Understanding VME Bus

Michael Davidsaver
mdavidsaver@bnl.gov

Rev. 1

1 Basics

1.1 Overview

Introduction

- Goals
  - Become familiar with language of VME operations
  - Interpret VMetro bus analyzer data.

The purpose of this document is to give the reader enough knowledge to begin using the Vanguard VMetro VME bus protocol analyzer. The VMetro is an excellent tool for non-invasive monitoring of bus activity. It can be used to observe software operations for debugging and optimization, as well as detecting problems with bus communications.

Language

VME VERSAmodule Eurocard

Backplane The connectors (slots) and wiring at the back of a VME crate

System Controller Card in slot 1. Special role in bus arbitrartion. Must be populated. Usually a Master.

Master The initiator of a transfer. Usually a CPU card. This is where software "lives".

Slave Other party in a transfer.

In it most commonly used configuration a VME bus has one Master, which is also the system controller, in slot 1. However, VME is a multi-master bus and it is possible to have several cards acting as Masters. Since most devices which act as Masters can also act as Slaves it is possible to use the VME bus to transfer data between two such devices.

VME vs. PCI

VME

- Asynchronous
  - Ordering is key
- Un-clocked
  - Max. bandwidth not well defined

PCI

- Synchronous
  - Timing is key
- Clocked (33 or 66 MHz)
  - Max. bandwidth well defined

While they perform similar functions, the electrical signalling of these two busses is completely difference. This difference arises from the presence or absence of a clock signal. With a clock signal it is possible to say that two signals change "at the same time" in that they both change during one clock period. Without a clock this is impossible.

VME signaling depends on the strict ordering of bus signals. For example, the 31 address lines must all be set before the Address Strobe line is set, and must remain set until after the AS line is cleared.

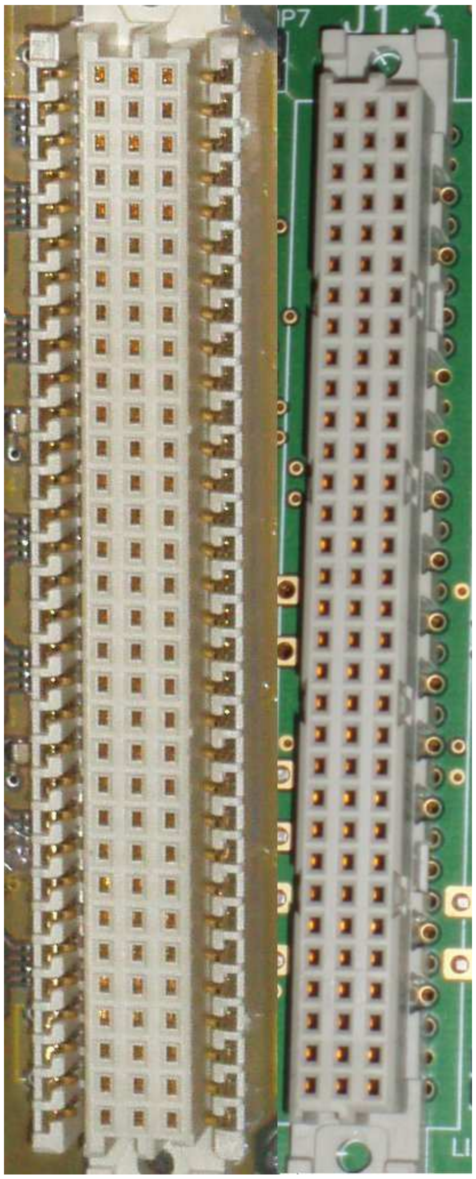
Actions can only be triggered by the rising/falling edge of a single signal.

1.2 Physical

Physical Wires

- Two connectors: P1 and P2
- VME32 (right)
  - 3 rows of 32 pins each.
  - $32 \times 3 \times 2 = 192$
- VME64 (left)
  - 5 rows of 32 pins each.
  - $32 \times 5 \times 2 = 320$

- Can plug VME64 card into VME32 crate and vice versa



The VME standard mandated one 96 pin connector (P1). This supported only 16 and 24 bit operations. The the P2 connector adds lines for 32 bit operations. It also added dedicated lines for rear transition modules. VME64 modified both connectors to add two more rows of pins using a connector which can also accept VME32 cards.

VME32 Signal Groups

The signals mentioned here will be mentioned many times later so it is important to know the names, and relationships between them.

- Addressing (38 pins)
  - Address lines (31 pins)
  - The numerical address (0x1000 or 0x22000000). Depending on the AM not all 31 pins are used.
  - Address modifier (6 pins)
  - Code governing the interpretation of the address lines, and selecting the protocol used (data or block modes)
  - Address Strobe (1 pin)
  - Indicates that all Address and AM have valid values
- Data (36 pins)
  - Data lines (32 pins)
  - The numerical value (0x1000 or 0x22000000). Depending on Data Strobe not all 32 pins are used.
  - Data Strobe (3 pins)
  - Used to select 8, 16, or 32 bit data.
  - Data Ack. (1 pin)
  - During a read/write cycle set to indicate that has been received.
- Interrupt (10 pins)
  - Level lines (7 pins)
  - Acknowledge (3 pins)
- Special (2 pins)
  - Write (1 pin)
  - The value of this line differentiates between read and write cycles
  - Bus Error (1 pin)
  - Set by either Master or Slave to abort the current cycle in reset.
- Total 86 of 192
- Not discussed
  - Multi master arbitration
  - SYS/AC Fail
  - Voltages and Ground
  - Rear transition module

Electrical Signals

- Most VME lines are direct connections between the same pin on every connector.
  - Exception: Interrupt Acknowledge daisy chain line (IACKIO).
- Two state logic
- Can be driven high (1) or low (0) by one (or more) VME devices.
- Lines float high when un-driven.
  - Control signals use inverted logic (True=0) for this reason.
  - eg. Address, data strobes, interrupt, and IACK.

1.3 Logical

Bus Addressing

- VME supports many different Addressing Modifiers.
- AM determines:
  - Address length (16, 24, or all 32 pins)
  - Selects process for read/write cycles.
    * Single cycle (Data mode)
    * Sequential (Block mode)
  - Permissions
    * supervisory/normal
    * Relic of original Motorola 68k
- Slave cards usually implement only some AMs.
- Does not specify data width

It is important to note that each AM is specifying several somewhat unrelated pieces of information. The one thing they have in common is that they can not be specified for each access.

Individual access is via single CPU instructions (load/store on x86) which can in general specify only the exact address, and the data width.

Most VME to host bridges define several “windows” into the the VME bus. Each window maps a block of CPU addresses to a block of VME addresses. Each window takes an Address Modifier as part of its configuration. Each load/store operations on a mapped CPU address causes the bridge to execute a read/write on the VME bus.

Address Modifiers

Code	Name	Address size	Priv.	Cycle type
0x3f	A24sB	24-bit	sup.	block
0x3d	A24sD	24-bit	sup.	data
0x3b	A24nB	24-bit	norm.	block
0x39	A24nD	24-bit	norm.	data
0x29	A16nD	16-bit	norm.	data
0x09	A32nD	32-bit	norm.	data
0x0B	A32nB	32-bit	norm.	block
...	...	...	...	...

This table is an incomplete list of address modifiers.

2 Operations

2.1 Overview

Signal Ordering

- VME bus has no clock.
- Each operation is a sequence of steps.
- A transition between steps is the rising/falling edge of a single control signal.
  - eg. Most operations start on the $1 \rightarrow 0$ transition of the address strobe line.

VME Operations

- VME bus supports many different operations. We will talk about the three most common.

Read Master takes data from Slave

Write Master pushes data to Slave

Interrupt Slave requests service from Master

- The following slides describe the single cycle (data mode) read/write operations. Block mode will not be discussed.

The cycles not discussed here include atomic Read-Modify-Write and A40 lock operations.

2.2 Data Mode Read

Single Cycle Read

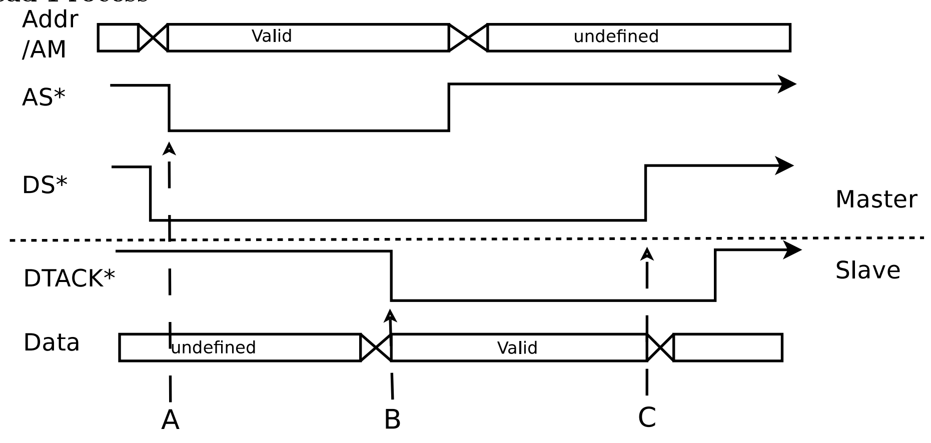
- Operation to read data from a single address.
- Address modifiers: A16nD, A24nD, A32nD, A16sD, A24sD, A32sD
- Data widths: 8-bit, 16-bit, or 32-bit

Note that the common denominator is the 'D' which indicates "Data mode". The alternative is 'B' for "Block mode". Block mode uses an optimized timing for faster reads/writes to sequential addresses.

Memory mapped VME access will almost always be done in Data mode since, even on modern CPU running a tight loop, there is significant dead time between succesive VME accesses. This dead time removes all the benefits of faster modes. Block mode operations are usually reserved for DMA operations.

- Master signals
  - Address Strobe
  - Address and Address modifer
  - Data strobes
- Slave signals:
  - Data Acknowledge
  - Data

Read Process



A Master requests data

1. Asserts Write line to false (1).

2. Master sets Address and Addr. modifier.
3. Uses data strobes to select data width.
4. There are actually 3 DS (DS0, DS1, LWORD) lines which are always asserted in a particular order, and released in the opposite order.
5. Master asserts ($1 \rightarrow 0$) Address Strobe.

B Slave provides data

1. Slave puts Data on Data lines.
2. Asserts ($1 \rightarrow 0$) Data Acknowledge.
    - Master may release AS, addr, and AM after this point

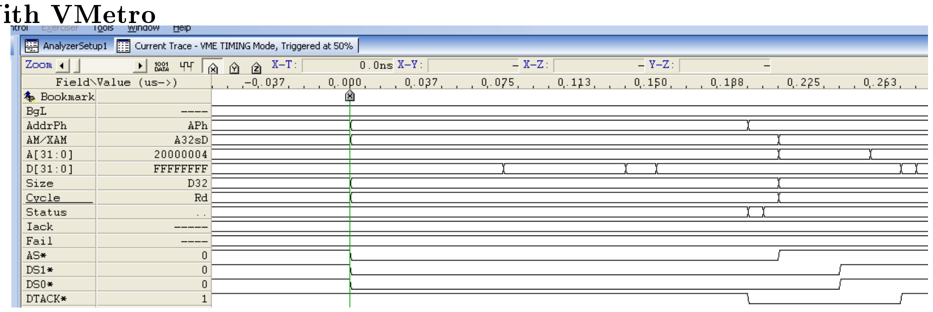
C Master ends cycle

1. Master samples data lines
2. Master releases Data Strobe.
3. Slave releases DTACK and data lines.

Control Signals

- Address Strobe
    - Controlled by Master
    - Asserted to trigger Slave action.
    - Released after DTACK asserted (by Slave)
- Data Strobe
    - Controlled by Master
    - Asserted before Address Strobe.
    - Released after DTACK asserted (by Slave)
- Data Acknowledge
    - Controlled by Slave
    - Asserted after AS asserted (by Master)
    - Asserted after Data lines driven (by Slave)
    - Released after release of DS (by Master)

With VMetro

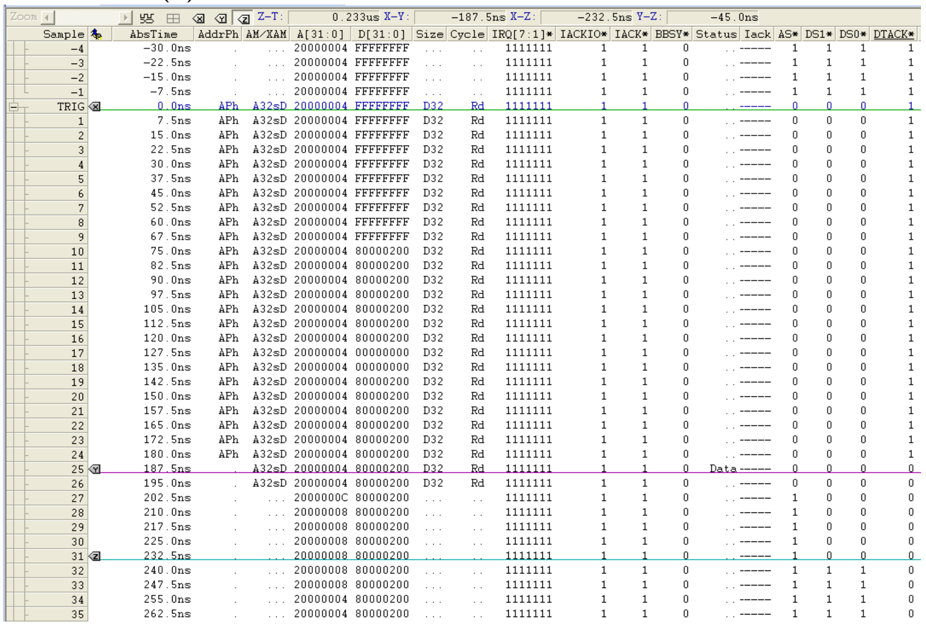


X-T: 0.0ns X-Y: – X-Z: – Y-Z: –

Field\Value (us->)	
Bookmark	
BgL	----
AddrPh	APh
AM/XAM	A32sD
A[31:0]	20000004
D[31:0]	FFFFFFFF
Size	D32
Cycle	Rd
Status	..
Iack	-----
Fail	----
AS*	0
DS1*	0
DS0*	0
DTACK*	1

This screen shot shows the result of capturing a data mode read with the VMetro analyzer and displaying it is waveform mode.

With VMetro (2)



Z-T: 0.233us X-Y: -187.5ns X-Z: -232.5ns Y-Z: -45.0ns

Sample		AbsTime	AddrPh	AM/XAM	A[31:0]	D[31:0]	Size	Cycle	IRQ[7:1]*	IACKIO*	IACK*	BBSY*	Status	Iack	AS*	DS1*	DS0*	DTACK*
-4		-30.0ns	.	...	20000004	FFFFFFFF	...	..	1111111	1	1	0	..	-----	1	1	1	1
-3		-22.5ns	.	...	20000004	FFFFFFFF	...	..	1111111	1	1	0	..	-----	1	1	1	1
-2		-15.0ns	.	...	20000004	FFFFFFFF	...	..	1111111	1	1	0	..	-----	1	1	1	1
-1		-7.5ns	.	...	20000004	FFFFFFFF	...	..	1111111	1	1	0	..	-----	1	1	1	1
TRIG	X	0.0ns	APh	A32sD	20000004	FFFFFFFF	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
1		7.5ns	APh	A32sD	20000004	FFFFFFFF	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
2		15.0ns	APh	A32sD	20000004	FFFFFFFF	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
3		22.5ns	APh	A32sD	20000004	FFFFFFFF	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
4		30.0ns	APh	A32sD	20000004	FFFFFFFF	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
5		37.5ns	APh	A32sD	20000004	FFFFFFFF	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
6		45.0ns	APh	A32sD	20000004	FFFFFFFF	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
7		52.5ns	APh	A32sD	20000004	FFFFFFFF	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
8		60.0ns	APh	A32sD	20000004	FFFFFFFF	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
9		67.5ns	APh	A32sD	20000004	FFFFFFFF	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
10		75.0ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
11		82.5ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
12		90.0ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
13		97.5ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
14		105.0ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
15		112.5ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
16		120.0ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
17		127.5ns	APh	A32sD	20000004	00000000	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
18		135.0ns	APh	A32sD	20000004	00000000	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
19		142.5ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
20		150.0ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
21		157.5ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
22		165.0ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
23		172.5ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
24		180.0ns	APh	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	1
25	Y	187.5ns	.	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	Data	-----	0	0	0	0
26		195.0ns	.	A32sD	20000004	80000200	D32	Rd	1111111	1	1	0	..	-----	0	0	0	0
27		202.5ns	.	...	2000000C	80000200	...	..	1111111	1	1	0	..	-----	1	0	0	0
28		210.0ns	.	...	20000008	80000200	...	..	1111111	1	1	0	..	-----	1	0	0	0
29		217.5ns	.	...	20000008	80000200	...	..	1111111	1	1	0	..	-----	1	0	0	0
30		225.0ns	.	...	20000008	80000200	...	..	1111111	1	1	0	..	-----	1	0	0	0
31	Z	232.5ns	.	...	20000008	80000200	...	..	1111111	1	1	0	..	-----	1	0	0	0
32		240.0ns	.	...	20000008	80000200	...	..	1111111	1	1	0	..	-----	1	1	1	0
33		247.5ns	.	...	20000008	80000200	...	..	1111111	1	1	0	..	-----	1	1	1	0
34		255.0ns	.	...	20000004	80000200	...	..	1111111	1	1	0	..	-----	1	1	1	0
35		262.5ns	.	...	20000004	80000200	...	..	1111111	1	1	0	..	-----	1	1	1	0

Shows the same read cycle in table mode. I personally find this easier to follow, and it can more quickly be transfered to a spreadsheet and sent to a third party.

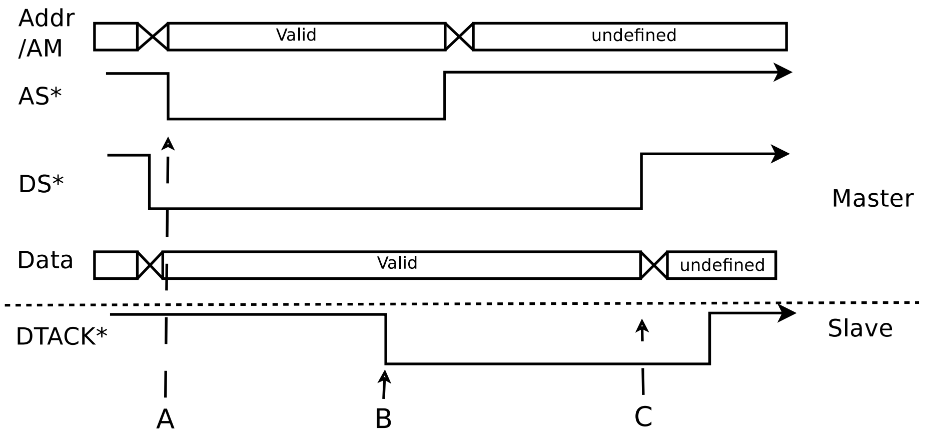
2.3 Data Mode Write

Single Cycle Write

- Operation to write data to a single address.
- Address modifiers: A16nD, A24nD, A32nD (also sup. modes)
- Data widths: 8-bit, 16-bit, or 32-bit

- Master signals
  - Address Strobe
  - Address and Address modifer
  - Data strobes
  - Data
- Slave signals:
  - Data Acknowledge

Write Process



A Master provides data

1. Asserts Write line to true (0).
2. Master sets Address and Addr. modifier.
3. Sets Data and asserts Data Strobes.
4. Master asserts Address Strobe.

B Slave accepts data

1. Slave samples Data.
2. Slave asserts ($1 \rightarrow 0$) Data Acknowledge.
   - Master may release AS, addr, and AM after this point

C Master ends cycle

1. Master releases Data and Data Strobe.
2. Slave releases DTACK.

Control Signals

- Same order as Read cycle
- Different meaning

2.4 Interrupt

VME Interrupts

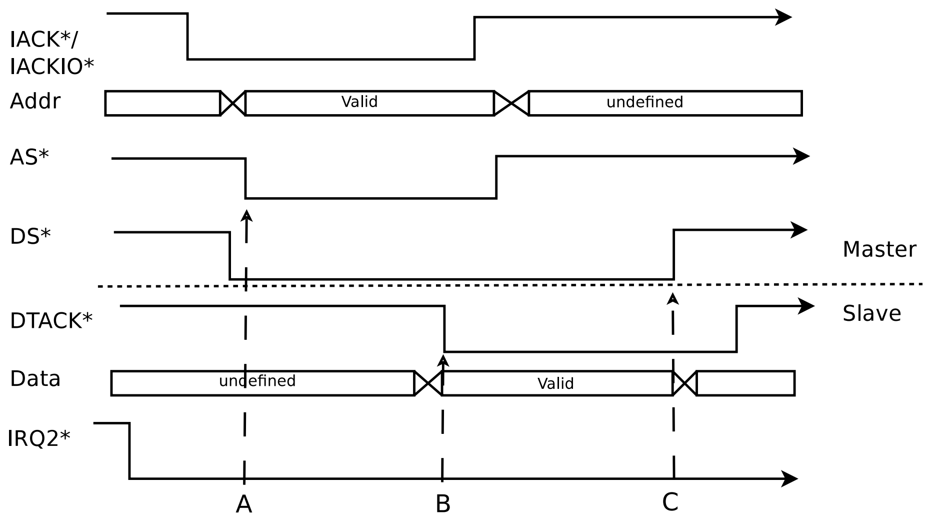
- A VME interrupt is identified by two pieces of information.
  - Level: [1, 7]
  - Vector: [0, 255]
- Levels are physical wires.
- Vectors are read from the device interrupting a given level.
- The Interrupt Acknowledge cycle is similar to a normal data mode read cycle.
- IACK* is set and Address modifier is ignored
- Address bits $1 \rightarrow 3$ are used to indicate which level is being acknowledged

When comparing VME to PCI consider that despite the apparent differences they are basically equivalent. VME has 7 interrupt levels. PCI has 255.

At first glance it might seem that there is no equivalent of VME IRQ vectors. Indeed for PCI devices which do not support "shared interrupts" there isn't. However, PCI devices which do must have a per-device interrupt status register to indicate if it is signaling an interrupt. This is the same information conveyed by the VME vector code.

Note that PCI ≥ 2.2 includes Message Signaled Interrupts (MSI) which are even closer to the VME model. The message data is even called a "vector".

Interrupt Acknowledge Cycle

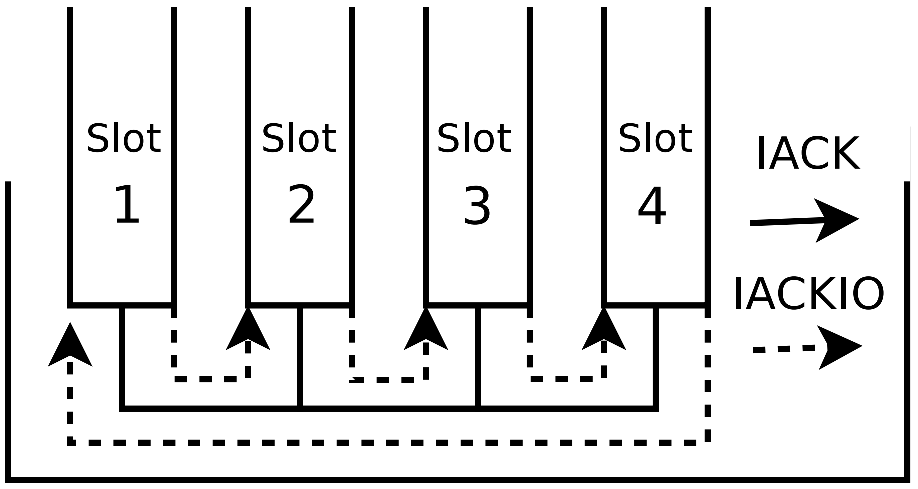


The IACK cycle timing is the same as a data mode read cycle if the Address Modifier lines are replaced with the IACK line. In fact IACK can be considered part of the address modifer. Only three address bits are used (ADDR[1:3]) for the Level (1-7).

It should be noted that the VME standard allows for Vector codes to be 8, 16, or 32 bits. The most common is 8-bit.

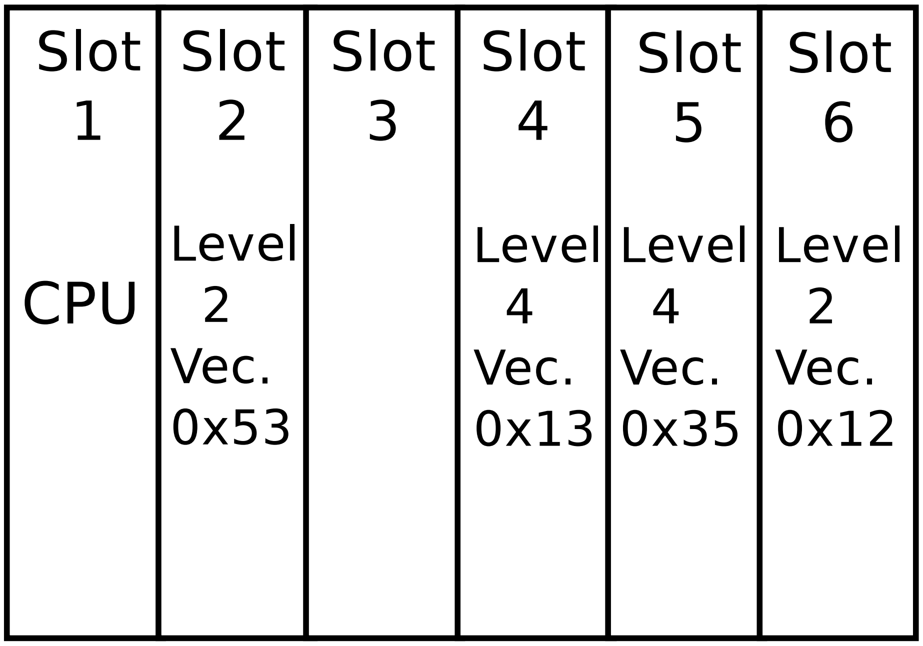
IACKIO

- Each slot has 2 pins: IACKIN and IACKOUT
- The IACKOUT of slot N is connected to IACKIN of slot N+1.
- Non-interrupting devices pass IACKIN through to IACKOUT.
  - Interrupting devices do not.
- Empty slots must have IACKIN shorted to IACKOUT.
  - Newer crates do this automatically
  - Older crates must add/remove jumpers



Interrupt Priority

- Higher levels are serviced first
- When two interrupters have the same level
  - First to receive IACKIN
  - Closest to Acknowledger on the right
- All four devices interrupt at the same time
- What order are they serviced in?



1. Slot 4
2. Slot 5
3. Slot 2
4. Slot 6

3 Errors

When something goes wrong

- VME operations are based on order.
- Control alternates between Master and Slave.
- When something unexpected happens, or doesn't happen, either party can assert a bus error.
- BERR aborts the current cycle and resets both Master and Slave so that a new cycle can begin.
- Most common error is read/write with no response.
  - Timeout waiting for DTACK.
  - Bad for performance
- Bus errors are not normal and should be investigated!

What do bus errors look like?

- Read
  - All bits set (0xffffffff)
  - Usually caused by read of unused address
  - Extra bits set (0xf3ff0443)
  - Timing error caused data lines to be released early. Master sampling data lines at approximately the same time sees noise.
- Write
  - Write has no effect
  - Readback gives unexpected value
- IACK
  - Interrupt on vector 0xff
  - After interrupt service Slave delays release interrupt line causing Master to attempt another IACK cycle, but Slave does not respond.

4 VMetro

4.1 Overview

VMetro VME Bus analyzer

- What is it?
- ~200 channel logic analyzer
- Onboard processor w/ specific knowledge of VME Protocol
- Useful for
  - Non-invasive monitoring of software actions
    - How long does the ISR take?
    - How often is register X read?
    - When is value Y written to register Z?
  - Detecting VME operation errors
- Operates in State or Timing modes.
- The manual is surprisingly complete.
- http://www-cdfonline.fnal.gov/daq/commercial/VG-VME_User_Guide.pdf

State Mode

- Onboard monitoring of VME cycles
- Report summary of operations
  - A32 read from address X
  - A16 write to address Y
- Can buffer a large number of operations
  - Time depends on bus load
- Useful when debugging software

Timing Mode

- Directly store bus signals at sampling rate ($\leq 133MHz$)
- Reports timing
  - AS* became 0 at T_0
  - DTACK became 0 at $T_0 + 2\,\mu s$
- Buffers for a short (fixed) time
- Useful when debugging hardware

Triggering

- The analyzer is constantly sampling like a DSO
- Triggers are specified by patterns involving real signals (AS*) and computed (Cycle type and Status).
- Triggers can be level or edge (0/1 or r/f)

4.2 Setup

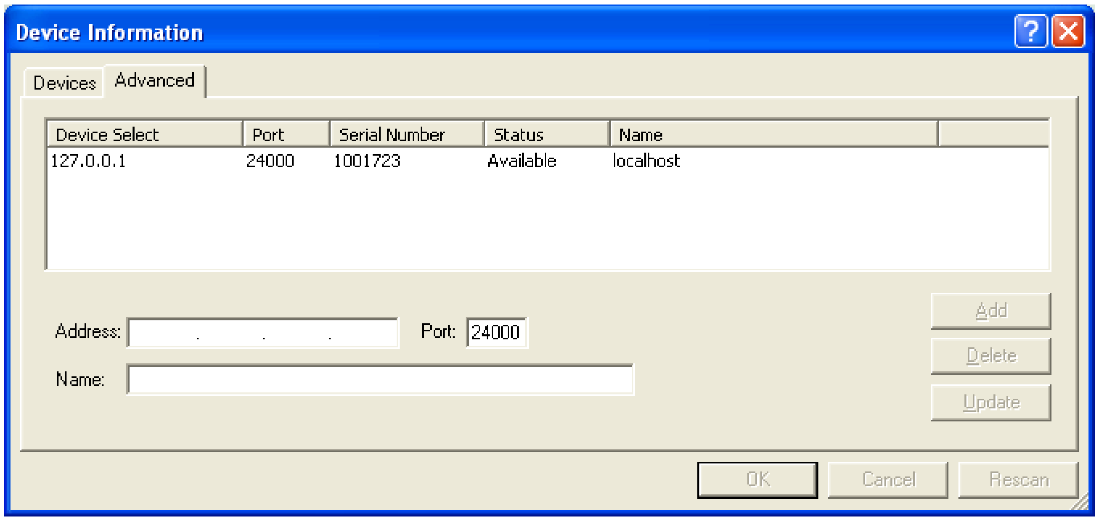
Remote Access

- Uses TCP 24000
  - Also displayed on front panel
- ssh -L 24000:192.168.90.60:24000 controldev32

Connecting Through SSH

1. Ensure that the VMetro has received an IP address by looking at the front panel LED display
2. Start Busview software
3. From *Tools* menu select *Hardware Connection.*
4. Under the *Advanced* tab add an entry for 127.0.0.1 port 24000. Select a name of 'localhost' and click add. This only needs to be done once.
5. In the *Devices* tab. Check next to 127.0.0.1 and click OK.
6. The sequencer controls should now appear.

Connecting Through SSH (2)

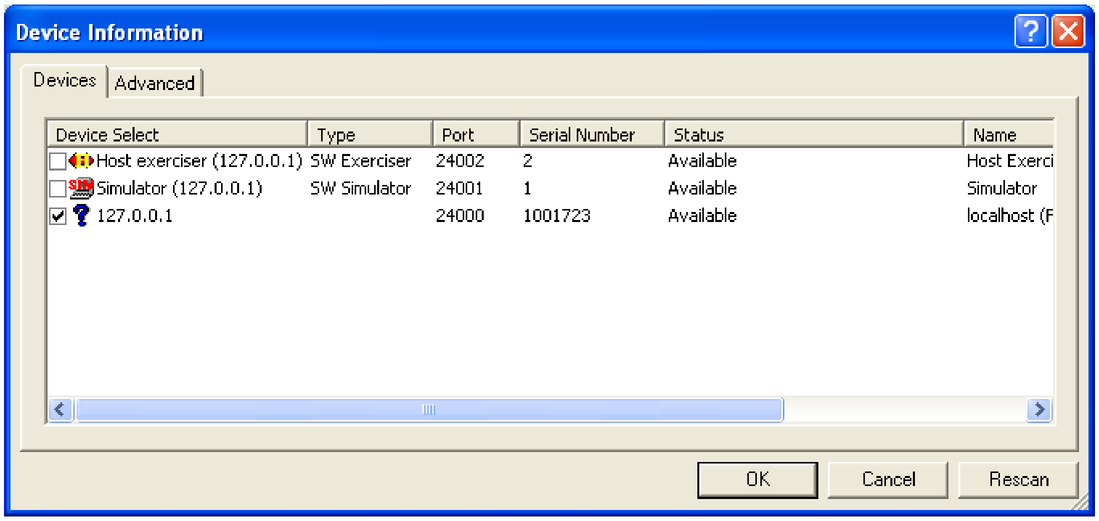


Device Select	Port	Serial Number	Status	Name
127.0.0.1	24000	1001723	Available	localhost

Address: . . .

Port: 24000

Name:



Device Information

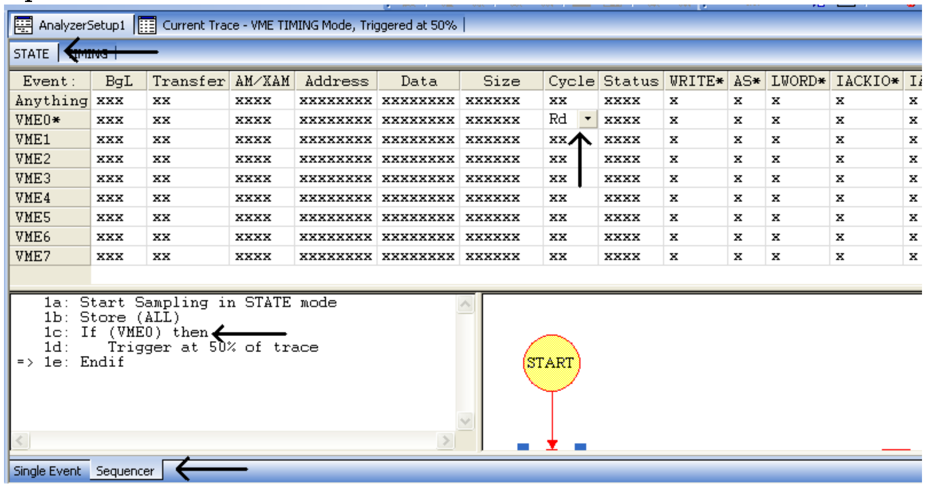
Devices | Advanced

Device Select	Type	Port	Serial Number	Status	Name
☐ Host exerciser (127.0.0.1)	SW Exerciser	24002	2	Available	Host Exerci
☐ Simulator (127.0.0.1)	SW Simulator	24001	1	Available	Simulator
☑ 127.0.0.1		24000	1001723	Available	localhost (F

OK | Cancel | Rescan

Once configured normal operation will be to check *127.0.0.1* and click OK.

Setup



AnalyzerSetup1 | Current Trace - VME TIMING Mode, Triggered at 50%

STATE | TIMING

Event:	BgL	Transfer	AM/XAM	Address	Data	Size	Cycle	Status	WRITE*	AS*	LWORD*	IACKIO*	I
Anything	xxx	xx	xxxx	xxxxxxxx	xxxxxxxx	xxxxxx	xx	xxxx	x	x	x	x	x
VME0*	xxx	xx	xxxx	xxxxxxxx	xxxxxxxx	xxxxxx	Rd	xxxx	x	x	x	x	x
VME1	xxx	xx	xxxx	xxxxxxxx	xxxxxxxx	xxxxxx	xx	xxxx	x	x	x	x	x
VME2	xxx	xx	xxxx	xxxxxxxx	xxxxxxxx	xxxxxx	xx	xxxx	x	x	x	x	x
VME3	xxx	xx	xxxx	xxxxxxxx	xxxxxxxx	xxxxxx	xx	xxxx	x	x	x	x	x
VME4	xxx	xx	xxxx	xxxxxxxx	xxxxxxxx	xxxxxx	xx	xxxx	x	x	x	x	x
VME5	xxx	xx	xxxx	xxxxxxxx	xxxxxxxx	xxxxxx	xx	xxxx	x	x	x	x	x
VME6	xxx	xx	xxxx	xxxxxxxx	xxxxxxxx	xxxxxx	xx	xxxx	x	x	x	x	x
VME7	xxx	xx	xxxx	xxxxxxxx	xxxxxxxx	xxxxxx	xx	xxxx	x	x	x	x	x

```
   1a: Start Sampling in STATE mode
   1b: Store (ALL)
   1c: If (VME0) then
   1d:     Trigger at 50% of trace
=> 1e: Endif
```

Single Event | Sequencer

1. Select State mode by clicking on the *State* tab at the top of the pane
2. Go to the *Sequencer* tab at the bottom of the pane.
3. In the **If** statement of sequencer program double-click on the condition (**Anything** by default). Change it to **VME0**.
4. On the **VME0** line in the *Event* table select a cycle type of **Rd** (read).
5. Start the sequencer from the *Run* menu or by pressing F9.

When the next VME read occurs the VMetro will be triggered. When the trace buffer is full the results will be displayed. To stop and display before the buffer is full select show sequencer from the *Run* menu, or press Shift+F9.

4.3 Stratagy

What to Look For?

- When first inspecting a new system, where to start?
- Check for Bus Errors
  - These should never happen.
- See what is happening most often
  - Does it need to?
  - Can it be more efficient?
- Interrupt handler run time
- All access of a specific register

When to Use

- Evaluating new hardware/driver
  - "But the manual says block mode reads should work."
- Performance measurements
  - Find targets for optimization
  - Measure bus time usage (idle and loaded)
- Wierd problems
  - "The CSR register is 0, but I'm sure I never set it to zero."
  - "Why do command errors only happen occasionally when we always send the same command"
  - "Spurious interrupt on vector 0xff"
- Learning
  - Until you know what it does, you won't know when to use it

End

Questions?