Understanding VME Bus

Michael Davidsaver
mdavidsaver@bnl.gov

Rev. 1
Introduction

Goals

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

Michael Davidsaver mdavidsaver@bnl.gov
Understanding VME Bus
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
VME32 Signal Groups

- Addressing (38 pins)
  - Address lines (31 pins)
  - Address modifier (6 pins)
  - Address Strobe (1 pin)

- Data (36 pins)
  - Data lines (32 pins)
  - Data Strobe (3 pins)
  - Data Ack. (1 pin)

- Interrupt (10 pins)
  - Level lines (7 pins)
  - Acknowledge (3 pins)

- Special (2 pins)
  - Write (1 pin)
  - Bus Error (1 pin)

- 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.
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
### Address Modifiers

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Address size</th>
<th>Priv.</th>
<th>Cycle type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3f</td>
<td>A24sB</td>
<td>24-bit</td>
<td>sup.</td>
<td>block</td>
</tr>
<tr>
<td>0x3d</td>
<td>A24sD</td>
<td>24-bit</td>
<td>sup.</td>
<td>data</td>
</tr>
<tr>
<td>0x3b</td>
<td>A24nB</td>
<td>24-bit</td>
<td>norm.</td>
<td>block</td>
</tr>
<tr>
<td>0x39</td>
<td>A24nD</td>
<td>24-bit</td>
<td>norm.</td>
<td>data</td>
</tr>
<tr>
<td>0x29</td>
<td>A16nD</td>
<td>16-bit</td>
<td>norm.</td>
<td>data</td>
</tr>
<tr>
<td>0x09</td>
<td>A32nD</td>
<td>32-bit</td>
<td>norm.</td>
<td>data</td>
</tr>
<tr>
<td>0x0B</td>
<td>A32nB</td>
<td>32-bit</td>
<td>norm.</td>
<td>block</td>
</tr>
</tbody>
</table>

... ... ... ...
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.
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

- Master signals
  - Address Strobe
  - Address and Address modifier
  - Data strobes

- Slave signals:
  - Data Acknowledge
  - Data
Read Process

A  Master requests data
B  Slave provides data
C  Master ends cycle
**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. Master asserts (1 → 0) Address Strobe.
Read Process

B  Slave provides data

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

C Master ends cycle

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

- Controlled by Master
- Asserted to trigger Slave action.
- Released after DTACK asserted (by Slave)
Control Signals

- Data Strobe
  - Controlled by Master
  - Asserted before Address Strobe.
  - Released after DTACK asserted (by Slave)
Control Signals

- **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)
Understanding VME Bus
With VMetro (2)

### Basics

### Operations

### Errors

### VMetro

#### Overview
- Data Mode Read
- Data Mode Write
- Interrupt

---

#### Michael Davidsaver
- mdavidsaver@bnl.gov

#### Understanding VME Bus
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 modifier
  - Data strobes
  - Data
- Slave signals:
  - Data Acknowledge
Write Process

A  Master provides data
B  Slave accepts data
C  Master ends cycle
**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.
Write Process

B Slave accepts data

1. Slave samples Data.
2. Slave asserts (1 → 0) Data Acknowledge.
   - Master may release AS, addr, and AM after this point
Write Process

C Master ends cycle

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

- Same order as Read cycle
- Different meaning
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 → 3 are used to indicate which level is being acknowledged
Interrupt Acknowledge Cycle

IACK*/IACKIO*
Addr: Valid -> undefined
AS*
DS*
DTACK*
Data: undefined -> Valid
IRQ2*

A -> B -> C

Master
Slave
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?

<table>
<thead>
<tr>
<th>Slot 1</th>
<th>Slot 2</th>
<th>Slot 3</th>
<th>Slot 4</th>
<th>Slot 5</th>
<th>Slot 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Level 2</td>
<td>Level 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vec. 0x53</td>
<td>Vec. 0x13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vec. 0x35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vec. 0x12</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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?

<table>
<thead>
<tr>
<th>Slot</th>
<th>Slot</th>
<th>Slot</th>
<th>Slot</th>
<th>Slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>CPU</td>
<td>Level 2</td>
<td>Level 4</td>
<td>Level 4</td>
<td>Level 4</td>
</tr>
<tr>
<td></td>
<td>Vec. 0x53</td>
<td>Vec. 0x13</td>
<td>Vec. 0x35</td>
<td>Vec. 0x12</td>
</tr>
</tbody>
</table>
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?

<table>
<thead>
<tr>
<th>Slot 1</th>
<th>Slot 2</th>
<th>Slot 3</th>
<th>Slot 4</th>
<th>Slot 5</th>
<th>Slot 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Level 2</td>
<td>Level 2</td>
<td>Level 4</td>
<td>Level 4</td>
<td>Level 2</td>
</tr>
<tr>
<td></td>
<td>Vec. 0x53</td>
<td>Vec. 0x13</td>
<td>Vec. 0x35</td>
<td>Vec. 0x12</td>
<td></td>
</tr>
</tbody>
</table>

1. Slot 4
2. Slot 5
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?

<table>
<thead>
<tr>
<th>Slot 1</th>
<th>Slot 2</th>
<th>Slot 3</th>
<th>Slot 4</th>
<th>Slot 5</th>
<th>Slot 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Level 2</td>
<td>Level 4</td>
<td>Level 4</td>
<td>Level 4</td>
<td>Level 2</td>
</tr>
<tr>
<td></td>
<td>Vec. 0x53</td>
<td>Vec. 0x13</td>
<td>Vec. 0x35</td>
<td>Vec. 0x12</td>
<td></td>
</tr>
</tbody>
</table>

1. Slot 4
2. Slot 5
3. Slot 2
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?

<table>
<thead>
<tr>
<th>Slot</th>
<th>Slot</th>
<th>Slot</th>
<th>Slot</th>
<th>Slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>CPU</td>
<td>Level</td>
<td>Level</td>
<td>Level</td>
<td>Level</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0x53</td>
<td>0x13</td>
<td>0x35</td>
<td>0x12</td>
</tr>
</tbody>
</table>

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

Michael Davidsaver  mdavidsaver@bnl.gov
What do bus errors look like?

**Read**
- All bits set (0xffffffff)
- Extra bits set (0xf3ff0443)

**Write**
- Write has no effect
- Readback gives unexpected value

**IACK**
- Interrupt on vector 0xff
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.

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 ($\leq 133 MHz$)
- 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)
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)
Setup

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

START
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
Questions?