field enables interrupt generation after the mailbox has been read while the *WI* field enables interrupt generation after write. Pending mailbox interrupts can be read from the system status register. A mailbox interrupt is cleared when the local processor reads or writes the corresponding mailbox register.



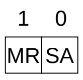
A16 addr.	Register
0000	Mailbox 0
0002	Mailbox 1
0004	Mailbox 2
0006	Mailbox 3
0008	System status

RI*n*	: enable read access interrupt for mailbox *n*
WI*n*	: enable write access interrupt for mailbox *n*
MBA	: mailbox start address (256-byte block)
SU	: enable supervisor access (AM=0x2D)
NP	: enable non-privileged access (AM=0x29)

Figure 1: *Mailbox configuration register*

3.5.2 VME status register

The VME status register is readable from the VME bus and contains two status bits from the local ERC32 system. It is mapped at address 0x8 (A16/D8/D16) directly after mailbox register 3.



SA	: ERC32 SYSAV pin
MR	: MRSTIN pin

Figure 5: *VME status register*

3.6 Interrupt handling

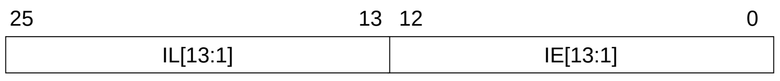
3.6.1 Interrupt controller

The EVI32 contains an interrupt controller that generates two local interrupts to the ERC32, **LIRQ0** and **LIRQ1**. The two local interrupts are generated by the interrupt controller from 13 internal sources. Table 7 shows the interrupt allocation. The two local interrupts are asserted as long as any pending interrupts are present, the corresponding MEC interrupts should therefore be programmed level-sensitive for correct operation.

Interrupt #	Interrupt source
13	ACFAIL
12	Mailbox 3
11	Mailbox 2
10	VME interrupt 7
9	VME interrupt 6
8	VME interrupt 5
7	VME interrupt 4
6	SYSFAIL
5	Mailbox 1
4	Mailbox 0
3	VME interrupt 3
2	VME interrupt 2
1	VME interrupt 1

Table 7: *Interrupt numbering*

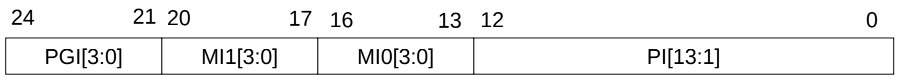
The interrupt level for each interrupt is programmed in the interrupt configuration register.



IE*n*	: Interrupt enable for interrupt *n* (0=disabled, 1=enabled)
IL*n*	: Interrupt level for interrupt *n* (0=level 0, 1=level 1)

Figure 6: *Interrupt configuration register*

The status of pending interrupts can be read from the interrupt status register.



PI*n*	: pending interrupt *n*
MI0	: highest priority pending interrupt in level 0
MI1	: highest priority pending interrupt in level 1
PGI*n*	: pending generated VME interrupt *n*

Figure 7: *Interrupt status register*

3.6.2 VME interrupt handler

The VME interrupt handler can handle all seven VME interrupts. The handling of a VME interrupt is enabled by enabling the corresponding bit in the interrupt configuration register. When an interrupt is detected, the corresponding interrupt pending bit is set in the interrupt status register and a local interrupt is generated to the ERC32. To acknowledge the VME interrupt, an interrupt acknowledge cycle needs to be performed to read the interrupter's STATUS/ID. This is done by doing a read cycle with *ASI* equal to 0x2 to an arbitrary location in the VME area. The read value contains the STATUS/ID of the interrupter that generated the interrupt. During the interrupt acknowledge cycle, address bits 4 - 2 of the local address bus are used to generate bits 3 - 1 of the VME address and are use to identify which interrupt is being acknowledged. The following sequence shows how a VME interrupt is handled:

- A VME interrupt is generated by asserting **VIRQOUT**.
- A local ERC32 interrupt is generated by asserting **LIRQ**.
- ERC32 takes an interrupt trap, disabling further interrupts (at this level).
- The ERC32 interrupt routine reads the EVI32 status register to identify the pending interrupt.
- The ERC32 interrupt routine generates a VME interrupt acknowledge cycle by reading the VME area with ASI=0x02. At this point, any VME ROAK interrupts will de-assert **VIRQOUT**.
- The ERC32 interrupt routine will read the interrupting device's register. At this point, any RORA interrupts will de-assert **VIRQOUT**.

It should be noted that for RORA interrupts, the interrupt service routine must allow at least 2 µs between the last step above and re-enabling of the corresponding MEC interrupt (VME specification rule 4.8).

A D08 interrupt acknowledge cycle must be performed with the least significant bit (A0) of the ERC32 address equal to one, since D08 interrupters respond only to DS(0).

If the VME acknowledge cycle is terminated with a bus error (**BERR** asserted), then the ERC32 load cycle will be terminated with **BUSERR** asserted.

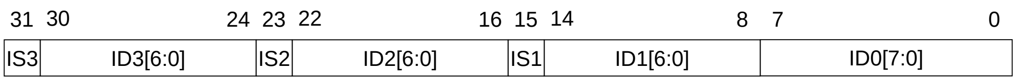
3.6.3 VME interrupter

The interrupter is capable of generating four VME interrupts. Interrupts are generated by writing to the Interrupt command register. Monitoring of generated interrupts is done through the interrupt status register. The VME STATUS/ID for each of the generated interrupts is programmed in the Interrupt identification register and is 7 bits long for VME interrupts 2 - 7, and 8 bits for VME interrupt 1. The VME interrupt number a generated interrupt is mapped on is defined by the interrupt select (*IS*) bits in the interrupt identification register. Pending generated interrupts (not yet acknowledged) can be cleared by writing to the interrupt command register. When a generated interrupt is acknowledged through a VME interrupt acknowledge cycle, the interrupter responds with the STATUS/ID, clears the pending bit and releases the interrupt line. This corresponds to a VME ROAK interrupter.

The interrupter will only respond to an interrupt acknowledge cycle when a generated interrupt is pending for that interrupt. The interrupter will respond to D8/D16/D32 acknowledge cycles, but only provide the least significant 7/8 bits. The remaining data bits will be high, as specified in the VME specification (AD4).

3.6.4 Interrupter identification register

The interrupt identification registers contains the STATUS/ID value for each generated interrupt. The registers are set to 00000000_H during reset.

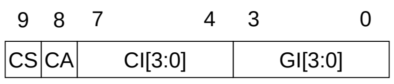


ID0[7:0]	: STATUS/ID code for generated interrupt 0 (VME irq# 1)
ID1[6:0]	: STATUS/ID code for generated interrupt 1 (VME irq# 2/3)
ID2[6:0]	: STATUS/ID code for generated interrupt 2 (VME irq# 4/5)
ID3[6:0]	: STATUS/ID code for generated interrupt 3 (VME irq# 6/7)
IS*n*	: IS*n* = 0 generate even VME interrupt, else odd

Figure 8: *Interrupt identification register*

3.6.5 Interrupt command register

The interrupt generate register is write-only register used to generate and clear interrupts.



GI*n*	: generate interrupt ID*n*
CI*n*	: clear pending generated interrupt ID*n*
CA	: clear ACFAIL interrupt
CS	: clear SYSFAIL interrupt

Figure 9: *Interrupt command register*

3.7 System status register

The system status register indicates the bus error cause and the status of some VME signals. A write to this register will only affect the *HE* and *BE* fields.

14	13 11	10	9	8	7	6 0
HE	BE[2:0]	SC	0	AC	SF	VIRQ[7:1]

BE[2:0]	Bus error cause
000	no error
001	VME bus grant not received (time-out)
010	VME access denied due to HE
011	Master/slave access conflict (try again)
100	VME bus time-out (slave did not release bus)
101	VME access terminated with BUSERR
110	VME access to own slave area
111	access to non-existing EVI32 register

VIRQ*n*	: value of VME IRQn signal
AC	: value of ACFAIL signal
SF	: value of SYSFAIL signal
SC	: value of SCON signal
BE	: bus error cause (ERC32 or VME)
HE	: hardware error

Figure 10: *EVI32 status register*

3.8 VME system controller functions

The arbiter, bus timer and **IACK** daisy-chain driver are only enabled if EVI32 acts as a system controller (when the **SCON** input is asserted).

3.8.1 Arbiter

The arbiter is a single level (SGL) arbiter as defined in the VME specification. To improve latency, bus parking can be enabled by setting the *BP* bit in the master configuration register, which will keep **BBSY** asserted between consecutive accesses until there is a bus request from another master.

3.8.2 Bus timer

A VME bus timer is provided in the EVI32. It consists of an 8-bit counter clocked by the system clock. The timer is reset and started when **VASIN** is asserted, indicating a bus access. If **DTACK** is not asserted before the counter reaches 248, **BERR** is asserted, indicating a bus error. During block transfers, the counter is reset after each **DTACK**.

3.8.3 IACK daisy-chain driver

The IACK daisy-chain driver generates a falling edge on the **IACKOUT** output when any interrupt handler on the VME bus acknowledges an interrupt.

3.9 Reset operation

3.9.1 General

EVI32 has three reset inputs; power-on reset (**PRST**), MEC reset (**MRSTIN**) and VME reset (**SYSRESETIN**). Two reset outputs are provided, one for the MEC (**MRSTOUT**) and one for the VME bus (**SYSRESET**).

EVI32 is internally reset when any of the following conditions are true:
- the **MRSTIN** input is asserted and the **MRSTOUT** output is not asserted;
- the **PRST** input is asserted;
- the **SYSRESETIN** input is asserted while the *SR* bit in the system reset register has the value zero (see section 3.9.2);
- the *ER* bit in the system reset register is set to the value 1 (in this case, neither **SYSRESET** nor **MRTSOUT** is asserted). This can for example be used after a hardware error has been detected, or similar.

During the internal EVI32 reset, all registers are cleared to their default zero values and all outputs (except the **MRSTOUT** output) are placed in their inactive state.

The **MRSTOUT** output is asserted on any of the following conditions:
- the **PRST** input is asserted;
- the **SYSRESETIN** input is asserted while the *SR* bit in the system reset register has the value 0 (see section 3.9.2).

The VME **SYSRESET** output is asserted on any of the following conditions:
- the **PRST** input is asserted;
- *SR* bit in the system reset register is set to the value 1 while the **SCON** input is being asserted;

If the VME **SYSRESET** is generated by setting the *SR* bit in the system reset register, the minimum assertion time of 200 ms for the **SYSRESET** signal (as per AD4) must be controlled via software.

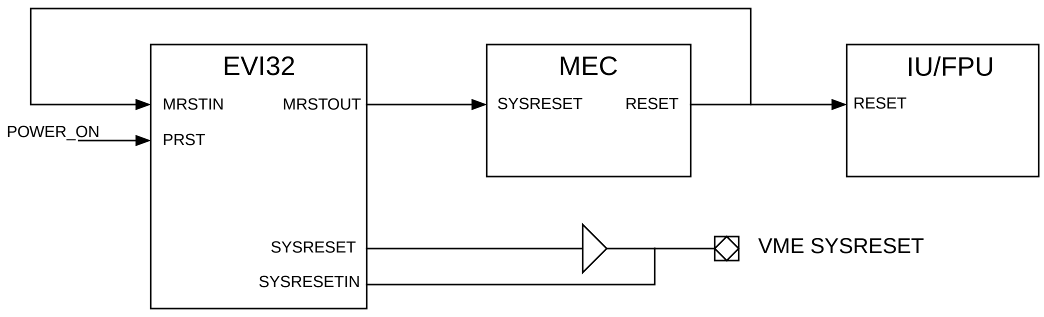


Figure 11: *ERC32/EVI32 reset generation*

3.9.2 System reset register

The system reset register is used to either generate VME **SYSRESET** or to reset EVI32. The *PO* bit can be written with any value, but will only be reset when **PRST** is asserted.

2	1	0
PO	ER	SR

SR	: generate VME SYSRESET (see section 3.9.1)
ER	: reset EVI32 (see section 3.9.1)
PO	: cleared on power-on reset

Figure 12: *System reset register*

3.9.3 Register reset values

All EVI32 registers are cleared to ‘all zeros’ during reset.

3.10 Error detection

To detect SEU errors, all registers are provided with a parity bit which is checked when the data in the corresponding register is used (registers which are used continuously, e.g. master configuration register, are therefore checked permanently). If an error is detected, the **ERROR** output is asserted for two clock cycles and the *HE* bit in the EVI32 status register is set. The **ERROR** signal can be connected to a MEC interrupt input or to an unused error signal (e.g. **IUCMPERR**). While the *HE* bit is set, all accesses to and from the VME bus will be terminated with **BUSERR**. The *HE* bit is cleared by writing to the status register or performing a reset.

The EVI32 is also provided with on-chip comparators to be able to work in master/checker mode. This feature is intended to be used during SEU testing. If the **MASTER** input is de-asserted, the EVI32 will disable its outputs and instead compare the value read from the outputs (except **ERROR**) with the internal value of the outputs. If a mismatch is detected, the **ERROR** output will be asserted.

4 PIN DESCRIPTION

The EVI32 device shall have 32 input signals, 34 output signals, 146 bidirectional signals, as specified in table 8, table 9 and table 10. Together with 27 power pins as specified in table 11, this gives a total pin count of 239 (to be confirmed).

Name	Type	Function	Active polarity
AL[31:0]	Bidir	Address bus (un-latched)	High
DMAAS	Tristate output	DMA address strobe	High
DL[31:0]	Bidir	Data bus	High
DPAR	Bidir	Data bus parity	High
ALE	Input	Address latch enable for IU address	Low
ASI[3:0]	Bidir	IU address space identifier	High
SIZE[1:0]	Bidir	SIZE indicator	High
RD	Bidir	Read strobe	High
WRT	Bidir	Early write strobe	High
LOCK	Bidir	Locked access	Low
WE	Tristate output	Write strobe	Low
LDSTO	Tristate output	Load/store strobe	High
DXFER	Tristate output	Data transfer strobe	High
DMAREQ	Output	DMA bus request	Low
DMAGNT	Input	DMA bus grant	Low
DRDY	Input	DMA data ready	Low
MEXC	Input	Memory exception	Low
BUSRDY	Output	Bus ready	Low
BUSERR	Output	Bus error	Low
IRQ[1:0]	Output	Local interrupt	Low
RSEL	Input	EVI32 register select	Low
IOBENIN	Input	I/O buffer enable input	Low
IOBENOUT	Output	I/O buffer enable output	Low
CLK	Input	MEC SYSCLK	High
PRST	Input	Power-on reset	Low
MRSTIN	Input	Reset input (connect to MEC RESET)	Low
MRSTOUT	Output	Reset output (connect to MEC SYSRESET)	Low
ASPAR	Tristate output	ASI and Size parity	High
APAR	Tristate output	Address parity	High
IMPAR	Tristate output	Control signals parity	High
NOPAR	Input	Disable parity generation	Low
SYSAV	Input	System available	High
ERROR	Output	Hardware error detected	Low
MASTER	Input	Enable master operation	Low
SEL16	Input	Enable accelerated 16 bit access	High
INULL	Input	Integer unit nullify	High

Table 8: *ERC32 interface*

Name	Type	Function	Active polarity
VASIN	Input	Address stable	Low
VASOUT	Output	Address stable	Low
DSIN[1:0]	Input	Data strobes	Low
DSOUT[1:0]	Output	Data strobes	Low
ACFAIL	Input	Power failure	Low
SYSFAIL	Input	System failure	Low
SYSRESET	Output	System reset	Low
SYSRESETIN	Input	System reset	Low
BBSY	Output	Bus busy	Low
BBSYIN	Input	Bus busy	Low
BERR	Output	Bus error	Low
BERRIN	Input	Bus error	Low
DTACK	Output	Data acknowledge	Low
DTACKIN	Input	Data acknowledge	Low
VIRQIN[7:1]	Input	Interrupts	Low
VIRQOUT[3:0]	Output	Interrupts	Low
A[31:1]	Bidir	Address bus	High
D[31:0]	Bidir	Data bus	High
AM[5:0]	Bidir	Address modifier	High
IACK	Bidir	Interrupt acknowledge	Low
LWORD	Bidir	Long word	Low
WRITE	Bidir	Write strobe	Low
BG3IN	Input	Bus grant in	Low
IACKIN	Input	Interrupt ack in	Low
BG3OUT	Output	Bus grant out	Low
IACKOUT	Output	Interrupt ack out	Low
BR3	Output	Bus request	Low
BR3IN	Input	Bus request	Low
SCON	Input	System controller	Low

Table 9: *VME interface*

Name	Type	Function	Active polarity
ABEN	Output	VME address buffer enable	Low
ABDIR	Output	VME address buffer direction	Low
DBLEN	Output	VME data buffer enable (low 16 bits)	Low
DBHEN	Output	VME data buffer enable (high 16 bits)	Low
DBDIR	Output	VME data buffer direction	Low

Table 10: *Buffer control*

Name	Function
VCC[10:0]	Power input
GND[15:0]	Ground

Table 11: *Power pins*

APPENDIX A: ABBREVIATIONS

AD	Applicable Document
ASIC	Application Specific Integrated Circuit
ASSP	Application Specific Standard Product
DMA	Direct Memory Access
EEPROM	Electrically Erasable Programmable Read Only Memory
ERC32	32-bit Embedded Real-time Computing core
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre
EVI32	ERC32 VMEbus Interface
FPU	Floating Point Unit, for ERC32
IEEE	Institute of Electrical and Electronics Engineers
ID	Identification
I/O	Input/Output
IU	Integer Unit, for ERC32
MEC	Memory Controller, for ERC32
ROAK	Release On Acknowledge
RORA	Release On Register Access
SEL	Single Event Latch-up
SEU	Single Event Upset
SGL	Single level
SRAM	Static Random Access Memory
WWW	World Wide Web